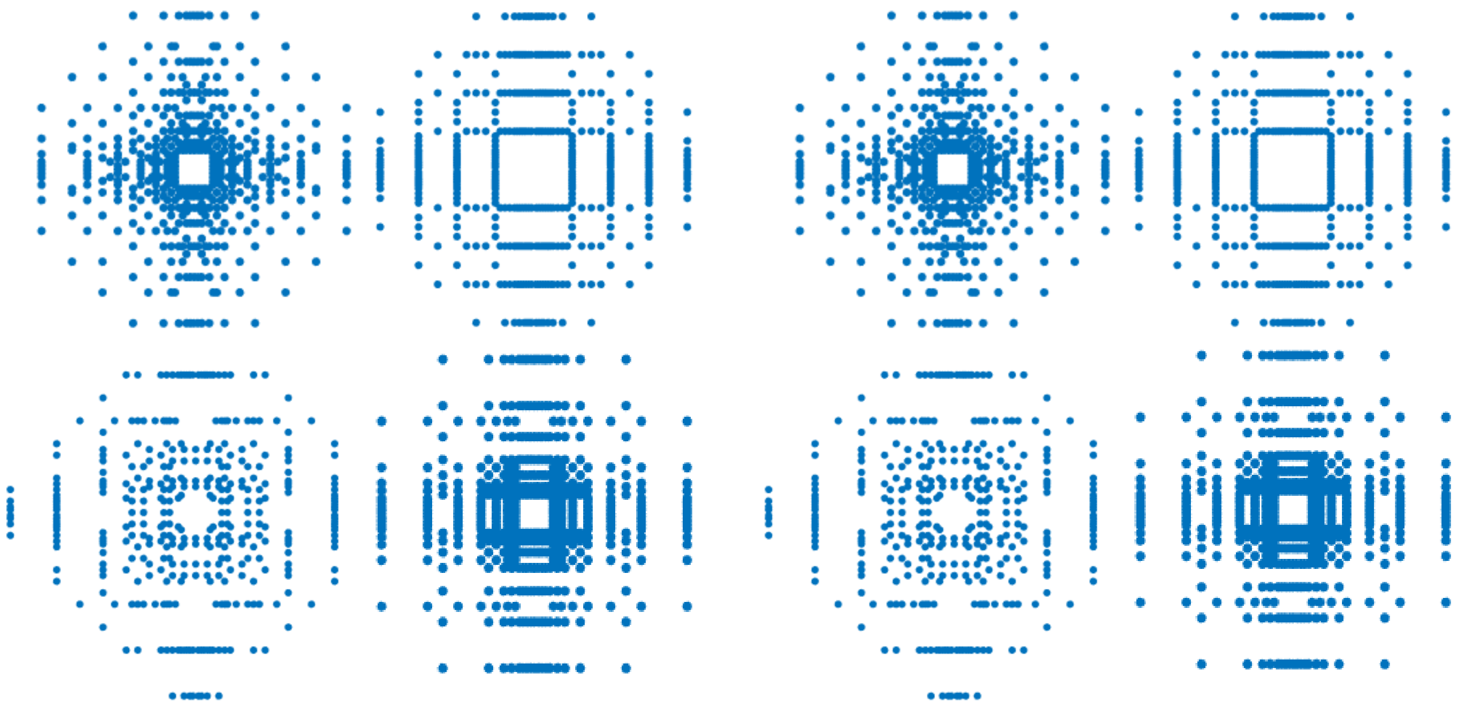
Wavelet-Based Adaptive Mesh Refinement

by

J. Knipping

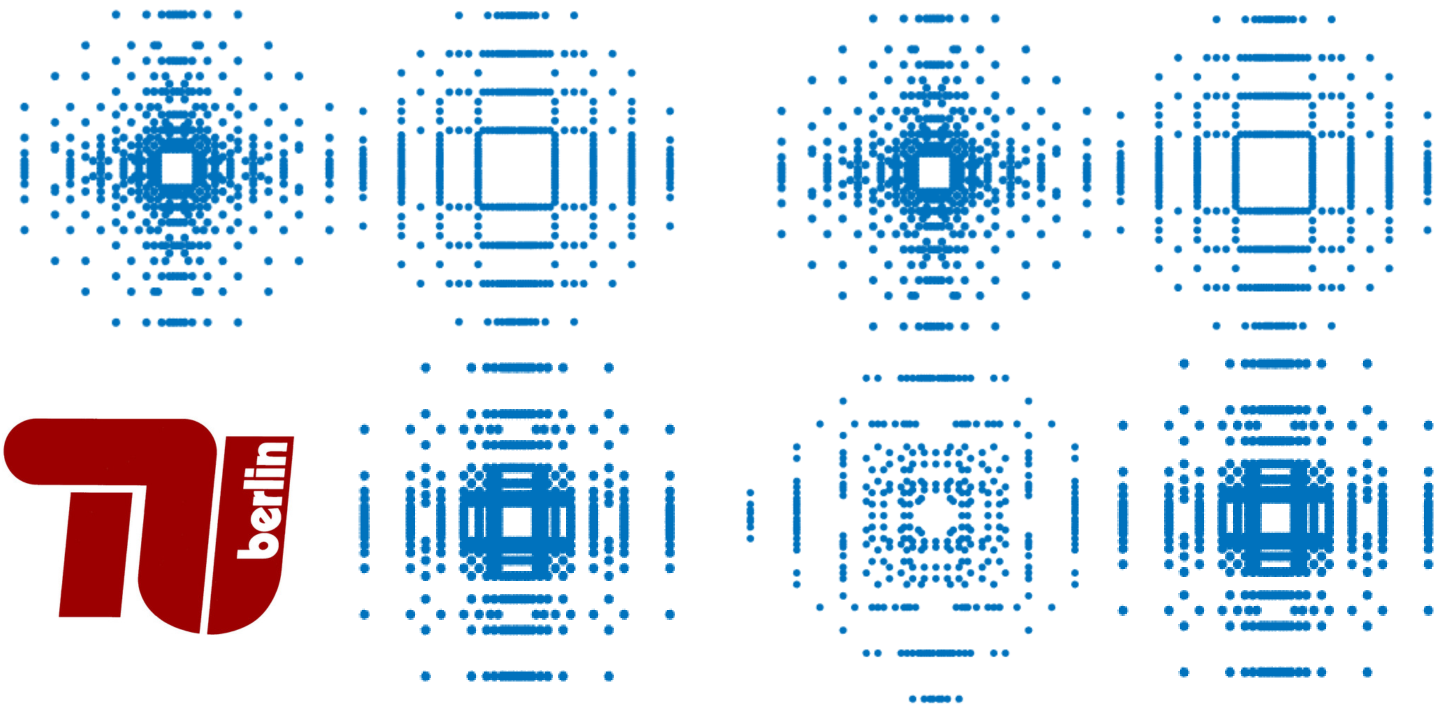
in partial fulfilment of the requirements for the degree of
Master of Science in Applied Mathematics
at the Delft University of Technology,
as part of the master program
COSSE (Computer Simulations for Science and Engineering).

Student number:	4212568	
Project duration:	April 3, 2018 – October 8, 2018	
Thesis committee:	Prof. dr. R. Nabben,	TU Berlin, supervisor
	Prof. dr. ir. K. Vuik,	TU Delft, supervisor
	Prof. dr. G. Bärwolff,	TU Berlin
	Dr. J. Du Toit,	Numerical Algorithms Group (NAG)



WAVELET-BASED ADAPTIVE MESH REFINEMENT

J. KNIPPING



Technische Universität Berlin

Institut für Mathematik
Fachgebiet Scientific Computing

Fakultät II
Straße des 17. Juni 136
10623 Berlin
http://www.math.tu-berlin.de

Master Thesis

Wavelet-Based Adaptive Mesh Refinement

Jorien Knipping

Saturday 6th October, 2018

Matriculation number:	0394686	
Project duration:	April 3, 2018 – October 8, 2018	
Thesis committee:	Prof. dr. R. Nabben,	TU Berlin, supervisor
	Prof. dr. ir. R. Vuik,	TU Delft, supervisor
	Prof. dr. G. Bärwolff,	TU Berlin
Daily supervisor	Dr. J. Du Toit,	Numerical Algorithms Group (NAG)

Hereby I declare that I wrote this thesis myself with the help of no more than the mentioned literature and auxiliary means.

Den Hoorn, Saturday 6th October, 2018

. . . .

(Signature Jorien Knipping)

Abstract

Wavelets have been very popular in the field of image compression and noise reduction. Another interesting application is adaptive mesh refinement. There are many wavelets with various properties, which will have different effects on different applications. There is no consensus on which wavelet is the best option for adaptive mesh refinement. Most commonly used wavelets for adaptive mesh refinement are Donoho's interpolating wavelet and Sweldens wavelet, the latter a lifted version of Donoho's interpolating wavelet. A detailed comparison of both wavelets is done on different data sets. Moreover, different manners of handling the boundaries are tested. An algorithm to construct the meshes using wavelets is tested and optimised.
Donoho's interpolating wavelet with the lower order boundary stencil implementation resulted to be the most accurate, whilst resulting in very high compression compared to the original mesh. Furthermore, changing the adaptivity of the algorithm, which constructs the meshes, turned out to be valuable for fast changing shapes. Lastly, an improvement on the inverse transform during the adaptive mesh refinement had promising results.

Abstrakte

In Bereichen der Bildkomprimierung und Rauschminderung sind Wavelets wohl bekannt. Ein weiteres interessantes, jedoch bisweilen noch weniger bekanntes Verwendungsgebiet ist die adaptive Netzverfeinerung. Es gibt eine Vielzahl von Wavelets, welche verschiedene Eigenschaften haben und in einer breiten Sparte unterschiedlicher Anwendungen ihren Einsatz finden. Bei der Frage, welche Wavelets zur adaptiven Netzverfeinerung am Besten geeignet sind, herrscht Uneinigkeit. In der vorliegend Arbeit, wurden hierfür Donoho's interpolierende Wavelets sowie eine geliftete Version dieser, die sogenannten Sweldens Wavelets gewählt. Zunächst erfolgte ein detaillierter Vergleich der beiden Methoden unter Nutzung unterschiedlicher Datensätze. Im Umgang mit den Randbereichen wurden verschiedene Verfahren angewandt und analysiert. Basierend hierauf wurde ein Algorithmus zur Konstruktion von Wavelets geprüft und optimiert.
Im Vergleich zum Ausgangsnetz erzielt Donoho's interpolierendes Wavelet die höchste Genauigkeit und Komprimierung. Als effizient erweist sich ebenso die Änderung der Anpassungsfähigkeit des Algorithmus für sich schnell ändernde Formen. Letztlich, führt eine verbesserte Rücktransformation während der Netzverfeinerung zu vielversprechenden Ergebnisse.

Preface

This thesis has been written as the final requirement to obtain the degree Master of Science in Scientific Computing at Technical University of Berlin and Master of Science in Applied Mathematics at Delft University of Technology. The degrees are a result from my participation in the master's program, Computer Simulations for Science and Engineering. The research took place in Manchester, England, in collaboration with the Numerical Algorithms Group (NAG). I have learned a lot the past six months, not only about wavelets but also about the perils of doing research. The research started off with the ambition to enhance an existing collocation method, IWOFD. However, the wavelets turned out to be a more comprehensive topic than expected. Moreover, the need arose to explore the performance of different wavelets. Throughout this process I have gotten a lot of guidance and support from many people, to whom I owe my gratitude.

First of all, I would like to thank Jacques Du Toit. All the results are a product of the many debates Jacques and me had. These always helped to put the mathematical problem in perspective and aim for what is important. Furthermore, Jacques was also there if I needed anything beyond my thesis. For example, within a week of moving to England, I was able to cycle through Manchester thanks to Jacques. Jacques took the time to produce interesting data sets to test the wavelets on. Which brings me to Kevin Olson, he has made a very valuable contribution to my thesis by providing more data sets to test. Furthermore, his knowledge and experience in adaptive mesh refinement helped to identify benchmarks in the industry. My gratitude also extends to Craig Lucas, for helping with anything I needed, Tim Schmielau, just for being a kind colleague, and all my other colleagues at NAG.

Second of all, I would like to thank my supervisors. Kees Vuik, thank you for guiding me throughout the project with the bi-weekly meetings. Furthermore, I deem the feedback you gave on my thesis as very valuable. I also like to thank Reinhard Nabben for guiding me in Berlin and giving me the freedom to fill in the thesis exactly as I liked.
I also want to thank Günter Bärwolff, for being part of my thesis committee and for taking the time to read and value this report.

Last of all, I would like to thank my family and friends for their constant support and their belief in my abilities. A big thanks to Marcel and Franziska for their help in the translation of my abstract.

Jorien Knipping,
October 2018

Contents

1 Introduction

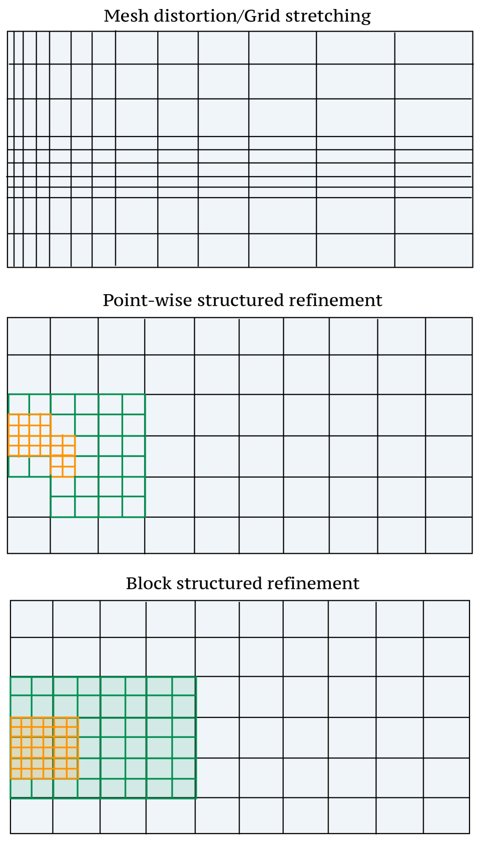
1.1 Motivation

1.1.1 Adaptive mesh refinement

In the field of Scientific Computing there is a big focus on solving Partial Differential Equations (PDEs) as efficiently and fast as possible. In some cases a function has certain regions with very high gradients. To approximate these regions a refined mesh is needed locally, while in other regions a sparser mesh is sufficient. To increase efficiency it makes sense to refine the mesh only in high gradient regions, these regions are also called 'regions of interest'. Finding such a mesh is called *mesh refinement*. Mesh refinement is used to develop a mesh which is sparse and results in an accurate approximation.
In order for such a sparse mesh to be beneficial, there are some important properties to consider. Firstly, with the sparse mesh an accurate approximation of the PDE should be found, i.e., the approximation found with the sparse mesh should be almost the same as the approximation found with a dense mesh. Secondly, the mesh should be highly sparse, i.e., the solving time should be reduced significantly and the memory usage should be decreased. Finally, it is important that the overhead caused by generating the sparse mesh is small.

For time dependent PDEs, there are two types of mesh refinements, namely static mesh refinement (SMR) and adaptive mesh refinement (AMR). Static mesh refinement is used when the overall behaviour of the PDE is known, i.e., the regions which might have a high gradient at certain time-steps are known in advance. In SMR a refined mesh is designed before solving the PDE. Throughout all the time-steps the used mesh stays the same. In adaptive mesh refinement the mesh will be adapted at certain predefined time-steps (this could be every time-step). Adaption means that points in the mesh can both be deleted or added, depending on certain criteria. The SMR approach is preferred if the regions of interest are known or predictable beforehand, as it is faster then AMR. SMR is faster because a mesh has to be designed only once and not multiple times. In this thesis wavelets, will be used for adaptive mesh refinement.
There are many algorithms designed for adaptive mesh refinement. Three main AMR methods are: mesh distortion, point-wise structured refinement, and block structured refinement. The following explanation is based on the lecture [11]. In the three methods the mesh is divided in cells, the size of the cells corresponds to the refinement level. If a cell is big then the cell represents a coarse approximation, if a cell is small this means that the cell represents a fine approximation. Various versions of the three methods exist. The overall idea per method is sketched here. The three methods are visualised

in Figure 1.1. Both the point-wise and block structured mesh refinements are based on different refinement levels. The different refinement levels are coloured green and orange in the figure. Where the orange coloured cells belong to the finest refinement.
In mesh distortion, also called stretching grids, the mesh surface is divided in cells which all have different sizes. The cells are created by lines put into the mesh, which are unevenly spaced. Many lines are placed in regions of interest and less lines in the other regions.
In point-wise structured refinement one starts with big cells, every cell is marked for refinement or not. After the marked cells are refined, these refined cells are again checked for refinement. This will go on until the densest level is reached. This particular refinement method is implemented using a tree structure.
In block structured refinement a whole block of cells is checked for refinement, if refined, then this block, which consists out of blocks in a finer level, is again checked for refinement.


Mesh distortion/Grid stretching
Point-wise structured refinement
Block structured refinement
Figure 1.1: Different adaptive mesh refinement methods.

Similar to the point-wise and block structured mesh refinements, the wavelet-based adaptive mesh refinement method is also based on different refinement levels. The wavelet based AMR is point based instead of cell based. A coarse level of points is added and points in between are checked for refinement. If certain criteria are satisfied for a point at level j that point is added to the mesh. In Figure 1.2 the wavelet based AMR is depicted schematically. The AMR methods described could also be used as SMR methods.

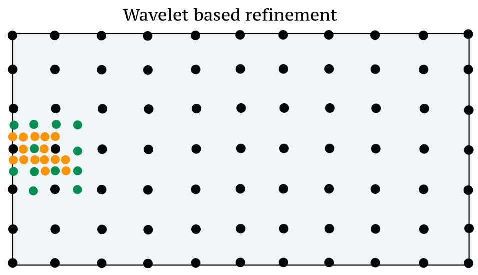


Figure 1.2: Wavelet based adaptive mesh refinement.

1.1.2 Wavelets

Over the last three decades there have been a lot of developments in the wavelet theory. Wavelets are extremely good in compression of data and allow for a fast computation of this compression. 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 utilisations. The two main applications are *adaptive wavelet Galerkin methods* and *adaptive wavelet collocation methods*. In the Galerkin methods wavelets are used to directly discretise the 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 the thesis the focus will be on the 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 fast. As mentioned before, it is important for adaptive mesh refinement that the mesh can be calculated quickly, a fast transform will ensure this. Moreover, because wavelets are good in compression of data they can be used to find a highly sparse mesh.

The thesis is in collaboration with the Numerical Algorithms Group (NAG). Therefore, the focus will be on solving financial parabolic PDEs. A lot of the research in collocation methods have focused mainly on 1 dimension, extensions to multiple dimensions have been mentioned, however they have not been extensively tested. Furthermore, when going in higher dimensions the computational costs can grow significantly. Therefore, it is important to find a wavelet transform which can easily and efficiently be extended to 1, 2 and 3 dimensions.
There are many different wavelet transforms. It is important to consider a particular

wavelet transform, which is fast to compute, has good compression rates and can easily be generalised to multiple dimensions, irregular meshes and different boundary conditions. Finding a perfect wavelet is not possible, therefore finding a wavelet which can easily be extended to multiple dimensions and has good compression rates is considered most important.
Furthermore, special attention should be paid to aliasing. Aliasing has been mentioned in literature as a negative effect of certain wavelets. If aliasing also has a negative result for AMR will be tested in this thesis.
Because meshes are defined on a finite domain, one should take special care of the boundaries. In this thesis two different boundary stencil implementations will be tested, the interpolating boundary stencil and the lower order boundary stencil.
In this thesis two types of wavelets will be considered, namely *Donoho's interpolating wavelet* and a second generation wavelet, *Sweldens wavelet*, which is special case of Donoho's interpolating wavelet. It is interesting to investigate the abilities of the mesh generation per wavelet, e.g., 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 and what is the sparsity rate of the generated meshes. The wavelets will be compared against each other.

1.1.3 Adaptive wavelet collocation methods

Applying adaptive wavelet collocation methods to solve time dependent PDEs exists out of multiple phases. In figure Figure 1.3 the phases are described in a block diagram.

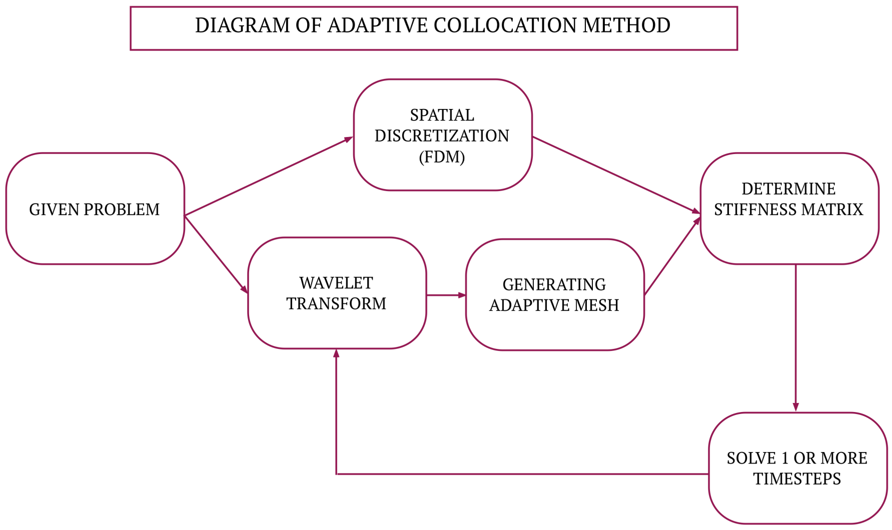


DIAGRAM OF ADAPTIVE COLLOCATION METHOD
GIVEN PROBLEM
SPATIAL DISCRETIZATION (FDM)
WAVELET TRANSFORM
GENERATING ADAPTIVE MESH
DETERMINE STIFFNESS MATRIX
SOLVE 1 OR MORE TIMESTEPS

Figure 1.3: Adaptive collocation method scheme.

Firstly, one starts of with a given problem existing out of a time dependent operator, boundary conditions and an initial condition. In the framework of the method of lines, spatial discretisation will be performed on the time dependent operator and boundary conditions. A discretisation method, which can be used in this phase, is Finite Differences. The initial conditions will be transformed with a wavelet transform, this transform will be used to generate a sparse mesh M. Next, the discretised operator and sparse mesh M will be combined to determine a stiffness matrix A. In the next phase one or more time steps will be solved. The amount of time steps, which will be solved in this phase, will depend on the different type points added in the generation of the mesh M. After this phase the current approximation will be used to determine a new wavelet transform. This wavelet transform will be used to generate a new mesh. A new stiffness matrix will be determined and some time steps will be solved. These last steps will loop until the final time is reached.

1.2 Scope

The focus in this thesis will be on extensively testing the AMR algorithm and the performance of two different wavelets, Donoho's interpolating wavelet and Sweldens wavelet. On both these wavelets two different boundary stencils will be tested, the interpolating boundary stencil and the lower order boundary stencil. All the implementations will be done in Matlab. The mesh algorithm will be based on [30]. The performance of the wavelets and the AMR algorithm will be checked by generating sparse meshes on data sets provided by NAG. These data sets exist out of the approximations of the PDE at every time step. Therefore, AMR can be applied every time step on the approximation restricted to the mesh of the previous time step. In this manner, it is possible to directly check the performance of the AMR through time. There are two different types of data sets. One type of sets are financial PDEs, there are 1 and 2 dimensional data sets of this type. The other type of data sets are based on a hydro PDE, the PDE calculates the pressure, which in turn depends on the density, energy and x, y, z velocity. The PDE has a 2 dimensional and 3 dimensional version.

The goal of this thesis is to extensively test different wavelets in the setting of adaptive mesh refinement.
This leads to the research question:

> *Which wavelet transform is most stable and has the best performance in adaptive mesh refinement?*

1.3 Outline

The report starts of in Chapter 2 with an overview of current literature in wavelet based adaptive mesh refinement. Thereafter, in Chapter 3, the theory behind wavelets will be discussed. In this chapter both Donoho's interpolating wavelet and Sweldens wavelet will be introduced. Subsequently, in Chapter 4 the adaptive mesh refinement algorithm is stated. This chapter will start with an explanation of mesh refinement using wavelets, after which the mesh refinement will be extended to adaptive mesh refinement. In Chapter 5, the tests will be explained and the results will be analysed and evaluated. Finally, Chapter 6 contains the conclusion. Furthermore, prospects for future research will be given.

2 Existing results in literature

In this thesis the focus will be on wavelet based adaptive mesh refinement. There are many applications for wavelets, therefore many wavelets have been developed. In section 2.1 a summary of existing literature on the different wavelets is given. Subsequently, literature on wavelet based mesh refinement methods are discussed in 2.2. Finally, an overview of collocation methods is given in 2.3. In this last section it will also be discussed which wavelets are used for AMR.

2.1 Wavelets

Cohen, Daubechies and Feauveau lay the fundamental backgrounds behind orthogonal and biorthogonal wavelets, [4, 5]. Furthermore, Cohen gives a detailed description of first generation wavelets in numerical analysis, [3]. Donoho introduces the interpolating wavelets in [9]. The ability of wavelets to perfectly represent functions is depended on the vanishing moments, [26, 28]. Sweldens introduced a faster and more intuitive manner to transform functions in their wavelet representation, [6, 23, 27, 25]. This new transform resulted in second generation wavelets, [15, 24].

2.2 Adaptive mesh refinement

Algorithms for wavelet based adaptive mesh refinement are introduced in [13] and [16]. Vasilyev and Bowman combine these methods to introduce an AMR, for non equidistant meshes and second generation wavelets, [30]. The algorithm can be changed in the way it thresholds, [14], or in the way it adapts, i.e., how adjacent points are added. Applying AMR on non equidistant meshes leads to stability issues, which are related to the update step, boundary stencil implementation and lack of orthogonality, [15, 22]. Equidistant meshes are tested, however considering the stability issues of non equidistant meshes turned out to be valuable for the equidistant case.

2.3 Collocation methods

There has not been a lot of comparison between the performance of different wavelets for AMR. Rather, a lot of research has been done in collocation methods and applying wavelet based AMR to all kind of PDEs. Wirasaet researched Donoho's interpolating wavelet with the interpolating boundary stencil for AMR, [32]. He considered the two type of adjacent points also considered in section 4.2.3. Furthermore, both equidistant

and non equidistant meshes have been tested on 1D and 2D problems. The mesh refinement algorithm is similar to [30] and the PDEs are solved using Finite Differences. This collocation method has been tested on different problems, [18, 19, 20].
A similar collocation method which focuses on Finance PDEs has been tested in [7, 8]. This collocation method does not use the lifting scheme for wavelet transformation. Further, Donoho's interpolating wavelet, the interpolating boundary stencil implementation, the mesh refinement of [30] and Finite Differences are used. The testing only has been performed on equidistant meshes.
Vasilyev and others have designed a similar collocation method, [1, 2, 17, 21, 29, 30]. The only difference is that Sweldens wavelet is used instead of Donoho's interpolating wavelet. Tests have been done on all kind of PDEs, mainly 1 dimensional problems and on occasion 2 and 3 dimensional problems and both equidistant and non equidistant. In [31] the possibility to use Multigrid instead of Finite Differences is investigated.

In the literature both Donoho's interpolating wavelet and Sweldens wavelet have been used for AMR. However, the two wavelets have not been directly compared. Because it is not clear which wavelet is favoured, both wavelets will be compared on various data sets in 1, 2 and 3 dimensions. Furthermore, in the literature the interpolating boundary stencil is always used. In this thesis the lower order boundary stencil will be tested, because [15] and [22] suggest that the behaviour of this boundary stencil implementation is more stable.

3 Wavelet analysis

3.1 Wavelets in general

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 the 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 the localised wavelets. This is typically done in a setting of a family of nested subsets, called multiresolution analysis (MRA), this will be described more thoroughly in the section 3.2. Wavelets can successfully represent functions which have certain smoothness properties, these particular properties differ per wavelet. However, wavelets are always able to represent polynomials up to a certain order N. This order can be steered in the initial phase of generating the wavelets.

There are many different wavelets with different properties. However, in the literature the focus was mostly on finding wavelets with a fast and parallelisable transform. This is an important property, because the usage of wavelets should deliver a significant speed up, hence the overhead of transforming a function should be small. The process of transforming a function to its wavelet transform will be called the fast transform, this will be discussed more detailed in section 3.3.
The wavelet transform exists out of two different transforms, namely the forward and inverse transform. The forward transform will present the function compactly. The inverse transform will use the compact representation to iteratively determine the original function. During the process of forming the fast transform, so called 'detail' coefficients are determined. Intuitively, the detail coefficient at a certain position represents the ability of the wavelet transform to present the actual value of the function by interpolation. If a point can be perfectly reconstructed the detail coefficient of that point will be zero. The higher the absolute value of the detail coefficient the less accurate the wavelet interpolation. Therefore, in the adaptive wavelet collocation methods the value of the detail coefficients is used for thresholding. In other words, if a detail coefficient is low the wavelet transform can accurately determine the value at that given position. Hence, it is not needed to keep the function value at that position. The mathematical derivation of this property can be easily demonstrated for Donoho's interpolating wavelet, see subsection 3.4.4.

3.1.1 Different types of wavelets

Two type of wavelets which have been researched extensively in the literature are orthogonal and biorthogonal wavelets. These wavelets will be introduced in this chapter, because they are helpful in defining the wavelets clearly. A typical example of an orthogonal wavelet is *Daubechies N wavelet*, where N indicates the amount of vanishing moments, more information on the vanishing moments is given in 3.5. This particular wavelet is generated using the Fourier transform, more information can be found in [5]. The *Haar wavelet* is an easy, almost trivial, example of a wavelet and has both orthogonal and biorthogonal versions. Although it is easy to understand and implement, the Haar wavelet is not very good in approximating functions. Another example of a biorthogonal wavelet is the *Cohen-Daubechies-Feauveau wavelet* as described in [4]. The Daubechies N wavelets, orthogonal and biorthogonal Haar wavelets and the Cohen-Daubechies-Feauveau wavelets will be used as examples in section 3.2.
Another famous wavelet is *Donoho's interpolating wavelet*, [9]. The Donoho's interpolating wavelet is based on the Deslauriers and Dubuc interpolation and is closely related to biorthogonal wavelets. Donoho designed his wavelet such that the fast transform is highly parallelisable. All the coefficients in the fast transform can be calculated independently of the other coefficients in the same level. This wavelet will be explained more extensively in section 3.4.

Donoho's interpolating wavelet gave an introduction to a new type of wavelets, as the wavelets did not need to be generated by the use of the Fourier transform. Sweldens explicitly defined this difference and called the wavelets, generated without the use of a Fourier transform, *second generation wavelets*. Implying that the other wavelets will be called *first generation wavelets*. The second generation wavelets are based on biorthogonal wavelets. The fast transform looks significantly different, Sweldens altered the fast transform by introducing *the lifting scheme*, [23], [24] , [25] and [27]. More information on this type of wavelets and their construction is given in section 3.6.

3.2 Multiresolution analysis

In this section mathematical fundaments of the wavelets are introduced, the setting will be restricted to 1D.
Wavelets are used to approximate functions, this is done by projecting the function to a *multiresolution space*. In the general wavelet theory the multiresolution space is an approximation of the Hilbert space $L^2(\mathbb{R})$. This approximation is done by an infinite set of nested subspaces:

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

Where a higher value for j implies a more precise space.
The manner in which I will define the multiresolution analysis is based on the works of Cohen, Sweldens, Dempster and de Wiart, [3], [24], [8] and [7]. Note that there is more than one way to define the MRA.

Definition 1 - Multiresolution analysis (MRA)

A multiresolution analysis is a sequence of closed subspaces of $L^2(\mathbb{R})$, such that the following properties are satisfied:

(i) The sequence is nested, i.e., for all $j \in \mathbb{Z}$,

$$V_j \subset V_{j+1}. \tag{3.1}$$

(ii) The spaces are related to each other by dyadic scaling, i.e.

$$f \in V_j \iff f(2\cdot) \in V_{j+1} \iff f(2^{-j}\cdot) \in V_0. \tag{3.2}$$

(iii) The union of the spaces is dense, i.e. for all f in $L^2(\mathbb{R})$

$$\lim_{j\to+\infty} ||f - P_j^0 f||_{L^2} = 0,$$

where P_j^0 is the orthonormal projection onto V_j.

(iv) The intersection of the spaces is reduced to the null function, i.e.

$$\lim_{j\to-\infty} ||P_j^0 f||_{L^2} = 0. \tag{3.3}$$

(v) There exists a function $\varphi \in V_0$ such that the family

$$\varphi(\cdot - k), \qquad k \in \mathbb{Z},$$

is a Riesz basis of V_0.

Note, because $\varphi(x) \in V_0$, from (3.1) it follows that $\varphi(x) \in V_1$, which implies $\varphi(x/2) \in V_0$ due to (3.2). The $\varphi(\cdot - k)$ form a Riesz basis of V_0, which yields:

$$\varphi(x/2) = \sum_{k\in\mathbb{Z}} \alpha_k \varphi(x-k),$$

with $(\alpha_k)_{k\in\mathbb{Z}} \in l^2(\mathbb{R})$. This leads to the *dilation equation*,

$$\varphi(x) = \sum_{k\in\mathbb{Z}} h_k \varphi(2x-k), \tag{3.4}$$

the solution $\varphi(x)$ is called the *mother scaling function* and $H = (h_k)_{k\in\mathbb{Z}}$ is called the *scaling filter*.

In order to have normality the integral of the mother scaling function should be 1, which gives a condition on the scaling filter H.

$$\begin{aligned}1 &= \int \varphi(x)dx = \int \sum_{k\in\mathbb{Z}} h_k \varphi(2x-k) = \sum_{k\in\mathbb{Z}} h_k \int \varphi(2x-k)dx \\ &= \sum_{k\in\mathbb{Z}} h_k \int \frac{1}{2}\varphi(u)du = \frac{1}{2}\sum_{k\in\mathbb{Z}} h_k \\ &\implies \sum_k h_k = 2.\end{aligned}$$

Due to (3.2) any $f_j \in V_j$ can be written as $f_j = 2^{j/2} f_0(2^j\cdot)$, where $f_0 \in V_0$ so $f_0 = \sum_k \alpha_k \varphi(\cdot - k)$. Hence any $f_j \in V_j$ can be written as $f_j = 2^{j/2}\sum_k \alpha_k \varphi(2^j \cdot - k)$. This leads to the following basis functions for V_j, $j \in \mathbb{Z}$:

$$\varphi_{j,k}(x) = 2^{j/2}\varphi(2^j x - k) \qquad k \in \mathbb{Z}.$$

The $\varphi_{j,k}$ form a Riesz basis for the space V_j, a property for a Riesz basis is that:

$$C_1 ||f||^2_{L^2} \leq \sum_{k\in\mathbb{Z}} |\langle f, \varphi_{j,k}\rangle|^2 \leq C_2 ||f||^2_{L^2}. \tag{3.5}$$

Therefore, any $f \in V_j$ can be written as:

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

where $\langle\cdot,\cdot\rangle$ denotes the dot product of $L^2(\mathbb{R})$.
Thus define the projection P_{V_j} of any $f \in L^2(\mathbb{R})$ as

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

In order to present any function f as a sum of scaling functions, the so called *smoothing coefficients*, $s_{j,k}$, need to be determined. These coefficients will be determined using the fast transform, which will be introduced in a later stage.

3.2.1 Orthogonal wavelets

The MRA description described above will be extended for the case of orthogonal wavelets. In this case orthogonality implies that two basis functions $\varphi_{j,k}$ of V_j are orthogonal. In other words $< \varphi_{j,k}, \varphi_{j,l} >= \delta_{k,l}$, where $k, l \in \mathbb{Z}$ and δ the Kronecker delta. The orthogonality condition leads to a condition on the filter H, namely $\sum_{k\in\mathbb{Z}} h_k h_{k-2l} = \delta_{0,l}$

for all $l \in \mathbb{Z}$, which followed from:

$$\begin{aligned}
\delta_{0,l} &= \langle \varphi_{0,0}, \varphi_{0,l} \rangle = \langle \varphi(x), \varphi(x-l) \rangle \\
&= \langle \sum_{k \in \mathbb{Z}} h_k \varphi(2x-k), \sum_{k' \in \mathbb{Z}} h_{k'} \varphi(2(x-l)-k') \rangle \\
&= \sum_{k \in \mathbb{Z}} \sum_{k' \in \mathbb{Z}} h_k h_{k'} \langle \varphi(2x-k), \varphi(2(x-l)-k') \rangle \\
&= \sum_{k \in \mathbb{Z}} \sum_{k' \in \mathbb{Z}} h_k h_{k'-2l} \langle \varphi(2x-k), \varphi(2x-k') \rangle \\
&= \sum_{k \in \mathbb{Z}} \sum_{k' \in \mathbb{Z}} h_k h_{k'-2l} \delta_{k,k'} = \sum_{k \in \mathbb{Z}} h_k h_{k+2l}.
\end{aligned}$$

Construction wavelets

In this subsection the wavelet will be introduced in an orthogonal setting. Consider the *detail space* W_j (in literature often referred to as innovation space), which is the orthogonal complement of V_j in V_{j+1}, i.e. $V_{j+1} = V_j \oplus W_j$. From the MRA definition (3.3) it can be concluded that the union of the detail spaces is dense, i.e.,

$$\cup_{i=-\infty}^{+\infty} W_i = L^2(\mathbb{R}).$$

Next, a mother wavelet function $\psi \in W_0 \subset V_1$ is defined as

$$\psi(x) = \sum_{m \in \mathbb{Z}} g_m \varphi(2x-m), \tag{3.6}$$

with $g_k = (-1)^k h_{1-k}$ and $G = (g_m)_{m \in \mathbb{Z}}$ the *wavelet filter*. Similar to the scaling functions the family $\psi(\cdot - m)$ forms a Riesz basis for W_0. Moreover, the basis functions for W_j are

$$\psi_{j,m}(x) = 2^{j/2} \psi(2^j x - m) \qquad j, m \in \mathbb{Z}.$$

Furthermore, due to the orthogonality the relation $\langle \psi_{j,m}, \psi_{j,n} \rangle = \delta_{m,n}$ holds and leads to the condition $\sum_{m \in \mathbb{Z}} g_m g_{m-2n} = \delta_{0,n}$. Because W_j is the orthogonal complement of V_j, it holds that $\langle \varphi_{j,k}, \psi_{j,m} \rangle = 0$, which leads to $\sum_k h_k g_{k-2m} = 0$:

$$\begin{aligned}
0 &= \langle \varphi_{0,0}, \psi_{0,m} \rangle = \langle \varphi(x), \psi(x-m) \rangle \\
&= \langle \sum_{k \in \mathbb{Z}} h_k \varphi(2x-k), \sum_{k' \in \mathbb{Z}} g_{k'} \varphi(2(x-m)-k') \rangle \\
&= \sum_{k \in \mathbb{Z}} \sum_{k' \in \mathbb{Z}} h_k g_{k'} \langle \varphi(2x-k), \varphi(2(x-m)-k') \rangle \\
&= \sum_{k \in \mathbb{Z}} \sum_{k' \in \mathbb{Z}} h_k g_{k'-2m} \langle \varphi(2x-k), \varphi(2x-k') \rangle \\
&= \sum_{k \in \mathbb{Z}} \sum_{k' \in \mathbb{Z}} h_k g_{k'-2m} \delta_{k,k'} = \sum_{k \in \mathbb{Z}} h_k g_{k-2m}.
\end{aligned}$$

Since $\psi_{j,m}$ form a basis for W_j, $f \in W_j$ can also be written as:

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

Furthermore, the projection $P_{W_j}$of any $f \in L^2$ is defined as

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

In order to present any function f as a sum of wavelet functions, the so called *detail coefficients*, $d_{j,m}$, need to be determined. These coefficients will be determined using the fast transform, which will be introduced in a later stage.

