International University Bremen

School of Engineering and Science

Bachelor's Thesis - Mathematics 2005

Stability Analysis of Nonlinear Subdivision

Student: Christian Kühn
Course: Guided Research Mathematics II
Supervisor: Professor Peter Oswald

Stability Analysis of Nonlinear Subdivision

Christian Kühn

May 27, 2005

Abstract

This paper has been written as part of the course "Guided Research Mathematics II" at International University Bremen during the Spring Semester 2005 under the supervision of Professor Peter Oswald. First we give a short introduction to subdivision algorithms and associated notions, which are relevant for the understanding of the investigated questions in this paper. The main question is the stability analysis for subdivision schemes. For linear subdivision this problem has been solved and we are concerned with possible ways, how to deal with stability properties of nonlinear subdivsion schemes. We focus on the case of subdivision in 1D. An explicit criterion for stability of nonlinear subdivision is proven and investigated numerically for a suitable way to apply this result to median subdivision. In addition we propose a way how to linearize median subdivision in a special case.

Contents

1 Introduction

The basic idea of subdivision can be roughly summarized as follows:

Given a (coarse) set of data, a subdivision scheme produces a refined data set by the repeated application of a set of subdivision rules.

Given a set of data points (x_i, y_i) in $\mathbb{R}^2$ and trying to refine this set of points can be related to finding a function f s.t. $f(x_i) = y_i$ for all i, i.e. an interpolation problem. A typical situation would be to fix a regular refinement rule for x_i. For example a dyadic grid would have the following refinement rule:

$$x_{2k}^{j+1} = x_k^j \qquad \text{and} \qquad x_{2k+1}^{j+1} = \frac{x_k^j + x_{k-1}^j}{2} \tag{1}$$

This means at each refinement step j we obtain elements of the refined data $j+1$ by taking midpoints. The lower subscript k signals the ordering of the points in a sequence. Then we only need a rule, how to assign new y-values to newly inserted nodes in x-direction. Formally the framework can be defined as being given a sequence of y-values, which we will denote by $v \in l_\infty(\mathbb{Z})$, where lower subscripts will denote the respective elements of v as above, e.g. v_k is the element in the sequence with index k. Upper subscrips like v^j formally denote the subdivision **level**, i.e. the refined data set after j subdivision steps; v^0 denotes the initial data/sequence. Then one defines a map S:

$$S : l_\infty(\mathbb{Z}) \to l_\infty(\mathbb{Z}) \tag{2}$$

$$Sv^j = v^{j+1} \tag{3}$$

The map S might be **linear** or **nonlinear**. If S is a nonlinear map then it might be expressible as several linear rules depending on the data v. In this case a **data dependent** subdivision rule is an operator valued function S, which associates to each $v \in l_\infty(\mathbb{Z})$ a linear operator $S(v)$. Usually such rules are also called **quasi-linear**. A subdivision scheme is called **interpolatory** if for all j one has $v_k^j \in v^{j+1}$. A scheme, which does not have this property is called **approximating**. The limit of the data refinement process is written as v^∞, which is the following limit:

$$\lim_{j\to\infty} \underbrace{S \circ S \circ S \circ \ldots}_{j-times} v^0 := S^\infty v^0 = v^\infty \tag{4}$$

Note that this framework of a fixed grid in x-direction and an operator S for the sequence of y-values is not the most general case of a subdivision scheme in one dimension. Clearly one could think of refining the x-values as well or dropping the restriction that f is a function and consider the case of curves. Note that in this case one should assume that f will be a 1-manifold. The reader should be aware of the fact that even in these cases the subdivision scheme is most often simply denoted by S, but is then not given in general by a regular refinement of the grid and a map from $l_\infty(\mathbb{Z})$ to $l_\infty(\mathbb{Z})$. In this paper the main focus is on subdivision schemes for the case of functions on regular grids.

Note that whereas in interpolation problems, one e.g. uses splines and looks for a way to insert new **control points** to improve the spline, subdivision is based on the fact that the sequence of control points as such converges to an actual limit curve making the machinery of splines superfluous. Therefore one needs a notion of convergence: Given the inital data/control points v^0 and the subdivision rule S, consider the function f^j, which is the linear interpolation of the points v^j. If there exists a function f such that for any $\epsilon > 0 \quad \exists j_0 \in \mathbb{N}$ s.t. for all $j > j_0$ and any x

$$\left| f(x) - f^j(x) \right| < \epsilon \tag{5}$$

then f^j converges **uniformly**, which implies f being continuous. The smoothness of the limit curve is one of the major properties. One tries to check whether $f \in C^s(\mathbb{R})$ for some s. Note that if $0 < s \leq 1$, then a function is called **Hölder continuous** if there exists a constant $C > 0$ s.t. for all x, y

$$|f(x) - f(y)| < C|x - y|^s \tag{6}$$

In this case s is also called the **Hölder exponent**. Convergence results will only be stated and not proven in this paper as the focus is to derive stability properties of subdivision schemes.