3.2.2 Biorthogonal wavelets

Next the MRA setting will be extended for the biorthogonal wavelets. Biorthogonal wavelets have been developed, due to the difficulty of finding appropriate orthogonal wavelets. Naturally, it is preferred for implementation purposes that H and G have a finite support, however if a support is too small there can be performance issues. As a result of the strong orthogonality restrictions the filters H and G typically have an extremely small amount of nonzero values. Which results in non smooth scaling and wavelet functions. Therefore, a weaker criterion, biorthogonallity, will be introduced.

In the biorthogonal setting next to the primal MRA M there is also a dual MRA $\tilde{M}$. The dual MRA is defined in a similar manner as the primal MRA. The subsets introduced will be denoted as

$$\{0\} \subset \ldots \subset \tilde{V}_{j-1} \subset \tilde{V}_j \subset \tilde{V}_{j+1} \subset \ldots \subset L^2(\mathbb{R}) \qquad j \in \mathbb{Z},$$

with basis functions $\tilde{\varphi}_{j,k}$ for $k \in \mathbb{Z}$. The imposed biorthogonallity condition is

$$\langle \varphi_{j,k}, \tilde{\varphi}_{j,k'}\rangle = \delta_{k,k'} \text{ for } k, k' \in \mathbb{Z}.$$

A dual mother scaling function is defined with the dual filter $\tilde{H}$:

$$\tilde{\varphi} = \sum_k \tilde{h}_k \tilde{\varphi}(2x-k) \qquad (\tilde{h}_k) \in l^2(\mathbb{Z}). \tag{3.7}$$

The extra scaling functions can be used to redefine the projections P_{V_j} of any $f \in L^2$ as

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

Note, the smoothing coefficients $s_{j,k}$ now depend on the dual scaling functions instead of the primal scaling functions.

Construction wavelets

Next, the wavelet will be introduced in the biorthogonal setting. Consider two detail spaces W_j and $\tilde{W}_j$. The detail spaces are not the orthogonal complements of respectively V_j and $\tilde{V}_j$, but they are the complements with cross orthogonality conditions:

$$\begin{aligned} &V_j \oplus W_j = V_{j+1} \qquad && \tilde{V}_j \oplus \tilde{W}_j = \tilde{V}_{j+1}, \\ &\tilde{V}_j \perp W_j && V_j \perp \tilde{W}_j \end{aligned}$$

The mother wavelets can be defined as follows:

$$\begin{aligned} \psi(x) &= \sum_m g_m \varphi(2x-m) \qquad & g_k &= (-1)^k \tilde{h}_{1-k}, \\ \tilde{\psi}(x) &= \sum_m \tilde{g}_m \tilde{\varphi}(2x-m) & \tilde{g}_k &= (-1)^k h_{1-k}. \end{aligned}$$

Thus there are two wavelet filters $G = (g_m)_{m\in\mathbb{Z}}$ and $\tilde{G} = (\tilde{g}_m)_{m\in\mathbb{Z}}$. In addition, the basis wavelets for W_j and $\tilde{W}_j$ are

$$\psi_{j,m}(x) = 2^{j/2}\psi(2^j x - m) \qquad \tilde{\psi}_{j,m}(x) = 2^{j/2}\tilde{\psi}(2^j x - m) \qquad \text{for } m \in \mathbb{Z}.$$

The duality and cross orthogonality lead to the following conditions on inproducts and implications for the filters:

$$\begin{aligned} \langle \varphi_{j,k}, \tilde{\varphi}_{j,k'} \rangle = \delta_{k,k'} &\implies \sum_k h_k \tilde{h}_{k-2l} = \delta_{0,l}, \\ \langle \psi_{j,m}, \tilde{\psi}_{j,m'} \rangle = \delta_{m,m'} &\implies \sum_m g_m \tilde{g}_{m-2n} = \delta_{0,n}, \\ \langle \varphi_{j,k}, \tilde{\psi}_{j,m} \rangle = 0 &\implies \sum_k h_k \tilde{g}_{k-2m} = 0, \\ \langle \psi_{j,m}, \tilde{\varphi}_{j,k} \rangle = 0 &\implies \sum_m g_m \tilde{h}_{m-2k} = 0, \\ &\text{for } k, k', l, m, m', n \in \mathbb{Z}. \end{aligned}$$

When a set of biorthogonal scaling functions and wavelets have been implemented, the above conditions on the filters can be used to check the mathematical correctness of the implementation.

Only the first derivation will be showed, the others can be done in a similar manner:

$$\begin{aligned}
\delta_{0,l} &= \langle \varphi_{0,0}, \tilde{\varphi}_{0,l} \rangle = \langle \varphi(x), \tilde{\varphi}(x-l) \rangle, \\
&= \langle \sum_k h_k \varphi(2x-k), \sum_{k'} \tilde{h}_{k'} \tilde{\varphi}(2(x-l)-k') \rangle \qquad \text{from (3.4) and (3.7)}, \\
&= \sum_k \sum_{k'} h_k \tilde{h}_{k'} \langle \varphi(2x-k), \tilde{\varphi}(2x-2l-k') \rangle, \\
&= \sum_k \sum_{k'} h_k \tilde{h}_{k'-2l} \langle \varphi(2x-k), \tilde{\varphi}(2x-k') \rangle, \\
&= \sum_k \sum_{k'} h_k \tilde{h}_{k'-2l} \langle \varphi_{0,k}, \tilde{\varphi}_{0,k'} \rangle, \\
&= \sum_k \sum_{k'} h_k \tilde{h}_{k'-2l} \delta_{k,k'} = \sum_k h_k \tilde{h}_{k-2l}.
\end{aligned}$$

The projection P_{W_j} of any $f \in L^2$ is defined using the dual wavelet for the detail coefficient:

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

3.2.3 Finite precision

The sums described 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 $J2$ and $J1$. Where $J2$ and $J1$ are integers and $J2 > J1$. $J2$ is considered to be a fine approximation space, it is assumed no finer space exists. Before a projection P_{V_j} for the orthogonal scaling function was defined as

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

Furthermore, the projection P_{W_j} for the orthogonal wavelet was defined as

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

Due to the definition of the detail space W_j, V_{J2} can be rewritten as

$$V_{J2} = V_{J1} \oplus \cup_{j=J1}^{J2-1} W_j.$$

Combining these definitions will lead to an approximation of $f \in L^2$ on the finest space V_{J2}:

$$\begin{aligned}
f(x) \approx P_{V_{J2}} f(x) &= P_{V_{J1}} f(x) + \sum_{j=J1}^{J2-1} P_{W_j} f(x) \\
&= \sum_{k \in \mathbb{Z}} s_{J1,k} \varphi_{J1,k}(x) + \sum_{j=J1}^{J2-1} \sum_{m \in \mathbb{Z}} d_{j,m} \psi_{j,m}(x).
\end{aligned}$$

3.2.4 Construction of the basis functions

If an appropriate wavelet has been determined the scaling functions and the wavelet functions should be generated. The manner in which wavelets are generated depends on the wavelet system used. In the case of Donoho's interpolating wavelet an interpolating scheme based on the Deslauriers-Dubuc function is used. When describing the wavelets, the specific construction algorithm needed will be mentioned.

3.2.5 Examples

In this subsection 4 examples of orthogonal and biorthogonal wavelets will be given.

Daubechies wavelet

The Daubechies wavelet is an example of an orthogonal wavelet. The wavelet is generated by applying the Fourier transform as described in [5]. One starts off with an interval function which is equal to 1 in the interval $[-\frac{1}{2}, \frac{1}{2}[$. Thereafter, this function is dilated and scaled multiple times. In Algorithm 1 the construction of the scaling function and wavelet function is given. Note that the formula for the scaling and wavelet functions differ from (3.4) and (3.6) in the sense that they are scaled with $\sqrt{2}$. The G filter is constructed by the formula $g_k = (-1)^k h_{1-k}$. Different Daubechies wavelets can be constructed by applying different H filters.

Algorithm 1 Construction of Daubechies

```
$\eta_0(x) = \chi_{[-\frac{1}{2},\frac{1}{2}[}(x)$
**for** $i = 1$ upto $\infty$ **do**
$\eta_i(x) = \sqrt{2}\sum_k h_k \eta_{i-1}(2x-k)$
**end for**
$\varphi(x) = \eta_\infty(x)$
$\psi(x) = \sqrt{2}\sum_k g_k \varphi(2x-k)$
```

The *Daubechies 2* wavelet, of order $N = 2$, is constructed with the filter $H^2 = (h^2_k)_{k\in\mathbb{Z}}$ in the Table 3.1 extracted from [5].

Table 3.1: The H^N filter coefficients of the different Daubechies wavelets.

k	h^1_k	h^2_k
0	1	$\frac{1+\sqrt{3}}{4\sqrt{2}}$
1	1	$\frac{3+\sqrt{3}}{4\sqrt{2}}$
2		$\frac{3-\sqrt{3}}{4\sqrt{2}}$
3		$\frac{1-\sqrt{3}}{4\sqrt{2}}$

This particular wavelet has been generated using Matlab, the code can be found in the appendix, in Figure 3.1 both the mother scaling function φ and the mother wavelet function ψ are shown.

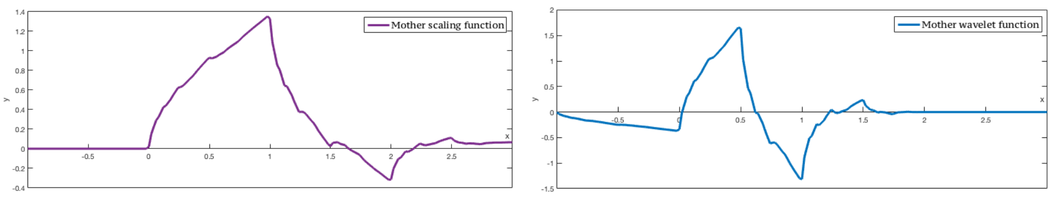


Figure 3.1: The Daubechies wavelet with $N = 2$.

Another famous wavelet which can be constructed as a Daubechies wavelet is the *orthogonal Haar wavelet*. By applying the same construction as Daubechies on the filter $H^1 = (h^1_k)_{k\in\mathbb{Z}}$ in the Table 3.1 the orthogonal Haar wavelet is generated. In Figure 3.2 both the mother scaling function φ and the mother wavelet function ψ are shown.

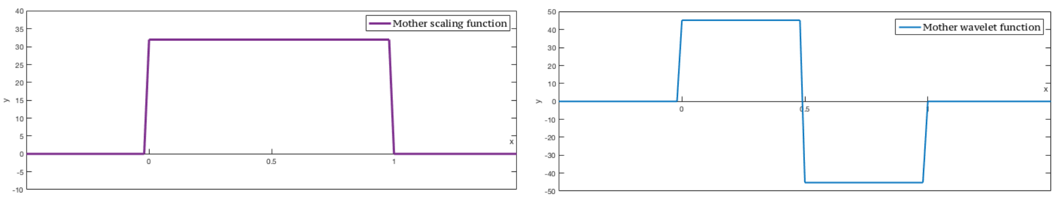


Figure 3.2: The orthogonal Haar wavelet or Daubechies wavelet with $N = 1$.

Cohen-Daubechies-Feauveau wavelet

Cohen, Daubechies and Feauveau described multiple biorthogonal wavelets in [4]. The wavelets are generated by applying the Fourier transform in a similar way as in [5]. One of the example wavelets given by Cohen, Daubechies and Feauveau is a wavelet based on splines. These wavelets will be denoted as *CDF($N, \tilde{N}$)*, for ease of notation, where N denotes the order and $\tilde{N}$ the dual order. In this case the primal mother scaling function is equal to a spline of a certain order, which influences the order N. The dual mother scaling function is a recursive application of dilations and scalings of the same spline, the filter used for this determines the dual order $\tilde{N}$. In Algorithm 2 the construction of the CDF$(N, \tilde{N})$ wavelet is given.

If the CDF mother wavelet is based on a linear spline, the *CDF(2,2)* wavelet is generated. This wavelet has order $N = 2$ and dual order $\tilde{N} = 2$ and can be constructed using the filters in table 6.1 in [4]. In Figure 3.3 the primal mother scaling function φ, the primal mother wavelet function ψ, the dual mother scaling function $\tilde{\varphi}$ and the dual mother wavelet function $\tilde{\psi}$ are shown.

Basing the CDF mother wavelet on a constant spline will result in the *biorthogonal Haar wavelet*. In Figure 3.4 the Haar wavelet with order $N = 1$ and dual order $\tilde{N} = 3$, using the filters in table 6.1 in [4], is shown.

Algorithm 2 Construction of CDF

```
$\varphi(x) = \text{spline}(x)$
$\eta_0(x) = \text{spline}(x)$
**for** $i = 1$ upto $\infty$ **do**
$\eta_i(x) = \sqrt{2}\sum_k \tilde{h}_k \eta_{i-1}(2x-k)$
**end for**
$\tilde{\varphi}(x) = \eta_\infty(x)$
$\psi(x) = \sqrt{2}\sum_k g_k \varphi(2x-k)$
$\tilde{\psi}(x) = \sqrt{2}\sum_k \tilde{g}_k \tilde{\varphi}(2x-k)$
```

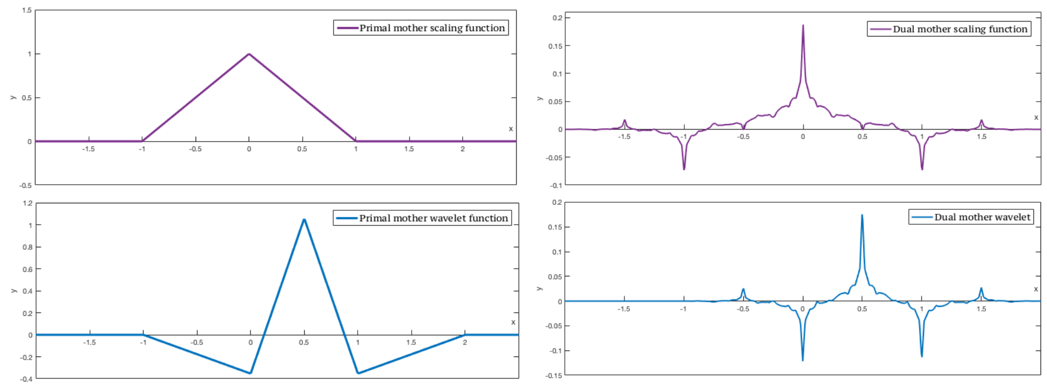


Figure 3.3: The Cohen-Daubechies-Feauveau wavelets with $N = 2$ and $\tilde{N} = 2$.

3.3 Fast transform

When one wants to determine the wavelet transform of a function the following definition can be used:

$$f(x) \approx P_{V_{J2}} f(x) = \sum_{k\in\mathbb{Z}} s_{J1,k}\varphi_{J1,k}(x) + \sum_{j=J1}^{J2-1}\sum_{m\in\mathbb{Z}} d_{j,m}\psi_{j,m}(x).$$

Consider the scaling functions and wavelet functions to be known. Then only the coefficients $s_{J1,k}$ and $d_{j,m}$, for $k, m \in \mathbb{Z}$ and $J1 \leq j \leq J2-1$, need to be determined to reconstruct f. The coefficients can be determined by rewriting the projection definitions. Calculating these coefficients will be called the *forward transform*. If $s_{j,k}$ are needed at any other level than $J1$ the projections can be rewritten in another matter to determine the values. This is called the *inverse transform*. The forward and inverse transform combined is called the *fast transform*. Typically when new wavelets are designed one tries to find a fast transform which can be solved in linear time ($\mathcal{O}(N)$). In this section the forward and inverse transform will be determined both for the orthogonal and biorthogonal wavelets. When a new wavelet is introduced the appropriate fast transform will be explained in that section.
There are multiple scaled versions of the mother scaling and wavelet functions. The ones

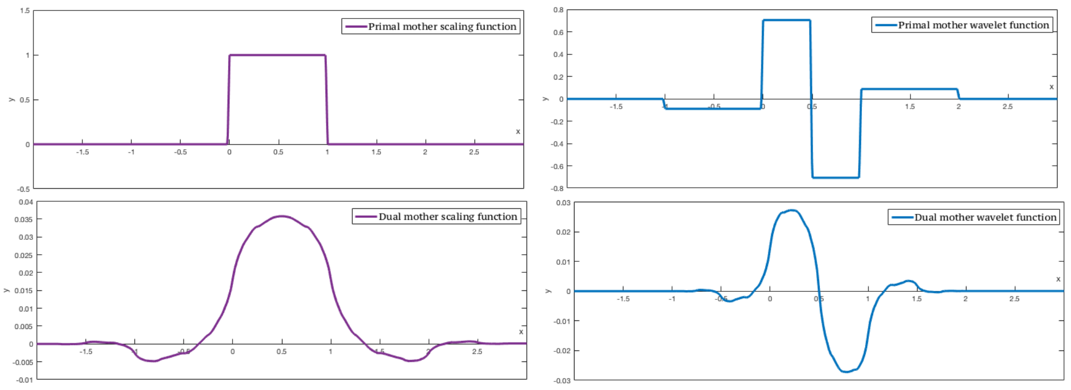


Figure 3.4: The biorthogonal Haar wavelet or Cohen-Daubechies-Feauveau wavelet with $N = 1$ and $\tilde{N} = 3$.

for which the fast transform will be calculated are:

$$
\begin{aligned}
\varphi(x) &= 2\sum_k h_k \varphi(2x - k) \\
\psi(x) &= 2\sum_m g_m \varphi(2x - m) \\
\varphi_{j,k}(x) &= 2^{j/2}\varphi(2^j x - k) \\
\psi_{j,m}(x) &= 2^{j/2}\psi(2^j x - m)
\end{aligned}
$$

The mother functions have a different scaling then defined in section 3.2, this because these are the equations which will be used later in section 3.6. The new scaling results in a new condition on the H filter, $\sum_k h_k = 1$.

3.3.1 Forward transform

In this transform one wants to determine the coefficients $s_{J1,k}$ and $d_{j,m}$, for $k, m \in \mathbb{Z}$ and $J1 \leq j \leq J2 - 1$. Firstly, assume that the coefficients $s_{J2,k}$ are known or have an accurate approximation, i.e., the smoothing values at the finest level of approximation are given.
Starting with the orthogonal wavelets a new expression for the detail coefficients can be found by simply rewriting the inner product $d_{j-1,k} = \langle f, \psi_{j-1,k}\rangle$.

$$
\begin{aligned}
d_{j-1,m} &= \langle f, \psi_{j-1,m}\rangle = \langle f, 2^{(j-1)/2}\psi(2^{j-1}x - m)\rangle, \\
&= \langle f, 2^{(j-1)/2} 2\sum_n g_n \varphi(2(2^{j-1}x - m) - n)\rangle = 2^{-1/2} 2\sum_n g_n \langle f, 2^{j/2}\varphi(2(2^{j-1}x - m) - n)\rangle, \\
&= 2^{1/2}\sum_n g_n \langle f, 2^{j/2}\varphi(2^j x - (2m + n))\rangle = 2^{1/2}\sum_n g_n \langle f, \varphi_{j,2m+n}\rangle, \\
&= 2^{1/2}\sum_n g_n s_{j,2m+n}.
\end{aligned}
$$

Hence, the detail coefficients at a coarser level can be determined completely by the smoothing coefficients of a finer level. Typically, the filter G is finite, hence the sum needed to determine $d_{j-1,k}$ is finite. Similarly for the smoothing coefficient at a coarser level it holds that:

$$\begin{aligned} s_{j-1,k} &= \langle f, \varphi_{j-1,k} \rangle = \langle f, 2^{(j-1)/2}\varphi(2^{j-1}x-k)\rangle, \\ &= \langle f, 2^{(j-1)/2} 2 \sum_l h_h \varphi(2(2^{j-1}x-k)-l)\rangle = 2^{-1/2} 2 \sum_l h_l \langle f, 2^{j/2}\varphi(2(2^{j-1}x-k)-l)\rangle, \\ &= 2^{1/2} \sum_l h_l \langle f, \varphi(2^j x - (2k+l))\rangle = 2^{1/2} \sum_l h_l \langle f, \varphi_{j,2k+l}\rangle, \\ &= 2^{1/2} \sum_l h_l s_{j,2k+l}. \end{aligned}$$

Thus the forward transform for the orthogonal wavelets is:

$$\begin{cases} d_{j-1,m} = \sqrt{2} \sum_n g_n s_{j,2m+n}, \\ s_{j-1,k} = \sqrt{2} \sum_l h_l s_{j,2k+l}. \end{cases} \tag{3.8}$$

The forward transform for the biorthogonal wavelets is completely the same, it only differs in the fact that the coefficients depend on the dual filters. Hence, for the biorthogonal wavelets the forward transform is:

$$\begin{cases} d_{j-1,m} = \sqrt{2} \sum_n \tilde{g}_n s_{j,2m+n}, \\ s_{j-1,k} = \sqrt{2} \sum_l \tilde{h}_l s_{j,2k+l}. \end{cases} \tag{3.9}$$

The equations of (3.8) and (3.9) can be applied iteratively to determine $s_{J1,k}$ and $d_{j,m}$, for $k, m \in \mathbb{Z}$ and $J1 \leq j \leq J2-1$. Typically, the s values will be overwritten every iteration in order to minimise the amount of data which needs to be stored. The algorithm is written down in Algorithm 3.

Algorithm 3 Forward transform

```
**for** j=J2 downto J1 **do**
$\forall m$    $d_{j-1,m} = \sqrt{2} \sum_n g_n s_{j,2m+n}$
$\forall k$    $s_{j-1,k} = \sqrt{2} \sum_l h_l s_{j,2k+l}$
**end for**
```

3.3.2 Inverse transform

In order to find the inverse transform, one can rewrite the projection

$$P_{V_j} f(x) = P_{V_{j-1}} f(x) \oplus P_{W_{j-1}} f(x). \tag{3.10}$$

Firstly, note:

$$\begin{aligned}
\varphi_{j-1,l}(x) &= 2^{(j-1)/2}\varphi(2^{j-1}x-l) = 2^{(j-1)/2}2\sum_m h_m\varphi(2(2^{j-1}x-l)-m)\\
&= 2^{1/2}\sum_m h_m 2^j\varphi(2^jx-(2l+m)) = 2^{1/2}\sum_m h_m\varphi_{j,2l+m}(x) = \sqrt{2}\sum_k h_{k-2l}\varphi_{j,k}(x),\\
\psi_{j-1,l}(x) &= 2^{(j-1)/2}\psi(2^{j-1}x-l) = 2^{(j-1)/2}2\sum_m g_m\varphi(2(2^{j-1}x-l)-m)\\
&= 2^{1/2}\sum_m g_m 2^j\varphi(2^jx-(2l+m)) = 2^{1/2}\sum_m g_m\varphi_{j,2l+m}(x) = \sqrt{2}\sum_k g_{k-2l}\varphi_{j,k}(x).
\end{aligned}$$

These expressions will be used to rewrite (3.10).

$$\begin{aligned}
P_{V_j}f(x) &= P_{V_{j-1}}f(x) \oplus P_{W_{j-1}}f(x)\\
\sum_k s_{j,k}\varphi_{j,k}(x) &= \sum_l s_{j-1,l}\varphi_{j-1,l}(x) + \sum_l d_{j-1,l}\psi_{j-1,l}(x)\\
\sum_k s_{j,k}\varphi_{j,k}(x) &= \sqrt{2}\sum_l s_{j-1,l}\sum_k h_{k-2l}\varphi_{j,k}(x) + \sqrt{2}\sum_l d_{j-1,l}\sum_k g_{k-2l}\varphi_{j,k}(x)\\
\sum_k s_{j,k}\varphi_{j,k}(x) &= \sum_k(\sqrt{2}\sum_l s_{j-1,l}h_{k-2l} + \sqrt{2}\sum_l d_{j-1,l}g_{k-2l})\varphi_{j,k}(x)\\
\implies s_{j,k} &= \sqrt{2}\sum_l s_{j-1,l}h_{k-2l} + \sqrt{2}\sum_l d_{j-1,l}g_{k-2l}
\end{aligned}$$

This equation is the same for the biorthogonal wavelet, as φ is rewritten and not the coefficients s and d.
Hence, for the orthogonal and biorthogonal wavelets the inverse transform is:

$$s_{j,k} = \sqrt{2}\sum_l s_{j-1,l}h_{k-2l} + \sqrt{2}\sum_l d_{j-1,l}g_{k-2l}. \tag{3.11}$$

The algorithm is written down in Algorithm 4.

Algorithm 4 Inverse transform

```
for j=J1  downto  J2 do
$\forall k$      $s_{j,k} = \sqrt{2}\sum_l s_{j-1,l}h_{k-2l} + \sqrt{2}\sum_l d_{j-1,l}g_{k-2l}$
end for
```

3.4 Donoho's interpolating wavelet

To find the wavelet transform of a function the fast transform has to be computed. The wavelet introduced by Donoho, [9], was designed to increase the speed of the fast transform. The three main properties Donoho tried to optimise are mentioned below:

1. Each smoothing coefficient $s_{j,k}$ can be calculated independent of all other coefficients. This in order to decrease computational costs of the forward transform.
2. The smoothing and detail coefficients have decay properties which are comparable to the decay properties of orthogonal wavelets.
3. A simple thresholding rule should hold in order to minimise the coefficients needed to reconstruct a function. This in order to decrease memory usage.

3.4.1 Construction basis functions

The wavelet designed by Donoho is based on the family of Deslauriers-Dubuc fundamental functions. These functions are formed by interpolating the Kronecker delta at the integers. The interpolation is done by fitting polynomials of order $N-1$. Repeating this interpolation step will lead to the primal mother scaling function φ of order N. In Algorithm 5 the construction of generating the Donoho mother scaling function is given. Note that the H filter is not needed for this construction.

Algorithm 5 Construction of Donoho's wavelet

```
For $k \in \mathcal{K}_0 \subset \mathbb{Z}$, $\varphi(k) = \begin{cases} 1 & \text{if } k == 0 \\ 0 & \text{else} \end{cases}$
for $i = 1$ upto $\infty$ do
    for $k \in \mathcal{K}_{i-1}$ do
        Add $\frac{k}{2}$ to $\mathcal{K}_i$.
    end for
    for $\frac{k}{2} \in \mathcal{K}_i$ do
        Fit $N-1$ order polynomial $p$ to the $N$ points $k \in \mathcal{K}_{i-1}$ surrounding $\frac{k}{2} \in \mathcal{K}_i$.
        Set $\varphi(\frac{k}{2}) = p(\frac{k}{2})$.
    end for
end for
```

This Donoho wavelet has been generated using Matlab, the code can be found in the appendix. In Figure 3.5 different Donoho wavelets φ^N are depicted.

3.4.2 MRA

The MRA setting will be similar as in section 3.2. If the mother wavelet is constructed from the Deslauriers-Dubuc fundamental functions, the φ can be represented as a linear combination of dilates and translates of itself:

$$\varphi^N(x) = \sum_k \varphi^N(\frac{k}{2})\varphi^N(2x-k).$$

Therefore, the H^N filter coefficients can be determined by checking the values of the half integers. Thus for the case $N=4$ the value at 0 is 1, hence $h_0^4 = 1$. Furthermore,

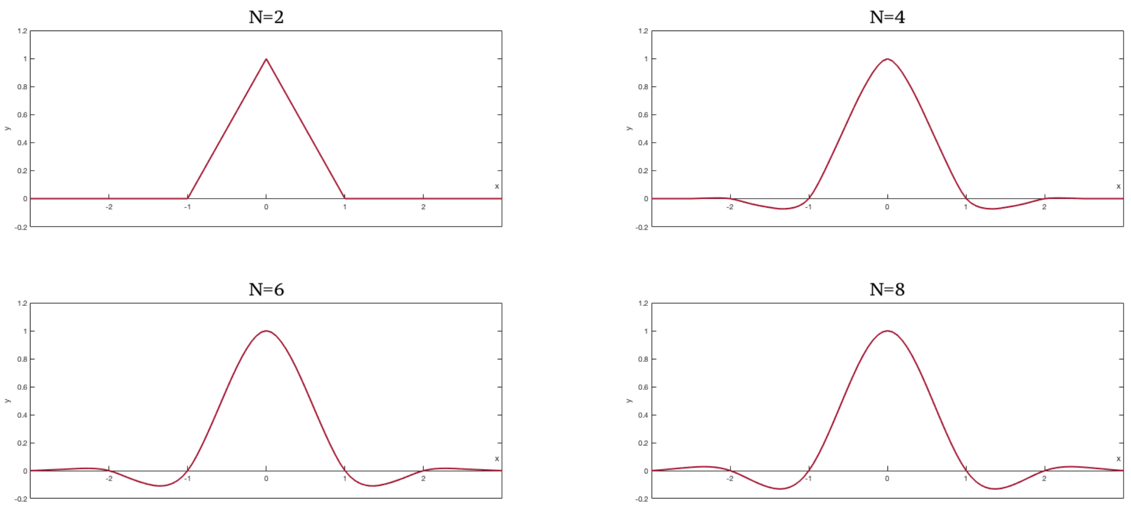


Figure 3.5: The Donoho mother scaling function of order $N = 2, 4, 6, 8$.

$\varphi^4(0.5) = \varphi^4(0.5) = \frac{9}{2^4}$, $\varphi^4(1.5) = \varphi^4(1.5) = \frac{-1}{2^4}$, hence $h^4_{-1} = h^4_1 = \frac{9}{2^4}$ and $h^4_{-3} = h^4_3 = \frac{-1}{2^4}$. At all other half integer locations the scaling function has a value equal to zero. These values and the filters of the other order Donoho functions can be found in the Table 3.6.

The wavelet ψ is constructed by $\psi(x) = \varphi^N(2(x - \frac{1}{2}))$. Furthermore, the bases for the nested families V_j and W_j are:

$$\varphi_{j,k}(x) = 2^{\frac{j}{2}}\varphi^N(2^j x - k)$$
$$\psi_{j,k}(x) = 2^{\frac{j}{2}}\psi(2^j x - k)$$

With these definitions an interpolating wavelet transform can be defined, which can reconstruct any f which is the sum of a polynomial of degree $\leq N-1$. This construction will look like:

$$f = \sum_k s_{J_1,k}\varphi_{J_1,k} + \sum_{j\geq J_1}\sum_k d_{j,k}\psi_{j,k}.$$

In [9] the proof can be found, which states that indeed this transform can reconstruct any function which has a polynomial degree $\leq N-1$. Note that the s coefficients relate to the β coefficients and d to α in [9].

The smoothing coefficient has the special property that $s_{j,k} = 2^{-\frac{j}{2}} f(2^{-j}k)$. The projection of a continuous function f on V_j is defined as $(P_{V_j} f)(x) = 2^{-\frac{j}{2}} \sum_k f(2^{-j}k)\varphi_{j,k}(x)$. From Lemma 2.4 in [9] it follows that for all continuous functions vanishing at ∞, the projection $P_{V_j} f$ converges to f as j increases.

Table 3.2: The H^N filter coefficients of the different Donoho wavelets.

k	h_k^2	h_k^4	h_k^6	h_k^8
-7				$-\frac{5}{2^{11}}$
-6				0
-5			$\frac{3}{2^8}$	$\frac{49}{2^{11}}$
-4			0	0
-3		$-\frac{1}{2^4}$	$-\frac{25}{2^8}$	$-\frac{245}{2^{11}}$
-2		0	0	0
-1	$\frac{1}{2}$	$\frac{9}{2^4}$	$\frac{155}{2^8}$	$\frac{1225}{2^{11}}$
0	1	1	1	1
1	$\frac{1}{2}$	$\frac{9}{2^4}$	$\frac{155}{2^8}$	$\frac{1225}{2^{11}}$
2		0	0	0
3		$-\frac{3}{2^4}$	$-\frac{25}{2^8}$	$-\frac{245}{2^{11}}$
4			0	0
5			$\frac{3}{2^8}$	$\frac{49}{2^{11}}$
6				0
7				$-\frac{5}{2^{11}}$

3.4.3 Fast transform

Forward transform

In order to obtain the forward transform two useful properties will be derived:

$$\begin{aligned}
\psi_{j,k}(x) &= 2^{\frac{j}{2}}\psi(2^j x - k) = 2^{\frac{j}{2}}\varphi(2(2^j x - k - \frac{1}{2})) \\
&= \frac{1}{\sqrt{2}}2^{\frac{j+1}{2}}\varphi(2^{j+1}x - 2k - 1) = \frac{1}{\sqrt{2}}\varphi_{j+1,2k+1}(x) \\
s_{j+1,2k} &= 2^{-\frac{j+1}{2}}f(2^{-j-1}2k) = \frac{1}{\sqrt{2}}2^{-\frac{j}{2}}f(2^{-j}2^{-1}2k) \\
&= \frac{1}{\sqrt{2}}s_{j,k}
\end{aligned}$$

This leads to the first part of the fast forward transform, namely $s_{j,k} = \sqrt{2}s_{j+1,2k}$. In order to determine the second half of the fast transform, determining the detail coefficients at coarser levels, a property leading to a new formulation of the detail coefficient will be derived.

$$\begin{aligned}
\varphi_{j,l}(x) &= 2^{\frac{j}{2}}\varphi(2^j x - l) \\
&= 2^{\frac{j}{2}}\sum_m \varphi(\frac{m}{2})\varphi(2(2^j x - l) - m) \\
&= \sum_m \varphi(\frac{m}{2})2^{-\frac{1}{2}}2^{\frac{j+1}{2}}\varphi(2^{j+1}x - 2l - m) \\
&= \sum_m \varphi(\frac{m}{2})2^{-\frac{1}{2}}\varphi_{j+1,2l+m}(x) \\
&= \frac{1}{\sqrt{2}}\sum_k \varphi(\frac{k-2l}{2})\varphi_{j+1,k}(x) \qquad (3.12)
\end{aligned}$$

$$\begin{aligned}
\sum_l d_{j,l}\psi_{j,l} &= \frac{1}{\sqrt{2}}\sum_l d_{j,l}\varphi_{j+1,2l+1} \\
&= \frac{1}{\sqrt{2}}\sum_k d_{j,\frac{k-1}{2}}\varphi_{j+1,k} \qquad (3.13)
\end{aligned}$$

For $f \in V_{j+1}$ we have two representations, namely $f = \sum_k s_{j+1,k}\varphi_{j+1,k}$ and $f = \sum_l s_{j,l}\varphi_{j,l} + \sum_l d_{j,l}\psi_{j,l}$. Setting them equal will give a nice property for the coefficients out of which the forward and inverse transform can be constructed.

$$\begin{aligned}
\sum_k s_{j+1,k}\varphi_{j+1,k} &= \sum_l s_{j,l}\varphi_{j,l} + \sum_l d_{j,l}\psi_{j,l} \\
&= \sum_l s_{j,l}\frac{1}{\sqrt{2}}\sum_k \varphi(\frac{k-2l}{2})\varphi_{j+1,k}(t) + \frac{1}{\sqrt{2}}\sum_k d_{j,\frac{k-1}{2}}\varphi_{j+1,k} \\
&= \frac{1}{\sqrt{2}}\sum_k(\sum_l s_{j,l}\varphi(\frac{k-2l}{2}) + d_{j,\frac{k-1}{2}})\varphi_{j+1,k} \\
\implies \sqrt{2}s_{j+1,k} &= \sum_l s_{j,l}\phi(\frac{k-2l}{2}) + d_{j,\frac{k-1}{2}} \qquad (3.14)
\end{aligned}$$

Rewriting the indices of (3.14) and using $s_{j,k} = \sqrt{2}s_{j+1,2k}$, the forward transform of the detail coefficient can be derived.

$$\begin{aligned}
d_{j,k} &= \sqrt{2}s_{j+1,2k+1} - \sum_l s_{j,l}\varphi(\frac{2k+1-2l}{2}) \\
&= \sqrt{2}s_{j+1,2k+1} - \sum_l \sqrt{2}s_{j+1,2l}\varphi(k+\frac{1}{2}-l) \\
&= \sqrt{2}(s_{j+1,2k+1} - \sum_l s_{j+1,2l}\varphi(k+\frac{1}{2}-l))
\end{aligned}$$

Hence, the simplified forward transform of the Donoho wavelet is:

$$\begin{cases} s_{j,k} = \sqrt{2}s_{j+1,2k} \\ d_{j,k} = \sqrt{2}(s_{j+1,2k+1} - \sum s_{j+1,2l}\varphi(k+\frac{1}{2}-l)) \end{cases}$$

Because φ only has a finite support the sum $\sum_l s_{j+1,2l}\varphi(k+\frac{1}{2}-l)$ is finite. Remember, that a standard forward transform looks as follows:

$$\begin{cases} s_{j-1,k} = \sum_l \tilde{h}_l s_{j,2k+l}, \\ d_{j-1,m} = \sum_n \tilde{g}_n s_{j,2m+n}. \end{cases}$$

Hence the filter $\tilde{H}$ only has a non-zero value at $k=0$, namely $\tilde{h}_0 = 1$. Furthermore, $\tilde{G}$ is a shifted version of the earlier mentioned H filter, which is defined on the half integers of φ. And the minus sign indicates that the filter H is multiplied with a (-1) term on the odd indices. This all corresponds with the definition of a general $\tilde{G}$ filter, as defined in the sections 3.2 and 3.3.

Inverse transform

For the reconstruction of the smoothing coefficients at finer levels a distinct will be made between the even and odd indices. Firstly, note that $s_{j,k} = \sqrt{2}s_{j+1,2k}$, so if k is even then $s_{j+1,k} = \frac{1}{\sqrt{2}}s_{j,\frac{k}{2}}$.
Secondly, rewriting (3.14) and assuming that the indices k are odd, leads to:

$$\begin{aligned} s_{j+1,k} &= \frac{1}{\sqrt{2}}(\sum_l s_{j,l}\varphi(\frac{k-2l}{2}) + d_{j,\frac{k-1}{2}}) \\ &= \frac{1}{\sqrt{2}}(\sum_l s_{j,l}\varphi(\frac{k-1}{2}+\frac{1}{2}-l) + d_{j,\frac{k-1}{2}}) \\ &= \frac{1}{\sqrt{2}}(\sum_l s_{j,l+\frac{k-1}{2}}\varphi(\frac{1}{2}-l) + d_{j,\frac{k-1}{2}}) \end{aligned}$$

Hence, the following inverse transform is constructed:

$$s_{j+1,k} = \begin{cases} \frac{1}{\sqrt{2}}s_{j,\frac{k}{2}} & \text{if } k \text{ is even} \\ \frac{1}{\sqrt{2}}(\sum_{l=-N+1}^{N} s_{j,l+\frac{k-1}{2}}\varphi(\frac{1}{2}-l) + d_{j,\frac{k-1}{2}}) & \text{if } k \text{ is odd} \end{cases}$$

3.4.4 Note on detail coefficient

In order to retrieve an intuitive interpretation of the detail coefficient and to understand the thresholding, (3.14) will be rewritten.
Note that:

$$\varphi(\frac{k-1}{2}+\frac{1}{2}-l) = \varphi(2^j(\frac{k+\frac{1}{2}}{2^j}) - l) = 2^{-\frac{j}{2}}\varphi_{j,l}(\frac{k+\frac{1}{2}}{2^j})$$

Using the fact that $s_{j,k} = 2^{-\frac{j}{2}} f(2^{-j}k)$ and $(P_{V_j}f)(t) = 2^{-\frac{j}{2}} \sum_k f(2^{-j}k)\varphi_{j,k}(t)$, results in:

$$\begin{aligned}
d_{j,k} &= \sqrt{2}s_{j+1,2k+1} - \sum_l s_{j,l}\varphi(\frac{2k+1-2l}{2}) \\
&= \sqrt{2}s_{j+1,2k+1} - 2^{-\frac{j}{2}} \sum_l s_{j,l}\varphi_{j,l}(\frac{k+\frac{1}{2}}{2^j}) \\
&= \sqrt{2}(2^{-\frac{j+1}{2}} f(\frac{2k+1}{2^{j+1}})) - 2^{-\frac{j}{2}} \sum_l (2^{-\frac{j}{2}} f(\frac{l}{2^j}))\varphi_{j,l}(\frac{k+\frac{1}{2}}{2^j}) \\
&= 2^{-\frac{j}{2}} f(\frac{k+\frac{1}{2}}{2^j}) - 2^{-\frac{j}{2}} \sum_l (2^{-\frac{j}{2}} f(\frac{l}{2^j}))\varphi_{j,l}(\frac{k+\frac{1}{2}}{2^j}) \\
&= 2^{-\frac{j}{2}} (f(\frac{k+\frac{1}{2}}{2^j}) - 2^{-\frac{j}{2}} \sum_l f(\frac{l}{2^j})\varphi_{j,l}(\frac{k+\frac{1}{2}}{2^j}) \\
&= 2^{-\frac{j}{2}} (f(\frac{k+\frac{1}{2}}{2^j}) - (P_{V_j}f)(\frac{k+\frac{1}{2}}{2^j}))
\end{aligned}$$

Hence the detail coefficient d measures the lack of approximation of f by $P_{V_j}f$. So in order to compress using this wavelet it makes sense to threshold on the value of the detail coefficients. Each detail coefficient corresponds to a smoothing coefficient on a higher level. So if the detail coefficient is almost equal to zero the corresponding smoothing coefficient can be correctly interpolated, thus the smoothing coefficient does not need to be saved. If the function is a polynomial of order $N-1$ all the detail coefficients will be zero, thus by only using the smoothing coefficient on the coarsest level the whole function can be reconstructed perfectly.
Consider the threshold value ε and denote the function with thresholded detail coefficients as $f^{(\varepsilon)}$. Donoho described the difference between $f^{(\varepsilon)}$ and $P_{V_j}f$ intuitively as: *"The slogan is that $f^{(\varepsilon)}$ contains the terms which are important, while $P_{V_j}f$ contains all terms which might possibly be important."*
In [9] theorem 3.8 gives a bound on the infinity norm of the difference between f and $f^{(\varepsilon)}$ for f sufficiently smooth.

$$||f - f^{(\varepsilon)}||_\infty \leq C_1\varepsilon,$$

for some constant C_1. The theorem also results in a bound for the amount of non zero detail coefficients $\mathcal{N}(\varepsilon)$:

$$\mathcal{N}(\varepsilon) \leq C_2\varepsilon^{-p},$$

for some constant C_2 and p corresponding to the Besov space as defined by Donoho. For more details [9] can be consulted.

3.5 Vanishing moments

Whenever an example of a wavelet system has been discussed, an order N was given. In this section, the order N and dual order $\tilde{N}$ will be introduced and their properties will be analysed. This will give insights on choosing the order and dual order of a wavelet system in such a manner that the approximation error will be minimised. More precisely choosing $N = \tilde{N}$ will result in the best results. Furthermore, this section will give insights on the weakness of the Donoho wavelet. The latter will result in the motivation behind second generation wavelets.

Firstly, the order N and dual order $\tilde{N}$ of a wavelet are defined:

Definition 2 - Vanishing moments

Consider a (bi)-orthogonal wavelet system, with scaling function φ and wavelet ψ. Then the wavelet system has order N if the scaling function has N vanishing moments, i.e.:

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

The wavelet system has dual order $\tilde{N}$ if the wavelet function has $\tilde{N}$ vanishing moments, i.e.:

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

Note that the integral of the wavelet, $\int_{-\infty}^{\infty} \psi(x)dx$, needs to be zero in order to have $\tilde{N} > 0$, whilst $\int_{-\infty}^{\infty} \varphi(x)dx = 1$ in many cases. The order N has a direct relation with the polynomial functions a wavelet system can reproduce. In other words a wavelet system of order N can reproduce any polynomial of order $N - 1$, see [9] and [26]. Furthermore, the order N of the wavelet system directly influences the size of the filter H. Theorem 2.2 in [12], gives a lower bound on the support of any wavelet depending on the order N, the support size directly relates to the filter size.
Moreover, imposing vanishing moments on the scaling function and dual vanishing moments on the wavelet will reduce the approximation error of f with the wavelet system. To illustrate this, theorems from [28] will be introduced. The theorems are used to depict the properties of the vanishing moments, for more details on these theorems and their proofs one can consult [28].

Firstly, denote by $C_0^{N,1}(\mathbb{R})$ the set of compactly supported functions having derivatives of order $\leq N$ and whose N-th derivative is Lipschitz. Furthermore, if an orthonormal wavelet system is used, an orthogonal wavelet system with the property $\int_{-\infty}^{\infty} \varphi(x)dx = 1$ is implied.

Theorem 3.5.1 - Orthogonal projection error

Consider an orthonormal wavelet system with scaling function $\varphi(x)$ and wavelet function $\psi(x)$. For $f(x) \in C_0^{\tilde{N},1}(\mathbb{R})$, define the orthogonal projection

$$P^j(f) := \sum_{k\in\mathbb{Z}} (\int_{-\infty}^{\infty} f(t)\varphi_{j,k}(t)dt)\varphi_{j,k}(x).$$

If $\psi(x)$ has vanishing moments up to degree $\tilde{N}$, then

$$||f(x) - P^j(f)||_{L^2} \leq C_1 2^{-j(\tilde{N}+1)},$$

where C_1 is a constant independent of j and $\tilde{N}$.

Hence, the higher the dual order $\tilde{N}$ the closer the orthogonal projection is to the function f. Next, a similar theorem will be applied to the vanishing moments of φ. The orthogonal projection as defined in theorem 3.5.1 has an integral, namely the inner product between f and the scaling functions. This integral can be approximated using the wavelet sampling approximation:

Definition 3 - Wavelet sampling approximation

The wavelet sampling approximation of an $L^2(\mathbb{R})$ function $f(x)$ at the j-th level is

$$S^j(f) := 2^{\frac{-j}{2}} \sum_{k\in\mathbb{Z}} f(\frac{k}{2^j})\varphi_{j,k}(x).$$

Note, that this manner of representing the inner product is similar to the manner in which Donoho defined the smoothing coefficients. The vanishing moments of φ can be used to express the approximation error of the wavelet sampling approximation.

Theorem 3.5.2 - Wavelet sampling approximation error

Consider an orthonormal wavelet system with scaling function $\varphi(x)$ and wavelet function $\psi(x)$. If $\varphi(x)$ has vanishing moments up to degree N and $f(x) \in C_0^{N,1}(\mathbb{R})$, then

$$||P^j(f) - S^j(f)||_{L^2} \leq C_2 2^{-j(N+1)},$$

where C_2 is a constant independent of j and N.

Using the wavelet sampling approximation method to determine the inner products, as is done by Donoho, leads to the following bound on the approximation error:

$$\begin{aligned} ||f(x) - S^j(f)||_{L^2}^2 &\leq ||f(x) - P^j(f)||_{L^2}^2 + ||P^j(f) - S^j(f)||_{L^2}^2 \\ &\leq C_1 2^{-j(\tilde{N}+1)} + C_2 2^{-j(N+1)}. \end{aligned}$$

Naturally, one tends to minimise this error. The order of the error is determined by the smallest of the terms N and $\tilde{N}$. Hence, an optimal choice would be $N = \tilde{N}$. This because a higher (dual) order leads to more computations and only increasing one of the orders will not have a huge influence on the behaviour of the error of the approximation. The above setting is defined for orthogonal wavelets, however in [28] it is shown that it also holds for the biorthogonal setting.

Concluding, a wavelet system should be designed in such a manner that the order N and dual order $\tilde{N}$ are equal. Furthermore, the higher the orders the smaller the approximation error, however a higher order results in a bigger filter H. Thus there is a trade-off between the order of the wavelet system and the computational costs.

3.5.1 Example

In section 3.2 an example of a biorthogonal wavelet was discussed, the CDF(2,2). This is a biorthogonal wavelet with $N = \tilde{N} = 2$. In this subsection the order and dual order will be derived.
Firstly, the mother scaling function and mother wavelet function are:

$$\varphi(x) = \begin{cases} 1+x & -1 \leq x \leq 0, \\ 1-x & 0 \leq x \leq 1, \\ 0 & \text{else}, \end{cases}$$
$$\psi(x) = \sqrt{2}\sum_k g_k \varphi(2x-k).$$

With $g_{-1} = g_3 = \frac{1}{8}$, $g_0 = g_2 = \frac{1}{4}$ and $g_1 = -\frac{3}{4}$. Secondly, the integrals of the scaling function will be calculated.

$$\begin{aligned}
\int_{-\infty}^{\infty} \varphi(x)dx &= \int_{-1}^{0}(1+x)dx + \int_{0}^{1}(1-x)dx \\
&= [x + \frac{1}{2}x^2]_{-1}^{0} + [x - \frac{1}{2}x^2]_{0}^{1} = 1 \\
\int_{-\infty}^{\infty} x\varphi(x)dx &= \int_{-1}^{0} x(1+x)dx + \int_{0}^{1} x(1-x)dx \\
&= [\frac{1}{2}x^2 + \frac{1}{3}x^3]_{-1}^{0} + [\frac{1}{2}x^2 - \frac{1}{3}x^3]_{0}^{1} = -(\frac{1}{2} - \frac{1}{3}) + (\frac{1}{2} - \frac{1}{3}) = 0 \\
\int_{-\infty}^{\infty} x^2\varphi(x)dx &= \int_{-1}^{0} x^2(1+x)dx + \int_{0}^{1} x^2(1-x)dx \\
&= [\frac{1}{3}x^3 + \frac{1}{4}x^4]_{-1}^{0} + [\frac{1}{3}x^3 - \frac{1}{4}x^4]_{0}^{1} = -(-\frac{1}{3} + \frac{1}{4}) + (\frac{1}{3} - \frac{1}{4}) = \frac{1}{6} \neq 0
\end{aligned}$$

Hence, the order N of CDF(2,2) is indeed 2. Next, the integrals of the wavelet function will be calculated. Note, that the integral of the wavelet should also be zero. First some useful integrals will be calculated.

$$\begin{aligned}
\int_{-\infty}^{\infty} \psi(2x-k)dx &= \frac{1}{2}\int_{-\infty}^{\infty} \psi(u)du = \frac{1}{2} \\
\int_{-\infty}^{\infty} x\psi(2x-k)dx &= \frac{1}{4}\int_{-\infty}^{\infty} (u+k)\psi(u)du \\
&= \frac{1}{4}[\int_{-1}^{0}(u+k)(1+u)du + \int_{0}^{1}(u+k)(1-u)du] = \frac{k}{4} \\
\int_{-\infty}^{\infty} x^2\psi(2x-k)dx &= \frac{1}{8}\int_{-\infty}^{\infty} (u^2+2uk+k^2)\psi(u)du \\
&= \frac{1}{8}[\int_{-1}^{0}(u^2+2uk+k^2)(1+u)du + \int_{0}^{1}(u^2+2uk+k^2)(1-u)du] \\
&= \frac{1}{8}(\frac{1}{6}+k^2)
\end{aligned}$$

$$\begin{aligned}
\int_{-\infty}^{\infty} \psi(x)dx &= \sqrt{2}\sum_k g_k \int_{-\infty}^{\infty} \varphi(2x-k)dx \\
&= \frac{\sqrt{2}}{2}\sum_k g_k = \frac{\sqrt{2}}{2}(\frac{1}{8}+\frac{1}{4}-\frac{3}{4}+\frac{1}{4}+\frac{1}{8}) = 0 \\
\int_{-\infty}^{\infty} x\psi(x)dx &= \sqrt{2}\sum_k g_k \int_{-\infty}^{\infty} x\varphi(2x-k)dx \\
&= \frac{\sqrt{2}}{4}\sum_k g_k k = \frac{\sqrt{2}}{4}(-1*\frac{1}{8}+0*\frac{1}{4}+1*(-\frac{3}{4})+2*\frac{1}{4}+3*\frac{1}{8}) = 0 \\
\int_{-\infty}^{\infty} x^2\psi(x)dx &= \sqrt{2}\sum_k g_k \int_{-\infty}^{\infty} x^2\varphi(2x-k)dx \\
&= \frac{\sqrt{2}}{8}\sum_k g_k(\frac{1}{6}+k^2) = \frac{\sqrt{2}}{8}(\frac{1}{8}(\frac{1}{6}+1)+\frac{1}{4}\frac{1}{6}+-\frac{3}{4}(\frac{1}{6}+1)+\frac{1}{4}(\frac{1}{6}+4) \\
&\quad +\frac{1}{8}(\frac{1}{6}+9)) > 0
\end{aligned}$$

Indeed the dual order $\tilde{N}$ of CDF(2,2) is 2.

3.5.2 Donoho's interpolating wavelet

In section 3.4 Donoho's interpolating wavelet is described. This wavelet can not detect non smooth behaviour in every level due to aliasing. Remember, the forward transform:

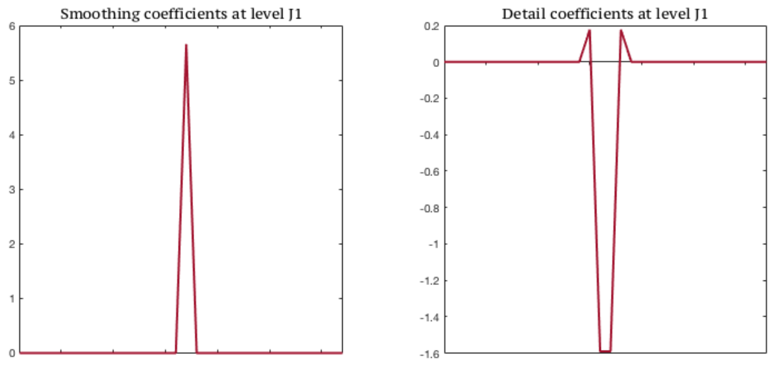
$$\begin{cases} s_{j,k} = \sqrt{2}s_{j+1,2k}, \\ d_{j,k} = \sqrt{2}(s_{j+1,2k+1} - \sum s_{j+1,2l}\varphi(k+\frac{1}{2}-l)). \end{cases}$$

This implies that the filter $\tilde{H}$ only has a non zero value at the position 0, $\tilde{h}_0 = 1$. The filter for the wavelet G directly corresponds to the filter $\tilde{H}$, thus $\tilde{H}$ relates directly to the vanishing moments of ψ. To determine the amount of dual vanishing moments of the Donoho wavelet, consider the integral of the wavelet ψ.

$$\begin{aligned}\int_{-\infty}^{\infty} \psi(x)dx &= \int_{-\infty}^{\infty} \varphi(2(x - \frac{1}{2}))dx \\ &= \frac{1}{2}\int_{-\infty}^{\infty} \varphi(u)du = \frac{1}{2}.\end{aligned}$$

Hence the dual order $\tilde{N} = 0$, this leads to aliasing, which will be shown with the unitary impulse. The unitary impulse is a test function which is everywhere zero expect for 1 location. Depending on the position of this 1 value the Donoho wavelet gives very different results. In Figure 3.6 the forward transform is shown, both the smoothing coefficients and the detail coefficients at the coarsest level are depicted. In this example the unitary impulse is exactly located at a position of the mesh which corresponds to a smoothing value at the coarsest level.

Figure 3.6: Correct forward transform of Donoho's interpolating wavelet.

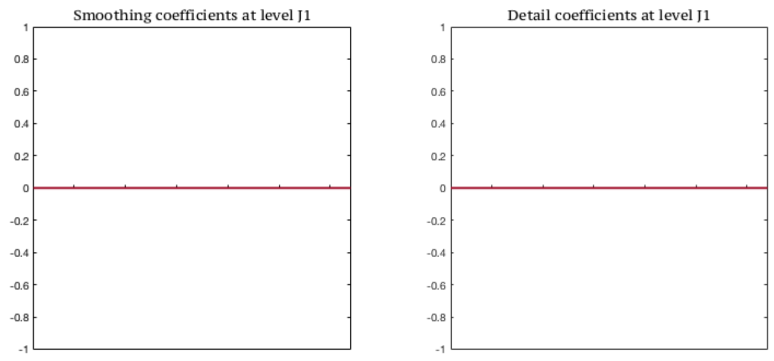


The forward transform was performed on the unitary impulse (even location) with Donoho's interpolating wavelet of order $N = 4$.

However, if the unitary impulse is located at a position which does not correspond with the smoothing locations of the coarsest mesh, the smoothing and detail coefficients at the coarsest level look completely different. This can be seen in Figure 3.7.

This illustrates a drawback of Donoho's interpolating wavelet. However, it is not clear if this will also be a drawback in adaptive mesh refinement. On one of the layers the detail coefficients will contain the peak of the unitary impulse. So the unitary impulse is detected, but the wavelet coefficients no longer represent the information in certain frequency bands, e.g., in the coarsest level the unitary impulse is undetected. If this has effect in the AMR setting will be tested.

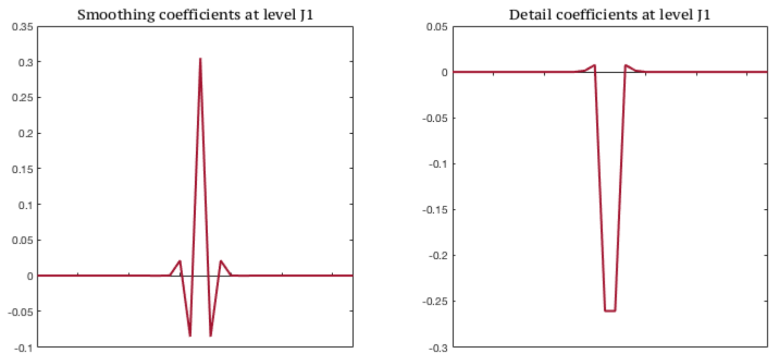
Figure 3.7: Incorrect forward transform of Donoho's interpolating wavelet.



The forward transform was performed on the unitary impulse (odd location) with Donoho's interpolating wavelet of order $N = 4$.

Increasing the dual order will solve this aliasing problem. In the next section the second generation wavelets will be introduced, these wavelets provide a tool with which wavelets can be custom designed. In order to illustrate the benefits of increasing the dual order, the same example as above will be considered for a different wavelet. In Figure 3.8 the forward transform of the unitary impulse is shown using the Sweldens wavelet. This wavelet is the donoho wavelet adjusted such that the dual order $\tilde{N} = 4$. In this example the unitary impulse is exactly located at a position of the mesh which corresponds to a smoothing value at the coarsest level.

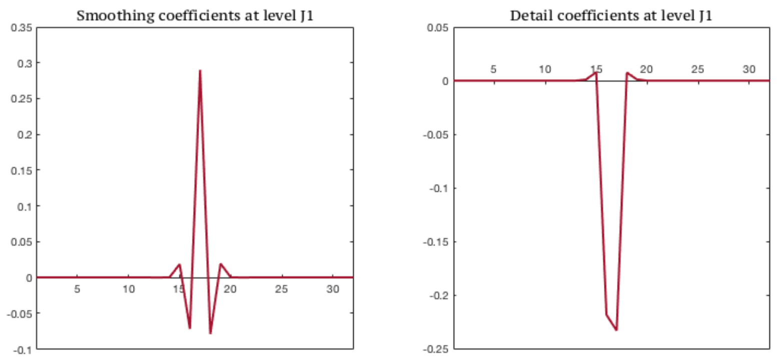
Figure 3.8: Forward transform of the Sweldens wavelet.



The forward transform was performed on the unitary impulse (even location) with Sweldens wavelet of order $N = \tilde{N} = 4$.

The Sweldens wavelet still detects the impulse correctly at the coarsest level, if the unitary impulse is located at a position which does not correspond with the smoothing locations of the coarsest mesh. Note that the the forward transform does not result in a symmetric function due to the location of the impulse. This can be seen in Figure 3.9.

Figure 3.9: Forward transform of the Sweldens wavelet.



The forward transform was performed on the unitary impulse (odd location) with Sweldens wavelet of order $N = \tilde{N} = 4$.

3.6 The lifting scheme and second generation wavelets

Starting in 1994 Sweldens introduced a new type of wavelets,*second generation wavelets*, in the papers [23], [24] , [25] and [27]. The wavelets described until now are always translates and dilates of one function, this kind of wavelet system is called *first generation wavelets*. The construction of the wavelets was always based on the Fourier transform. Sweldens introduced a new way to construct wavelets, *the lifting scheme*. The lifting scheme can be used to construct the traditional first generation wavelets. More importantly, the lifting scheme can be used to construct a new class of wavelets (second generation wavelets).
In this section both the lifting scheme and the second generation wavelets will be discussed. Thereafter, the lifting scheme will be applied to construct a better version of Donoho's interpolating wavelet.

3.6.1 Second generation wavelets

Traditionally, wavelets are dyadic translates and dilates of the mother wavelet. The second generation wavelets however, are not necessarily translates and dilates of each other. The second generation wavelets are constructed using the lifting scheme. Although they differ from the first generation wavelets some important properties are kept:

- ◇ The wavelets form a Riesz basis for $L^2(\mathbb{R})$.
- ◇ The wavelets fit within the multiresolution analysis. The wavelets are orthogonal or biorthogonal, in which case the dual wavelets are known.

◇ The wavelets and their duals are local in space and frequency.

These properties are important as they ensure that the essential information of a function is captured by a small fraction of detail coefficients. Sweldens' motivation for the second generation wavelets are the following three properties which are missing in the first generation setting.

◇ Wavelet bases that are defined on arbitrary, possibly non-smooth, domains of $\mathbb{R}^n$ are needed.

◇ Because of diagonalisation of differential forms, a basis adapted to weighted measures is needed.

◇ First generation wavelets need regular sampled data, while sometimes data is sampled on irregular locations.

To ensure the last three properties, the translation and dilations property is lost.
The second generation wavelets are constructed using the lifting scheme. One starts off with a fairly simple wavelet, in many cases the Lazy wavelet, after which prediction and update steps are performed to lift the wavelet to a multiresolution analysis with particular properties. Using the lifting scheme one can custom design the filters needed.

3.6.2 Lifting scheme

The lifting scheme can be used to construct wavelets, and to gain a fast implementation of the fast transform. The advantages of using the lifting scheme to construct the first generation wavelets are summed up below:

1. The lifting scheme is a faster version of the standard fast transform, because the lifting scheme makes optimal use of the similarities between the filters to speed up the calculation. Smaller filters can be used, so less points need to be called from memory.

2. The lifting scheme only uses in-place calculations and thus no extra memory is needed, i.e. the smoothing coefficients can be overwritten. This will be further explained at the end of this section in 3.6.7.

3. The inverse transform becomes trivial.

Moreover, because the lifting scheme does not depend on the Fourier transform, it can be used to construct second generation wavelets.
The forward transform of the lifting scheme consists out of three steps, the split, predict and update steps. In the *split* step the finer set of smoothing coefficients s_j is split into two disjoint sets s_{j-1} and d_{j-1}. The *Lazy wavelet* is a wavelet which has a fast transform that only consists out of the split step. Hence, the Lazy wavelet is only a split of a data set. The Lazy wavelet splits the set into even and odd points, this is the type of split which will be used in this section.

The second step is the *predict* step. In this step smoothing coefficients at the same level, s_{j-1}, are used to predict the other set, d_{j-1}. The predict step is executed with the prediction operator $\mathcal{P}$. This gives the following new equation for the detail coefficients:

$$d_{j-1} = d_{j-1} - \mathcal{P}(s_{j-1}).$$

Hence, the detail coefficients now imply how much the data deviates from the interpolation model $\mathcal{P}$. The operator $\mathcal{P}$ is linear and can be mathematically written as $\mathcal{P}(s_{j-1,k}) = \sum_l p_l s_{j-1,k+l}$. Thus the predict step has its own new filter, P.
In the last step of the forward transform the smoothing coefficients are *updated* in order to preserve certain properties of the smoothing coefficients, e.g. the average value of the smoothing coefficients stays the same during the coarsening process. This is done by applying an operator $\mathcal{U}$ to the detail coefficients at the same level:

$$s_{j-1} = s_{j-1} + \mathcal{U}(d_{j-1}).$$

Note that this step is calculated after all the detail coefficients have been predicted. The operator $\mathcal{U}$ is also linear and can be mathematically written as $\mathcal{U}(d_{j-1,k}) = \sum_l u_l d_{j-1,k+l}$, the update filter is thus U.
In Algorithm 6, the fast transform using the lifting scheme is given.

Algorithm 6 Lifting scheme: forward transform

```
for j=J2  downto  J1 do
    $[s_{j-1}, d_{j-1}] = \text{split}(s_j)$
    $d_{j-1} = d_{j-1} - \mathcal{P}(s_{j-1})$
    $s_{j-1} = s_{j-1} + \mathcal{U}(d_{j-1})$
end for
```

The inverse transform is simply the reverse of the forward transform. The inverse transform exists out of three steps, undo update, undo predict and merge. Firstly, the update will be undone, followed by undoing the prediction and in the end the two subsets are merged back to one set. Note, the order in which the steps are performed is important. This leads to Algorithm 7.

Algorithm 7 Lifting scheme: invers transform

```
for j=J1  upto  J2 do
    $s_{j-1} = s_{j-1} - \mathcal{U}(d_{j-1})$
    $d_{j-1} = d_{j-1} + \mathcal{P}(s_{j-1})$
    $s_j = \text{merge}(s_{j-1}, d_{j-1})$
end for
```

In Figure 3.10 the wiring diagram of the full fast transform is given.

In section 3.2 two examples of biorthogonal wavelets are given, the CDF(2,2) and the Haar wavelet. Here, another version of the Haar wavelet (different dual order) and the

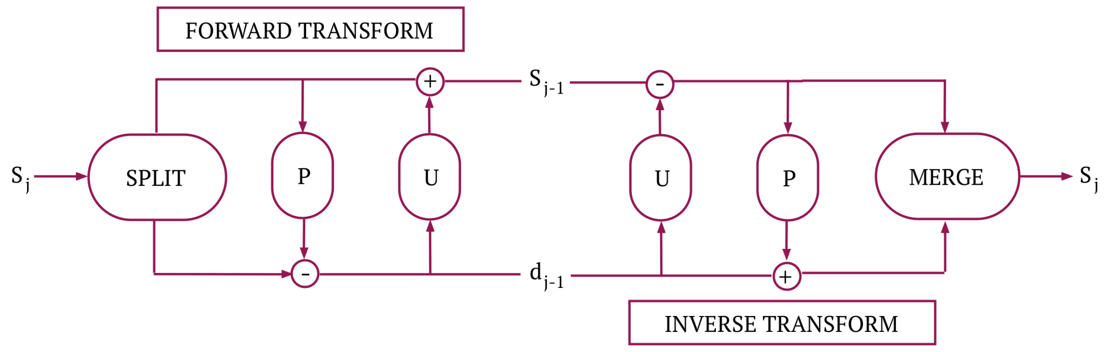


Figure 3.10: Wiring diagram of the lifted scheme.

CDF(2,2) will be considered, however this time in the lifting scheme setting. There is no need to construct the scaling and dual functions, the functions will be generated during the forward transform of the lifted scheme. In order to compare the lifting scheme forward transform with the classical forward transform, the filters of the Haar and CDF(2,2) wavelets are given in Table 3.3.

Table 3.3: The $\tilde{H}$ and $\tilde{G}$ filter coefficients of a biorthogonal Haar wavelet and CDF(2,2).

	Haar			CDF(2,2)	
k	$\tilde{h}_k$	$\tilde{g}_k$	k	$\tilde{h}_k$	$\tilde{g}_k$
-2			-2	$-\frac{1}{8}$	
-1			-1	$\frac{1}{4}$	
0	$\frac{1}{2}$	-1	0	$\frac{3}{4}$	$-\frac{1}{2}$
1	$\frac{1}{2}$	1	1	$\frac{1}{4}$	1
2			2	$-\frac{1}{8}$	$-\frac{1}{2}$

Biorthogonal Haar wavelet

The classical forward transform of the Haar wavelet is:

$$\begin{cases} s_{j-1,k} = \frac{1}{2}s_{j,2k} + \frac{1}{2}s_{j,2k+1} \\ d_{j-1,k} = -s_{j,2k} + s_{j,2k+1} \end{cases}$$

In the lifting scheme one starts with the Lazy wavelet, i.e. splitting the finer level into two subsets. The split is done by dividing s_j into a set of even points (s_{j-1}) and odd points (d_{j-1}). The goal is to reduce the value of the detail coefficients such that thresholding can be efficiently applied. Hence, the detail coefficients (d_{j-1}) will be predicted using the smoothing coefficients (s_{j-1}). In the case of the Haar wavelet the predictor operator is equal to $\mathcal{P}(s_{j-1,k}) = s_{j-1,k}$:

$$d_{j-1,k} = d_{j-1,k} - s_{j-1,k}.$$

Thus the detail coefficients are simply predicted by the value of the neighbouring smoothing coefficient. Because $s_{j-1,k} = s_{j,2k}$ and $d_{j-1,k} = s_{j,2k+1}$, in classical form the prediction would look like:

$$d_{j-1,k} = s_{j,2k+1} - s_{j,2k}.$$

This corresponds to the classical forward transform of the biorthogonal Haar wavelet. Next, the smoothing coefficients are updated. One property of the Haar wavelet is that the average value of the smoothing coefficients stays the same:

$$\frac{1}{2}\sum_{k=0}^{2^j} s_{j,k} = \sum_{k=0}^{2^{j-1}} s_{j-1,k}, \tag{3.15}$$

where the $\frac{1}{2}$ comes from the fact that level j has twice as many points as $j-1$. The property 3.15 can be used to find the update operator $\mathcal{U}$. Choosing the update operator as $\mathcal{U}(d_{j-1,k}) = \frac{1}{2}d_{j-1,k}$ will ensure the average to be preserved. This can be verified by substituting the following equation into 3.15:

$$s_{j-1,k} = s_{j-1,k} + \frac{1}{2}d_{j-1,k}.$$

Substituting the detail coefficient gives:

$$s_{j-1,k} = s_{j,2k} + \frac{1}{2}s_{j,2k+1} - \frac{1}{2}s_{j,2k}.$$

Again, this is the same result as the classical forward transform. In Table 3.4 the filters P and U are given. The gain is in the fact that no additional data storage is needed. The finer level does not have to be kept and can be completely overwritten. While in the classical case s_j needs to be kept in order to make s_{j-1} and d_{j-1}, only after these coefficients have been created the s_j coefficients can be removed.
In short, the lifted forward transform of the biorthogonal Haar wavelet is:

$$\begin{cases} s_{j-1,k} = s_{j,2k}, \\ d_{j-1,k} = s_{j,2k+1}, \\ d_{j-1,k} = d_{j-1,k} - s_{j-1,k}, \\ s_{j-1,k} = s_{j-1,k} + \frac{1}{2}d_{j-1,k}. \end{cases}$$

CDF(2,2)

The classical forward transform of the CDF(2,2) is:

$$\begin{cases} s_{j-1,k} = -\frac{1}{8}s_{j,2k-2} + \frac{1}{4}s_{j,2k-1} + \frac{3}{4}s_{j,2k} + \frac{1}{4}s_{j,2k+1} - \frac{1}{8}s_{j,2k+2} \\ d_{j-1,k} = -\frac{1}{2}s_{j,2k} + s_{j,2k+1} - \frac{1}{2}s_{j,2k+2} \end{cases}$$

Table 3.4: The P and U filter coefficients of a biorthogonal Haar wavelet and CDF(2,2).

	Haar			CDF(2,2)	
k	p_k	u_k	k	p_k	u_k
-1			-1		$\frac{1}{4}$
0	1	$\frac{1}{2}$	0	$\frac{1}{2}$	$\frac{1}{4}$
1			1	$\frac{1}{2}$	

After starting with the Lazy wavelet, the detail coefficient will be predicted by averaging the neighbouring smoothing coefficients (linear prediction), i.e. $\mathcal{P}(s_{j-1,k}) = \frac{1}{2}s_{j-1,k} + \frac{1}{2}s_{j-1,k+1}$:

$$d_{j-1,k} = d_{j-1,k} - \frac{1}{2}(s_{j-1,k} + s_{j-1,k+1}),$$

which corresponds to:

$$d_{j-1,k} = -\frac{1}{2}s_{j,2k} + s_{j,2k+1} - \frac{1}{2}s_{j,2k+2}.$$

Again, the update operator ensures that the average is preserved. The operator which achieves this is equal to $\mathcal{U}(d_{j-1,k}) = \frac{1}{4}d_{j-1,k-1} + \frac{1}{4}d_{j-1,k}$:

$$s_{j-1,k} = s_{j-1,k} + \frac{1}{4}(d_{j-1,k-1} + d_{j-1,k}),$$

which corresponds to:

$$\begin{aligned} s_{j-1,k} &= s_{j,2k} + \frac{1}{4}(s_{j,2k-1} - \frac{1}{2}s_{j,2k-2} - \frac{1}{2}s_{j,2k} + s_{j,2k+1} - \frac{1}{2}s_{j,2k} - \frac{1}{2}s_{j,2k+2}), \\ &= -\frac{1}{8}s_{j,2k-2} + \frac{1}{4}s_{j,2k-1} + \frac{3}{4}s_{j,2k} + \frac{1}{4}s_{j,2k+1} - \frac{1}{8}s_{j,2k+2} \end{aligned}$$

The averaging property can be verified by substituting the above in equation 3.15. The lifting steps result in the same forward transform as the classical forward transform. The predict and update filters can be found in Table 3.4.
In short, the lifted forward transform of CDF(2,2) is:

$$\begin{cases} s_{j-1,k} = s_{j,2k}, \\ d_{j-1,k} = s_{j,2k+1}, \\ d_{j-1,k} = d_{j-1,k} - \frac{1}{2}(s_{j-1,k} + s_{j-1,k+1}), \\ s_{j-1,k} = s_{j-1,k} + \frac{1}{4}(d_{j-1,k-1} + d_{j-1,k}). \end{cases}$$

3.6.3 Lifting to construct second generation wavelets

In this subsection it is explained how to built new wavelets using the lifting scheme. Earlier, lifting was introduced as adding and subtracting of linear operators, these operators resulted in new filters P and U. Combining these filters with the filters of the starting wavelet will result in the filters of the new wavelet.

Primal lifting

In the forward transform of the lifting scheme the update step adds the linear operator $\mathcal{U}$ to the smoothing coefficients. This corresponds to changing the $\tilde{H}^{old}$ filter of the original wavelet to the filter $\tilde{H}$ of the newly built wavelet. This process is also called *'primal lifting'*. Because the $\tilde{H}$ wavelet is directly connected to the G filter, this filter also changes. How the filters change is stated in the theorem below extracted from [25].