Some more vocabulary and definitions are necessary. Let N be a fixed integer, then a subdivision rule S **reproduces polynomials** of order N if for any polynomial P of degree less or equal to N there exists a polynomial Q of degree at most N s.t. $P-Q$ has degree $N-1$ and one has:

$$Sp = q \qquad \text{where } p_k = P(k) \text{ and } q_k = Q\left(\frac{k}{2}\right) \tag{7}$$

Roughly speaking, this means that the space of polymials of degree up to N is invariant under the subdivision rule. Also one has to recall the **n-th order forward finite difference operator**:

$$\left(\triangle^n v\right)_k = \sum_{m=0}^{n} (-1)^{n-m} \begin{pmatrix} n \\ m \end{pmatrix} v_{k+m} \tag{8}$$

In addition to the notation Δ^k it is common to use D^k for the same operator. In general it has been shown (see [5]) that polynomial reproduction of order N for a linear subdivision scheme implies the existence of an **associated difference scheme** for each order $k = 1, \ldots, N+1$. More precisely, there exist subdivision schemes $S_k : l_\infty(\mathbb{Z}) \to l_\infty(\mathbb{Z})$ such that $\Delta^k(Sv) = S_k\left(\Delta^k v\right)$. We define the usual norm for the space of bounded sequences $v = \{v_k\}_{k\in\mathbb{Z}}$

$$\|v\|_{l_\infty} = \sup_k |v_k| \tag{9}$$

In addition to this notation, if S is a map then $\|.\|_{l_\infty}$ denotes the associated **operator norm**:

$$\|S\|_{l_\infty} = \sup_{x\in l_\infty} \{\|Sx\|_{l_\infty} : \|x\|_{l_\infty} = 1\} \tag{10}$$

Also $\|.\|_{L_\infty}$ is the standard supremum norm on the function space L_∞. Note that all norms in the remaining sections are infinity-norms and since it is clear from the context if an operator norm, a norm on sequences or a norm on functions is used we abbreviate $\|.\|_{L_\infty} = \|.\|$ and $\|.\|_{l_\infty} = \|.\|$.

2 Linear Subdivision

If $S = (s_{ij})$ is a linear map, the analysis of the subdivision scheme simplifies significantly. First one notes that S is a **bi-infinite** matrix as it is an endomorphism on $l_\infty(\mathbb{Z})$. It is reasonable to impose a locality condition on S by requiring for any fixed j:

$$s_{ij} = 0 \qquad \text{for } |c\cdot j - i| \geq L \quad \text{for some (small) finite } L \in \mathbb{N} \tag{11}$$

where c is a constant depending on the subdivision scheme. Based on the **spectral radius** $\rho(S_{local}) = max_i|\lambda_i|$ of S one can then decide if the scheme converges. In this case λ_i denote the eigenvalues of a finite matrix S_{local} since the analysis of convergence can be reduced to a finite neighborhood of one point as subdivision scheme obeys a locality condition as described above. Basically the results of this analysis are that by checking the spectral radius of a local part of S, convergence and stability for a linear subdivision scheme can be verified. For the technical details of the proofs see [3], [14] or [12].

Example 2.1. Consider the 4-point scheme, which is an interpolatory subdivision scheme. It is used for the subdivision of curves or the refinement of 2-D data. One determines new points by considering 4 points and inserting a new point based on the weights:

$$-\frac{1}{16}, \frac{9}{16}, \frac{9}{16}, -\frac{1}{16} \tag{12}$$

Note that in this case both coordinates of points x and y are refined according to the given weights. These coefficients are also sometimes called the **(subdivision) mask**. Since the scheme is interpolatory, one has

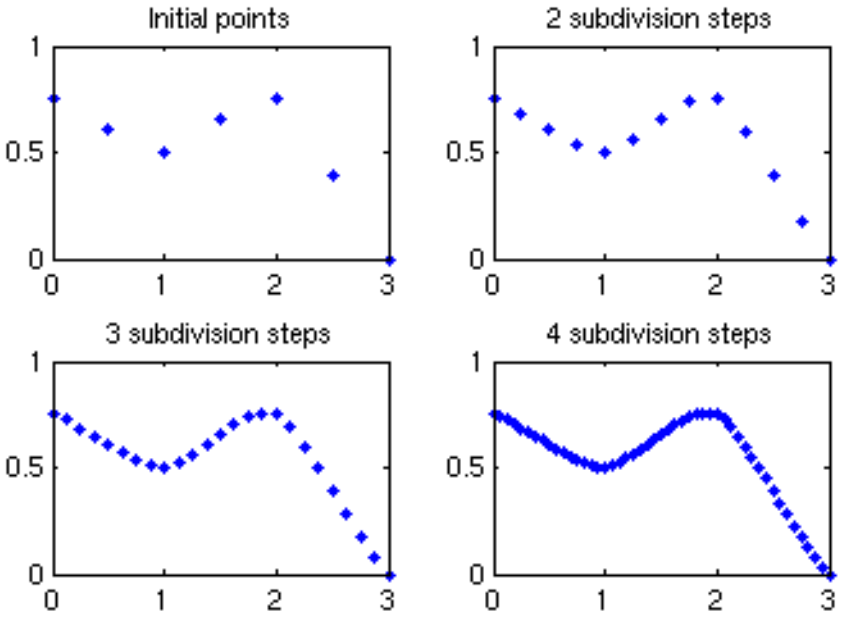

Figure 1: Four-point subdivision for on a dyadically refined grid

$v_k^j = v_{2k}^{j+1}$. This means the points remain as even-indexed points, once they have been fixed. The odd entries are determined by the weights above to yield:

$$v_{2k+1}^{j+1} = -\frac{1}{16}v_{k-1}^j + \frac{9}{16}v_k^j + \frac{9}{16}v_{k+1}^j - \frac{1}{16}v_{k+2}^j \tag{13}$$

Clearly one can write this in matrix form. One also notices that the entries in each row sum to 1, which implies $S1 = 1$. This property is equivalent to polynomial reproduction of constants and to **affine invariance**. Affine invariance simply means that e.g. translating points first and then applying the subdivision rule or first applying the subdivision rule and then translating the points yield the same result. Using these facts together with an analysis of the spectral radius of S_{local} yields that the 4-point scheme converges uniformly to a C^1 limit curve and is stable.

One can also transfer the idea of the four-point scheme to the easier setting of dyadically refining a grid in x-direction and using the subdivision mask for the y-coordinate only. Figure 2 shows an example of this procedure.

If a linear subdivision scheme converges then we have a relation of the form $\|S^j v\| \leq C\|v\|$ for all j with a constant C. For two sequences v and $\tilde{v}$ it follows by linearity of S that:

$$\|v^\infty - \tilde{v}^\infty\| \leq C\|v^0 - \tilde{v}^0\| \tag{14}$$

The relation (14) is the fundamental idea, which motivates this paper. One asks, whether the following holds or not: *'Small' deviations in the intial data induce only 'small' deviations in the limit.*

Note that equation (14) confirms exactly this property for linear subdivision schemes. One should note that linearity is the key to directly conclude that a result for one sequence implies the same relation for two sequences. It turns out that for nonlinear subdivision the stability analysis is more complicated.

3 Quasilinear Subdivision

3.1 Basic Theory

Recall that quasilinear subdivision schemes are based on a family of linear maps $S(v)$, which is dependent on the data v. Note that we can view $S(v)$ either as a linear map on a sequence space or as the result of an operator

associating to a sequence v a linear map $S(v)$. The analysis can be based on similar techniques as for linear schemes and is developed in detail in [1]. Instead of analyzing the spectral radius one now analyzes the **joint spectral radius** $\rho_\infty(S)$ defined as:

$$\rho_\infty(S) = \lim_{j\to\infty} sup \quad sup_{(v^0,\ldots,v^{j-1})\in(l_\infty(\mathbb{Z}))^j} \|S\left(v^{j-1}\right)\ldots S\left(v^0\right)\|^{1/j} \tag{15}$$

In the case of linear subdivision, this is just the usual spectral radius of the matrix S. Furthermore a data-dependent subdivision operator $S(v)$ is called **bounded** if $\exists B > 0$ s.t. $\forall v \in l_\infty(\mathbb{Z})$ one has $\|S(v)\| \leq B$.

The notion of polynomial reproduction carries over in the obvious fashion to require in equation (7) to hold for all $v \in l_\infty(\mathbb{Z})$ and $S(v)$ instead of S. Also one needs to have for many results that the quasilinear scheme is **(Lipschitz) continuously dependent on data**, i.e. $\forall v, w \in l_\infty(\mathbb{Z})$ the subdivision operators $S(v)$ and $S(w)$ satisfy

$$\|S(v) - S(w)\| \leq C\|v - w\| \tag{16}$$

where $C > 0$ depends in a non-increasing way on $max\{\|v\|, \|w\|\}$. Then under the hypothesis of polynomial reproduction and bounds on the joint spectral radii of the associated difference schemes one can derive the convergence of the scheme and an upper bound for the smoothness class of the limit function.

In general, a subdivision rule is called **stable** if $\exists C > 0$ s.t. $\forall v^0, w^0 \in l_\infty(\mathbb{Z})$ the following inequality holds:

$$\|v^\infty - w^\infty\| \leq C\|v^0 - w^0\| \tag{17}$$

where one regards v^∞ and w^∞ as functions with the appropiate supremum norm L_∞.

The following theorems have been proven in [1], here only the result for the smoothness of the limit function will be given. For the stability result we will give the idea of the proof as it is one standard technique, how to prove the stability of a subdivision scheme.

Theorem 3.1. *Let S be a data dependent subdivision rule, which reproduces polynomials up to degree N. If the rule for the differences satisfies $\rho_\infty(S_{n+1}) < 2^{-n}$ for some $n \in \{0,\ldots,N\}$, then the quasilinear subdivision rule based on S is uniformly convergent and the limit function $S^\infty v^0$ is C^s for all $s < \frac{log\rho_\infty(S_{n+1})}{log2}$.*

Note that one has implicitly used a lemma in the statement, which says that polynomial reproduction for quasilinear subdivision also implies existence of associated schemes for the differences as for linear rules. This illustrates that the theory for quasilinear schemes is a generalization for the linear theory. For the stability of quasilinear subdivision schemes one has the following result:

Theorem 3.2. *Let S be a quasilinear subdivision rule, which reproduces constants. Assume that S is continuously dependent on data and that $\rho_\infty(S_1) < 1$. Then for all initial data v^0 and $\tilde{v}_0$ it follows $\|S^\infty v^0 - S^\infty \tilde{v}^0\| \leq C\|v^0 - \tilde{v}^0\|$, where C depends in a continuous non-decreasing way on $max\left\{\|v^0\|, \|\tilde{v}^0\|\right\}$.*

Proof. (sketch) Note that it suffices to prove the theorem for $\|v^j - \tilde{v}^j\|$ with constants uniform in j and then take the limit $j \to \infty$. Define sequences $\alpha^j := \|v^j - \tilde{v}^j\|$ and $\beta^j = \|\Delta v^j - \Delta\tilde{v}^j\|$.

Get an upper bound on the sequences for each j, which will turn out in this case to be $\alpha^j \leq \alpha^{j-1} + D\beta^j$ and $\beta^j \leq C\rho^j\left(\alpha^{j-1} + \ldots + \alpha^0\right)$ where C and D are constants. This is the key step, which is a standard example for the required calculations. The goal is a bound on the behaviour of the differences of two sequences, which are initially 'close-by' in terms of the initial difference plus a term involving the differences 'Δ' with growth limited by a constant ρ^j at the j-th step, which gives a geometric series in the limit and so a fixed bound. This theme will occur again throughout this paper.