Theorem 3.6.1 - Primal lift

Take an initial set of biorthogonal filter operators $\{h^{old}, \tilde{h}^{old}, g^{old}, \tilde{g}^{old}\}$. Then a new set of biorthogonal filter operators $\{h, \tilde{h}, g, \tilde{g}\}$ can be found as

$$\begin{aligned} h_{j,k} &= h^{old}_{j,k}, \\ \tilde{h}_{j,k} &= \tilde{h}^{old}_{j,k} + \sum_l u_{j,l}\tilde{g}^{old}_{j,k+l}, \\ g_{j,k} &= g^{old}_{j,k} - \sum_l u^*_{j,l} h^{old}_{j,k+l}, \\ \tilde{g}_{j,k} &= \tilde{g}^{old}_{j,k}, \end{aligned}$$

where $u^*_{j,l}$ is some shifted version of $u_{j,l}$.

The filters can be written as 2π-periodic continuous functions instead of discrete functions. This notation can be useful for the construction and interpretation of the liftings.

Definition 4 - 2π-periodic filters

Consider H and G to be the filters of a wavelet system, then the 2π-periodic filters are defined as

$$\begin{aligned} h(\omega) &= \sum_k h_k e^{-ki\omega}, \\ g(\omega) &= \sum_k g_k e^{-ki\omega}. \end{aligned}$$

$\tilde{h}(\omega)$ and $\tilde{g}(\omega)$ are defined in a similar manner.

This leads to a new version of theorem 3.6.1.

Theorem 3.6.2 - Primal lift

Take an initial set of biorthogonal filter operators $\{h^{old}, \tilde{h}^{old}, g^{old}, \tilde{g}^{old}\}$. Then a new set of biorthogonal filter operators $\{h, \tilde{h}, g, \tilde{g}\}$ can be found as

$$\begin{aligned} h(\omega) &= h^{old}(\omega), \\ \tilde{h}(\omega) &= \tilde{h}^{old}(\omega) + \tilde{g}^{old}(\omega)\overline{u(2\omega)}, \\ g(\omega) &= g^{old}(\omega) - h^{old}(\omega)u(2\omega), \\ \tilde{g}(\omega) &= \tilde{g}^{old}(\omega), \end{aligned}$$

where $\overline{u(2\omega)}$ *indicates the complex conjugate of* $u(2\omega)$.

As a consequence of the primal lifting, the scaling and wavelet functions are altered, remember:

$$\begin{aligned}
\varphi(x) &= \sum_k h_k \varphi(2x-k), \\
\tilde{\varphi}(x) &= \sum_k \tilde{h}_k \tilde{\varphi}(2x-k), \\
\psi(x) &= \sum_k g_k \varphi(2x-k), \\
\tilde{\psi}(x) &= \sum_k \tilde{g}_k \tilde{\varphi}(2x-k).
\end{aligned}$$

Comparing this to theorem 3.6.1, it can be concluded that the dual scaling functions and primal wavelet functions are altered due to the updated filters. Because the dual wavelet functions are constructed from the dual mother scaling function, the dual wavelet functions also change although the filter does not change.

The result of these theorems is that if one starts off with a biorthogonal set of filters, then after lifting the new set of filters are again biorthogonal. This does not depend on the filter choice U, i.e. the filter coefficients u_k can be freely chosen. However, it is not ensured that the set of new filters give scaling and dual functions which belong to $L^2(\mathbb{R})$ and form a Riesz basis.
Typically, the filter coefficients are chosen such that the number of vanishing moments of the wavelet $\tilde{N}$ will be increased. In section 3.4, Donoho's interpolating wavelet of arbitrary order N and dual order $\tilde{N} = 0$ is discussed. This wavelet has aliasing problems due to the dual order, the primal lifting can therefore improve Donoho's interpolating wavelet.
Choosing the U filter such that the dual order is increased can be done by firstly noticing that $\psi^{\text{new}}(x) = \psi^{\text{old}}(x) - \sum_k u_k \varphi^{\text{old}}(x)$. This can be applied to the formal definition of the dual order to get a linear system which can be solved for u_k:

$$\int_{-\infty}^{\infty} x^n \psi^{\text{new}}(x)dx = \int_{-\infty}^{\infty} x^n \psi^{\text{old}}(x)dx - \sum_k u_k \int_{-\infty}^{\infty} x^n \varphi^{\text{old}}(x)dx.$$

Cakewalk construction

The prediction operator $\mathcal{P}$ in the forward transform tends to predict the values of the detail coefficients such that thresholding can be applied to these values. The prediction operator is subtracted from the detail coefficients, i.e. the filter $\tilde{G}^{\text{old}}$ is lifted to the new filter $\tilde{G}^{\text{new}}$. This manner of lifting is called *'dual lifting'*. Dual lifting also preserves the biorthogonallity of a set of filters:

Theorem 3.6.3 - Dual lift

Take an initial set of biorthogonal filter operators $\{h^{old}, \tilde{h}^{old}, g^{old}, \tilde{g}^{old}\}$*. Then a new set of biorthogonal filter operators* $\{h, \tilde{h}, g, \tilde{g}\}$ *can be found as*

$$\begin{aligned}
h(\omega) &= h^{old}(\omega) + g^{old}(\omega)\overline{p(2\omega)},\\
\tilde{h}(\omega) &= \tilde{h}^{old}(\omega),\\
g(\omega) &= g^{old}(\omega),\\
\tilde{g}(\omega) &= \tilde{g}^{old}(\omega) - \tilde{h}^{old}(\omega)p(2\omega).
\end{aligned}$$

The lifting scheme is introduced with the predict and update operators, indicating that one starts with the prediction operator followed by the update operator. However, one can alternate the update and predict steps, i.e. the primal and dual lifting phases can be alternated in order to find a custom designed wavelet. Alternating these two different lifting phases is called a cakewalk construction. In [25] it is shown that primal lifting will not change the order N and that the dual lifting will not change the dual order $\tilde{N}$. So, after constructing a wavelet of order N, the primal lifting can be applied to increase the dual order $\tilde{N}$ without changing the order N.

3.6.4 Lifting Donoho

In section 3.4 Donoho's interpolating wavelet is introduced, in section 3.5 the aliasing problem of Donoho's interpolating wavelet is discussed. By applying a primal lifting to Donoho's interpolating wavelet, this aliasing problem can be solved by increasing the dual order $\tilde{N}$. The lifted version of Donoho's interpolating wavelet will be denoted as *Sweldens wavelet.*
Firstly, it will be shown how Donoho's interpolating wavelet can be constructed by lifting the Lazy wavelet. Thereafter, the Donoho's interpolating wavelet will be lifted to an arbitrary dual order.

Lifting the Lazy wavelet to Donoho's interpolating wavelet

In the construction of Donoho's interpolating wavelet a scaled version of the Lazy wavelet will be used:

$$\begin{aligned}
h^{\text{old}}(\omega) &= \frac{1}{2},\\
\tilde{h}^{\text{old}}(\omega) &= 1,\\
g^{\text{old}}(\omega) &= e^{-i\omega},\\
\tilde{g}^{\text{old}}(\omega) &= \frac{e^{-i\omega}}{2}.
\end{aligned}$$

Table 3.5: The filter coefficients of the scaled Lazy wavelet.

	Lazy wavelet			
k	h_k	$\tilde{h}_k$	g_k	$\tilde{g}_k$
0	$\frac{1}{2}$	1		
1			1	$\frac{1}{2}$

Which corresponds to the filters in Table 3.5.
In Table 3.6 the H^N filters of Donoho's interpolating wavelet are given, note that they are scaled such that $\sum_k h_k = 1$ holds. In continuous form this filter can written as:

$$h^N(\omega) = \frac{1}{2} + e^{-i\omega}\overline{p^N(2\omega)}, \tag{3.16}$$

for arbitrary N. In this notation it can be clearly seen that Donoho's interpolating wavelet is a lifted version of the Lazy wavelet. This $\overline{p^N(2\omega)}$ can be determined by rewriting the filter $h^N(\omega)$ functions of Donoho. For example, the case $N = 4$:

$$\begin{aligned} h^4(\omega) &= -\frac{1}{2^5}e^{3i\omega} + \frac{9}{2^5}e^{i\omega} + \frac{1}{2} + \frac{9}{2^5}e^{-i\omega} - \frac{1}{2^5}e^{-3i\omega} \\ &= \frac{1}{2} + e^{-i\omega}(-\frac{1}{2^5}e^{4i\omega} + \frac{9}{2^5}e^{2i\omega} + \frac{9}{2^5} - \frac{1}{2^5}e^{-2i\omega}) \\ \overline{p^N(2\omega)} &= -\frac{1}{2^5}e^{4i\omega} + \frac{9}{2^5}e^{2i\omega} + \frac{9}{2^5} - \frac{1}{2^5}e^{-2i\omega} \\ p^N(2\omega) &= -\frac{1}{2^5}e^{2i\omega} + \frac{9}{2^5} + \frac{9}{2^5}e^{-2i\omega} - \frac{1}{2^5}e^{-4i\omega} \end{aligned}$$

The dual lifting filters P^N can be found in Table 3.6.
By performing a dual lifting with $p^N(2\omega)$, the Lazy wavelet is lifted to Donoho's interpolating wavelet. The continuous filters will look as follows:

$$\begin{aligned} h^{\text{new}}(\omega) &= \frac{1}{2} + e^{-i\omega}\overline{p^N(2\omega)}, \\ \tilde{h}^{\text{old}}(\omega) &= 1, \\ g^{\text{old}}(\omega) &= e^{-i\omega}, \\ \tilde{g}^{\text{new}}(\omega) &= \frac{e^{-i\omega}}{2} - p^N(2\omega). \end{aligned}$$

Or in lifting form:

$$\begin{aligned} h^N(\omega) &= h^{\text{old}}(\omega) + g^{\text{old}}(\omega)\overline{p^N(2\omega)}, \\ \tilde{h}^{\text{old}}(\omega) &= \tilde{h}^{\text{old}}(\omega), \\ g^{\text{old}}(\omega) &= g^{\text{old}}(\omega), \\ \tilde{g}(\omega) &= \tilde{g}^{\text{old}}(\omega) - \tilde{h}^{\text{old}}(\omega)p^N(2\omega). \end{aligned}$$

Table 3.6: The scaled Donoho filter H^N, dual filter P^N and Sweldens filter $\tilde{H}$ for $N = 4, \tilde{N} = 2$.

	Donoho H^N				P^N				Sweldens $\tilde{H}$
k	$N=2$	$N=4$	$N=6$	k	$N=2$	$N=4$	$N=6$	k	$\tilde{h}_k$
-5			$\frac{3}{2^9}$	-4			$\frac{3}{2^9}$	-5	
-4			0	-3			0	-4	$\frac{1}{2^6}$
-3		$-\frac{1}{2^5}$	$-\frac{25}{2^9}$	-2		$-\frac{1}{2^5}$	$-\frac{25}{2^9}$	-3	0
-2		0	0	-1		0	0	-2	$-\frac{1}{2^3}$
-1	$\frac{1}{2^2}$	$\frac{9}{2^5}$	$\frac{150}{2^9}$	0	$\frac{1}{2^2}$	$\frac{9}{2^5}$	$\frac{150}{2^9}$	-1	$\frac{1}{2^2}$
0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1	0	0	0	0	$\frac{23}{5}$
1	$\frac{1}{2^2}$	$\frac{9}{2^5}$	$\frac{150}{2^9}$	2	$\frac{1}{2^2}$	$\frac{9}{2^5}$	$\frac{150}{2^9}$	1	$\frac{1}{2^2}$
2		0	0	3		0	0	2	$-\frac{1}{2^3}$
3		$-\frac{1}{2^5}$	$-\frac{25}{2^9}$	4		$-\frac{1}{2^5}$	$-\frac{25}{2^9}$	3	0
4			0	5			0	4	$\frac{1}{2^6}$
5			$\frac{3}{2^9}$	6			$\frac{3}{2^9}$	5	

Lifting Donoho's interpolating wavelet to Sweldens wavelet

The mother wavelet of Donoho's wavelet system has zero vanishing moments. This leads to inconvenient aliasing and because there is no vanishing integral there is no Riesz basis for $L^2(\mathbb{R})$. Therefore, primal lifting will be applied to increase the number of vanishing moments of the wavelet. Only the case where $\tilde{N} \leq N$ will be considered. In [24] Sweldens gives a theorem to lift Donoho's interpolating wavelet to the Sweldens wavelet of dual order $\tilde{N} \leq N$.

Theorem 3.6.4 - Lifting Donoho

Consider Donoho's interpolating wavelet of order N. If $\tilde{N} \leq N$, lifting with

$$\overline{u(2\omega)} = 2p^{\tilde{N}}(-2\omega),$$

results in the shortest wavelet with $\tilde{N}$ vanishing moments which is symmetric around $\frac{1}{2}$.

Note that $p^{\tilde{N}}(-2\omega) = \overline{p^{\tilde{N}}(2\omega)}$. Applying this primal lift to Donoho's interpolating wavelet, gives:

$$\begin{aligned}
h^{\text{new}}(\omega) &= \frac{1}{2} + e^{-i\omega}\overline{p^N(2\omega)},\\
\tilde{h}^{\text{old}}(\omega) &= 1 + (\frac{e^{-i\omega}}{2} - p^N(2\omega))2\overline{p^{\tilde{N}}(2\omega)},\\
g^{\text{old}}(\omega) &= e^{-i\omega} - (\frac{1}{2} + e^{-i\omega}\overline{p^N(2\omega)})2p^{\tilde{N}}(2\omega),\\
\tilde{g}^{\text{new}}(\omega) &= \frac{e^{-i\omega}}{2} - p^N(2\omega).
\end{aligned}$$

Or in lifting form:

$$\begin{aligned}
h^N(\omega) &= h^{\text{old}}(\omega) + g^{\text{old}}(\omega)\overline{p^N(2\omega)},\\
\tilde{h}(\omega) &= \tilde{h}^{\text{old}}(\omega) + \tilde{g}(\omega)\overline{u(2\omega)},\\
g(\omega) &= g^{\text{old}}(\omega) - h(\omega)u(2\omega),\\
\tilde{g}(\omega) &= \tilde{g}^{\text{old}}(\omega) - \tilde{h}^{\text{old}}(\omega)p^N(2\omega).
\end{aligned}$$

In order to make this more concrete the Sweldens $\tilde{H}$ filter of order $N = 4$ and dual order $\tilde{N} = 2$ will be calculated. Note that normally these calculations do not have to be performed as the direct filter is not needed in the lifting scheme. However, this filter can be used in order to check the implementation of the lifted scheme and it can give clarification on the formulas written above.

$$\begin{aligned}
\overline{p^2(2\omega)} &= \frac{1}{2^2}e^{2i\omega} + \frac{1}{2^2},\\
p^4(2\omega) &= -\frac{1}{2^5}e^{2i\omega} + \frac{9}{2^5} + \frac{9}{2^5}e^{-2i\omega} - \frac{1}{2^5}e^{-4i\omega},\\
\tilde{h}(\omega) &= 1 + (\frac{e^{-i\omega}}{2} - p^4(2\omega))2\overline{p^2(2\omega)},\\
&= 1 + (\frac{1}{2^5}e^{2i\omega} - \frac{9}{2^5} + \frac{1}{2}e^{-i\omega} - \frac{9}{2^5}e^{-2i\omega} + \frac{1}{2^5}e^{-4i\omega})(\frac{1}{2}e^{2i\omega} + \frac{1}{2}),\\
&= 1 + \frac{1}{2}(\frac{1}{2^5}e^{4i\omega} - \frac{9}{2^5}e^{2i\omega} + \frac{1}{2}e^{i\omega} - \frac{9}{2^5} + \frac{1}{2^5}e^{-2i\omega}\\
&\quad + \frac{1}{2^5}e^{2i\omega} - \frac{9}{2^5} + \frac{1}{2}e^{-i\omega} - \frac{9}{2^5}e^{-2i\omega} + \frac{1}{2^5}e^{-4i\omega}),\\
&= \frac{1}{2^6}e^{4i\omega} - \frac{1}{2^3}e^{2i\omega} + \frac{1}{4}e^{i\omega} + \frac{23}{2^5} + \frac{1}{4}e^{-i\omega} - \frac{1}{2^3}e^{-2i\omega} + \frac{1}{2^6}e^{-4i\omega}
\end{aligned}$$

In discrete form the $\tilde{H}$ filter can be found in Table 3.6. The big advantage of designing a wavelet system in this manner is that the implementation of the fast transform is trivial. The forward transform is almost exactly the same as in Algorithm 6, with $\mathcal{P}(s_{j-1,k}) = \sum_l p_l^N s_{j-1,k+l}$ and $\mathcal{U}(d_{j-1,k}) = 2\sum_l p_{-l}^{\tilde{N}} d_{j-1,k+l}$. The only difference is that directly after the split all the detail coefficients should be halved after which both the detail coefficients and the smoothing coefficients will be subject to downsampling. Downsampling means that the coefficients are multiplied by $\sqrt{2}$, this because the fast forward transform related to this MRA setting looks as follows:

$$\begin{cases} d_{j-1,m} = \sqrt{2}\sum_n \tilde{g}_n s_{j,2m+n}, \\ s_{j-1,k} = \sqrt{2}\sum_l \tilde{h}_l s_{j,2k+l}. \end{cases}$$

The algorithm of the fast forward transform can be found in Algorithm 8. Furthermore, the wiring diagram of the Sweldens wavelet is given in Figure 3.11.

Algorithm 8 Lifting scheme: forward transform of Sweldens wavelet

```
for j=J2  downto  J1 do
    $[s_{j-1}, d_{j-1}] = \text{split}(s_j)$
    $d_{j-1} = \frac{1}{2} d_{j-1}$
    $s_{j-1} = \sqrt{2} s_{j-1}$
    $d_{j-1} = \sqrt{2} d_{j-1}$
    $d_{j-1} = d_{j-1} - \mathcal{P}(s_{j-1})$
    $s_{j-1} = s_{j-1} + \mathcal{U}(d_{j-1})$
end for
```

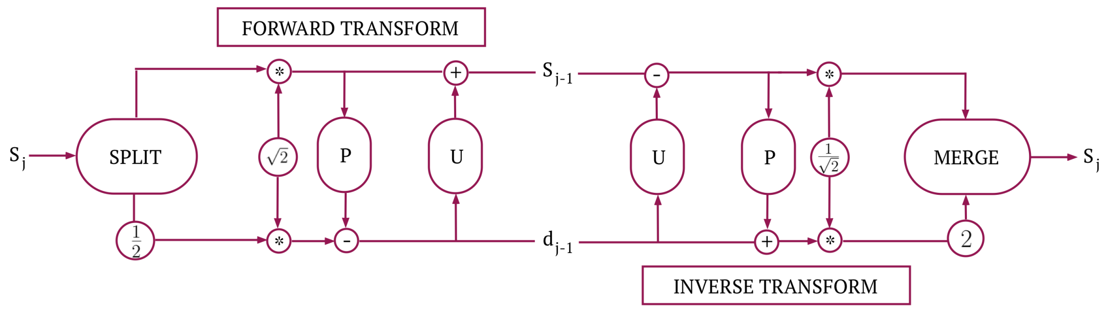


Figure 3.11: Wiring diagram of the Sweldens wavelet.

3.6.5 Sweldens wavelet

The Sweldens wavelet is formed by lifting the Donoho wavelet, this lifting is done by adding an update step to the smoothing coefficients. This update step increases the dual order of the wavelet. In this subsection the effect of lifting this dual order is discussed shortly.

As mentioned before in the sections 3.4 and 3.5, the Donoho wavelet has difficulties with aliasing, because the smoothing coefficients are simply translates of the actual function values. To improve this behaviour, the smoothing coefficients are updated. Because of the update step the smoothing coefficients at level j are not simply the values of the

corresponding coefficient in the higher level $j+1$, but they are an average of the detail coefficients and the corresponding smoothing coefficient at the higher level. Due to this fact, the smoothing coefficients actually 'smooth' the function, while going from the dense level $J2$ to the coarse level $J1$.
To illustrate this new smoothing property, an example is introduced. The function which will be discussed is:

$$f(x) = \cos 2\pi x + \text{noise}.$$

In order to illustrate the difference between the Donoho and Sweldens smoothing coefficients, the added noise is high, between -1 and +1. In Figure 3.12 the function is depicted, it can be seen that there is a lot of noise. The forward transform has been applied to the function with noise. Both with Donoho's wavelet and Sweldens wavelet. The densest level used is $J2 = 18$ and the coarsest level $J1 = 5$. In Figure 3.13 one step of the forward transform using Sweldens wavelet has been applied. It can be seen that already in the first step smoothing is visible. In Figure 3.14 the smoothing coefficients of Sweldens wavelet at level $J1$ after the forward transform are given, it is clear that the Sweldens wavelet has been able to remove the noise and retrieve the cosine function. In Figure 3.15 the smoothing coefficients of Donoho are given after the forward transform, the function is highly irregular and it can not be concluded from these smoothing coefficients that the original signal was a cosine function. These results suggest that Sweldens wavelet will be able to reduce noise, this could be interesting when an initial solution of a partial differential equation is based on measured data.

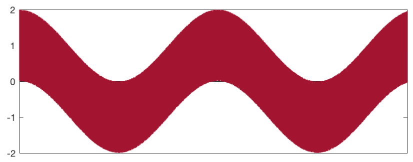


Figure 3.12: $\cos(2\pi x)$ with noise.

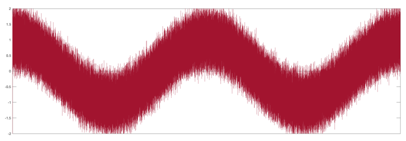


Figure 3.13: 1 Sweldens smoothing step.

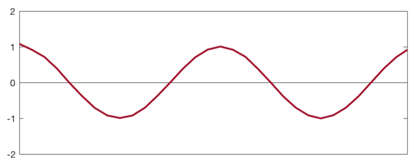


Figure 3.14: Sweldens forward transform.

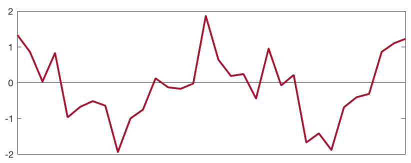


Figure 3.15: Donoho forward transform.

3.6.6 Boundary stencil

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 which are not defined. There are multiple ways to deal with the boundary. In this subsection two methods will be discussed. Note that in the Sweldens case 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. The only differences are that the predict filter depends on N, while the update filter depends on $\tilde{N}$. Furthermore, the predict filter works on the smoothing coefficients, while the update filter uses the detail coefficients. Because the cases are so similar only the boundary handling of the predict step will be discussed.

In the case of order $N = 2$ the predict step is:

$$d_{j,k} = d_{j,k} - (\frac{1}{4}s_{j,k} + \frac{1}{4}s_{j,k+1}).$$

Hence, on the right boundary an issue will arise if the point $s_{j,k+1}$ is needed. In Figure 3.16 the predict step for $N = 2$ is schematically given. The boundary problem occurs here for the detail coefficient $d_{j,3}$, as mentioned before, two methods have been used to solve this issue.

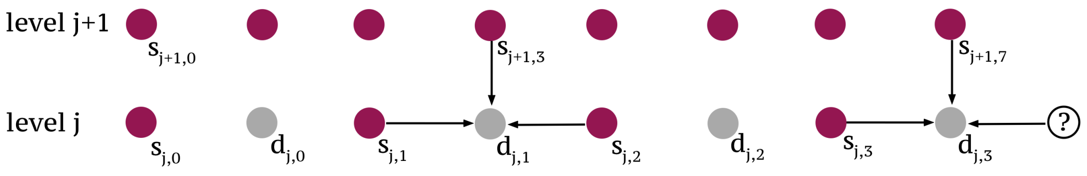


Figure 3.16: Predict step of order $N = 2$.

Cascade algorithm

In this subsection figures of the wavelets will be shown. These wavelets were generated by using the *cascade algorithm*, in the cascade algorithm one puts all the smoothing coefficients at level $J1$ to zero, all the detail coefficients at levels $J1$ up until $J2 - 1$ will be set to zero as well. Every smoothing coefficient and detail coefficient corresponds to either a scaling or wavelet function. If a value of a detail coefficient at an arbitrary level is set to 1 the wavelet corresponding to that point can be retrieved by performing the inverse transform on the set of smoothing and detail coefficients which are all zero. This has been done for the detail coefficients near the boundary at level $J1$ in the figures.

Interpolating boundary stencil

Both the predict and update step are based on fitting a polynomial on N points. In other words for the predict step, a polynomial is fitted on the points $s_{j,k-N/2+1}$ until $s_{j,k+N/2}$. Thereafter the predict value $d_{j,k}$ is found by filling the location into the polynomial. One manner to treat the boundary is to use N smoothing coefficients closest to the boundary and fit a polynomial on these points. Again, the predict value $d_{j,k}$ is found by filling the location into the polynomial. In Figure 3.17 this is depicted schematically. It can be observed that both $d_{j,2}$ and $d_{j,3}$ in the figure will have the same polynomial, namely the polynomial fitted on $s_{j,2}$ and $s_{j,3}$. The difference is the location where this polynomial is evaluated. For the predict step of $d_{j,2}$ the polynomial will be evaluated at location 2.5, while for $d_{j,3}$ the polynomial will be evaluated at 3.5.

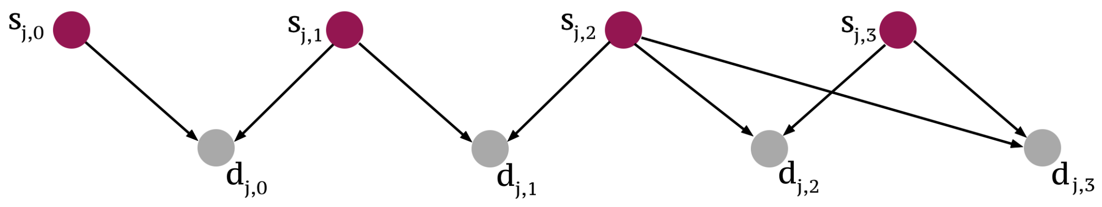


Figure 3.17: Predict step of order $N = 2$ using the interpolating boundary stencil.

The boundary predict step can be written as a filter, similar to the predict filter. This is done by using the Lagrange polynomial:

$$L(x) = \sum_{i=0}^{N-1} y_i l_i(x), \text{ for the samples } (x_0, y_0), \ldots, (x_{N-1}, y_{N-1}),$$

$$\text{and } l_i(x) = \prod_{\substack{0 \le m \le N-1 \\ m \ne i}} \frac{x - x_m}{x_i - x_m}.$$

To show how the Lagrange polynomial can be used to create a filter, the right boundary of the Sweldens wavelet of order $N = 2$ is considered. Assume that at level j there are $m + 1$ detail coefficients, i.e., $d_{j,0}$ until $d_{j,m}$. The predict step of $d_{j,m}$ is:

$$d_{j,m} = d_{j,m} - L^{R,N=2}(m + \frac{1}{2}).$$

$L^{R,N=2}(x)$ denotes the polynomial fitted on the samples $(x_0, y_0) = (m - 1, s_{j,m-1})$ and $(x_1, y_1) = (m, s_{j,m})$. This polynomial is evaluated in $m+\frac{1}{2}$, because the detail coefficient $d_{j,m}$ is located a half step to the right of the smoothing coefficient $s_{j,m}$. The polynomial is given by:

$$\begin{aligned} L^{R,N=2}(x) &= s_{j,m-1} l_0(x) + s_{j,m} l_1(x), \\ l_0(x) &= \frac{x - m}{m - 1 - m} = -x + m, \\ l_1(x) &= \frac{x - (m - 1)}{m - (m - 1)} = x - m + 1. \end{aligned}$$

Evaluating this polynomial at $m + \frac{1}{2}$, gives $L^{R,N=2}(m + \frac{1}{2}) = s_{j,m-1}(-\frac{1}{2}) + s_{j,m}(\frac{3}{2})$. So the predict step at the right boundary becomes

$$d_{j,m} = d_{j,m} - (-\frac{1}{2} s_{j,m-1} + \frac{3}{2} s_{j,m}).$$

Table 3.7: The interpolating boundary stencil predict filters $P^{L,N}$ and $P^{R,N}$.

	Left Predict $P^{L,N}$				Right Predict $P^{R,N}$					
	$N=4$	$N=6$			$N=2$	$N=4$		$N=6$		
k	d_0	d_0	d_1	k	d_m	d_{m-1}	d_m	d_{m-2}	d_{m-1}	d_m
0	$\frac{5}{2^5}$	$\frac{63}{2^9}$	$-\frac{7}{2^9}$	m-5				$-\frac{3}{2^9}$	$\frac{7}{2^9}$	$-\frac{63}{2^9}$
1	$\frac{15}{2^5}$	$\frac{315}{2^9}$	$\frac{105}{2^9}$	m-4				$\frac{21}{2^9}$	$-\frac{45}{2^9}$	$\frac{385}{2^9}$
2	$-\frac{5}{2^5}$	$-\frac{105}{2^8}$	$\frac{105}{2^8}$	m-3		$\frac{1}{2^5}$	$-\frac{5}{2^5}$	$-\frac{35}{2^8}$	$\frac{63}{2^8}$	$-\frac{495}{2^8}$
3	$\frac{1}{2^5}$	$\frac{63}{2^8}$	$-\frac{35}{2^8}$	m-2		$-\frac{5}{2^5}$	$\frac{21}{2^5}$	$\frac{105}{2^9}$	$-\frac{105}{2^8}$	$\frac{693}{2^8}$
4		$-\frac{45}{2^9}$	$\frac{21}{2^9}$	m-1	$-\frac{1}{4}$	$\frac{15}{2^5}$	$-\frac{35}{2^5}$	$\frac{105}{2^9}$	$\frac{315}{2^9}$	$-\frac{1155}{2^9}$
5		$\frac{7}{2^9}$	$-\frac{3}{2^9}$	m	$\frac{3}{4}$	$\frac{5}{2^5}$	$\frac{35}{2^5}$	$-\frac{7}{2^9}$	$\frac{63}{2^9}$	$\frac{693}{2^9}$

This leads to the predict filter $P^R = (-\frac{1}{2}, \frac{3}{2}, 0)$.
It is important to note that if wavelets of higher order are used, e.g., $N = 4$, then there are more points which need a boundary filter. In the case of $N = 4$ this leads to one boundary filter on the left side and two different boundary filters on the right side. Both the right points are predicted using the same polynomial, however the polynomial is evaluated on different locations. Furthermore, the Sweldens and Donoho wavelet half the size of the detail coefficient during the Lazy wavelet split, so the filter should also be multiplied by a half otherwise the prediction is incorrect. The predict filters at the boundary for $N = 2, 4, 6$ are given in Table 3.7.
Using the cascade algorithm 4 wavelets near the left boundary for the case $N = \tilde{N} = 4$ are generated and given in Figure 3.18. On the left side of this wavelet there is one boundary filter for the predict step and there are two boundary filters for the update step. As can be seen in the figure this leads to 4 altered wavelets at the lowest level $J1$. At the boundary, polynomials of order N are fitted, hence the wavelet is able to reconstruct polynomials up to order $N - 1$ near the boundary, which leads to zero detail coefficients at the boundary for polynomials up to order $N - 1$.

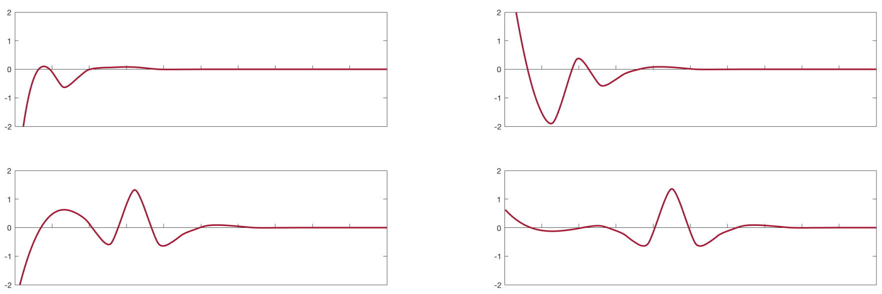


Figure 3.18: Left boundary for the interpolating boundary stencil with $N = \tilde{N} = 4$.

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 filter. The closer the wavelet or scaling function is to the boundary, the lower the order of this function. To illustrate this, the Sweldens wavelet of order $N = 4$ will be considered. In Figure 3.19 the predict step is schematically given. There are three different boundary points, namely $d_{j,0}$, $d_{j,2}$ and $d_{j,3}$. Consider boundary point $d_{j,0}$, it misses one smoothing coefficient $s_{j,-1}$. However, if the predict filter $P^{N=2}$ is used instead of $P^{N=4}$, then $d_{j,0}$ has all the smoothing neighbours it needs. Similarly, $d_{j,2}$ will be predicted with the $P^{N=2}$ filter. The boundary point $d_{j,3}$ does not have any right neighbours. Here, one can use the predict filter for $N = 1$, which is simply $\frac{1}{2}$ times the corresponding smoothing coefficient $s_{j,3}$.

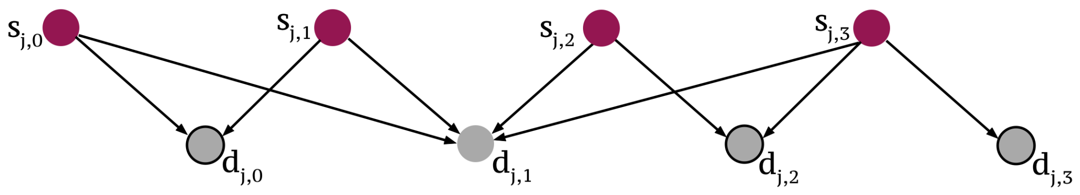


Figure 3.19: Predict step of order $N = 4$ using the lower order boundary stencil, all the detail coefficients with a black border are boundary points.

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. In figure Figure 3.20 the boundary wavelets of Sweldens with orders $N = \tilde{N} = 4$ are given.

Comparison of the boundaries

The downside of the lower order boundary stencil is the loss of precision near the boundary. However, if the interpolating boundary wavelets in Figure 3.18 are looked at closely, it can be seen that the tail towards the left boundary takes on extreme values, compared to the normal wavelet. This because the detail coefficients use smoothing coefficients further away than usual. If one of the smoothing coefficients is non zero than the smoothing coefficient relatively far away influences all the boundary wavelets. This can lead to higher detail coefficients near the boundary. Another way to look at this, is by inspecting the coefficients in Table 3.7. The boundary coefficients are around a factor 4 bigger than the internal wavelet coefficients. Hence, possible errors will be blown up. Both of the boundary stencil implementations will be tested and compared in Chapter 5.

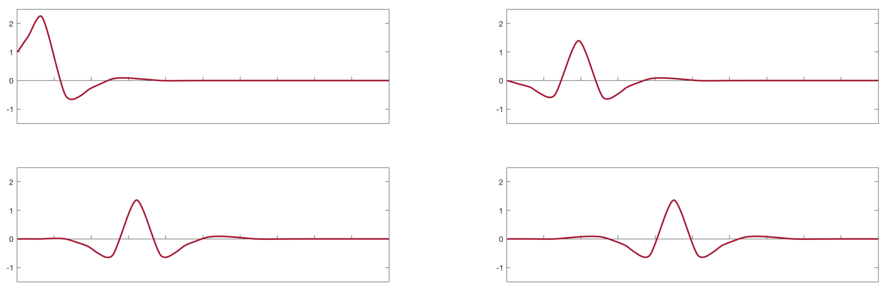


Figure 3.20: $N = \tilde{N} = 4$ lowerOrder.

3.6.7 In-place calculations

As mentioned earlier one of the main advantages of the lifting scheme compared to the classical fast transform is that in-place calculations can be performed. This will be explained in the 1d case. Classically, a forward transform step is:

$$\begin{cases} s_{j,k} = \sqrt{2}\sum_l \tilde{h}_l s_{j+1,2k+l}, \\ d_{j,m} = \sqrt{2}\sum_n \tilde{g}_n s_{j+1,2m+n}. \end{cases}$$

Only after the forward transform step has been performed s_{j+1} can be removed from memory. This means that the detail coefficients, which are being kept at every level need to be stored in separate arrays.
A forward transform step using the lifting scheme is:

$$\begin{cases} [s_j, d_j] = \text{split}(s_{j+1}) \\ d_j = d_j - P(s_j) \\ s_j = s_j - U(d_j). \end{cases}$$

After the split step, s_{j+1} is not needed any more, thus s_{j+1} can be overwritten with s_j and d_j. Hence, in the case of the lifting scheme the densest array can be kept, and certain values in the array can be altered during the forward transform. In Figure 3.21 a schematic forward transform is depicted from level $J2 = 3$ down to $J1 = 1$. After the forward transform both the smoothing coefficients at level $J1$ and all the detail coefficients in the intermediate layers are stored in one array. Detail coefficients at a particular layer can be retrieved by stepping through the array with step-size$= 2^{J2-j}$ starting from the offset$= \frac{1}{2}*$step-size. The smoothing coefficients at level $J1$ can be retrieved by stepping through the array with step-size $= 2^{J2-J1}$ starting from the offset $= 0$. The lifting scheme can store all the detail coefficients and the smoothing coefficients at level $J1$ in one array as it exploits the mathematical property described in 3.2:

$$V_{J2} = V_{J1} \oplus \cup_{j=J1}^{J2-1} W_j.$$

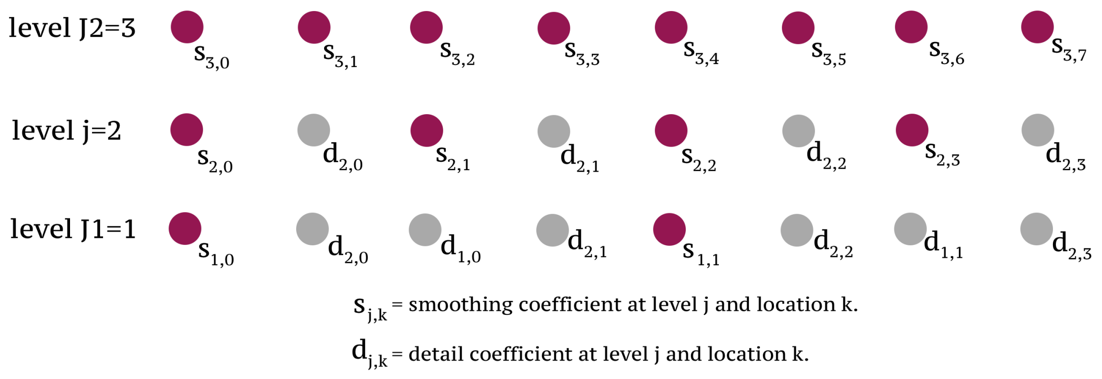


Figure 3.21: In-place 1D forward transform from $J2 = 3$ down to $J1 = 1$.