Now one can show an inequality for α^j, namely $\|\Delta^n v^j\| \leq 2^{-n}\|\Delta^n v^{j-1}\| + D\|\Delta^{n+1}v^j\|$ for all $n < N$, where N is the polynomial reproduction. This proceeds by showing first that $\left(\Delta^n v^{j-1}\right)_k$ can be expressed as a linear combination of elements of $\Delta^n v^j$, where the coefficients in the linear combination are just from a local (bounded) stencil around the index $2k$. Then one can simply write out the desired estimate and collect terms,

where the finite number of coefficients from the linear combination yields the constant $D > 0$.

The main tools to show the result for the sequences β^j involves using the continuous dependence on data $\|S_1(v) - S_1(\tilde{v})\| \leq C_1\|v - tildev\|$ of the difference scheme and its boundedness, which follow from the same properties for S. Here C_1 is a constant. The main complication is derive from this basic properties a contraction property of the form $\|\Delta^{n+1}v^{j+1} - \Delta^{n+1}\tilde{v}^{j+1}\| \leq \rho^j \|$ "terms involving only v^j" $\|$, where $\rho < 1$ gives a contraction and is related to the subdivision operator, which is particularly easy in this case as one can generalize the spectral radius approach to the joint spectral radius. Now apply a contraction argument for ρ^j as $\rho^j < 1$. □

The proof contains a general strategy, which has turned out to be useful in the analysis of subdivision schemes in general. Having polynomial reproduction one first constructs a scheme for the differences themselves and then uses estimates for each subdivision step to derive a contraction property. This idea is applicable to nonlinear schemes, which are not quasilinear as well. We are going to show this in detail (see Theorem 7.1) as one main result of this paper.

3.2 ENO

This quasilinear method is based on esentially non-osciallatory (ENO) prediction techniques. The ENO technique is a subdivision algorithm for functions. Given the data v^j on some level j as real numbers $v_k^j \in \mathbb{R}$, one considers them as values of a function v on a **dyadic grid** $\Gamma^j = \left(2^{-j}k\right)_{k\in\mathbb{Z}}$; in particular one may write $v\left(2^{-j}k\right) = v_k^j$. The scheme is interpolatory and the points remain as even-indexed $v_{2k}^{j+1} = v_k^j$. Then one associates to each k a so-called **prediction stencil** $S_r(k)$ of length M, where $r \in \mathbb{Z}$ defines the position of the stencil w.r.t. k:

$$S_r(k) = \left\{(k-r)\,2^{-j}, \ldots, (k-r+M-1)\,2^{-j}\right\} \tag{18}$$

Now one uses the function values $(v(\gamma))_{\gamma\in S_r(k)}$ to find by Lagrange interpolation the unique polynomial p_r of degree M, which interpolates v on the stencil points. The odd-indexed points are defined as:

$$v_{2k+1,r}^{j+1} = p_r\left((2k+1)\,2^{-j}\right) \tag{19}$$

One obtains a data-dependent subdivision scheme $S(v)$; note that r can still be chosen for each k. The choice of r is related to the goal of choosing the least osciallatory polynomial p_r in a neighborhood of $\left(2^{-j}k, v_k^j\right)$. This is usually done by a fixed numerical criterion. As an example, the masks for a cubic interpolation polynomial $(M = 4)$ are given below for $r = 0, 1, 2$:

$$\begin{aligned}
v_{2k+1,0}^{j+1} &= -\frac{5}{16}v_k^j + \frac{15}{16}v_{k+1}^j - \frac{5}{16}v_{k+2}^j + \frac{1}{16}v_{k+3}^j \\
v_{2k+1,1}^{j+1} &= -\frac{1}{16}v_{k-1}^j + \frac{9}{16}v_k^j + \frac{9}{16}v_{k+1}^j - \frac{1}{16}v_{k+2}^j \\
v_{2k+1,2}^{j+1} &= -\frac{1}{16}v_{k-2}^j - \frac{5}{16}v_{k-1}^j + \frac{15}{16}v_k^j + \frac{5}{16}v_{k+1}^j
\end{aligned} \tag{20}$$

It is interesting to note that the case $r = 1$ is just the four-point scheme as described above as an example for a linear interpolation scheme. Once the r is fixed for each k one simply drops the r index and has $v_{k,r}^{j+1} = v_k^{j+1}$.

3.3 WENO

The weighted-ENO subdivision builds upon the principles of ENO, but does not only use one of the stencils, but one uses a **convex combination** of the refined values for different stencils:

$$\begin{aligned}
v_k^{j+1} &= \sum_{r=0}^{M-1} \alpha_r v_{k,r}^{j+1} \\
\text{with } \alpha_r \geq 0 \text{ and } &\sum_{r=0}^{M-1} \alpha_r = 1
\end{aligned} \tag{21}$$

A choice for the coefficients α_r is developed in [13]. WENO is introduced since the ENO scheme has undesirable properties with respect to the stability of stencil selection, i.e. a small perturbation in the initial data can cause a different stencil to be selected, which causes the scheme to be no longer continuously dependent of data. This makes the ENO subdivision algorithm unstable as a result.

But for the analysis of WENO the general theory for quasilinear subdivision applies as developed in [1]. In particular the limit function of a WENO subdivision is bounded and belongs to C^s for $s < \frac{log(9/16\sqrt{2})}{log2} \approx 0.6601449$. Stability follows from the fact that the WENO scheme is continuously dependent on data, which is not true for the ENO scheme.

4 A Note on Multiresolution

Having defined subdivision for elements in the sequence space $l_\infty(\mathbb{Z})$ one primary application is the construction of an associated **multiresolution transform**. Assume that we are given v as a continuous function on $\mathbb{R}$.

One first introduces a grid Γ^j of $\mathbb{R}$, where the index j signals that the grid will be refined on different levels, e.g. one could take a dyadic decomposition $\Gamma^j = \left(2^{-j}k\right)_{k\in\mathbb{Z}}$. Then define a **discretization operator**, which evaluates the function on the gridpoints and produces a sequence $v_k^j \in l_\infty(\mathbb{Z})$. Formally write this operator as $\mathcal{D}_j : C(\mathbb{R}) \to l_\infty(\mathbb{Z})$. Now a **reconstruction operator** $\mathcal{R}_j$ is any map, which is a right inverse to $\mathcal{D}_j$, i.e. $\mathcal{D}_j \circ \mathcal{R}_j = Id$.

If one chooses a reconstruction operator $\mathcal{R}_j$ then the composition $\mathcal{D}_{j+1} \circ \mathcal{R}_j = S$ is a subdivision scheme, which computes data at the scale $j+1$. In the setting of multiresolution algorithms this subdivision method is also called a **prediction** operator. The idea of a multiresolution algorithm is to reconstruct the function v from the coarsest data set v^0 and so-called **details** $\left\{d^0, d^1, d^2, ..., d^j\right\}$.

First decompose f using the discretization operator by the procedure of applying D_j to get v^j. Note that the discretization operator and the grid Γ^j cannot be chosen arbitrarily in this case as the subdivision scheme S, which one uses in the following might be defined only a regular grid, e.g. on a dyadic grid. Having obtained v^j the process proceeds iteratively:

1. Define the details at step j as $d^j = Sv^j - v\left(\Gamma^j\right)$. If $j = 0$, stop the decomposition.
2. Consider a coarser grid Γ^{j-1} thereby obtaining a coarser sequence v^{j-1}. Go to 1.

So the result is a set $\left\{v^0, d^0, d^1, d^2, ..., d^j\right\}$, which is sufficient to reconstruct the initial sequence v^j. Note that the subdivision algorithm is crucial for the reconstruction process and one can ask the question, how the stability of the subdivision scheme is related to the stability of the multiresolution algorithm or even more generally if the multiresolution transform is stable. This means whether the following property holds for two different given decompositions $\left\{v^0, d^0, ..., d^j\right\}$ and $\left\{\tilde{v}^0, \tilde{d}^0, ..., \tilde{d}^j\right\}$:

$$\|v^L - \tilde{v}^L\| \leq C_1\|v^0 - \tilde{v}^0\| + C_2 \sum_{j=0}^{L-1} \|d^j - \tilde{d}^j\| \tag{22}$$

where C_1 and C_2 are constants. If this property holds we will call the multiresolution transform **stable**. Note that this property is essential to use the algorithm for pratical applications like compression since instead of the exact decomposition $\left\{v^0, d^0, d^1, d^2, ..., d^j\right\}$ one must work with a slightly perturbed decomposition $\left\{\tilde{v}^0, \tilde{d}^0, ..., \tilde{d}^j\right\}$ anyway due to numerical error and to achieve good compression rates, one also wants to set details $d_k^j = 0$ for all those k, which "don't contribute much". So one needs to be assured that the result of the reconstruction closely resembles the intial function/data.

The question remains, why one should actually bother about the stability of the subdivision algorithm since it is easy to show that for the linear schemes the multiresolution transform is stable. Unfortunately linear multiresolution techniques do not work for all applications like e.g. the removal of non-Gaussian noise, which

motivated the development of a nonlinear multiresolution transform in [4]. Therefore we will give in the following section an overview, which nonlinear schemes are available.

5 Nonlinear subdivision

5.1 Median Interpolation

This subdivision scheme, using the median of a function, was introduced in [4]. Currently the scheme is used for the subdivision of sequences on regular grids. Let f be a real-valued function defined on an interval I. Then the **median** of f on I is denoted $med\,(f|I)$ and defined as

$$med\,(f|I) = arg\min_{m\in\mathbb{R}} \int_I |f\,(t) - m|dt \tag{23}$$

Given initial data v^j, one assumes that the data set consists of median values of a function v on **triadic intervals** $I_{j,k}$

$$med\,(v|I_{j,k}) = v_k^j \tag{24}$$

We fix an even non-negative natural number $D = 2A$. Then for each interval $I_{j,k}$ find a polynomial $p_{j,k}$ of degree D, which 'interpolates the medians'; this means it satisfies

$$med\,(p_{j,k}|I_{j,k+l}) = v_{k+l}^j \qquad \text{for } -A \le l \le A \tag{25}$$

It has been shown in [6] that such a polynomial $p_{j,k}$ exists and is unique. Unfortunately the proof is non-constructive. Despite this fact one has obtained an explicit formula for the case $D = 2$; for the details and the proof of this formula see [4]. Just suppose from now on that one has found the polynomial $p_{j,k}$. Then one can proceed to the **imputation**, which gets medians at the finer scale by

$$v_{3k+l}^{j+1} = med\,(p_{j,k}|I_{j,3k+l}) \qquad l = 0, 1, 2 \tag{26}$$

This two-step process defines a nonlinear subdivision scheme S, which is also sometimes denoted as S_{med} in this case. The lower bound for the Hölder exponent s for the limit curve has been the topic of several recent papers ([10],[15]) and it has been shown that the critical Hölder exponent is $s = 1$. All proofs take advantage of the fact that there exist closed form expressions for triadic median subdivision with quadratic polynomials.