3.7 Multi-dimensional wavelets

In the literature there are 2 main manners in which multiple dimensions are applied to the wavelet setting. Firstly, triangulations are applied mainly to irregular meshes. More often, one chooses the tensor product due to the ease of implementation. For the problems which will be used in this thesis a regular mesh is adequate. Furthermore, exploring 2 and 3 dimensions is one of the goals of this thesis. Therefore, it makes sense to choose the tensor product as the implementation is straightforward.
If an n-dimensional function f is approximated using a wavelet transform, the tensor product simplifies down to applying the forward transform sequentially to dimension 1 up until dimension n. The inverse transform traverses the dimensions in a reversed order, starting at dimension n followed by $n-1$ down to dimension 1.
In this section first the 2D tensor product will be introduced in a mathematical setting. Thereafter, the implementation will be explained and in the last subsection higher dimensions will be discussed.

3.7.1 2D mathematically

The 2 dimensional setting will be introduced in the case of biorthogonal wavelets. In the 2D setting the nested family V_j can be written as a tensor product of the x- and y-direction. This tensor splitting leads to new detail spaces:

$$\begin{aligned}
V_j &= V_j^x \otimes V_j^y = (V_{j-1}^x \oplus W_{j-1}^x) \otimes (V_{j-1}^y \oplus W_{j-1}^y), \\
&= (V_{j-1}^x \otimes V_{j-1}^y) \oplus (W_{j-1}^x \otimes V_{j-1}^y) \oplus (V_{j-1}^x \otimes W_{j-1}^y) \oplus (W_{j-1}^x \otimes W_{j-1}^y), \\
&= V_{j-1} \oplus W_{j-1}^a \oplus W_{j-1}^b \oplus W_{j-1}^c.
\end{aligned}$$

Before, in the 1 dimensional setting there was one detail space, in 2 dimensions there are 3 different detail spaces, W_j^a, W_j^b and W_j^c. W_j^a is a tensor product of W_{j-1}^x and V_{j-1}^y, hence it is generated by $\psi(x)$ and $\varphi(y)$. The product of these functions will be denoted as ψ^a, in a similar manner the scaling and wavelet functions are defined as:

$$\begin{aligned}\varphi(x,y) &= \varphi(x)\varphi(y),\\ \psi^a(x,y) &= \psi(x)\varphi(y),\\ \psi^b(x,y) &= \varphi(x)\psi(y),\\ \psi^c(x,y) &= \psi(x)\psi(y).\end{aligned}$$

Remember, the orthogonal projection in the 1D setting is

$$f(x) \approx P_{V_{J2}}f(x) = P_{V_{J1}}f(x) + \sum_{j=J1}^{J2-1} P_{W_j}f(x),$$

where $J2$ is the densest level of the mesh and $J1$ the coarsest level. In the 2D setting there are more detail spaces, hence the orthogonal projection will look like:

$$f(x,y) \approx P_{V_{J2}}f(x,y) = P_{V_{J1}}f(x,y) + \sum_{j=J1}^{J2-1} P_{W_j^a}f(x,y) + \sum_{j=J1}^{J2-1} P_{W_j^b}f(x,y) + \sum_{j=J1}^{J2-1} P_{W_j^c}f(x,y),$$

where $P_{W_j^a}f(x,y) = \sum_{k,l\in\mathbb{Z}} d^a_{j,k,l}\psi^a(x,y)$ with $d^a_{j,k,l} = \langle f, \tilde{\psi}^a_{j,k,l}\rangle$.
In the 1 dimensional setting there was one type of detail coefficients. However, in the 2 dimensional setting there are three different types of detail coefficients, $d^a_{j,k,l}, d^b_{j,k,l}$ and $d^c_{j,k,l}$. Where $d^b_{j,k,l}$ corresponds to the inner product $\langle f, \tilde{\psi}^b_{j,k,l}\rangle$ and $d^c_{j,k,l}$ to $\langle f, \tilde{\psi}^c_{j,k,l}\rangle$. Hence, an approximation of f is:

$$\begin{aligned}f(x,y) \approx \sum_{k,l\in\mathbb{Z}} s_{j,k,l}\varphi(x,y) &+ \sum_{j=J1}^{J2-1}\sum_{k,l\in\mathbb{Z}} d^a_{j,k,l}\psi^a(x,y)\\ &+ \sum_{j=J1}^{J2-1}\sum_{k,l\in\mathbb{Z}} d^b_{j,k,l}\psi^b(x,y) + \sum_{j=J1}^{J2-1}\sum_{k,l\in\mathbb{Z}} d^c_{j,k,l}\psi^c(x,y).\end{aligned}$$

In order to determine the smoothing coefficients (s) and the detail coefficients (d^a, d^b, d^c) the forward transform can be performed. Consider the detail coefficient d^a, this is the inner product of f and the wavelet ψ^a.
$\psi^a = \psi(x)\varphi(y)$ is a wavelet in the x-direction, however it is a scaling function in the y-direction. Firstly, the forward transform will be applied in the first dimension, in this case the detail coefficient will be treated as a 1 dimensional wavelet. Secondly, the forward transform will be applied to the second dimension, in this case d^a will be treated as a smoothing coefficient. Similarly, d^b will be treated as a smoothing coefficient in the first dimension and as a detail coefficient in the second dimension. d^c will be treated twice as a detail coefficient.

In Figure 3.22 the positioning of the smoothing and detail coefficients are given for the 1D setting and the 2D setting. There are two grey boxes in both meshes, indicating the smoothing coefficient with its detail coefficient or coefficients. In the 1D setting the smoothing coefficient has 1 corresponding detail coefficient in the same level. In the 2D setting one smoothing coefficient has three detail coefficients, d^a in the x-direction, d^b in the y-direction and d^c in the x, y-direction. Therefore, thresholding on the detail coefficients will have a higher drop out rate in the 2D setting compared to the 1D setting. Because there are 3 times more detail coefficients than smoothing coefficients in the 2D case.

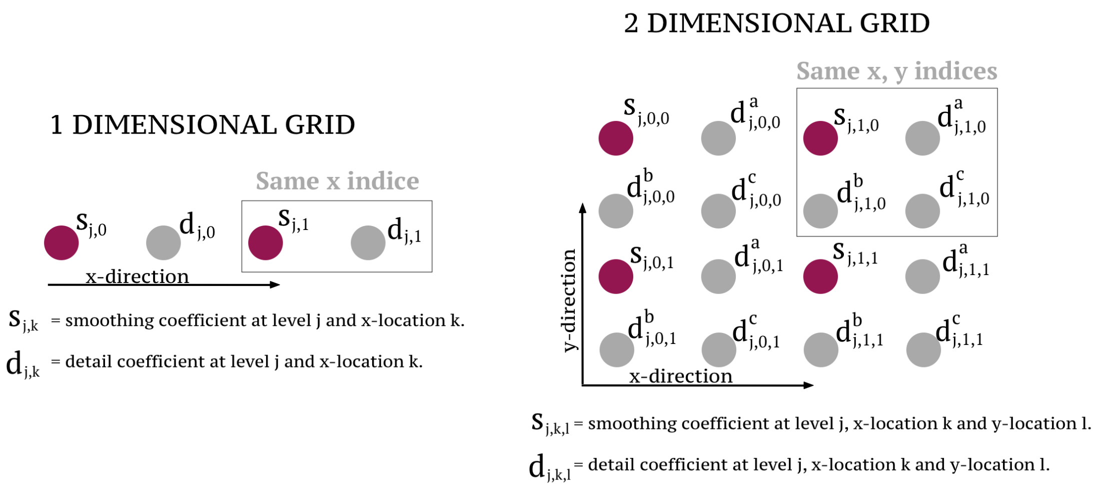
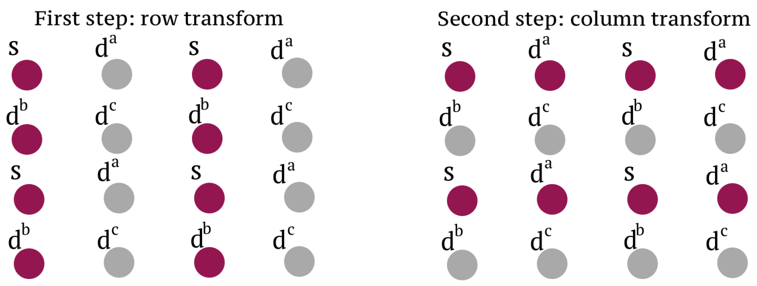

1 DIMENSIONAL GRID
Same x indice
$s_{j,0}$ $d_{j,0}$ $s_{j,1}$ $d_{j,1}$
x-direction
$s_{j,k}$ = smoothing coefficient at level j and x-location k.
$d_{j,k}$ = detail coefficient at level j and x-location k.
2 DIMENSIONAL GRID
Same x, y indices
$s_{j,0,0}$ $d^a_{j,0,0}$ $s_{j,1,0}$ $d^a_{j,1,0}$
$d^b_{j,0,0}$ $d^c_{j,0,0}$ $d^b_{j,1,0}$ $d^c_{j,1,0}$
$s_{j,0,1}$ $d^a_{j,0,1}$ $s_{j,1,1}$ $d^a_{j,1,1}$
$d^b_{j,0,1}$ $d^c_{j,0,1}$ $d^b_{j,1,1}$ $d^c_{j,1,1}$
y-direction
x-direction
$s_{j,k,l}$ = smoothing coefficient at level j, x-location k and y-location l.
$d_{j,k,l}$ = detail coefficient at level j, x-location k and y-location l.

Figure 3.22: Division of dyadic 2^{J2} mesh in 1D and 2D setting.

3.7.2 2D implementation

The derivation of the fast transform as in section 3.3 is straightforward but cumbersome. Instead, the fast transform will only be considered using the lifting scheme. In this setting the scaling and wavelet functions, $\varphi(x, y)$, $\psi^a(x, y)$, $\psi^b(x, y)$ and $\psi^c(x, y)$, will not be calculated directly.
Consider the 2D mesh in Figure 3.22 as a matrix. The general idea is to start applying the lifting forward transform to each row, after which the lifting forward transform can be applied to each column. The inverse transform starts of with applying the inverse transform to each column and finishes with every row. In Figure 3.23 the row and column transforms are depicted. In the first step both the coefficients s and d^b will be handled as 1D smoothing coefficients, while the d^a and d^c are handled as 1D detail coefficients. In the second step the columns are subject to a forward transform step. The coefficients s and d^a will be handled as 1D smoothing coefficients and the d^b and d^c are handled as 1D detail coefficients. Note, that the detail coefficient d^c is handled twice as a detail coefficient, both in the x- and y-direction.

Figure 3.23: The 2D forward transform.



The points handled as smoothing coefficients are red and the points handled as detail coefficients are grey.

Implementing the two steps in Matlab is fairly easy. The smoothing coefficients of the densest level can be stored in a matrix. One forward transform step from j to $j-1$ can then be applied to every row. Thereafter, one forward transform step can be applied to every column.
Due to the in-place properties of the lifting scheme, the next forward transform step can be applied again on this matrix using a different step-size. The forward transform steps can be applied until level $J1$ is reached. In Algorithm 9 the forward transform implementation is given in functions.

Algorithm 9 Lifting scheme: 2D forward transform

```
Safe smoothing coefficients of level $J2$ in the matrix $S$.
for j=J2  downto  J1 do
    S=forward_step_perRow(S)
    S=transpose(S)
    S=forward_step_perRow(S)
    S=transpose(S)
    [S, Da, Db, Dc]=split(S)
end for
```

3.7.3 More dimensions

Following the implementation as described above leads to a general extension in multiple dimensions. In the 2D setting, one forward step is first applied in the 1st dimension (x-direction) followed by a forward step in the 2nd dimension (y-direction). Extension in more dimensions is done by applying the forward step to the other dimensions as well. Note, that the inverse lifting transform always performs all steps in the opposite order. The manner in which the smoothing coefficients and detail coefficients are saved should be considered carefully. One can consider to threshold the detail coefficients already during the forward transform to reduce the storage.

4 Adaptive mesh refinement

In [30], C. Bowman and O. Vasilyev introduce a mesh adaption algorithm for wavelets. The mesh refinement and adaptive mesh refinement algorithms used will be based on this algorithm. However, as will be discussed later on it will differ in the way we threshold, add neighbours, and calculate the inverse transform. The changes are inspired after testing adaptive mesh refinement on different non smooth test functions.
Naturally, wavelets are defined in a dyadic setting, hence a dyadic mesh is needed this will be further explained in subsection 4.1.1. In the case of second generation wavelets the mesh is not restricted to be dyadic, however in this thesis a dyadic mesh is chosen. This because we can use the same wavelets on all the internal points of the meshes, which will lead to more stable behaviour, which is explained in [15]. Furthermore, mesh refinement will only be applied to square meshes, i.e., the same number of points in all dimensions. Within one dimension all the mesh points are equidistant, but between dimensions the distances can differ. Extending the mesh refinement to non square meshes is not difficult, however due to time constraints it is not considered in this thesis. In the first section 4.1 mesh refinement will be introduced. This will be extended in 4.2 to adaptive mesh refinement.

4.1 Mesh refinement

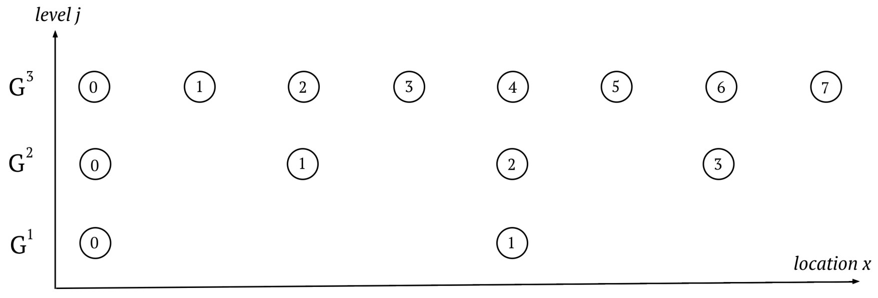
4.1.1 Mesh setup

Recall, in definition 1 the multiresolution analysis of the wavelet systems was defined. The wavelet system has a set of nested spaces $V_{J1} \subseteq V_{J1+1} \subseteq \cdots \subseteq V_{J2}$, where $J1$ corresponds to the coarsest level and $J2$ to the densest level. If we relate this to meshes V_{J1} corresponds to the coarsest mesh and V_{J2} to the finest mesh considered. In definition 1 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.
In Figure 4.1 a one dimensional example is depicted. There are three different meshes G^1, G^2 and G^3, each corresponding to a space V_j. The densest level in this case is 3, corresponding to mesh G^3, this mesh exists out of $2^3 = 8$ points. One level below, $j = 2$ so grid G^2, has $2^2 = 4$ points, essentially halving the number of points in mesh G^3.
In general, one fixes two levels of approximation, namely $J2$ and $J1$ corresponding respectively to the densest and coarsest meshes. This leads to $J2 - J1$ evenly spaced meshes G^j, where G^j exists out of 2^j points for $J1 \leq j \leq J2$, i.e.

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

The mesh is dyadic, hence a 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.

Figure 4.1: 1D dyadic mesh.



The location indices are depicted inside the mesh points.

Extending this definition to multiple dimensions is simply done by the tensor product, e.g., in the 2 dimensional 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.

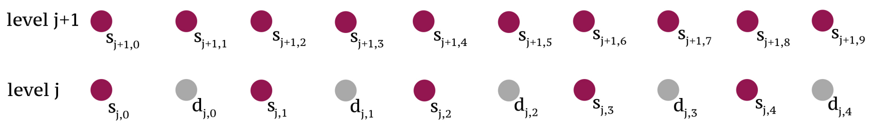
4.1.2 Smoothing and Detail coefficients

The representation of a function f by the second generation wavelets is calculated by the lifting scheme (a version of the fast transform). The lifting scheme starts with splitting the data set in smoothing coefficients and detail coefficients. In Figure 4.1 3 different levels of meshes are depicted. The points in the densest level G^3 are all smoothing points. Then this is split up, the even points will stay smoothing coefficients and the odd points will become detail coefficients. The smoothing coefficients after the split correspond to the points of mesh G^2. This splitting process is shown in Figure 4.2. After the splitting step, the update and predict step will be applied to the smoothing and detail coefficients. Then the remaining smoothing coefficients are split up again in smoothing coefficients and detail coefficients. This process will repeat until the coarsest level $J1$ is reached. After the forward transform one ends up with smoothing coefficients at the coarsest level and detail coefficients corresponding to every level j.

4.1.3 Thresholding

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

Figure 4.2: Splitting.



The location indices are depicted beneath points.

As mentioned before the value of detail coefficients indicates how well a mesh point can be estimated using the wavelet transform. If the absolute value of the detail coefficient is low, the function value can be easily determined using the neighbours. However, when the value is high, the function value is difficult to estimate and thus in that case it is better to keep the mesh point. The approximation of a function using the wavelet transform can be written as:

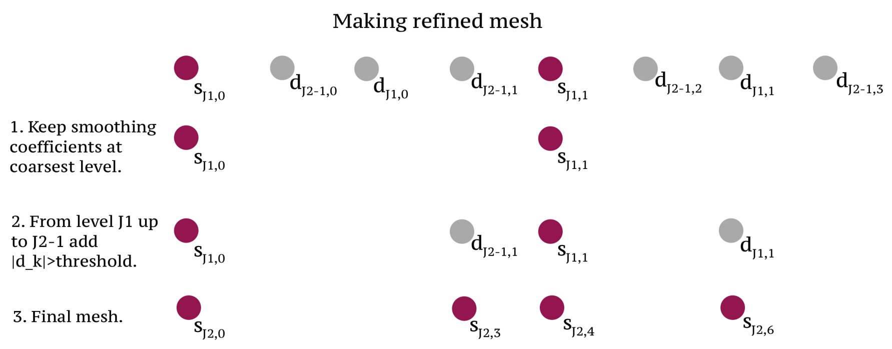
$$\begin{aligned} f(x) \approx P_{V_{J2}} f(x) &= \sum_k s_{J2,k} \varphi_{J2,k}(x) \\ &= P_{V_{J1}} f(x) + \sum_{j=J1}^{J2-1} P_{W_j} f(x) = \sum_k s_{J1,k} \varphi_{J1,k}(x) + \sum_{j=J1}^{J2-1} \sum_m d_{j,m} \psi_{j,m}(x). \end{aligned}$$

Every point in the dense mesh corresponds to a smoothing coefficient $s_{J2,k}$ in the densest level. In another representation, all the points correspond to either a smoothing coefficient $s_{J1,k}$ 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, therefore it can be dropped out of 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.
In Figure 4.3 the thresholding process is depicted. After the forward transform, the smoothing coefficients are kept. Then moving through the levels high detail coefficients are kept. After identifying which points to keep, all the points will be set to the function values at that location, i.e. they will function as smoothing coefficients of the densest level.

4.1.4 Perfect reconstruction

The points which have been added up until now are considered as important, i.e. all the points have a significant high detail coefficient. Because the points are significant, one wants to be able to fully construct the values of these points using the inverse 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 are dependent on the inverse transform, i.e., they are dependent on the used wavelet transform. Because Donoho's interpolating wavelet

Figure 4.3: Thresholding.



The location indices are depicted beneath points.

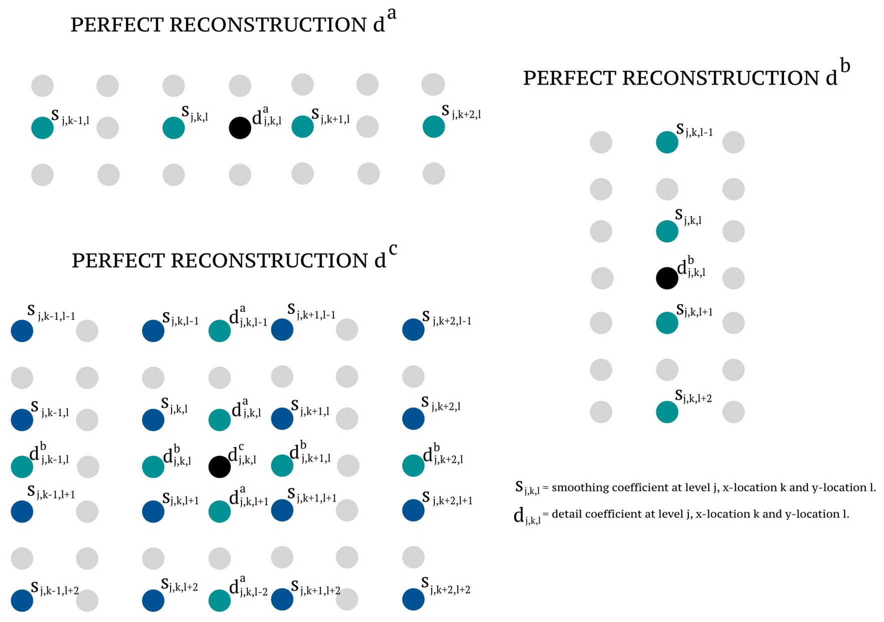
and Sweldens wavelet are considered in this thesis, the perfect reconstruction points for these wavelet transforms will be discussed. Because all the mesh points should be reconstructed perfectly, one should note that the perfect reconstruction check should be repeated until no new points are added.
In the one dimensional setting the added points are always detail coefficients generated by the wavelet ψ, therefore the 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 significantly low so it does not affect the value of the smoothing coefficient significantly. If also these detail coefficients are added the mesh will end up to be full. Which smoothing coefficients need to be added for the perfect reconstruction of a detail coefficient depends on the order of the wavelet used. Because the detail coefficient is constructed using the predict filter, both Sweldens and Donoho have the same neighbouring smoothing coefficients which need to be added for each detail coefficient. Namely, for $d_{j,k}$ the points $s_{j,k+n}$ need to be added, where $-N/2+1 \leq n \leq N/2$.

2 dimensional setting

In the 2 dimensional setting, the detail coefficients d^a and d^b are both depending on the scaling function φ and on the wavelet ψ. Furthermore, for the detail coefficients d^c the points which act as smoothing coefficients are the detail coefficients d^a and d^b. For the implementation, first the smoothing neighbours of the detail coefficients d^c need to be added. Thereafter, the perfect reconstruction can be applied to the detail coefficients d^a and d^b. The process needs to be repeated until no new points are added, as new d^a and d^b coefficients will be added during the perfect reconstruction process of d^c. These new d^a and d^b coefficients need to be perfectly reconstructed themselves.
For the reconstruction of $d^c_{j,k,l}$ the neighbours $d^a_{j,k,l+n}$ and $d^b_{j,k+n,l}$ need to be added, where

$-N/2+1 \leq n \leq N/2$. Similarly, for the reconstruction of $d^a_{j,k,l}$ the neighbours $s_{j,k+n,l}$ need to be added and for $d^b_{j,k,l}$ the neighbours $s_{j,k,l+n}$. For both d^a and d^b neighbours are only added in one direction, as discussed above only the smoothing coefficients needed for reconstruction of the detail coefficients need to be added, i.e., no extra detail coefficients need to be added in this case. In Figure 4.4 per type of detail coefficient the process of adding the perfect reconstruction points is depicted. In the figure it is assumed that Sweldens and Donoho wavelets have order $N = 4$. Note, that the perfect reconstruction of d^c is done in two steps. First, d^a and d^b will be added after which the s corresponding to these d^a and d^b will be added.

Figure 4.4: Perfect reconstruction for 2 dimensional mesh, $N = 4$.



The black dots are detail coefficients which have been thresholded. The green dots are points which have to be added for the perfect reconstruction of the black dot. The perfect reconstruction of d^c has blue dots which result from the perfect reconstruction of the detail points added, due to the perfect reconstruction of d^c.

4.1.5 The algorithm

The complete mesh refinement is as follows. Consider a function f on a closed domain Ω, density levels $J2$ and $J1$ and a set of layered meshes G^j. These layered meshes look similar as the meshes in Figure 4.1. Set $s_{J2,k} = f(x_{J2,k})$, for $0 \leq k \leq 2^{J2} - 1$. Then the forward transform can be performed up until the level $J1$. After the forward transform a set of smoothing coefficients on the coarsest level and a set of detail coefficients are retrieved. Thresholding will be applied to the detail coefficients, if the detail coefficients

are above a certain threshold the corresponding mesh point will be added to the mesh. Next, points will be added to be able to perfectly reconstruct all the points in the mesh. In Algorithm 10 the mesh generation is given. The M resulting from this algorithm is an indication of which points should be in the mesh. This computational mesh M includes all the smoothing coefficients of the coarsest level $J1$, all significant detail coefficients in the levels between $J1$ and $J2$ and all the points needed to perfectly reconstruct the mesh points. The algorithm is written for the 1 dimensional case, however extension to more dimensions is easily done, by simple checking more detail coefficients.
The error of the mesh refinement can be checked by performing an inverse transform on the function restricted to the mesh.

Algorithm 10 1D mesh refinement

```
Set $G^j = \{x_{j,k} : 0 \leq k \leq 2^j - 1\}$ for all $J1 \leq j \leq J2$.
$M^{J1} = G^{J1}$.
Perform the forward transform on $s_{J2} = f(G^{J2})$.
for j=J1 upto J2 do
    for k=0 upto $2^j - 1$ do
        if $|d_{j,k}| \geq \varepsilon^{\text{Threshold}}$ then add $x_{j+1,2k+1}$ to $M^j$.
        end if
    end for
end for
Add perfect reconstruction points to $M^j$.
$M = \cup_{j=J1}^{J2} M^j$.
```

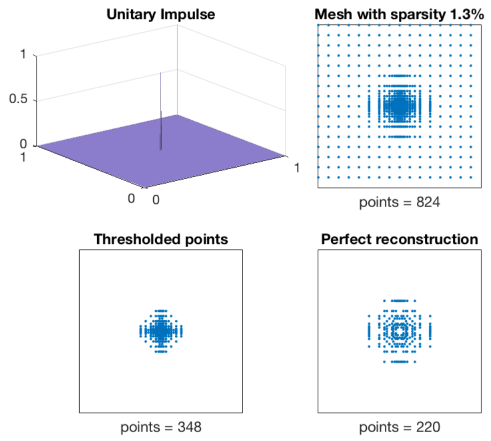
4.1.6 2D examples

Two simple test functions will be introduced, to show some mesh refinement examples. The wavelets which will be researched in this thesis are Donoho's interpolating wavelet and Sweldens wavelet. Both the interpolating and the lower order boundary stencil implementations are tested. These two different boundary types are described in section 3.6.6. The parameters used to generate the following examples are: $J2 = 8$, $J1 = 4$, $\varepsilon^{\text{Threshold}} = 0.001$, $N = 4$, $\tilde{N} = 4$. Hence, the finest mesh considered will have $p = 2^8 \times 2^8$ points, so for sparsity the following formula is used: $100 \times \frac{\text{Number of Points}}{p}$. The wavelets will be denoted as Don($N, \tilde{N}$,bounary) and Swel($N, \tilde{N}$,bounary), where boundary stencil is denoted as int or low corresponding respectively to the interpolating and lower order boundary stencil.

Unitary impulse

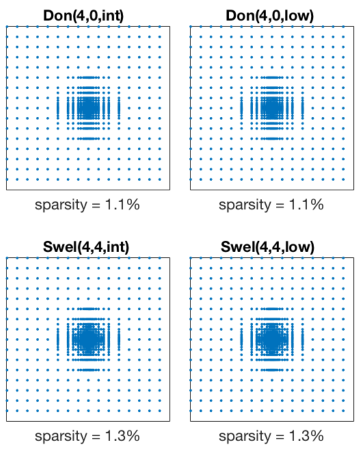
The unitary impulse is a function which is equal to zero everywhere except for 1 point, located in the middle of the domain. In Figure 4.5 the function is shown. Mesh refinement can be split in 3 steps, adding the smoothing coefficients of the coarsest level,

Figure 4.5: Mesh refinement on the unitary impulse.



thresholding detail coefficients and adding points for perfect reconstruction. The last 2 steps are inspected for Sweldens wavelet with the interpolating boundary stencil. In Figure 4.5 the thresholded and the perfect reconstruction points are shown.
Mesh refinement on the unitary impulse is tested on 4 different wavelets. Donoho's interpolating wavelet and Sweldens wavelet, both with the lower order and the interpolating boundary stencil implementations. The primal order of the wavelets tested is $N = 4$. The results of the meshes can be seen in Figure 4.6. The mesh refinement algorithm results in a very sparse grid, because the non smoothness of this function is very local. The unitary impulse has local behaviour far away from the boundary, this results in exactly the same meshes for the two different boundary stencil types. The performance of the two wavelets is very similar, both resulted in meshes with a sparsity of around 1%.

Figure 4.6: Meshes from different wavelets.



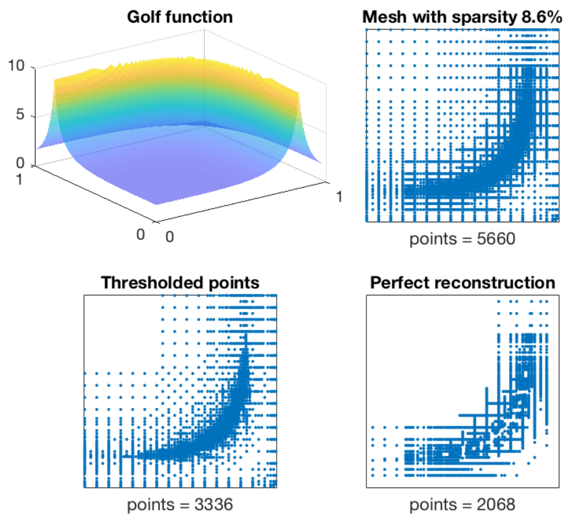
Golf function

The golf function is equal to

$$f(x, y) = \frac{1}{|0.5 - x^4 - y^4| + 0.1}.$$

The behaviour of this test function is less local, and close to the boundary. In Figure 4.7 the golf function is shown. Furthermore, the different steps of mesh refinement using Donoho's wavelet can be seen.

Figure 4.7: Mesh refinement on the golf function.



In Figure 4.8 meshes of both the interpolating and lower order implementations are given. The interpolating boundary stencil is adding more points near the boundary, thus is keeping more points.

Relative error versus sparsity

The sparsity of a mesh is important, however the accuracy of the mesh is also of great significance. In Figure 4.9 the error versus the sparsity is depicted. This graph was constructed by performing mesh refinement for different threshold values, namely 1×10^{-1}, 1×10^{-2}, 1×10^{-3}, 1×10^{-4}, 1×10^{-5}, 1×10^{-6}, 1×10^{-7}, 1×10^{-8}. The error was measured by performing an inverse transform on the function restricted to the mesh. After the inverse transform the result was compared to the actual function. The error measured is the relative Frobenius error. The calculation is done by the formula:

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

Figure 4.8: Meshes from different wavelets

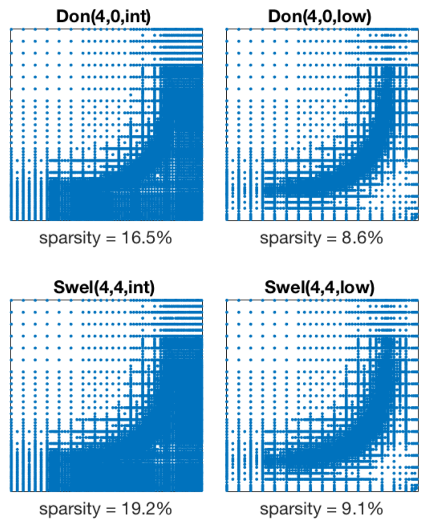
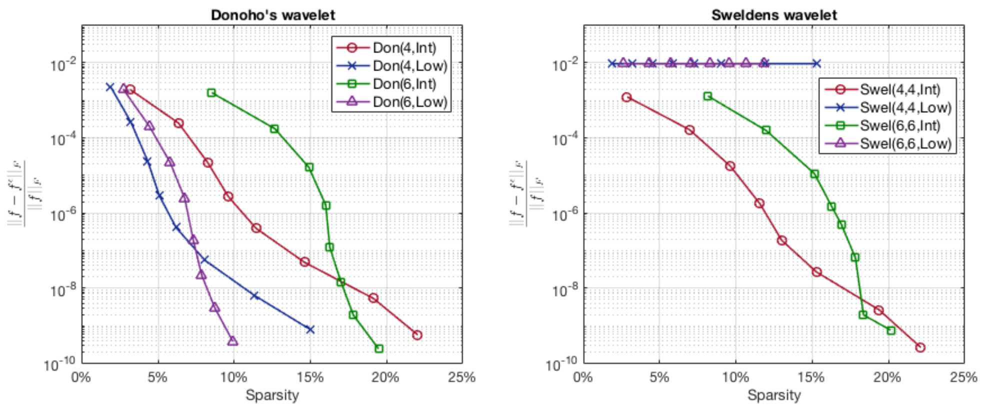


Figure 4.9: Sparsity versus relative error.



Where f^{ε} denotes the function values estimated by the inverse transform. The test function is the golf function The levels of accuracy used are $J2 = 8$ and $J1 = 4$. It is interesting to note that the lower order boundary stencil implementation is not able to reduce the error in the Sweldens case, while it gets a high accuracy with high sparsity in the Donoho case. The interpolating boundary stencil has similar performance for both the Donoho and Sweldens case.

4.2 Adaptive Mesh refinement

The wavelets perform good on mesh refinement, except for Sweldens wavelet with the lower order boundary stencil implementation. However, mesh refinement becomes more

delicate when it is done through time, adaptive mesh refinement. A PDE is solved over a certain time span, $[0, T]$. Mesh refinement will be applied in the first time step, on this refined mesh the PDE will be solved for one or more time steps. After which, the refined mesh will be refined again for the coming time step(s). In this new refining step, the wavelets should be able to both add and remove points. In order to do mesh refinement the forward transform should be adapted, this because there are no values for some coefficients. This special forward transform is called the adaptive forward transform. Performing the inverse transform will give the function f on the whole domain, for the inverse transform one can also use an adapted form, however this is not needed.
In order to be able to adapt through time adjacent points are added. If a point is thresholded it is considered as significant. Therefore, in the coming time step(s) its neighbouring points can become significant. Hence, one new aspect in adaptive mesh refinement will be adding the immediate neighbours of the thresholded points.

4.2.1 Adaptive forward transform

Normally, the forward transform is applied to all the points in the mesh. However, not all the values of the smoothing coefficients are known, because the mesh is sparse. Because of the special reconstruction check, all the smoothing coefficients needed to calculate the detail coefficients of the mesh points, are in the mesh. It is important that 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 the Sweldens wavelet is used, or another wavelet with an update step, then the smoothing coefficients are updated using detail coefficients which might not be in the mesh. Firstly, 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 because in the thresholding phase, only detail coefficients almost equal to zero were removed. Hence, the original value of the detail coefficient is considered insignificant for the update step.

4.2.2 Adaptive inverse transform

During the testing phase, it was noticed that the Sweldens wavelet had a larger error than the Donoho wavelet in the 2 dimensional test sets. In order to find out where the error was build up, the points with the highest errors were checked. This was done by printing out its value during the forward and inverse transform. In the 2 dimensional setting points were treated in two directions. It was noticed that in the last update step the update was different from the corresponding update step in the forward transform. In the inverse transform one wants to reform the original function f, 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 forward transform used zeros for the update step, if detail coefficients were used which were not in the mesh. In the inverse transform all the coefficients will receive a value, hence the update step will be different. Therefore, in the adapted setting one can choose to change the inverse transform. So during the update step it is checked if a detail coefficient used was in the mesh which was used for the forward transform. If not this detail coefficient will contribute the value of zero to the update step instead of its own value. The mesh used to restrict the inverse transform is not the last generated mesh, but the mesh of the time step before, the one which one was used in the forward transform. Hence, for the adaptive inverse transform one extra binary sparse matrix should be stored in memory. During testing the adaptive inverse transform will be compared to the normal inverse transform.

4.2.3 Adjacent points

If a mesh point $x_{j,k}$ is added to M^j for $j > J1$, then the absolute value of the detail coefficient at that level is significantly big. The mesh generated will be used to solve a partial differential equation for one or more time steps. It makes sense that the computational mesh should consist of those mesh points associated with detail coefficients which *are or can possibly become significant during the period of time when the grid 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 in this thesis. Type 1 refers to adding neighbours only in the direction the coefficient functions as a detail coefficient. For example, if a point in a 2 dimensional mesh is a detail coefficient in direction 1 and a smoothing coefficient in direction 2, then only in direction 1 neighbours will be added. Commonly, points which have been thresholded are detail coefficients in both directions but in different layers j, then neighbours in the corresponding direction will be added in that layer. 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 layer j are added. In both types of neighbouring points, one can also add points in the neighbouring layers. For example, type 1 neighbours can be added in the layer below, the same layer and the layer above.

Type 1

In the one dimensional 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 layers in which adjacent points are added, and $L^{\text{neighbour}}$ are the number of adjacent points added per layer 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 layer below, the same layer and the layer above.
In [30] it is suggested that the optimal values of the adjacent zone are $L^{\text{level}} = L^{\text{neighbour}} = 1$. During the testing phase these parameters will be changed, to investigate if $L^{\text{level}} = L^{\text{neighbour}} = 1$ is indeed optimal. The higher the adjacent criteria values are the more

Algorithm 11 Adaptive mesh refinement

```
Perform the adapted forward transform on $s_{J2} = f(G^{J2}|_{M_{t-1}})$.
Add the points corresponding to $s_{J1}$ to $M_t$.
Add thresholded points to $M_t$.
Add adjacent points to $M_t$.
Add perfect reconstruction points to $M_t$.
```

mesh points will be in the mesh M. However, if the adjacent criteria values are too low solving the partial differential equation with the computational mesh M for t time-steps can give inaccurate results. So for optimality conditions the trade-off between these properties should be considered.
In the 2 dimensional setting, the one dimensional approach is used. If a coefficient is a detail coefficient in two directions and different levels this means neighbours will be added twice, in the corresponding layers. This approach is similar in a 3 dimensional case.