We give the details of $S_{med} : l_\infty(\mathbb{Z}) \to l_\infty(\mathbb{Z})$ in this special case by its action on triples of points (m_1, m_2, m_3), which represent medians on the intervals $[0, 1]$, $[1, 2]$ and $[2, 3]$. Let $Q : \mathbb{R}^3 \to \mathbb{R}^3$ denote the map which maps (m_1, m_2, m_3) to three new medians $(m_1^{(1)}, m_2^{(1)}, m_3^{(1)})$, which are defined on the intervals $[1, 4/3]$, $[4/3, 5/3]$ and $[5/3, 2]$ respectively. To see that this map Q covers all cases we need two properties of median subdivision. First, triadic median interpolation with quadratic polynomilas is **reversal equivariant**, which means that $Q(m_1, m_2, m_3) = -rev(Q(-m_1, -m_2, -m_3))$ where we define $rev(a, b, c) := (c, b, a)$. Second, it also is **affine invariant**, which means that for all $a, b \in \mathbb{R}$ we have $Q(a + bm_1, a + bm_2, a + bm_3) = a + bQ(m_1, m_2, m_3)$. So by translation and scaling invariance Q defines all cases. To see that those two properties hold obeserve that median interpolation by a quadratic polynomial is a symmetric operation around the middle interval and for affine invariance note that scaling all medians by b will scale all coefficients of the polynomial interpolating the medians by the same factor. Translation invariance is obivous from the definition of median subdivision.

The explicit formulas for Q can be given as follows: Set $d = \frac{m_3 - m_2}{m_2 - m_1}$ then by using reversal equivariance we can assume $m_2 - m_1 \neq 0$ since if $m_2 - m_1 = 0$ we can exchange m_1 and m_3. If $m_3 - m_2 = 0$ as well then clearly $Q(m_1, m_2, m_3) = (m_1, m_1, m_1)$. Hence we obtain the main representation formula:

$$\begin{aligned} Q(m_1, m_2, m_3) &= (m_1, m_1, m_1) + (m_2 - m_1)Q(0, 1, 1 + \frac{m_3 - m_2}{m_2 - m_1}) \\ &:= (m_1, m_1, m_1) + (m_2 - m_1)(q_1(d), q_2(d), q_3(d)) \end{aligned} \tag{27}$$

where the relevent coordinate functions $q_i(d)$ turn out to be (see [4]):

$$q_1(d) = \begin{cases} \frac{59}{27} + \frac{7}{27}d - \frac{8}{27}\sqrt{16+16d+d^2} & \text{if } d \in [\frac{7}{3}, 5] \\ \frac{26}{27} + \frac{16}{27}d - \frac{4}{27}\sqrt{1+16d+16d^2} & \text{if } d \in [\frac{1}{5}, \frac{3}{7}] \\ \frac{77}{135} + \frac{13}{135}d + \frac{8}{135}\sqrt{1-62d+d^2} & \text{if } d \in [-3, -\frac{1}{3}] \\ \frac{1}{288}\frac{323-214d+35d^2}{1-d} & \text{if } d \in [-11, -3] \\ \frac{7}{9} - \frac{d}{9} & \text{otherwise} \end{cases}$$

$$q_2(d) = \begin{cases} -\frac{1}{270}\frac{1097-1174d+17d^2+(278-8d)\sqrt{1-62d+d^2}}{-4+4d-\sqrt{1-62d+d^2}} & \text{if } d \in [-\frac{10}{7}, -\frac{7}{10}] \\ \frac{23}{30} + \frac{7}{30}d + \frac{1}{15}\sqrt{1-62d+d^2} & \text{if } d \in [-3, -\frac{1}{3}] \setminus [-\frac{10}{7}, -\frac{7}{10}] \\ 1 & \text{otherwise} \end{cases} \tag{28}$$

$$q_3(d) = \begin{cases} 1 + d - dq_1(\frac{1}{d}) & \text{if } d \neq 0 \\ \frac{10}{9} & \text{if } d = 0 \end{cases}$$

Although convergence properties have been extensively investigated in this case and also an explicit formula is available, it is not known if the scheme is stable. It has been conjectured by several researchers in the field that it is stable (e.g. [4]). It is excepted that for many other nonlinear subdivision algorithms similar problems with respect to proving stability arise.

5.2 Continuous M-Estimators

We now generalize the concept of median subdivision by using **M-estimators**. Let f be a real-valued function defined on the interval I, then the general M-estimator is defined as

$$m(f; I, \varphi) = arg \min_{m \in \mathbb{R}} \int_I \varphi(f(t) - m)\, dt \tag{29}$$

where φ can be chosen as any univariate **convex** function with a unique minimum. Recall that a real-valued function φ is called convex on I if $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y)$ whenever $x, y \in I$ and $0 < \lambda < 1$. The other parts of the subdivision with medians carry over directly. It is instructive to list a few special cases of M-estimators, i.e. of the function φ.

- $\varphi(t) = |t|$. This leads again to the median, namely $med(f; I) = arg \min_{m \in \mathbb{R}} \int_I |f(t) - m| dt$
- $\varphi(t) = \sqrt{t^2 + \epsilon^2} - \epsilon$, for some $\epsilon > 0$. So one uses a C^∞ approximation to the absolute value function. Also one notes that φ has positive second derivative.
- $\varphi(t) = t^2$. This simply gives the average value $ave(f; I) = \frac{1}{meas(I)} \int_I f(t)\, dt$. The average value is known to yield a linear subdivision scheme.
- $\varphi(t) = |t|^p$ for some $p \in (1, \infty)$. This estimator is also sometimes called the **p-mean** of f on I.