Type 2

Another way to add points, is adding all the points within a certain distance of the mesh point $x_{j,k}$. This does not give a difference in the 1 dimensional setting, but it does make a difference in higher dimensions. For example, in the 2 dimensional case, if point $x_{j,k,l}$ is a detail coefficient in layer j in direction 1, type 1 points will add points in the same layer with the coordinates $(j \pm L^{\text{level}}, k \pm L^{\text{neighbour}}, l)$. In the type 2 setting the points with the following coordinates will be added $(j \pm L^{\text{level}}, k \pm L^{\text{neighbour}}, l \pm L^{\text{neighbour}})$. If then the same mesh point is a detail coefficient in the second direction in layer J, again this type of boundaries will be added, $(J \pm L^{\text{level}}, p \pm L^{\text{neighbour}}, m \pm L^{\text{neighbour}})$.

Algorithm 12 AMR for timedependent PDE

```
t=0.
Perform the forward transform on $s_{J2} = f(G^{J2})$.
Perform Mesh refinement to retrieve $M$.
Solve time step(s) on mesh $M$.
**for** t=1  upto  T **do**
Perform the adapted forward transform on $s_{J2} = f(G^{J2}|_M)$.
Perform Mesh refinement to retrieve $M$.
Solve time step(s) on mesh $M$.
**end for**
Perform the (adapted) inverse transform on $f_M$.
```

4.2.4 Adaptive thresholding

The thresholding of points is strongly connected to the adjacent points, namely, only for the thresholded points adjacent points are added. This in a sense broadens the way in which thresholding can be done. For example, if a detail coefficient is very high it can be considered as very 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 and for all the thresholded points, adjacent points of type 1 will be added. With the parameters $L^{\text{level}} = L^{\text{neighbour}} = 1$.

Version 2

In version 2 two values for thresholding are used, again type 1 adjacent points is considered. The motivation behind two layers of thresholding, is that if a certain detail coefficient has a very high absolute value, then more neighbours need to be added. The parameters for the first threshold are $\varepsilon_1^{\text{threshold}} = 1$ and $L_1^{\text{level}} = L_1^{\text{neighbour}} = 2$. So if a detail value is high more neighbours will be added in more layers. The second threshold value $\varepsilon_2^{\text{threshold}}$ can be chosen arbitrary with $L_2^{\text{level}} = L_2^{\text{neighbour}} = 1$. During the testing phase, the results of version 1 and version 2 where very the similar. Therefore, version 2 will not be mentioned in the results, however the data can be found in the tables in the appendix.

Version 3

Some of the financial PDEs which will be considered have a very high peak which will spread out quickly. In this case a more extreme adding of neighbours might be considered. Setting the threshold value dependable on the test function can give an indication of the relative difficulty to predict the value. In this sense, the first threshold is defined as $\varepsilon_1^{\text{threshold}} = \frac{|\max(f)-min(f)|}{4}$, with type 2 adjacent points. So in 2D a square of points will be added around the thresholded detail coefficient. In the 3D setting this will be a cube of neighbours. The second threshold will be free to choose and the other parameters will be: $L_1^{\text{neighbour}} = 5$ and $L_1^{\text{level}} = L_2^{\text{level}} = L_2^{\text{neighbour}} = 1$.

4.2.5 The algorithm

Summarising, in adaptive mesh refinement the algorithm to determine the mesh is expanded with adjacent points. In Algorithm 11 the new mesh refinement is given. Furthermore, in the first time step the normal forward transformed is used, but after that a special version of the forward transform should be used. In Algorithm 12 the application of adaptive mesh refinement on a time dependent PDE is given.

4.2.6 2D examples

The unitary impulse and golf function will be used to show the effect of adding adjacent points in the mesh refinement. Version 1 of the AMR will be tested. In Figure 4.10 the mesh refinement using Sweldens wavelet with the interpolating boundary stencil can be seen. The mesh is split up in the thresholded points, adjacent points and perfect reconstruction points. It can be seen that more points have been added, and the area covered by the mesh has expanded. Similar results can be seen in Figure 4.12. Note that the amount of perfect reconstruction is not increased. This because some points added in the adjacent step, would otherwise have been added in the perfect reconstruction. Finally, in Figure 4.11 and Figure 4.13 the resulting AMR meshes for both Sweldens and Donoho's wavelet are given.

Figure 4.10: Adaptive mesh refinement on the unitary impulse.

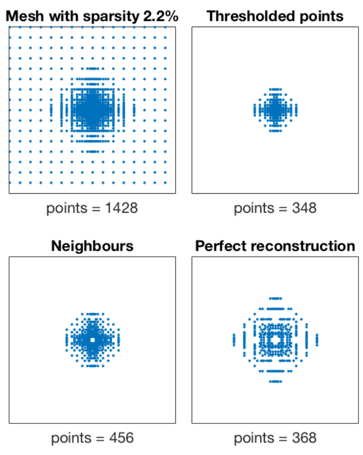


Figure 4.11: Meshes from different wavelets

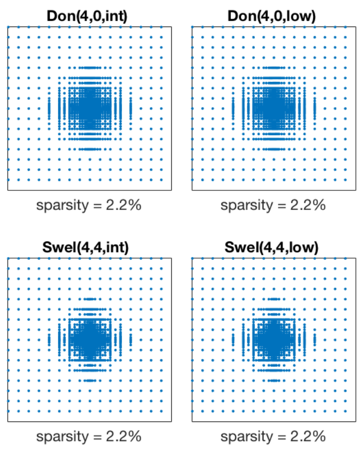


Figure 4.12: Adaptive mesh refinement on the golf function.

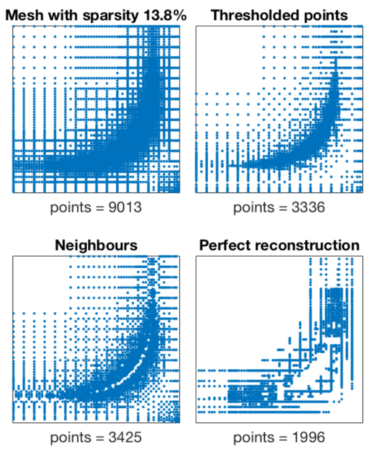
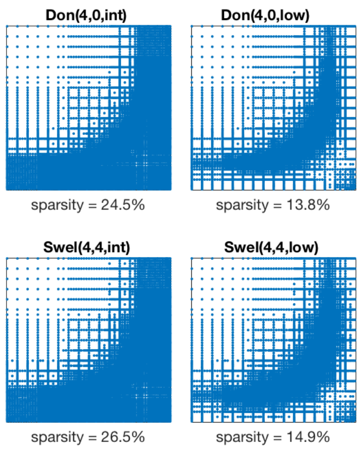


Figure 4.13: Meshes from different wavelets



5 Comparison of wavelets

5.1 Problem setup

NAG (Numerical Algorithms Group) has provided data sets to evaluate the performance of the wavelets. The PDEs which are tested are Finance PDEs in 1 and 2 dimensions provided by Jacques Du Toit and a Hydro PDE in 2 and 3 dimensions provided by Kevin Olson. The PDEs are approximated through time, every time step the intermediate approximation is stored. Therefore, the data on which the wavelets will be tested are approximations of PDEs at every time step. This gives the ability to evaluate the performance of the wavelets at every time step. Only a part of the tested data is evaluated in this chapter, for more results the Appendix can be consulted.

Two types of wavelets will be extensively tested, Donoho's interpolating wavelet and Sweldens wavelet. Their performance will be investigated in 1, 2 and 3 dimensions. Both wavelets will be tested with two different boundary stencil implementations, lower order and interpolating boundaries. These two different boundary types are described in section 3.6.6. Furthermore, two orders will be checked, $N = 4$ and $N = 6$. In order to indicate which wavelet is tested the following notation will be used: Wavelet(primal order N, dual order $\tilde{N}$, boundary). Where *Don* indicates Donoho's interpolating wavelet, *Swel* Sweldens wavelet, *int* the interpolating boundary stencil and *low* the lower order boundary stencil. Two examples are Don(6,0,int) and Swel(4,4,low).

5.1.1 Benchmarks

Kevin Olson is experienced in AMR, therefore the standards he and his colleagues use will be considered. A mesh is evaluated on its sparsity and the accuracy with which it can approximate a function. AMR is only beneficial if the meshes are sparse enough. Typically, 2 dimensional AMR is used if the sparsity is below 40% and below 30% for 3 dimensional AMR. There are no standards in accuracy, because as this depends on the accuracy of the numerical solver. During testing 3 different accuracies were considered, 0.1%, 1% and 10% relative Frobenius error. This relative error is calculated as follows. Define f as the approximation retrieved from the data set and f^{ε} as the approximation generated by the wavelets. In order to calculate the relative Frobenius error, the Frobenius norm on $f - f^{\varepsilon}$ and f are calculated. Then the relative error is $\frac{||f-f^{\varepsilon}||_F}{||f||_F}$. The Frobenius norm for a 2 dimensional matrix A is $||A||_F = \sqrt{\sum_i \sum_j a_{ij}^2}$.

5.1.2 AMR in short

The provided data sets are approximations of the PDE at every time step. The AMR starts with applying a forward transform on the initial time step. Subsequently, mesh refinement results in a mesh M. After the mesh refinement the initial solution is restricted to the mesh M and an inverse transform is applied. The inverse transform will be compared to the initial solution and the relative error is calculated.
Next, the approximation at the subsequent time step is restricted to M. On this restricted version of the data, mesh refinement will be applied, resulting in a new mesh M. Thereafter the relative error will be calculated by performing an inverse transform. These last steps proceed until the last time step is reached.

5.1.3 Financial PDEs

Below the three different types of PDEs are shortly described, for more information https://www.nag.co.uk/doc/techrep/pdf/tr3_16.pdf can be consulted.

SLV

SLV is the probability density function for the 2D Stochastic Local Volatility model with Heston volatility dynamics. The SLV model is widely used in finance, especially in foreign exchange (FX) markets. The model has to be calibrated to market data, and a key stage in this calibration is solving for the probability density function of the process. The 1 dimensional version is called LV.

Price

Call is the price of a call option driven by SLV. The payoff of a call options is $\max(S_T - K, 0)$ where S_T is the underlying asset at time T and K is the strike price. This data set is called price.

Digital

Digital is the price of a digital option driven by SLV. The payoff of a digital option is 1 if $S_T > K$, 0 otherwise, where S_T and K are the underlying asset at time T and the strike price.

5.1.4 Hydro PDE

The PDE computes pressure, which depends on the density, energy, x velocity, y velocity and z velocity. The code which delivered the approximations, solves the equations of compressible hydrodynamics assuming an ideal gas equation of state using a finite volume technique. 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 [10].

The code applies AMR only on the pressure and density. In the wavelet setting this would entail generating meshes for both approximations and using the union of these meshes to solve the next time step(s). Therefore, these two data sets will be considered in the results. Because the other variables provide more surfaces to test the wavelets on, they have also been investigated. The results of this can be found in the appendix.

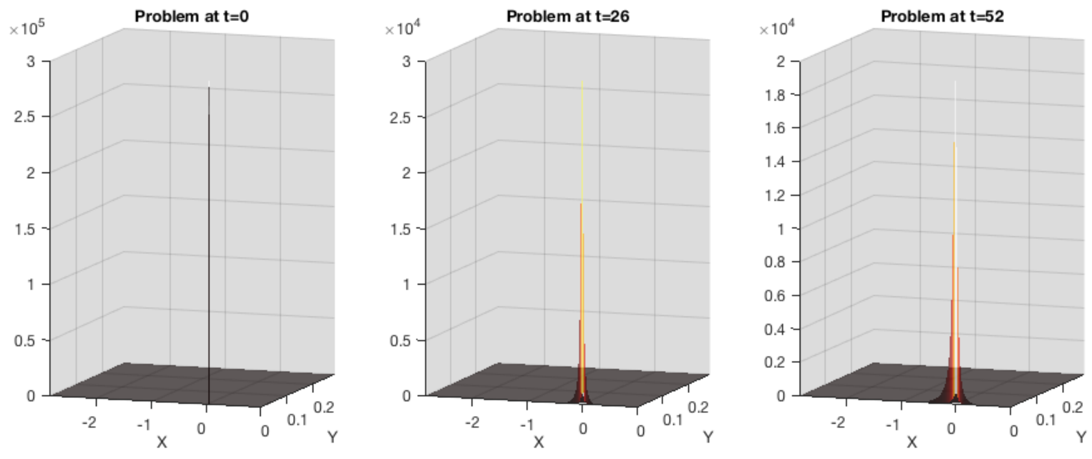
5.2 Finance PDEs

All the 1D data is tested with $J2 = 11$ as the densest level and $J1 = 5$ as the coarsest level. In the 2D tests, $J2 = 8$ was assumed to be the finest level of refinement and $J1 = 4$ the coarsest level.

5.2.1 2D SLV 17

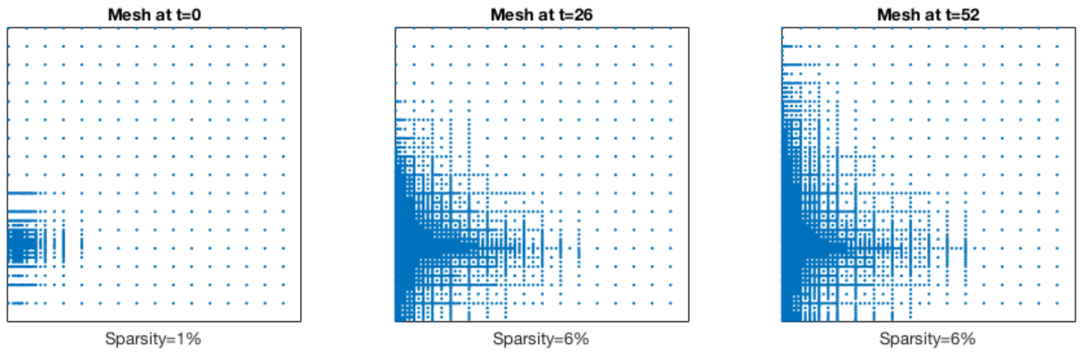
There are three versions of the 2 dimensional SLV method which have been tested, SLV 5, SLV 17 and SLV 53. The difference is caused by different initial solutions. SLV 17 will be considered in the results. In Figure 5.1 SLV 17 is depicted at time step 0 (the initial solution), time step 26 and the final time step 52. The PDE starts off with a high peak on one point, the rest of the approximation is zero. This peak is located near the boundary, this makes the data set interesting as non smoothness near the boundary is more difficult for the wavelets. The high peak spreads out during time.

Figure 5.1: SLV 17 through time.



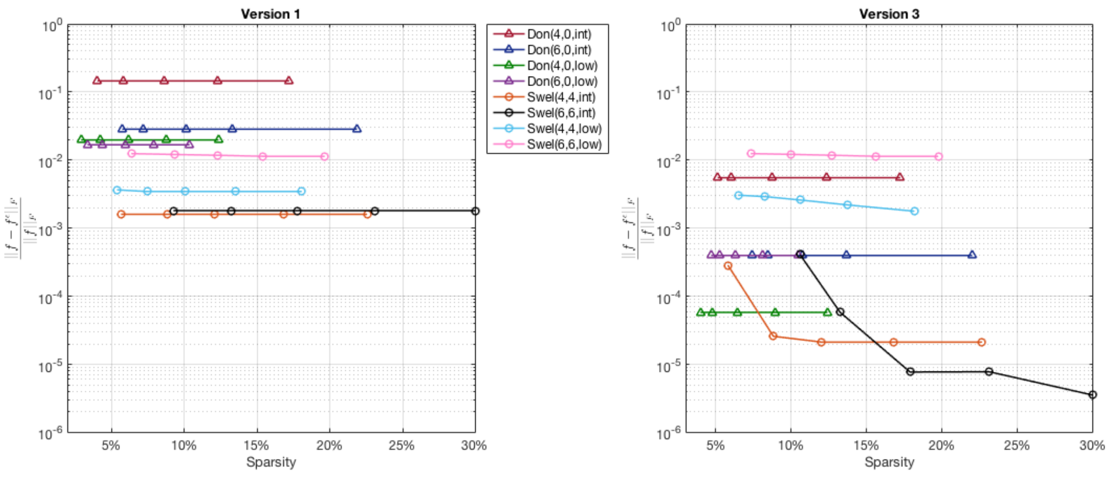
In Figure 5.2 the three different meshes corresponding to time steps 0, 26 and 52 are shown. The wavelet used in this AMR is Donoho's interpolating wavelet with the lower order boundary stencil, primal order $N = 4$ and dual order $\tilde{N} = 4$. Version 1 of the AMR is used, this means that the there is one threshold value, namely 0.001, and neighbours are added with the values $L^{\text{level}} = L^{\text{neighbour}} = 1$. Version 1 of the AMR is explained in more detail in 4.2.4. The figure shows that the AMR is capable of growing its mesh if the non smooth area grows in time.

Figure 5.2: Meshes of SLV 17 using Don(4,0,low).



In order to visualise the performance of all the different wavelets, 5 tests have been run. Every wavelet is tested on the data set 5 times per version by varying the threshold value, $\varepsilon^{\text{threshold}} = [1 \times 10^{-1}, 1 \times 10^{-2}, 1 \times 10^{-3}, 1 \times 10^{-4}, 1 \times 10^{-5}]$. After each test the maximum number of points and the maximum error are kept. This results in the plots of Figure 5.3. Version 1 and version 3 of the AMR are evaluated.

Figure 5.3: Comparing all the wavelets on SLV 17.



More information on the third version of the AMR is given in 4.2.4. In the figure it can be observed that in the first version of the AMR all the wavelets have difficulties in reducing the error, whilst this error is significantly reduced in version 3 of the AMR. Moreover,

reducing the threshold results in more points but not a higher accuracy. To investigate this further it is convenient to observe the error at every time step, see Figure 5.4. There are some remarkable items to be observed in this figure.
Firstly, the error of Donoho's interpolating wavelet is for a long time constant at 1×10^{-15}. In reality this error is not constant, however many times Donoho was so accurate that the squaring in the relative error was causing arithmetic underflow. In order to catch away NAN numbers the relative error was manually set to 1×10^{-15}, in case of arithmetic underflow.

Secondly, it can be seen that the maximum error is occurring in the second time step. SLV 17 starts of with a very high peak on one point. Naturally, this will lead to adding points around this point. However, the number of neighbours added in this case is not high enough. This adaptive malfunction lead to testing different versions of the AMR. One successful version was version 3. In this setting two different threshold values are used. The first depends on the current approximation and the second is static, chosen beforehand. The dynamic threshold is relative to the approximation. 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 second step can be approximated, however this leads to adding more points. The adding of more points is not a problem for SLV 17, as the maximum number of points is reached at later time steps. However, to make this version suitable for more problems $L_{\text{dynamic}}^{\text{neighbour}} = 5$ was chosen. A benefit of version 3 is that extra points will be added only if there is a very high detail coefficient. This can be observed in Figure 5.5, at the first time steps Sweldens and Donoho add more points than in version 1 Figure 5.4, but after some time steps they grow to the same number of points.

Furthermore, Sweldens structurally has more points than Donoho's interpolating wavelet. This is caused by the interpolating boundary stencil, in all the test cases more points are added for the interpolating boundary stencil. More points are added near the boundary, because the interpolating boundary wavelets have a very high tail. Due to this tail points fairly far away from the boundary will influence the boundary. When going to the lower scales, e.g. level $J1$, this means that points relatively far away from the boundary influence the boundary. Hence, high detail coefficients further away from the boundary will result in higher detail coefficients close to the boundary. In the case of SLV17, Sweldens will also add more points because the peak is smoothed out so it will cover a larger area. One beneficial aspect of this smoothing is that Sweldens wavelet has less problems to adapt in the second time step.

However, although the interpolating boundary stencil is adding more points it works better than the lower order boundary stencil for Sweldens wavelet. The lower order implementation in Sweldens case is highly unstable, many times it has difficulties obtaining a high accuracy. Due to the update step, this is a problem with the Sweldens wavelet and not for Donoho's interpolating wavelet. Sweldens wavelet has to use the boundary stencil implementation both for the smoothing coefficients (update step) and for the detail coefficients.

Figure 5.4: Version 1, performance per time step on SLV 17.

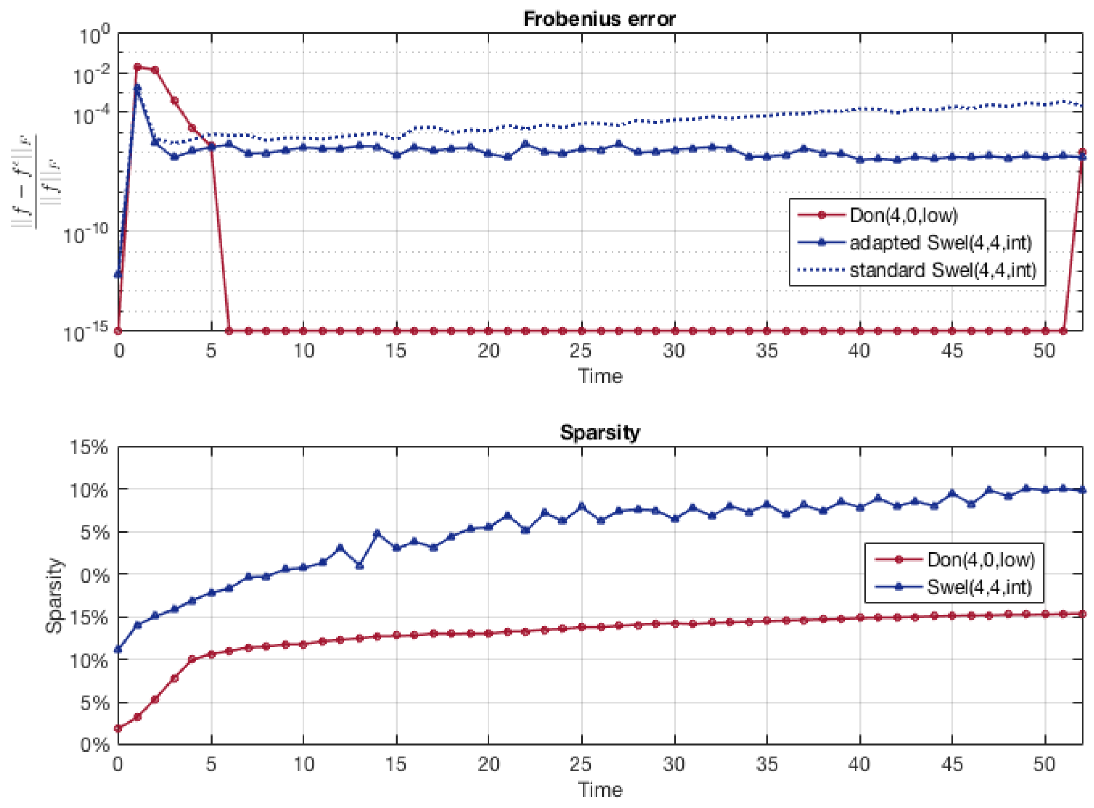
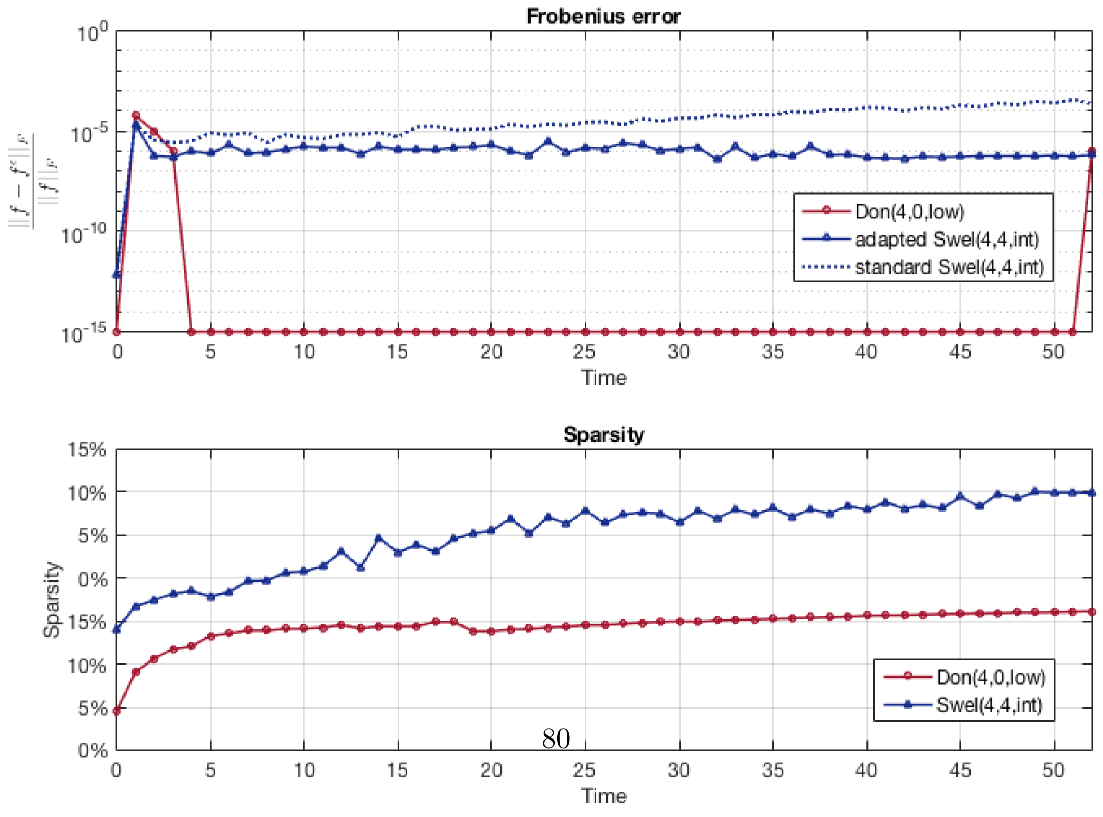


Figure 5.5: Version 3, performance per time step on SLV 17.

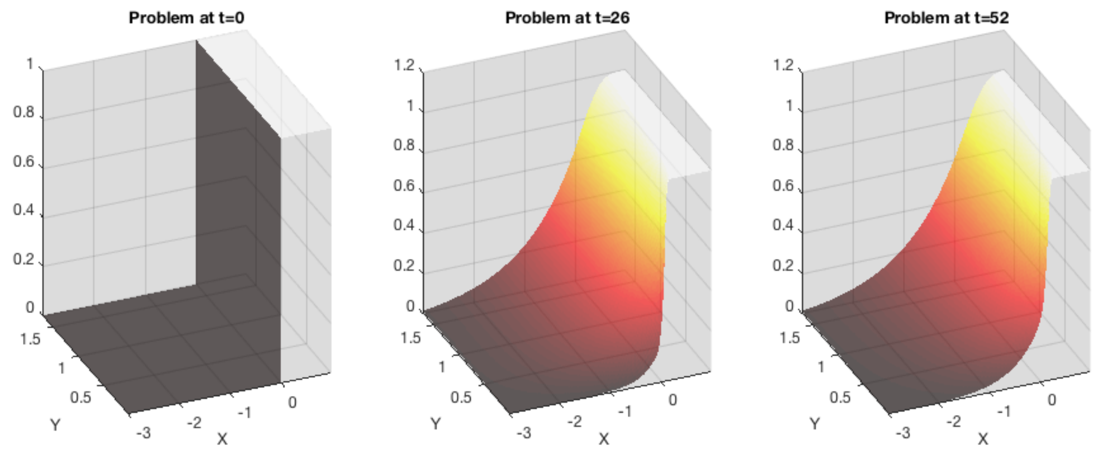


Whereas Donoho's wavelet only needs to deal with boundary points for the detail coefficients. Moreover, the update step works as a smoother, i.e., it averages the smoothing coefficients. Reducing the order near the boundary can cause an error. Due to the smoothing this error will be transported through the grid.

Lastly, in the error plots two different graphs for the Sweldens wavelet are depicted. In the testing phase it was discovered that high errors for the Sweldens wavelet were caused by the update step in the inverse transform. Therefore, we designed our own version of the inverse transform, the adapted inverse transform. The adapted inverse is described in section 4.2.2.This leads to only changing the last step in the AMR, hence both versions have the same meshes.

5.2.2 2D Digital

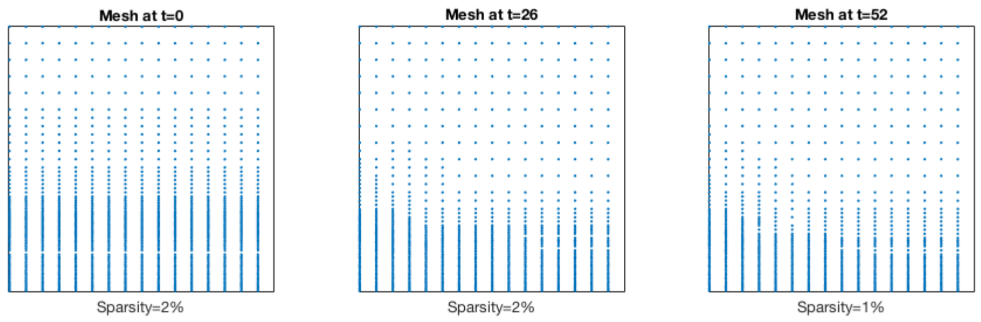
Figure 5.6: Digital through time.



Digital is a financial PDE with a completely different surface. It starts of as a step function, thereafter it smoothens out. Digital at time step 0, 26 and the last time step 52, can be seen in Figure 5.6. It can be observed that on one side of the domain the step function smoothens out more then on the other side. In Figure 5.7 the meshes generated by version 1 of the AMR using Sweldens interpolating wavelet can be seen. The type of neighbours added in version 1, only add neighbours in the direction that the point is acting as a detail coefficient. This can be clearly seen in the meshes. The meshes are highly sparse, however only points in the y direction are added. Although this PDE is different from SLV 17 version 1 of the AMR has difficulties with the error.

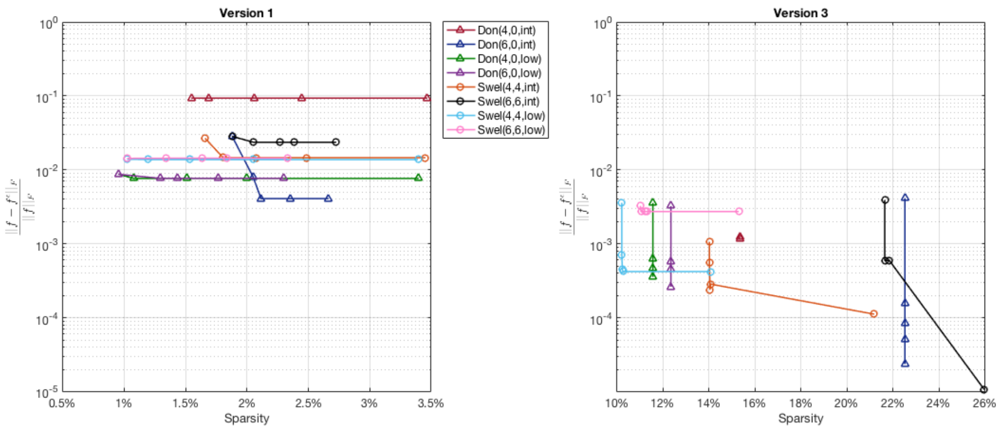
In Figure 5.8 a comparison of all the wavelets can be found. Indeed, version 1 of the AMR has difficulties reducing the errors. It is also worth to note that reducing the threshold, does not result in significantly more points. In the case of version 3 of the AMR a lot more points are used. However, version 3 of the AMR is capable of reducing the error. For example, Don(4,0,low) is reducing the error to below 1×10^{-3}, whilst keeping just 11% of the points. It is interesting that version 3, designed to deal with highly non smooth areas quickly spreading out, also has such a positive effect on the Digital PDE.

Figure 5.7: Meshes of digital using Swel(4,4,int).



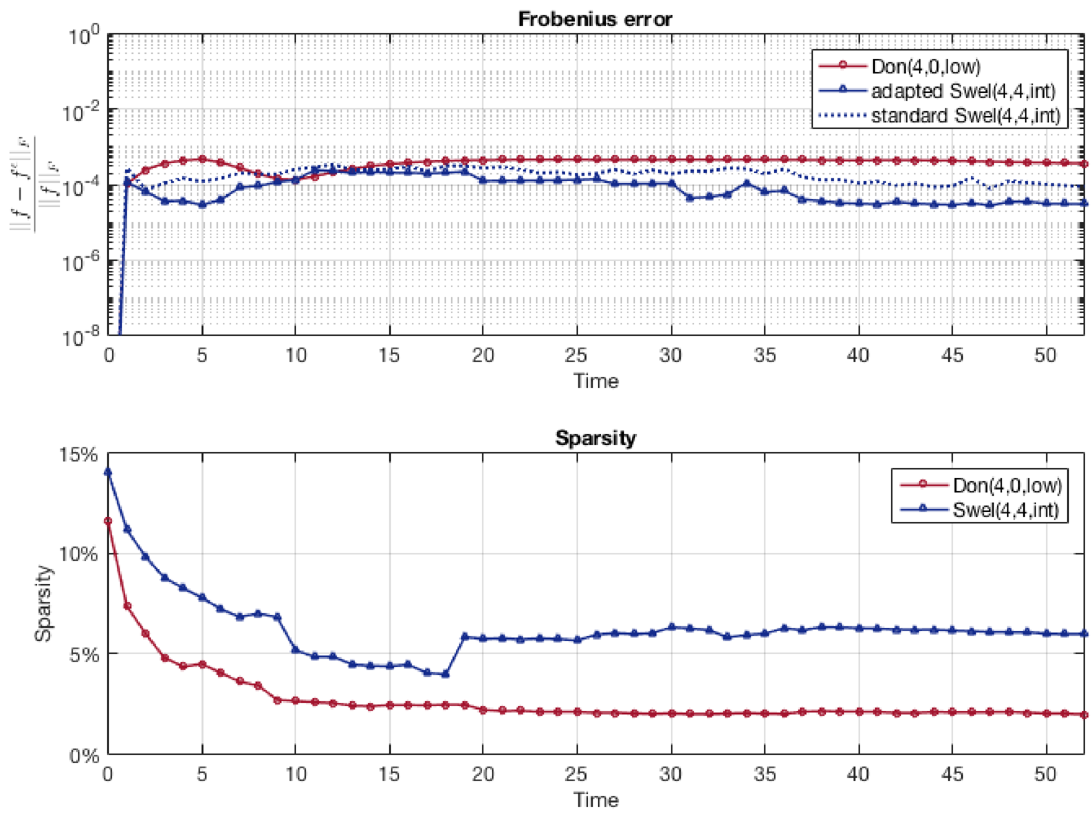
The 6th order wavelets with interpolating boundary stencil are adding a lot of points. This occurs in more data sets, although more points also result in a higher accuracy, it is not beneficial. In many cases, a PDE solver will not have an accuracy of 1×10^{-5}, hence it does not make sense to aim for an AMR which is that accurate. A better heuristic method would be to look for wavelets with the lowest number of points, whilst having an accuracy of 1×10^{-3}. Another notable aspect is that Don(4,0,int) appears as a point in version 3. This means that the maximum error and number of points are stirred by 1 or 2 time steps, hence lower threshold values do not result in better performance.

Figure 5.8: Comparing all the wavelets on digital.



In Figure 5.9 Don(4,0,low) and Swel(4,4,int) are compared using version 3 of the AMR. In this case Sweldens is slightly more accurate, whilst Donho's wavelet is a bit sparser. Contrary to SLV 17, it is not obvious which wavelet performs best.

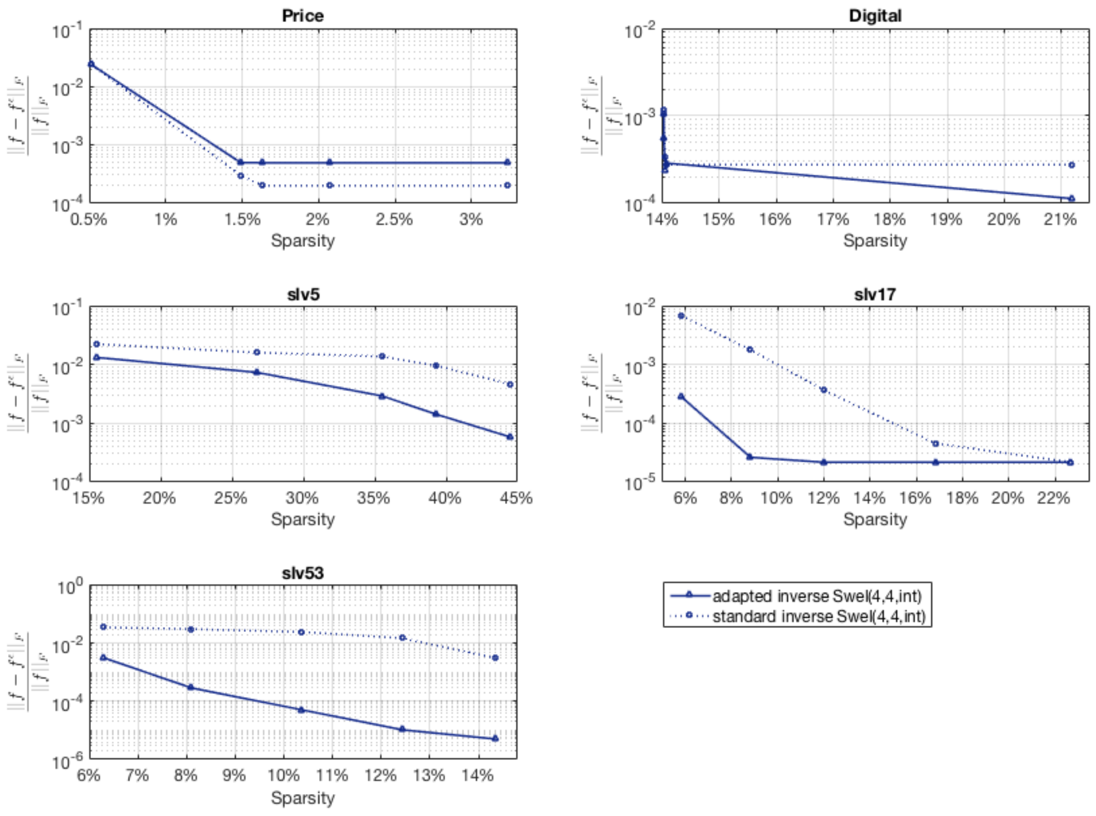
Figure 5.9: Performance per time step on digital.