The generalization to M-Estimators has been grown out of the general problem for median interpolation, i.e. to find a polynomial, which "interpolates the medians". This can be formalized as trying to find the inverse map M^{-1} to:

$$\begin{aligned} & & M : \Pi_n \to \mathbb{R}^{n+1} \\ p \in \Pi_n & & M(p) = (med(p, I_i))_{i=0}^n \end{aligned} \tag{30}$$

where Π_n denotes the space of polynomials of degree less or equal to n. In general the inversion map M^{-1} is known to be a homeomorphism (see [2]). To find the inverse numerically the second example for an M-estimator introduced above is especially interesting since the smoothness of $\varphi(t) = \sqrt{t^2 + \epsilon^2} - \epsilon$ guarantees that M is actually a diffeomorphism and one can use e.g. a Newton's method to solve equations of the type (25). For more details on results of this analysis see [11]. Nevertheless there are still many open problems related to the theoretical properties and implementation issues for M-estimator based subdivision including the stability analysis for all cases.

5.3 PPH

The PPH subdivision algorithm is based on a refinement of data on a dyadic grid $\Gamma^j = \left(2^{-j}k\right)_{k\in\mathbb{Z}}$. It is based on piecewise polynomial interpolation of degree 3, where instead of an arithmetic mean a harmonic mean is used. The scheme is interpolatory and one tries to find at each level j for two neighboring points in the grid x_k and x_{k+1} the value at the midpoint $x_{k+1/2} = (x_k + x_{k+1})/2$, which we will call v^j_{2k+1}. As usual for schemes based on polynomial interpolation one defines v^j_{2k+1} as the value of a polynomial P^j_k at $x_{k+1/2}$. In this case P^j_k is given as the unique polynomial of degree 3 satisfying:

$$P^j_k(x_t) = \begin{cases} v^j_t & if\ t = k-1, k, k+1 \\ \tilde{v}^j_t & if\ t = k+2 \end{cases} \tag{31}$$

where $\tilde{v}^j_{k+2}$ is defined as

$$\tilde{v}^j_{k+2} := v^j_{k+1} + v^j_k - v^j_{k-1} + 2H\left(\left(\Delta^2 v^j\right)_k, \left(\Delta^2 v^j\right)_{k+1}\right) \tag{32}$$

$$H(x,y) = \frac{xy}{x+y}\left(sgn(xy) + 1\right) \tag{33}$$

In this case the sgn function is defined as $sgn(x) = 1$ if $x \geq 0$ and $sgn(x) = -1$ for $x < 0$. This means the function H is well defined and continuous on $\mathbb{R}^2 \backslash \{(0,0)\}$. But by noting that for the limit as $(x,y) \to (0,0)$ one has $H(x,y) \to 0$ the function H can be extended to a continuous function on $\mathbb{R}^2$ by setting $H(0,0) = 0$.

Also note that we can view the PPH subdivision as a perturbation of the linear subdivision process originating from Lagrange interpolation of order 3. In particular the following explicit formula holds:

$$v^{j+1}_{2k+1} = \frac{v^j_k + v^j_{k+1}}{2} + \frac{1}{8}H\left(\Delta^2 v^j_{k+1}, \Delta^2 v^j_k\right) \tag{34}$$

This shows again the general pattern that many (though not all) nonlinear schemes can be viewed as a perturbation of linear schemes with a nonlinear term, which in this specific case is given by $H\left(\Delta^2 v^j_{k+1}, \Delta^2 v^j_k\right)$. Stability for the associated multiresolution algorithm for PPH subdivision, which is constructed as given in previous section on multiresolution, has recently been proved in [2]. An implementation of PPH in MatLab can be found in the Appendix A.1.

5.4 Normal subdivision

In general normal subdivision applies to curves. In addition to the initial data v^0 (or in general v^j for some j) one needs to know the curve Γ, which one wants to approximate in advance since the normal subdivision scheme is only part of a multiresolution algorithm. This separates normal subdivision from many other subdivision schemes. For more details on the multiresolution analysis one might consider [8], where normal subdivision is developed in detail.

The scheme is interpolating and one has $v^{j+1}_{2k} = v^j_k$. First use a subdivision scheme S, e.g. midpoint subdivision $\tilde{v}^{j+1}_{2k+1} = \left(v^j_k + v^j_{k+1}\right)/2$, to construct a so-called **base point** $\tilde{v}^{j+1}_{2k+1}$. Then join the two points v^j_k and v^j_{k+1} by a line. Construct the line l normal/orthogonal to the line just constructed so that l contains the basepoint $\tilde{v}^{j+1}_{2k+1}$. Then for many initial choices of subdivision algorithms S it is guaranteed that l intersects Γ in at least one point. Choosing any one of these intersection points, we declare it as the new value v^{j+1}_{2k+1}. Figure 2 illustrates the process of normal subdivision graphically.

Many basic properties of this scheme are proved in [8], where the authors study the multiresolution algorithm associated to the normal subdivision scheme.