5.2.3 Adapted inverse for Finance 2D PDEs

As mentioned before, there are two ways to extrapolate the approximation in the AMR. The normal inverse transform, which is currently the standard in literature, and an adaptive inverse transform can be used. This adapted version is not specifically designed for Sweldens wavelet, it is for any second generation wavelet with an update step. The idea for the adapted inverse transform resulted from investigating the SLV results. Hence, it is important to see the effect on all the data sets. In Figure 5.10 the results for Swel(4,4,int) on all the 2D data sets are depicted. Both the normal and adapted inverse transform were calculated for each threshold. Major improvement is observed in the SLV sets. This was to be expected as the problems are similar and they motivated the design of the adapted inverse transform. Especially, for SLV 53 the reduction in error is between 100x and 1000x better. In the case of SLV17 it is interesting to note that the adapted inverse has an accuracy $< 1 \times 10^{-4}$ with a sparsity of 6%. Meanwhile, the normal inverse transform needs around 17% points to reach that accuracy.

Figure 5.10: Comparing the different inverses for Sweldens.



There is almost no difference in the other two sets. For the Price set the error is even worse for the adapted inverse transform. However, it is important to note that the reduction in error is less than 10x. Furthermore, both versions of the inverse transform are capable to reduce the error to $< 1 \times 10^{-3}$, whilst maintaining a high sparsity.

5.2.4 1D Finance PDEs

In 1 dimension the wavelets behave very different. In the literature wavelet based AMR has mostly been tested on 1 dimensional PDEs. A lot of difficulties encountered in 2 dimensions are simply not important in 1 dimension. For example, different implementations of the AMR result in exactly the same values. Therefore, only the data of version 1 will be shown. Furthermore, the adapted inverse transform is exactly the same as the normal inverse in the 1 dimensional setting. 2 different PDEs will shortly be discussed, SLV 5 and Price.

Figure 5.11: LV 5 using Swel(4,4,int).

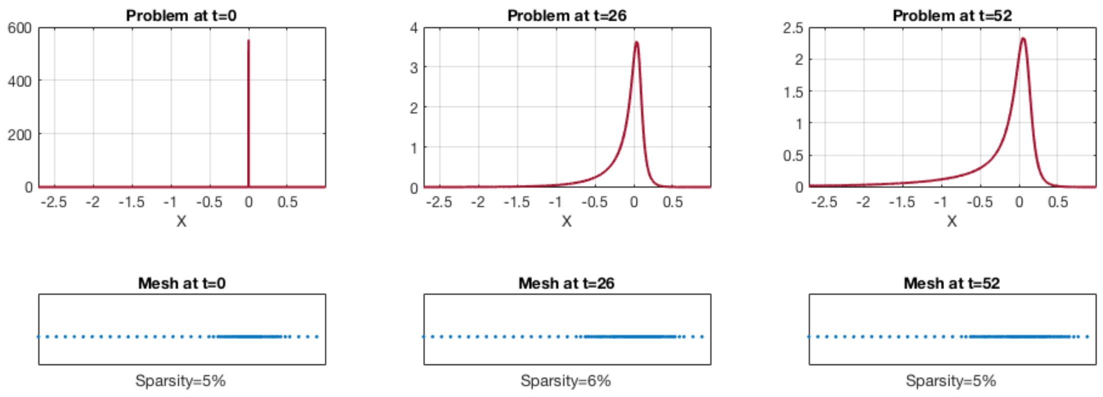
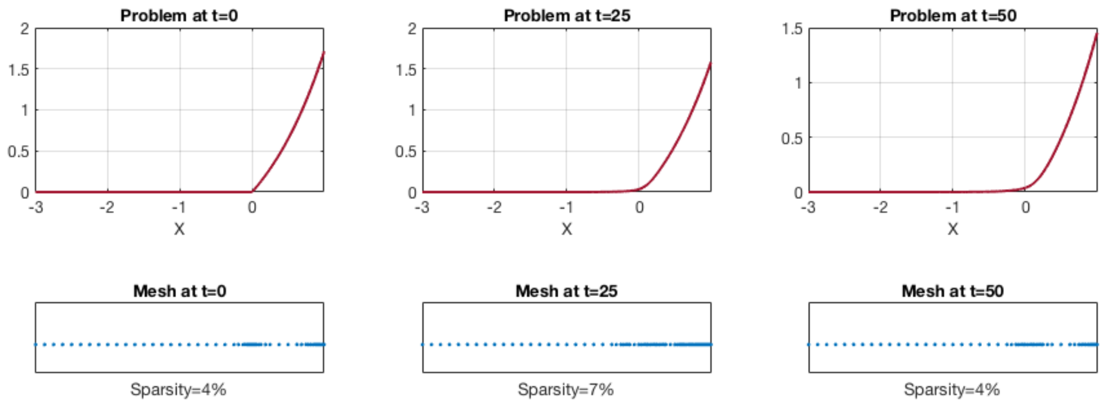
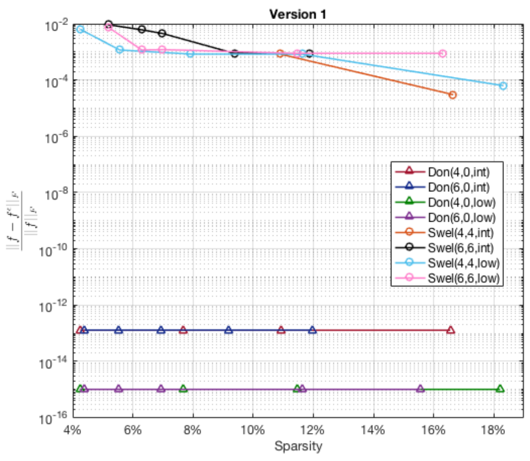


Figure 5.12: Price using Don(4,0,low).

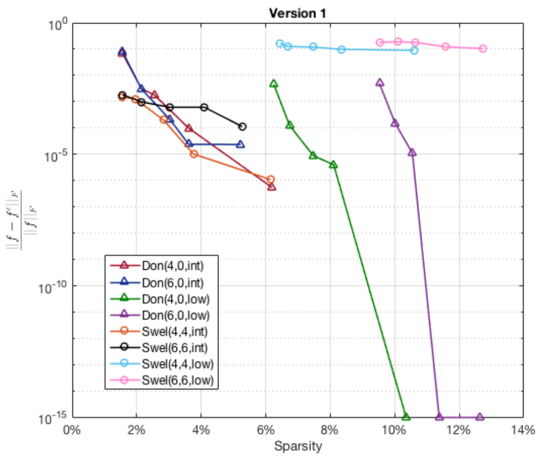


In Figure 5.11 LV 5 is shown at time steps 0, 26 and 52, below the plots the meshes are shown. Note that the peak of SLV 5 starts off at around 600 and then quickly reduces to around 4. In Figure 5.12 the Price set is depicted, which has just 51 time steps, hence $t = 0$, $t = 25$ and $t = 50$ are shown. In both the figures it can be seen that the grid is not only able to grow, but also able to shrink if the solution becomes smoother. In Figure 5.14a and Figure 5.14b all the wavelets are compared. It is interesting to note that Donoho's wavelets are highly accurate. As discussed earlier it only makes sense to consider an accuracy of 1×10^{-3}. In that setting Sweldens wavelets are also performing very good.

Figure 5.13: Comparing all the wavelets.



(a) lv 5.



(b) Price.

5.3 Hydro PDE

The 2D Hydro data is tested with finest scale $J2 = 8$ and coarsest scale $J1 = 4$. For the 3D test sets $J2 = 7$ was assumed to be the finest level of refinement and $J1 = 3$ the coarsest level.

5.3.1 2D Pressure

In Figure 5.14 the pressure is given at time step 0, 257 and the last time step 514. The surface and heat plots are depicted for every time step. The solution starts of with a local blob, which results in a wave that interacts with the boundaries. In Figure 5.15 the meshes at these time steps are shown. Donoho's lower order wavelet was 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 only around the shock waves, points are added. Furthermore, although the shocks are spread over the whole domain the wavelet is able to make a sparse grid, with just 19% of the points kept.

All the wavelets are tested on pressure for version 1 and 3 of the AMR, the results are shown in Figure 5.16. As mentioned earlier, in practise AMR is only considered worthwhile in 2D if less than 40% of the points are kept. Only Donoho's interpolating wavelet with the lower order boundary stencil implementation is capable to have this sparsity whilst having an error around 1×10^{-3}. Furthermore, it can be observed that there is not a big difference between the version 1 and 3. It is only important to note that Donoho's interpolating wavelet in version 3 will need around 30% of the points in order to reach an error around 1×10^{-3}. Meanwhile, in version 1 only 20% of the points are needed.

Figure 5.14: The pressure through time.

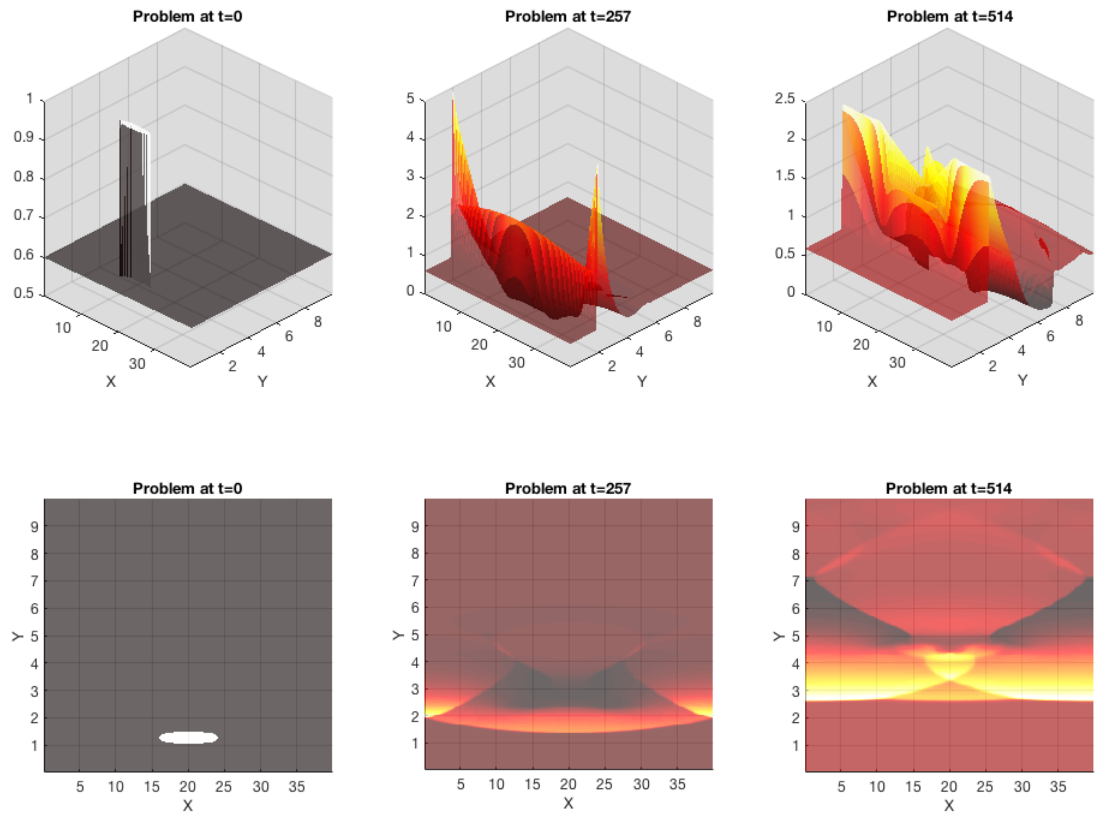
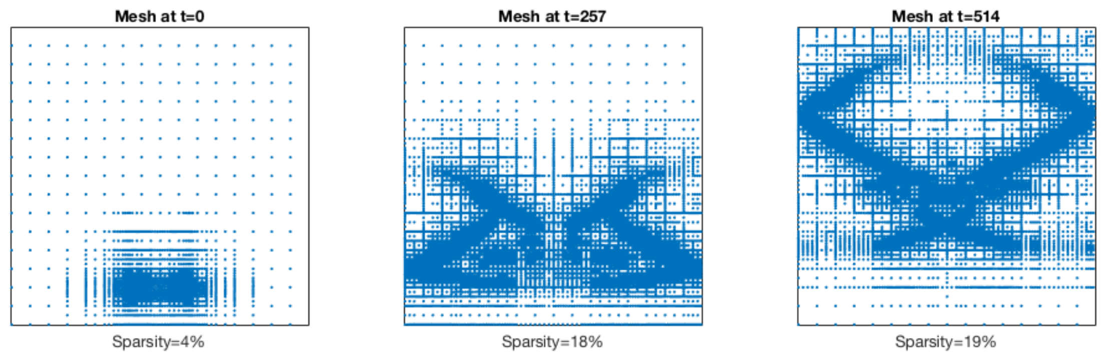


Figure 5.15: Meshes of pressure using Don(4,0,low).



In Figure 5.17 the Frobenius error and sparsity of pressure through time are given. The used threshold value is 1×10^{-3}, this leads to adding too many points. In the error plot it can be observed that the error of Donoho's wavelet is more stable. Furthermore, the standard inverse transform for Sweldens leads to a larger error, than using the adapted inverse. Although the maximal Frobenius errors are similar.

Figure 5.16: Comparing all the wavelets on pressure.

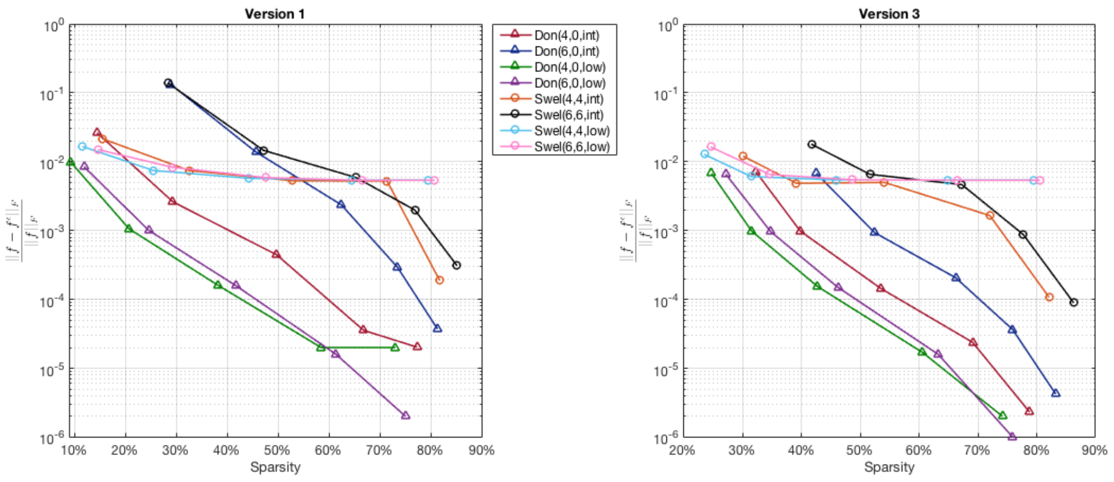
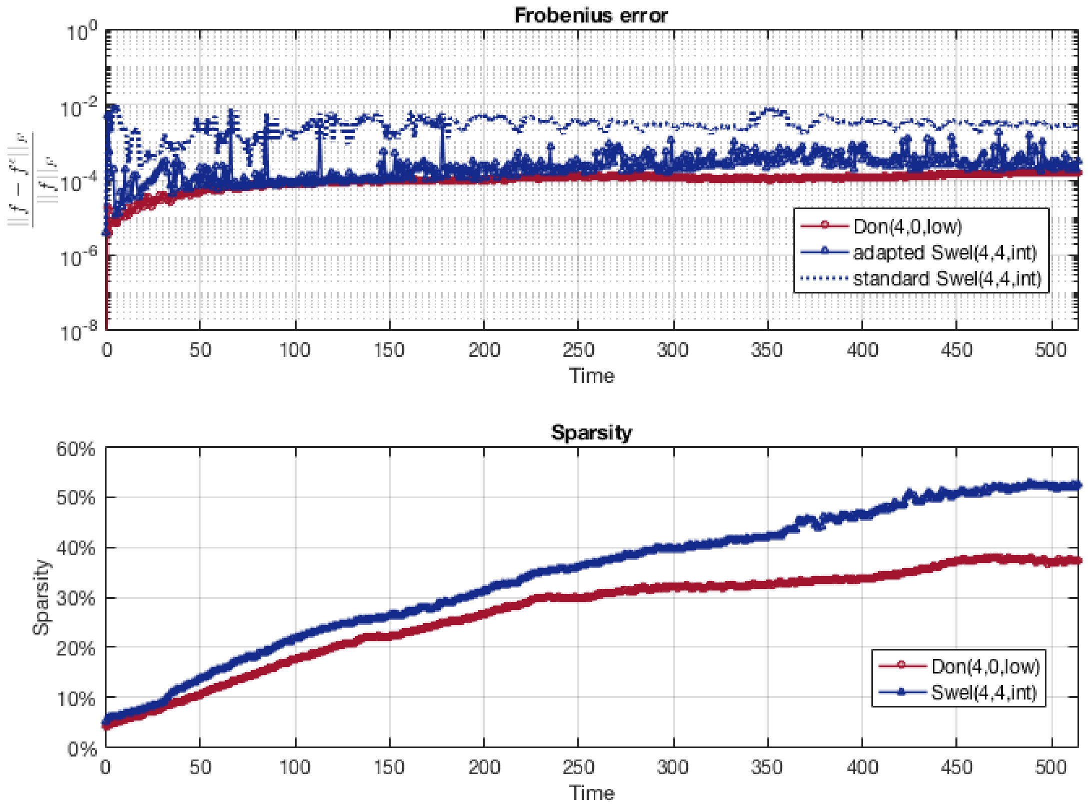


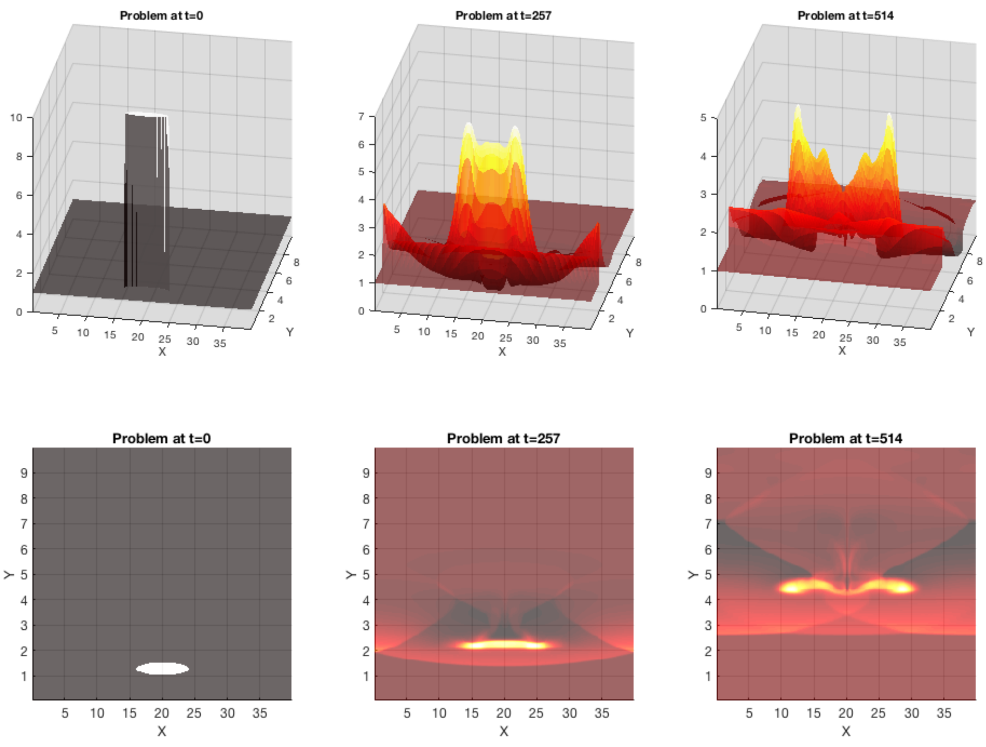
Figure 5.17: Performance per time step on pressure.



5.3.2 2D Density

The hydro PDE is solved using an AMR. This AMR is only applied to pressure and density. Therefore, the density data set is considered. In Figure 5.18 surfaces of density through time are given. The surfaces look very similar to the pressure surface, hence the union of these two grids will not contain too many extra points. In Figure 5.19 the meshes at the corresponding time steps are given. Sweldens wavelet with the interpolating boundary stencil is used. In the second and third mesh it is visible that too many points are added near the boundary. There are more mesh points on the right and upper side, this is caused by the manner in which the data is split. On the right and top all the boundary points are detail coefficients, so at this side the boundary stencil implementation is more sensitive for thresholding. At the left and lower side smoothing coefficients are at the boundary. As discussed earlier, the tail of the interpolating boundary stencil wavelet is causing high detail coefficient values to influence the boundary points, especially in the lower scales.

Figure 5.18: The density through time.



In Figure 5.20 and Figure 5.21 the plots for comparing all the wavelets and the behaviour through time can be seen. The results are very similar to the results of pressure, again Donoho combined with the lower order boundary stencil is having the best performance.

Figure 5.19: Meshes of density using Swel(4,4,int).

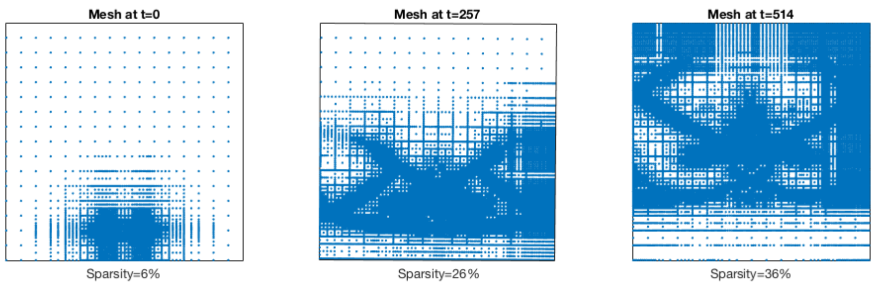


Figure 5.20: Comparing all the wavelets on density.

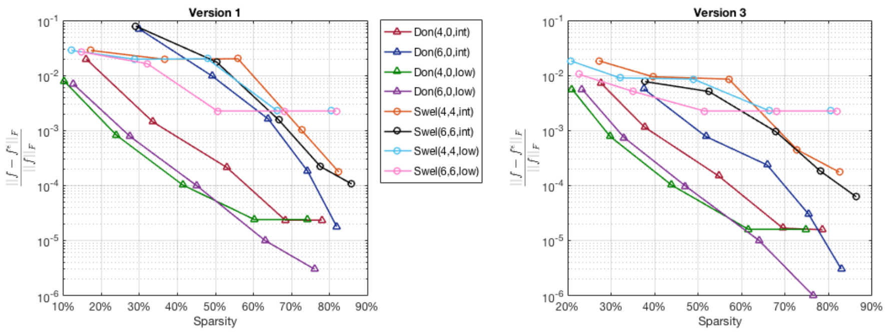
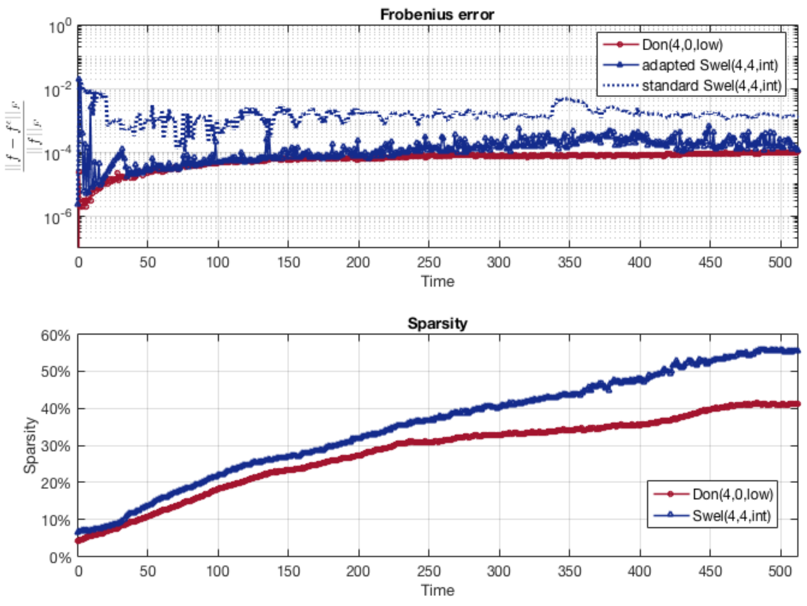


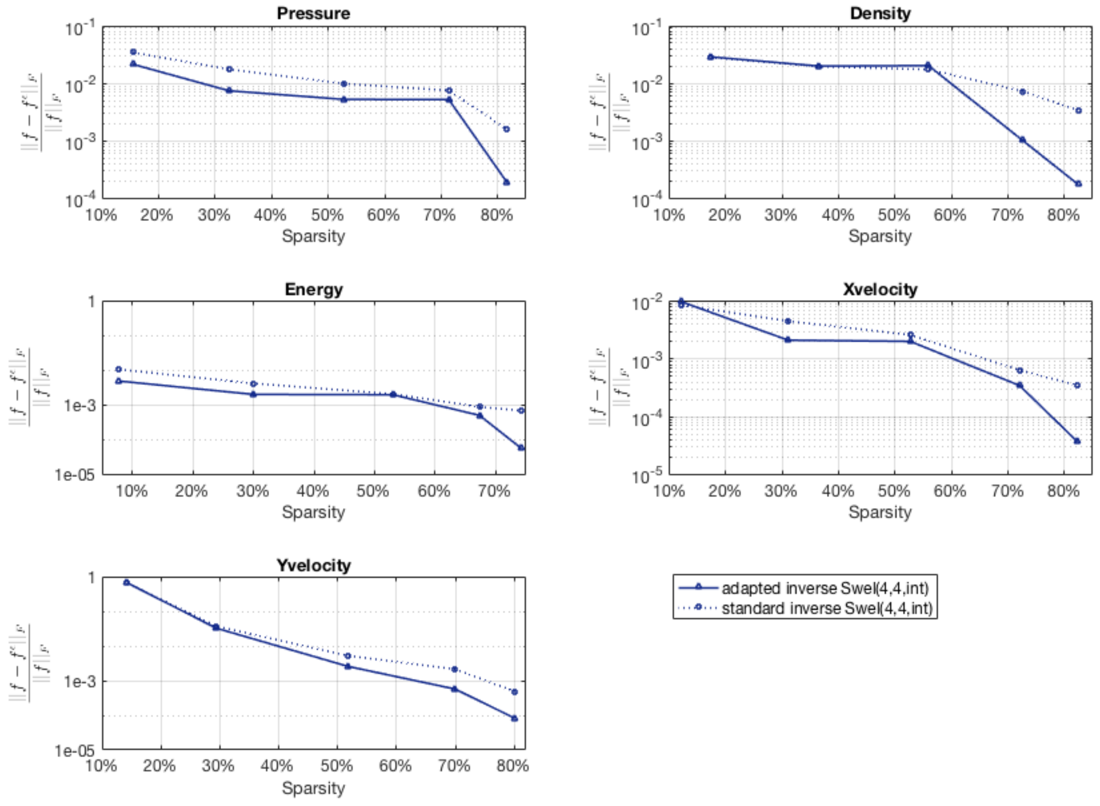
Figure 5.21: Performance per time step on density.



5.3.3 Adapted inverse for Hydro 2D PDE

The performance of the adapted inverse on all the data sets is given in Figure 5.22. Although the differences are not big, the adapted version of the inverse transform constantly results in a lower Frobenius error.

Figure 5.22: Comparing the different inverses for Sweldens.



5.3.4 3D results

In 3 dimensions it is more difficult to depict the behaviour of the solution in a plot. Therefore, these are not shown. The solution slices look very similar to the 2D test case. In Figure 5.23 and Figure 5.24 the overall performance of the wavelets is depicted. 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 only for a sparsity of 30% or less AMR is considered useful. Like before, only Donoho's interpolating wavelet with the lower order boundary stencil is able to reach this sparsity with a small Frobenius error.

Figure 5.23: Comparing all the wavelets on density.

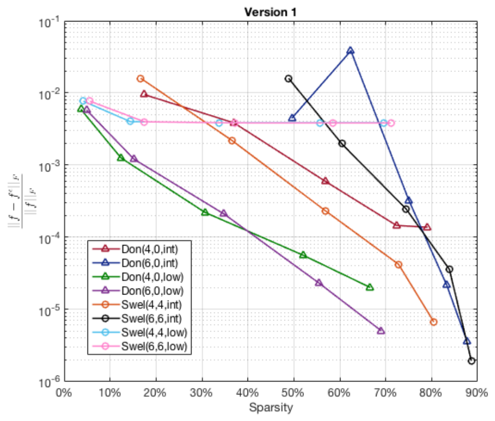
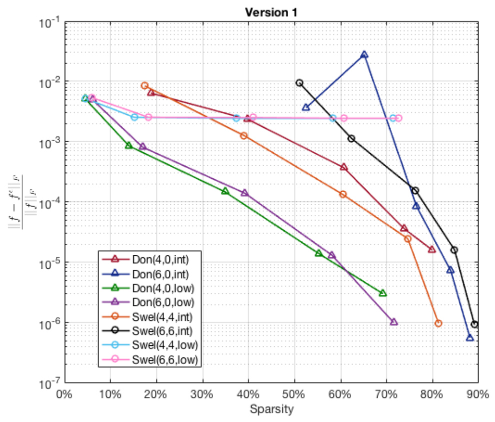


Figure 5.24: Comparing all the wavelets on pressure.



In Figure 5.25 and Figure 5.26, Don(4,0,low) and Swel(4,4,int) are plotted for every time step. The errors of Donoho and Sweldens are very similar, only the normal inverse of Sweldens is giving a higher error. Although the error behaves similar for Donoho and Sweldens, the number of points needed to achieve this error is different. Sweldens is adding many more points.

Figure 5.25: Performance per time step on pressure.

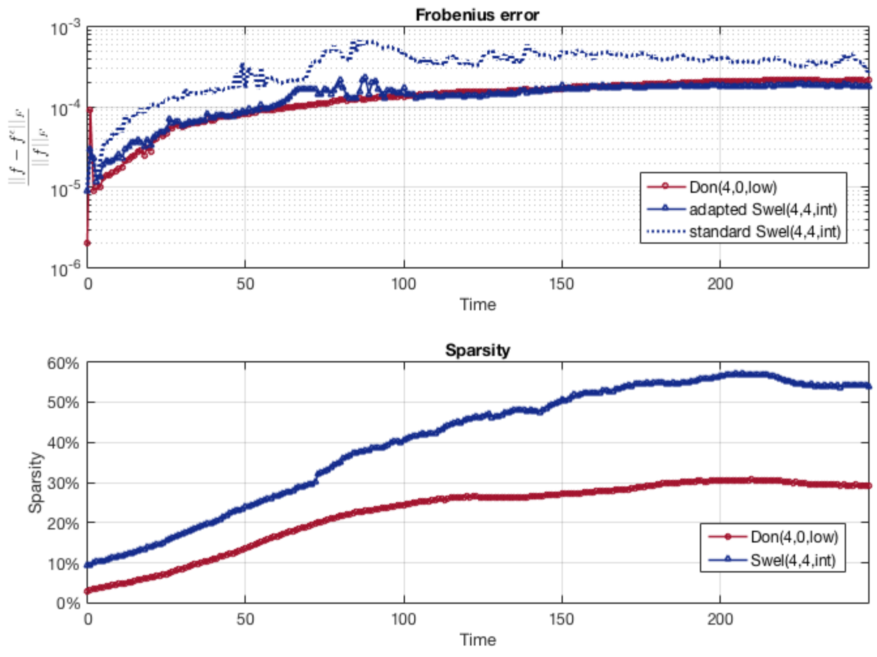
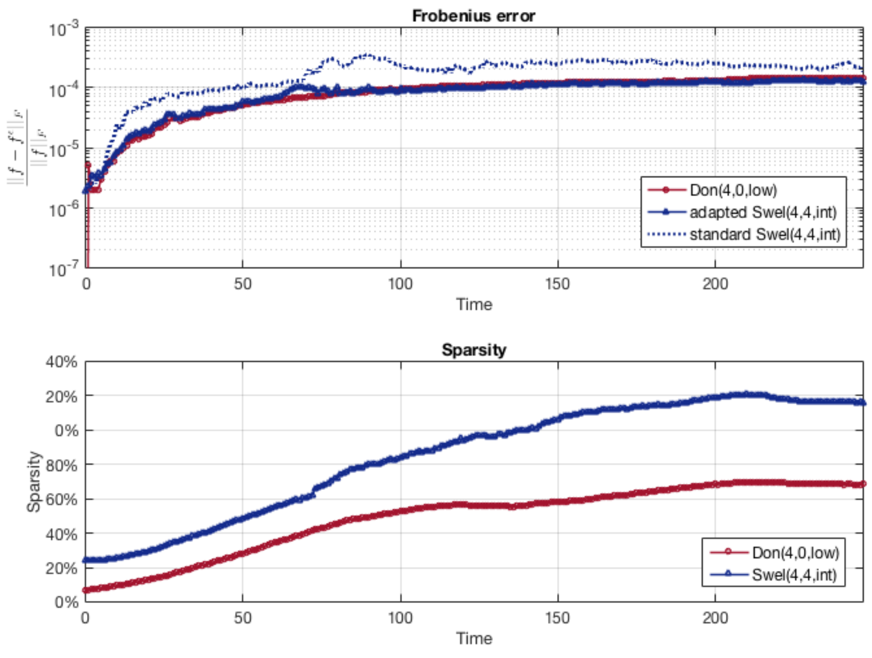


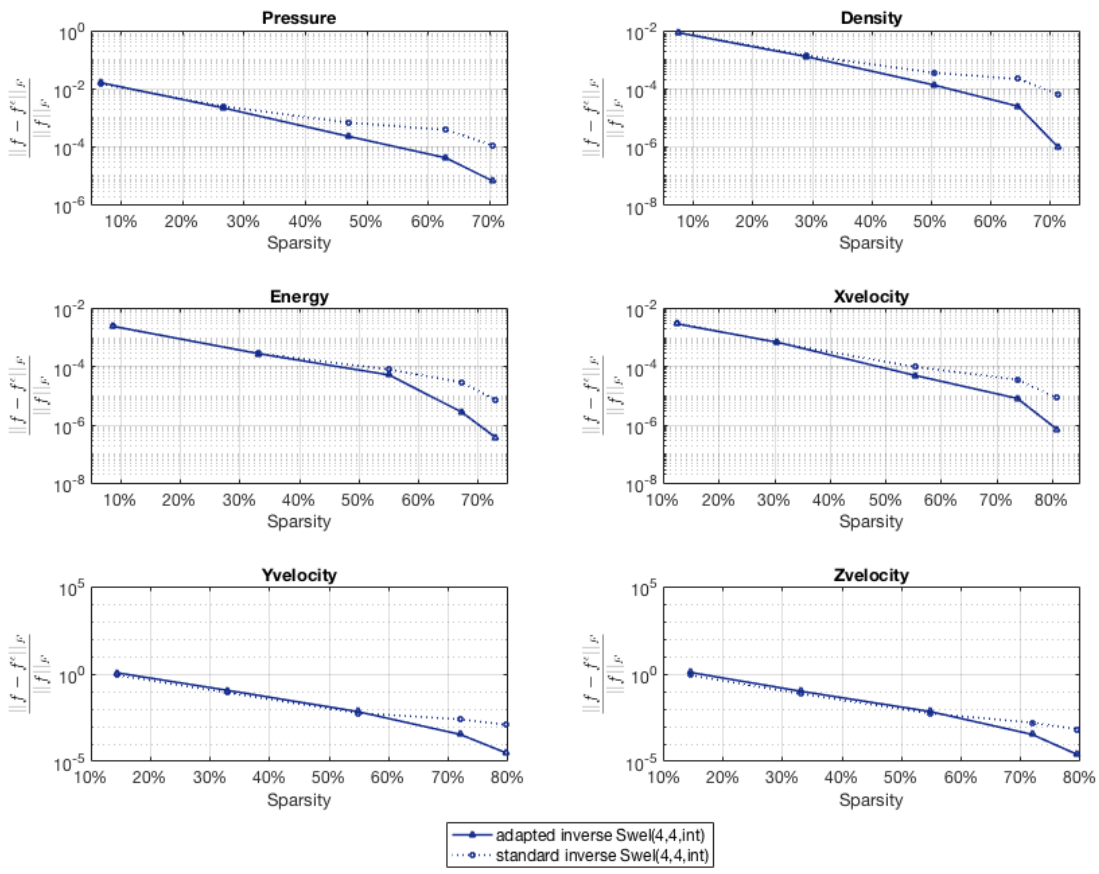
Figure 5.26: Performance per time step on density.



5.3.5 Adapted inverse for Hydro 3D PDE

In Figure 5.27 the different versions of the inverse transform are tested on all 3D hydro data sets. The maximum error is different for the very small thresholds, however there is not a big difference in the bigger threshold values. The difference for the smaller thresholds is not very important, as the wavelets are adding too many points there in order to consider AMR.

Figure 5.27: Comparing the different inverses for Sweldens.



6 Conclusion and Future work

6.1 Conclusion

The objective of this thesis is to research the performance of different wavelets for adaptive mesh refinement. In literature, Donoho's interpolating wavelet and Sweldens wavelet are used for adaptive mesh refinement, however it is not clear which wavelet is the best choice. Therefore, different versions of these wavelets are tested to answer the research question:

Which wavelet transform is most stable and has the best performance in adaptive mesh refinement?

The wavelets have been tested with different boundary stencil implementations, different versions of the AMR algorithm and different orders N. The tested orders N are 4 and 6. Although $N = 6$ outperforms $N = 4$ sometimes, for both Sweldens and Donoho's wavelet $N = 4$ ended up being the most stable choice.
Furthermore, for Sweldens wavelet the interpolating boundary stencil is the best boundary stencil implementation. It adds unnecessary points near the boundary due to the long tails of the boundary wavelets, but is accurate for all the 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.
Comparing Sweldens wavelet versus Donoho's interpolating wavelet, Donoho's interpolating wavelet with the lower order boundary stencil implementation is the best. Sweldens can compete with Donoho's interpolating wavelet on PDEs with very local behaviour, however when non smooth behaviour is occurring over the whole domain, Sweldens wavelet is adding too many points. This is caused both by the update step of Sweldens, which smoothens out the function, hence local behaviour becomes less local. Another cause is the interpolating boundary stencil, due to the long tail, on lower scales behaviour further away from the boundary influences the boundary wavelets.
Sweldens wavelet is a second generation wavelet with an update step, during testing it was noted that using an adapted inverse transform reduced the error significantly. This change in the inverse transform is only for wavelets with an update step, hence it does not effect Donoho's interpolating wavelet.
Concluding, throughout the different tests, Donoho's interpolating wavelet with the lower order boundary stencil was the most stable and best performing wavelet for equidistance meshes. This is remarkable as the lower order boundary stencil is not used in literature.

6.2 Recommendations for future work

Using wavelets for adaptive mesh refinement has shown promising results. In this thesis the focus is on testing the abilities of different wavelets for adaptive mesh refinement. However, there are still more aspects to be researched.

- ◇ The wavelets have been tested on equidistant meshes. Stability issues are known to occur for non equidistant meshes. Therefore, it would be interesting to compare the wavelets on non equidistant meshes. In [22] it is suggested that the update step is one of the causes for instability. This would suggest that Donoho's interpolating wavelet will also perform better for non equidistant meshes. However, this is a hypothesis which is not verified.
- ◇ The fast transform (forward and inverse transform) is suitable for parallelisation. Especially, the 3 dimensional fast transform is slow. In order for wavelet based AMR to be valuable in different fields of application, speeding up the code is necessary.
- ◇ Both Donoho's interpolating and Sweldens wavelets are not orthogonal. Donoho's interpolating wavelet could also be lifted using a different update step. For example, the update step could be utilised to orthogonalise the wavelets instead of raising the dual order. A hybrid version could also be considered, using one degree of freedom to have one dual vanishing moment and use the other degrees of freedom for orthogonalisation.
- ◇ Adaptive mesh refinement is a part of the collocation methods. Another topic to be researched is the actual solving of the PDEs. Multigrid would be interesting to explore. Because the wavelets divide a domain in different scales (layers), each scale could correspond to a mesh in Multigrid. In other words, the prolongation and restriction steps would be dictated by the different mesh points in the different scales.

Abbreviations and symbols

Abbreviations

AMR	Adaptive mesh refinement
MRA	Multiresolution analysis
NAG	Numerical Algorithms Group
PDE	Partial Differential Equations
SMR	Static mesh refinement

Symbols

$s_{j,k}$	Smoothing coefficient at level j and location k.
$d_{j,k}$	Detail coefficient at level j and location k.
$s_{j,k}$	Smoothing coefficient at level j and location k, l.
$d^a_{j,k,l}$	Detail coefficient in the x-direction at level j and location k, l.
$d^b_{j,k,l}$	Detail coefficient in the x-direction at level j and location k, l.
$d^c_{j,k}$	Detail coefficient in the x, y-direction at level j and location k, l.
V_j	Approximation space.
W_j	Detail space.
$\tilde{V}_j$	Dual approximation space.
$\tilde{W}_j$	Dual detail space.
W^a_j, W^b_j, W^c_j	Two dimensional detail spaces.
$\psi(\cdot)$	Mother scaling function.
$\tilde{\psi}(\cdot)$	Dual mother scaling function.
$\psi_{j,k}(\cdot)$	Basis scaling function at level j and location k.
$\tilde{\psi}_{j,k}(\cdot)$	Dual basis scaling function at level j and location k.
$\varphi(\cdot)$	Mother wavelet function.
$\tilde{\varphi}(\cdot)$	Dual mother wavelet function.
$\varphi_{j,k}(\cdot)$	Basis wavelet function at level j and location k.
$\tilde{\varphi}_{j,k}(\cdot)$	Dual basis wavelet function at level j and location k.
$\psi^a(\cdot), \psi^b(\cdot), \psi^c(\cdot)$	Two dimensional wavelets.
$J2$	Densest level, i.e. level with highest precision.
$J1$	Coarsest level, i.e. level with lowest precision.
N	Primal order of wavelet system.
$\tilde{N}$	Dual order of wavelet system.
$H = (h_k)_{k\in\mathbb{Z}}$	Filter of the primal scaling functions.
$\tilde{H} = (\tilde{h}_k)_{k\in\mathbb{Z}}$	Filter of the dual scaling functions.
$G = (g_k)_{k\in\mathbb{Z}}$	Filter of the primal wavelet functions.

$\tilde{G} = (\tilde{g}_k)_{k\in\mathbb{Z}}$	Filter of the dual wavelet functions.
$h(\omega), g(\omega)$	2π-periodic primal filters.
$\tilde{h}(\omega), \tilde{g}(\omega)$	2π-periodic dual filters.
$\mathcal{P}(s_{j,k})$	Predict operator.
$\mathcal{U}(d_{j,k})$	Update operator.
$P^N = (p_k)_{k\in\mathbb{Z}}$	Filter of the predict step, with order N.
$U^{\tilde{N}} = (u_k)_{k\in\mathbb{Z}}$	Filter of the update step, with order $\tilde{N}$.
$P^{N,R}, P^{N,L}$	Filter of the predict step on the right and left boundaries.
G^j	Dyadic grid with 2^j points.
M^j	Mesh at layer j resulting from the mesh refinement.
M	Union of the all the meshes M^j.
$L^{\text{neighbour}}$	The amount of neighbours added per layer.
L^{level}	The amount of levels in which adjacent points are added.

Bibliography

[1] E. B.-D. A. Nejadmalayeri, A. Vezolainen and O. V. Vasilyev. Parallel adaptive wavelet collocation method for pdes. *Journal of Computational Physics*, 298:237–253, 2015.

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

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

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

[5] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Communication on Pure and Applied Mathematics*, 41:909–996, 1988.

[6] I. Daubechies and W. Sweldens. Factoring wavelet transforms into lifting steps. 1996.

[7] B. C. de Wiart. *Wavelet optimised PDE methods for financial derivatives.* PhD thesis, Centre for Financial Research, Judge Institute of Management and St John's College, University of Cambridge, 2005.

[8] B. C. de Wiart and M. A. H. Dempster. Wavelet optimized valuation of financial derivatives. *International Journal of Theoretical and Applied Finance*, 14(7):1113–1137, 2011.

[9] D. L. Donoho. Interpolating wavelet transforms. *Presented at the NATO Advanced Study Institute Conference, Ciocco, Italy*, 1992.

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

[11] M. Guarrasi. An introduction to adaptive mesh refinement (amr): Numerical methods and tools.

[12] B. Hand and Z. Shen. Wavelets with short support. *SIAM J. Math. Analysis*, 38:530–556, 2006.

[13] A. Harten. Adaptive multiresolution schemes for shock computations. *Journal of Computational Physics*, 1994.

[14] M. Jansen, M. Malfait, and A. Bultheel. Generalized cross validation for wavelet thresholding. *Signal Processing*, 1997.

[15] M. Jansen and P. Oonincx. *Second Generation Wavelets and Applications.* Springer, 2005.

[16] J. Liandrat and P. Tchamitchian. Resolution of the 1d regularized burgers equation using a spatial wavelet approximation. Technical report, Nasa and ICASE, 1990.

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

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

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

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

[21] K. Schneider and O. V. Vasilyev. Wavelet methods in computational fluid dynamics. *Annual Review of Fluid Mechanics*, 42:473–503, 2010.

[22] J. Simoens and S. Vandewalle. On the stability of wavelet bases in the lifting scheme. Technical report, Department of Computer Science, K.U. Leuven, 2000.

[23] W. Sweldens. The lifting scheme: A new philosophy in biorthogonal wavelet constructions. In *Wavelet Applications in Signal and Image Processing III*, pages 68–79, 1995.

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

[25] W. Sweldens. The lifting scheme: A construction of second generation wavelets. *Society for Industrial and Applied Mathematics*, 011, 1998.

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

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

[28] J. Tian and R. O. Wells. A remark on vanishing moments. In *PROC. 30th asilomar conf. on signals, systems and computers*, 1996.

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

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

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

[32] D. Wirasaet. *Numerical Solutions of Multi-Dimensional Partial Differential Equations using an Adaptive Wavelet Method.* PhD thesis, University of Notre Dame, 2007.

Appendix

Not all the data was shared in the results. If one is interested to see more results, data is provided in this Appendix. Five different thresholds have been tested for the different data sets and versions. Only the most interesting threshold per data set and version is given in the tables. In 1D only version 1 is given as the data of version 2 and 3 are the same. In 3D the Z-velocity are in a separate table at the end.
If one inspects the tables it can be observed that version 1 and 2 are very similar. Furthermore, the wavelets have difficulties with the y- and z-velocity of the Hydro PDE.

A.1 Finance PDEs data

A.1.1 1D results

Price	Version 1			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	2.54	1.84E-03	-
1.00E-03	Don(6,0,int)	3.03	2.12E-04	-
	Don(4,0,low)	7.47	9.00E-06	-
	Don(6,0,low)	10.55	1.10E-05	-
	Swel(4,4,int)	2.83	2.01E-04	2.01E-04
	Swel(6,6,int)	3.03	6.14E-04	6.14E-04
	Swel(4,4,low)	7.47	1.19E-01	1.19E-01
	Swel(6,6,low)	10.64	1.79E-01	1.79E-01
Digital	Version 1			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	3.27	5.02E-15	-
1.00E-02	Don(6,0,int)	4.35	1.32E-14	-
	Don(4,0,low)	3.27	1.00E-16	-
	Don(6,0,low)	4.35	1.00E-16	-
	Swel(4,4,int)	3.81	8.87E-04	8.87E-04
	Swel(6,6,int)	4.20	4.77E-03	4.77E-03
	Swel(4,4,low)	3.81	5.22E-04	5.22E-04
	Swel(6,6,low)	4.20	1.05E-03	1.05E-03
LV 5	Version 1			
		Sparsity	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	10.94	1.27E-13	-
1.00E-04	Don(6,0,int)	9.18	1.28E-13	-
	Don(4,0,low)	11.47	1.00E-16	-
	Don(6,0,low)	11.62	1.00E-16	-
	Swel(4,4,int)	10.89	8.38E-04	8.38E-04
	Swel(6,6,int)	9.38	8.85E-04	8.85E-04
	Swel(4,4,low)	11.62	8.38E-04	8.38E-04
	Swel(6,6,low)	11.47	8.85E-04	8.85E-04
LV 17	Version 1			
		Sparsity	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	4.25	1.63E-19	-
1.00E-01	Don(6,0,int)	4.79	1.63E-19	-
	Don(4,0,low)	4.25	1.00E-16	-
	Don(6,0,low)	4.79	1.00E-16	-
	Swel(4,4,int)	4.44	9.78E-04	9.78E-04
	Swel(6,6,int)	5.18	3.91E-04	3.91E-04
	Swel(4,4,low)	4.44	9.78E-04	9.78E-04
	Swel(6,6,low)	5.18	3.91E-04	3.91E-04
LV 53	Version 1			
		Sparsity	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	5.71	1.27E-13	-
1.00E-01	Don(6,0,int)	6.05	1.28E-13	-
	Don(4,0,low)	5.71	1.00E-16	-
	Don(6,0,low)	6.05	1.00E-16	-
	Swel(4,4,int)	5.71	4.47E-02	4.47E-02
	Swel(6,6,int)	6.05	4.54E-02	4.54E-02
	Swel(4,4,low)	5.71	4.47E-02	4.47E-02
	Swel(6,6,low)	6.05	4.48E-02	4.48E-02

A.1.2 2D results

Version 1

Price	Version 1			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	1.63	1.96E-04	-
1.00E-03	Don(6,0,int)	1.93	1.42E-04	-
	Don(4,0,low)	1.35	1.38E-03	-
	Don(6,0,low)	1.44	1.38E-03	-
	Swel(4,4,int)	1.64	1.96E-04	4.86E-04
	Swel(6,6,int)	2.00	1.42E-04	1.19E-03
	Swel(4,4,low)	1.40	1.03E-02	1.68E-02
	Swel(6,6,low)	1.56	9.70E-03	1.70E-02
Digital	Version 1			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	3.47	9.34E-02	-
1.00E-05	Don(6,0,int)	2.66	4.08E-03	-
	Don(4,0,low)	3.40	7.65E-03	-
	Don(6,0,low)	2.30	7.67E-03	-
	Swel(4,4,int)	3.45	4.90E-03	1.46E-02
	Swel(6,6,int)	2.73	4.08E-03	2.36E-02
	Swel(4,4,low)	3.39	7.65E-03	1.38E-02
	Swel(6,6,low)	2.33	7.68E-03	1.44E-02
SLV 5	Version 1			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	29.58	1.36E-01	-
1.00E-05	Don(6,0,int)	28.27	8.37E-02	-
	Don(4,0,low)	22.80	2.58E-02	-
	Don(6,0,low)	16.24	1.41E-02	-
	Swel(4,4,int)	44.50	4.41E-03	4.95E-04
	Swel(6,6,int)	46.37	7.39E-03	6.38E-04
	Swel(4,4,low)	42.05	3.82E-03	2.61E-03
	Swel(6,6,low)	43.09	1.20E-02	1.20E-02
SLV 17	Version 1			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	8.63	1.45E-01	-
1.00E-03	Don(6,0,int)	10.13	2.84E-02	-
	Don(4,0,low)	6.16	1.97E-02	-
	Don(6,0,low)	5.93	1.68E-02	-
	Swel(4,4,int)	12.04	1.81E-03	1.60E-03
	Swel(6,6,int)	17.77	7.64E-03	1.81E-03
	Swel(4,4,low)	10.04	3.41E-03	3.46E-03
	Swel(6,6,low)	12.28	1.14E-02	1.18E-02
SLV 53	Version 1			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	5.14	1.48E-04	-
1.00E-02	Don(6,0,int)	6.91	2.02E-04	-
	Don(4,0,low)	3.30	1.48E-04	-
	Don(6,0,low)	3.87	1.35E-04	-
	Swel(4,4,int)	7.10	3.06E-02	2.93E-04
	Swel(6,6,int)	9.54	2.59E-02	2.29E-03
	Swel(4,4,low)	5.36	9.17E-03	2.86E-03
	Swel(6,6,low)	6.30	1.16E-02	1.15E-02

Version 2

Price	Version 2			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	1.63	1.96E-04	-
1.00E-03	Don(6,0,int)	1.93	1.42E-04	-
	Don(4,0,low)	1.35	1.38E-03	-
	Don(6,0,low)	1.44	1.38E-03	-
	Swel(4,4,int)	1.64	1.96E-04	4.86E-04
	Swel(6,6,int)	2.00	1.42E-04	1.19E-03
	Swel(4,4,low)	1.40	1.03E-02	1.68E-02
	Swel(6,6,low)	1.56	9.70E-03	1.70E-02
Digital	Version 2			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	3.47	9.34E-02	-
1.00E-05	Don(6,0,int)	2.66	4.08E-03	-
	Don(4,0,low)	3.40	7.65E-03	-
	Don(6,0,low)	2.30	7.67E-03	-
	Swel(4,4,int)	3.45	4.90E-03	1.46E-02
	Swel(6,6,int)	2.73	4.08E-03	2.36E-02
	Swel(4,4,low)	3.39	7.65E-03	1.38E-02
	Swel(6,6,low)	2.33	7.68E-03	1.44E-02
SLV 5	Version 2			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	29.58	3.43E-02	-
1.00E-05	Don(6,0,int)	28.28	8.32E-02	-
	Don(4,0,low)	22.80	1.25E-02	-
	Don(6,0,low)	16.24	4.64E-03	-
	Swel(4,4,int)	44.53	3.94E-03	4.27E-04
	Swel(6,6,int)	46.40	7.39E-03	6.45E-04
	Swel(4,4,low)	41.92	4.14E-03	2.61E-03
	Swel(6,6,low)	43.03	1.20E-02	1.20E-02
SLV 17	Version 2			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	8.66	8.45E-03	-
1.00E-03	Don(6,0,int)	10.17	2.33E-02	-
	Don(4,0,low)	6.19	6.35E-03	-
	Don(6,0,low)	6.04	3.77E-03	-
	Swel(4,4,int)	12.01	1.81E-03	1.60E-03
	Swel(6,6,int)	17.90	7.64E-03	1.81E-03
	Swel(4,4,low)	10.09	3.25E-03	3.27E-03
	Swel(6,6,low)	12.34	1.14E-02	1.19E-02
SLV 53	Version 2			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	5.14	1.14E-04	-
1.00E-02	Don(6,0,int)	6.92	2.02E-04	-
	Don(4,0,low)	3.32	1.34E-04	-
	Don(6,0,low)	3.89	1.35E-04	-
	Swel(4,4,int)	7.10	3.06E-02	2.93E-04
	Swel(6,6,int)	9.55	2.59E-02	2.29E-03
	Swel(4,4,low)	5.38	9.17E-03	2.90E-03
	Swel(6,6,low)	6.35	1.16E-02	1.15E-02

Version 3

Price	Version 3			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	1.49	3.10E-04	-
1.00E-02	Don(6,0,int)	9.79	5.35E-05	-
	Don(4,0,low)	3.20	6.14E-04	-
	Don(6,0,low)	3.56	5.12E-04	-
	Swel(4,4,int)	1.49	2.97E-04	4.91E-04
	Swel(6,6,int)	9.79	5.10E-05	1.12E-03
	Swel(4,4,low)	4.68	1.05E-02	1.01E-02
	Swel(6,6,low)	5.80	9.95E-03	9.58E-03
Digital	Version 3			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	15.34	1.15E-03	-
1.00E-02	Don(6,0,int)	22.51	1.54E-04	-
	Don(4,0,low)	11.57	6.34E-04	-
	Don(6,0,low)	12.35	5.72E-04	-
	Swel(4,4,int)	14.03	1.03E-03	5.50E-04
	Swel(6,6,int)	21.64	7.69E-04	5.96E-04
	Swel(4,4,low)	10.22	7.89E-04	7.16E-04
	Swel(6,6,low)	11.05	2.72E-03	2.72E-03
SLV 5	Version 3			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	29.58	1.98E-02	-
1.00E-05	Don(6,0,int)	28.77	9.15E-03	-
	Don(4,0,low)	22.80	3.46E-04	-
	Don(6,0,low)	16.25	6.32E-04	-
	Swel(4,4,int)	44.49	4.63E-03	5.77E-04
	Swel(6,6,int)	46.35	7.27E-03	6.37E-04
	Swel(4,4,low)	41.93	4.29E-03	2.61E-03
	Swel(6,6,low)	42.99	1.20E-02	1.20E-02
SLV 17	Version 3			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	5.12	5.51E-03	-
1.00E-01	Don(6,0,int)	7.42	3.98E-04	-
	Don(4,0,low)	3.99	5.80E-05	-
	Don(6,0,low)	4.70	3.98E-04	-
	Swel(4,4,int)	5.80	7.00E-03	2.86E-04
	Swel(6,6,int)	10.58	9.52E-03	4.09E-04
	Swel(4,4,low)	6.48	2.28E-03	3.05E-03
	Swel(6,6,low)	7.32	1.19E-02	1.25E-02
SLV 53	Version 3			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	7.08	1.06E-04	-
1.00E-02	Don(6,0,int)	10.96	2.01E-04	-
	Don(4,0,low)	4.65	1.29E-04	-
	Don(6,0,low)	5.24	1.35E-04	-
	Swel(4,4,int)	8.10	3.06E-02	2.90E-04
	Swel(6,6,int)	10.30	2.45E-02	2.29E-03
	Swel(4,4,low)	6.66	9.17E-03	2.90E-03
	Swel(6,6,low)	7.57	1.16E-02	1.15E-02

A.2 Hydro PDE data

A.2.1 2D results

Version 1

Pressure	Version 1			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	33.58	1.44E-03	-
1.00E-02	Don(6,0,int)	49.16	9.84E-03	-
	Don(4,0,low)	23.97	8.29E-04	-
	Don(6,0,low)	27.57	7.85E-04	-
	Swel(4,4,int)	36.55	1.96E-02	2.00E-02
	Swel(6,6,int)	50.32	1.52E-01	1.77E-02
	Swel(4,4,low)	28.83	1.96E-02	2.00E-02
	Swel(6,6,low)	32.12	1.55E-02	1.66E-02
Density	Version 1			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	29.28	2.62E-03	-
1.00E-02	Don(6,0,int)	45.61	1.38E-02	-
	Don(4,0,low)	20.68	1.06E-03	-
	Don(6,0,low)	24.54	1.01E-03	-
	Swel(4,4,int)	32.48	1.76E-02	7.44E-03
	Swel(6,6,int)	47.08	1.07E-01	1.46E-02
	Swel(4,4,low)	25.45	1.43E-02	7.44E-03
	Swel(6,6,low)	29.14	1.33E-02	8.32E-03
Energy	Version 1			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	36.63	4.76E-04	-
1.00E-02	Don(6,0,int)	50.94	3.58E-03	-
	Don(4,0,low)	26.25	2.74E-04	-
	Don(6,0,low)	29.72	2.65E-04	-
	Swel(4,4,int)	40.06	4.25E-03	2.05E-03
	Swel(6,6,int)	53.34	1.53E-02	4.27E-03
	Swel(4,4,low)	31.77	1.90E-03	2.05E-03
	Swel(6,6,low)	35.43	1.44E-03	2.21E-03
X velocity	Version 1			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	28.13	6.60E-04	-
1.00E-02	Don(6,0,int)	42.72	3.65E-03	-
	Don(4,0,low)	19.10	3.66E-04	-
	Don(6,0,low)	22.67	3.51E-04	-
	Swel(4,4,int)	31.07	4.53E-03	2.12E-03
	Swel(6,6,int)	43.94	4.41E-02	4.64E-03
	Swel(4,4,low)	23.83	2.63E-03	2.11E-03
	Swel(6,6,low)	27.36	1.78E-03	2.20E-03
Y velocity	Version 1			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	29.19	3.37E-02	-
1.00E-02	Don(6,0,int)	44.76	2.76E-01	-
	Don(4,0,low)	20.06	3.37E-02	-
	Don(6,0,low)	23.53	4.00E-02	-
	Swel(4,4,int)	29.38	3.85E-02	3.37E-02
	Swel(6,6,int)	43.09	2.00E-01	2.53E-01
	Swel(4,4,low)	21.37	3.85E-02	3.37E-02
	Swel(6,6,low)	24.87	4.14E-02	3.48E-02

Version 2

Pressure	Version 2			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	33.58	1.44E-03	-
1.00E-02	Don(6,0,int)	49.37	9.84E-03	-
	Don(4,0,low)	23.97	8.29E-04	-
	Don(6,0,low)	27.57	7.85E-04	-
	Swel(4,4,int)	36.55	1.96E-02	2.00E-02
	Swel(6,6,int)	50.32	1.52E-01	1.77E-02
	Swel(4,4,low)	28.83	1.96E-02	2.00E-02
	Swel(6,6,low)	32.12	1.55E-02	1.66E-02
Density	Version 2			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	29.28	2.62E-03	-
1.00E-02	Don(6,0,int)	45.85	1.38E-02	-
	Don(4,0,low)	20.68	1.06E-03	-
	Don(6,0,low)	24.54	1.01E-03	-
	Swel(4,4,int)	32.48	1.76E-02	7.44E-03
	Swel(6,6,int)	47.13	1.07E-01	1.46E-02
	Swel(4,4,low)	25.46	1.43E-02	7.44E-03
	Swel(6,6,low)	29.14	1.33E-02	8.32E-03
Energy	Version 2			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	36.63	4.76E-04	-
1.00E-02	Don(6,0,int)	50.99	3.58E-03	-
	Don(4,0,low)	26.25	2.74E-04	-
	Don(6,0,low)	29.72	2.65E-04	-
	Swel(4,4,int)	40.06	4.25E-03	2.05E-03
	Swel(6,6,int)	53.34	1.53E-02	4.27E-03
	Swel(4,4,low)	31.77	1.90E-03	2.05E-03
	Swel(6,6,low)	35.43	1.44E-03	2.21E-03
X velocity	Version 2			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	28.13	6.60E-04	-
1.00E-02	Don(6,0,int)	42.73	3.65E-03	-
	Don(4,0,low)	19.10	3.66E-04	-
	Don(6,0,low)	22.67	3.51E-04	-
	Swel(4,4,int)	31.07	4.53E-03	2.12E-03
	Swel(6,6,int)	43.94	4.41E-02	4.64E-03
	Swel(4,4,low)	23.83	2.63E-03	2.11E-03
	Swel(6,6,low)	27.36	1.78E-03	2.20E-03
Y velocity	Version 2			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	29.19	3.37E-02	-
1.00E-02	Don(6,0,int)	44.76	2.76E-01	-
	Don(4,0,low)	20.06	3.37E-02	-
	Don(6,0,low)	23.53	4.00E-02	-
	Swel(4,4,int)	29.38	3.85E-02	3.37E-02
	Swel(6,6,int)	43.09	2.00E-01	2.53E-01
	Swel(4,4,low)	21.37	3.85E-02	3.37E-02
	Swel(6,6,low)	24.87	4.14E-02	3.48E-02

Version 3

Pressure	Version 3			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	37.68	1.16E-03	-
1.00E-02	Don(6,0,int)	51.78	7.82E-04	-
	Don(4,0,low)	29.77	7.85E-04	-
	Don(6,0,low)	33.00	7.47E-04	-
	Swel(4,4,int)	39.59	1.91E-02	9.60E-03
	Swel(6,6,int)	52.49	1.69E-02	5.17E-03
	Swel(4,4,low)	32.06	1.91E-02	9.14E-03
	Swel(6,6,low)	35.06	1.69E-02	5.18E-03
Density	Version 3			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	39.83	9.82E-04	-
1.00E-02	Don(6,0,int)	52.46	9.33E-04	-
	Don(4,0,low)	31.56	9.84E-04	-
	Don(6,0,low)	34.77	9.70E-04	-
	Swel(4,4,int)	39.19	1.43E-02	4.85E-03
	Swel(6,6,int)	51.65	1.29E-02	6.55E-03
	Swel(4,4,low)	31.54	1.43E-02	6.06E-03
	Swel(6,6,low)	34.66	1.29E-02	6.50E-03
Energy	Version 3			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	39.39	2.56E-04	-
1.00E-02	Don(6,0,int)	53.92	2.64E-04	-
	Don(4,0,low)	30.05	2.68E-04	-
	Don(6,0,low)	33.07	2.59E-04	-
	Swel(4,4,int)	41.55	1.74E-03	8.03E-04
	Swel(6,6,int)	54.93	1.87E-03	5.25E-04
	Swel(4,4,low)	33.26	1.74E-03	8.01E-04
	Swel(6,6,low)	36.45	1.87E-03	5.82E-04
X velocity	Version 3			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	33.17	3.56E-04	-
1.00E-02	Don(6,0,int)	46.78	3.69E-04	-
	Don(4,0,low)	25.13	3.56E-04	-
	Don(6,0,low)	28.08	3.43E-04	-
	Swel(4,4,int)	34.19	1.92E-03	9.64E-04
	Swel(6,6,int)	46.47	1.83E-03	7.84E-04
	Swel(4,4,low)	26.97	1.92E-03	9.69E-04
	Swel(6,6,low)	30.17	1.83E-03	7.75E-04
Y velocity	Version 3			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	45.92	1.70E-02	-
1.00E-02	Don(6,0,int)	55.89	1.94E-02	-
	Don(4,0,low)	39.38	1.70E-02	-
	Don(6,0,low)	42.15	1.94E-02	-
	Swel(4,4,int)	42.75	2.52E-02	1.01E-02
	Swel(6,6,int)	53.67	2.49E-02	1.06E-02
	Swel(4,4,low)	35.53	2.52E-02	1.36E-02
	Swel(6,6,low)	37.69	2.48E-02	1.53E-02

A.2.2 3D results

Version 1

Pressure	Version 1			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	36.96	3.84E-03	-
1.00E-02	Don(6,0,int)	62.41	3.83E-02	-
	Don(4,0,low)	12.39	1.25E-03	-
	Don(6,0,low)	15.23	1.20E-03	-
	Swel(4,4,int)	36.52	2.50E-03	2.19E-03
	Swel(6,6,int)	60.43	5.32E-03	1.97E-03
	Swel(4,4,low)	14.40	4.35E-03	4.00E-03
	Swel(6,6,low)	17.51	4.37E-03	3.97E-03
Density	Version 1			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	39.84	2.35E-03	-
1.00E-02	Don(6,0,int)	65.12	2.77E-02	-
	Don(4,0,low)	14.04	8.47E-04	-
	Don(6,0,low)	16.96	8.11E-04	-
	Swel(4,4,int)	39.04	1.39E-03	1.26E-03
	Swel(6,6,int)	62.32	3.65E-03	1.14E-03
	Swel(4,4,low)	15.13	2.69E-03	2.54E-03
	Swel(6,6,low)	18.23	2.71E-03	2.54E-03
Energy	Version 1			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	43.85	5.37E-04	-
1.00E-02	Don(6,0,int)	69.52	1.39E-04	-
	Don(4,0,low)	18.74	2.54E-04	-
	Don(6,0,low)	22.61	2.40E-04	-
	Swel(4,4,int)	43.16	2.88E-04	2.83E-04
	Swel(6,6,int)	66.30	8.13E-04	2.75E-04
	Swel(4,4,low)	19.85	6.78E-04	6.51E-04
	Swel(6,6,low)	23.47	7.35E-04	7.03E-04
X velocity	Version 1			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	33.43	7.47E-04	-
1.00E-02	Don(6,0,int)	62.42	2.21E-04	-
	Don(4,0,low)	10.31	3.30E-04	-
	Don(6,0,low)	12.87	3.23E-04	-
	Swel(4,4,int)	30.47	6.80E-04	6.96E-04
	Swel(6,6,int)	59.36	1.22E-03	5.09E-04
	Swel(4,4,low)	11.35	6.05E-04	5.27E-04
	Swel(6,6,low)	14.05	6.13E-04	5.54E-04
Y velocity	Version 1			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	55.15	1.66E-02	-
1.00E-02	Don(6,0,int)	74.02	1.40E-02	-
	Don(4,0,low)	28.76	3.22E-03	-
	Don(6,0,low)	32.85	3.39E-03	-
	Swel(4,4,int)	54.78	5.99E-03	7.26E-03
	Swel(6,6,int)	73.47	8.16E-03	7.48E-03
	Swel(4,4,low)	30.91	2.02E-02	2.04E-02
	Swel(6,6,low)	34.65	2.34E-02	2.37E-02

Version 2

Pressure	Version 2			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	36.96	3.84E-03	-
1.00E-02	Don(6,0,int)	62.77	7.59E-04	-
	Don(4,0,low)	12.39	1.25E-03	-
	Don(6,0,low)	15.23	1.20E-03	-
	Swel(4,4,int)	36.52	2.50E-03	2.19E-03
	Swel(6,6,int)	60.44	5.31E-03	1.97E-03
	Swel(4,4,low)	14.40	4.35E-03	4.00E-03
	Swel(6,6,low)	17.51	4.37E-03	3.97E-03
Density	Version 2			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	39.84	2.35E-03	-
1.00E-02	Don(6,0,int)	65.12	5.06E-04	-
	Don(4,0,low)	14.04	8.47E-04	-
	Don(6,0,low)	16.96	8.11E-04	-
	Swel(4,4,int)	39.04	1.39E-03	1.26E-03
	Swel(6,6,int)	63.04	3.65E-03	1.14E-03
	Swel(4,4,low)	15.13	2.69E-03	2.54E-03
	Swel(6,6,low)	18.23	2.71E-03	2.54E-03
Energy	Version 2			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	43.98	5.37E-04	-
1.00E-02	Don(6,0,int)	70.29	1.39E-04	-
	Don(4,0,low)	18.74	2.54E-04	-
	Don(6,0,low)	22.61	2.40E-04	-
	Swel(4,4,int)	43.17	2.88E-04	2.83E-04
	Swel(6,6,int)	66.73	8.13E-04	2.75E-04
	Swel(4,4,low)	19.85	6.78E-04	6.51E-04
	Swel(6,6,low)	23.47	7.35E-04	7.03E-04
X velocity	Version 2			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	33.43	7.47E-04	-
1.00E-02	Don(6,0,int)	62.72	2.21E-04	-
	Don(4,0,low)	10.31	3.30E-04	-
	Don(6,0,low)	12.87	3.23E-04	-
	Swel(4,4,int)	30.47	6.80E-04	6.96E-04
	Swel(6,6,int)	59.51	1.22E-03	5.09E-04
	Swel(4,4,low)	11.35	6.05E-04	5.27E-04
	Swel(6,6,low)	14.05	6.13E-04	5.54E-04
Y velocity	Version 2			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	55.15	1.66E-02	-
1.00E-02	Don(6,0,int)	74.02	1.40E-02	-
	Don(4,0,low)	28.76	3.22E-03	-
	Don(6,0,low)	32.85	3.39E-03	-
	Swel(4,4,int)	54.78	5.99E-03	7.26E-03
	Swel(6,6,int)	73.47	8.16E-03	7.48E-03
	Swel(4,4,low)	30.91	2.02E-02	2.04E-02
	Swel(6,6,low)	34.65	2.34E-02	2.37E-02

Version 3

Pressure	Version 3			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	37.29	1.08E-03	-
1.00E-02	Don(6,0,int)	62.48	7.59E-04	-
	Don(4,0,low)	12.49	1.25E-03	-
	Don(6,0,low)	15.29	1.20E-03	-
	Swel(4,4,int)	36.78	2.15E-03	1.46E-03
	Swel(6,6,int)	60.67	5.31E-03	1.97E-03
	Swel(4,4,low)	14.49	4.35E-03	4.00E-03
	Swel(6,6,low)	17.59	4.37E-03	3.97E-03
Density	Version 3			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	40.15	7.48E-04	-
1.00E-02	Don(6,0,int)	65.15	5.06E-04	-
	Don(4,0,low)	14.08	8.47E-04	-
	Don(6,0,low)	17.00	8.11E-04	-
	Swel(4,4,int)	39.20	1.13E-03	8.28E-04
	Swel(6,6,int)	62.35	3.65E-03	1.14E-03
	Swel(4,4,low)	15.19	2.69E-03	2.54E-03
	Swel(6,6,low)	18.29	2.71E-03	2.54E-03
Energy	Version 3			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	43.93	2.14E-04	-
1.00E-02	Don(6,0,int)	69.65	1.39E-04	-
	Don(4,0,low)	18.77	2.54E-04	-
	Don(6,0,low)	22.63	2.40E-04	-
	Swel(4,4,int)	43.25	2.88E-04	2.46E-04
	Swel(6,6,int)	66.32	8.13E-04	2.75E-04
	Swel(4,4,low)	19.87	6.76E-04	6.51E-04
	Swel(6,6,low)	23.48	7.35E-04	7.03E-04
X velocity	Version 3			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	33.47	2.88E-04	-
1.00E-02	Don(6,0,int)	62.62	2.07E-04	-
	Don(4,0,low)	10.34	3.30E-04	-
	Don(6,0,low)	12.89	3.23E-04	-
	Swel(4,4,int)	30.63	3.87E-04	3.32E-04
	Swel(6,6,int)	59.38	1.22E-03	5.09E-04
	Swel(4,4,low)	11.37	6.05E-04	5.27E-04
	Swel(6,6,low)	14.07	6.13E-04	5.54E-04
Y velocity	Version 3			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	55.31	8.15E-03	-
1.00E-02	Don(6,0,int)	74.02	1.07E-02	-
	Don(4,0,low)	28.81	3.22E-03	-
	Don(6,0,low)	32.88	3.35E-03	-
	Swel(4,4,int)	54.90	5.55E-03	6.19E-03
	Swel(6,6,int)	73.47	8.16E-03	7.46E-03
	Swel(4,4,low)	30.94	2.02E-02	2.00E-02
	Swel(6,6,low)	34.67	2.33E-02	2.34E-02

Z velocity 3D

Z velocity	Version 1			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	55.26	1.66E-02	-
1.00E-02	Don(6,0,int)	74.10	1.19E-02	-
	Don(4,0,low)	28.86	3.01E-03	-
	Don(6,0,low)	32.88	3.57E-03	-
	Swel(4,4,int)	54.77	5.69E-03	7.30E-03
	Swel(6,6,int)	73.26	6.99E-03	6.74E-03
	Swel(4,4,low)	30.51	2.02E-02	2.05E-02
	Swel(6,6,low)	34.28	2.33E-02	2.38E-02
	Version 2			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	55.26	1.66E-02	-
1.00E-02	Don(6,0,int)	74.10	1.19E-02	-
	Don(4,0,low)	28.86	3.01E-03	-
	Don(6,0,low)	32.88	3.57E-03	-
	Swel(4,4,int)	54.77	5.69E-03	7.30E-03
	Swel(6,6,int)	73.26	6.99E-03	6.74E-03
	Swel(4,4,low)	30.51	2.02E-02	2.05E-02
	Swel(6,6,low)	34.28	2.33E-02	2.38E-02
	Version 3			
		Sparsity %	Normal inverse	Adapted inverse
Threshold	Don(4,0,int)	55.42	8.52E-03	-
1.00E-02	Don(6,0,int)	74.10	1.18E-02	-
	Don(4,0,low)	28.91	3.07E-03	-
	Don(6,0,low)	32.91	3.53E-03	-
	Swel(4,4,int)	54.88	4.77E-03	6.23E-03
	Swel(6,6,int)	73.26	6.99E-03	6.75E-03
	Swel(4,4,low)	30.54	2.02E-02	2.00E-02
	Swel(6,6,low)	34.29	2.33E-02	2.34E-02