5.5 Manifold-valued subdivision

Instead of trying to find a multiresolution transform for a function $f : \mathbb{R} \to \mathbb{R}$, one is interested in the generalization to a function $p : \mathbb{R} \to M$, where M is a manifold. For the following construction, one assumes that

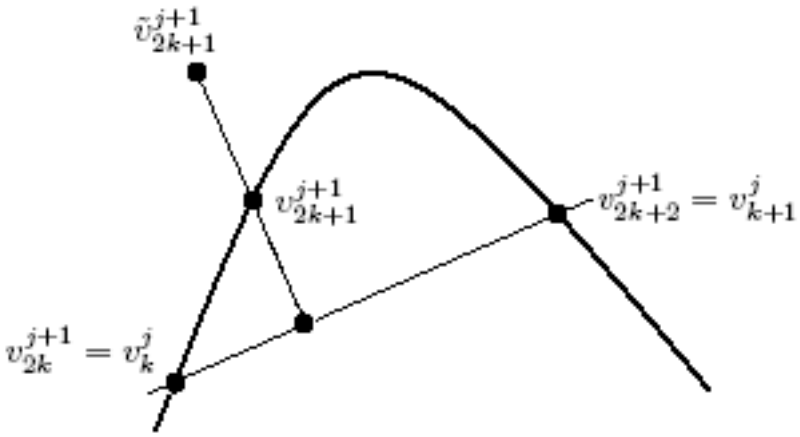

Figure 2: Notation and construction for normal subdivision for curves

M is a Riemannian manifold of dimension d. At each $p_0 \in M$ one can define the corresponding tangent space $T_{p_0}(M)$. Elements in the tangent space will be denoted by θ. Again we work for the refinement process on a dyadic grid. The following description is based on the scheme proposed in [7].

Assume that the function p is given on a coarse grid at level j, i.e. p is known at points $p_k^j = p\left(k/2^j\right)$ for $k \in \mathbb{Z}$. Now fix an odd natural number D. Consider the $D+1$ nearest grid points to p_k^j. Then use the map $Log_{p_k^j} : M \to T_{p_k^j}(M)$ to map these points into the tangent space at p_k^j; here the 'Log' is the usual inverse to the canonically defined expontential map 'Exp', which is a map from M to $T_{p_k^j}(M)$ and maps lines through the origin of the tangent space to geodesics on the manifold. In particular this yields:

$$\theta(l) = Log_{p_k^j}(p(l)) \qquad l = -\frac{D-1}{2^j}, \ldots, \frac{D+1}{2^j} \tag{35}$$

Since the results $\theta(l)$ are elements of the tangent space one can simply treat them as vectors as the tangent space is a vector space of dimension d. So chose a basis for $T_{p_k^j}(M)$, say $\{e_i\}_{i=1}^d$. Then one obtains coordinates/coordinate maps $(\tau_1, \ldots, \tau_d)$ such that:

$$\theta(l) = \sum_{i=1}^{d} \tau_i(l)\, e_i \tag{36}$$

Since there is no way to directly refine the data on the manifold, we construct a special presentation of data in the tangent space, which is defined by coordinate maps $\tau_i : T_{p_k^j} \to \mathbb{R}$. By noting that the family of maps consisting of the compositions $\tau_i \circ p$ are maps from $\mathbb{R}$ to $\mathbb{R}$ we use a refinement scheme on the following real-valued sequence of data on a dyadic grid:

$$(\tau_i(l))_{l=-(D-1)/2^j}^{(D+1)/2^j} \tag{37}$$

The authors of [7] propose to use a scheme of Deslauries-Dubuc. The Deslauries-Dubuc scheme is a linear interpolation scheme, which uses the given data points on the dyadic grid to fit a polynomial π to the points by Lagrange interpolation. Then one simply evaluates this polynomial π on the finer grid at scale $1/2^{j+1}$ to get the refined values. Both schemes proposed in [7] for manifold data are interpolatory in the usual sense for dyadic grids (here: $p_{2k}^{j+1} = p_k^j$).

The values for each coordinate $\tilde{\tau}_i\left(\frac{k}{2^j} + \frac{1}{2^{j+1}}\right)$ obtained by subdivision of the sequence in (37) now give an imputed vector:

$$\tilde{\theta}\left(\frac{k}{2^j} + \frac{1}{2^{j+1}}\right) = \sum_{i=1}^{d} \tilde{\tau}_i\left(\frac{k}{2^j} + \frac{1}{2^{j+1}}\right) e_i \tag{38}$$

Using the exponential map to transfer the points back from the tangent space to the manifold, the refined data is as follows:

$$p_{2k+1}^{j+1} = p\left(\frac{k}{2^j} + \frac{1}{2^{j+1}}\right) = Exp_{p_k^j}\left(\tilde{\theta}\left(\frac{k}{2^j} + \frac{1}{2^{j+1}}\right)\right) \tag{39}$$

The subdivision scheme is in general non-linear for many manifolds as the exponential and logarithm maps are nonlinear. Note that there are several aspects, which make the subdivision scheme probably not well-defined for all Riemannian manifolds and any initial data set. First of all the exponential and logarithm maps are only defined locally at each point and the domain, on which they are injective, might simply not be big enough to map all chosen points in the neighborhood of p_k^j to the tangent plane in an injective way. Also the choice of coordinates in the tangent space might influence the behaviour of the subdivision algorithm. Basically almost all theoretical questions about manifold-valued subdivision proposed here are currently open including the possible problems mentioned.

Another approach proposed in [7] for the subdivision scheme is an analog to average interpolation (discussed as a special case of continuous M-estimators above). This approach basically replaces the Deslauries-Dubuc scheme with average interpolation for the coordinates in equation (37).

6 An Example for Non-Linearizability

It might be appealing to try to fit the nonlinear schemes presented in the previous section into the framework of quasilinear schemes and then just benefit from the results about Hölder continuity and stability, which have been shown in [1]. In following we will show that this is not always directly possible without losing hypotheses needed for the convergence and stability results to hold. In particular we focus on the stability properties of a possible linearization.

As an example consider the interpolatory PPH subdivision S_{PPH} as given in the previous section with the explicit formula for the refined data v^{j+1} at step $j+1$:

$$v_{2k+1}^{j+1} = \frac{v_k^j + v_{k+1}^j}{2} + \frac{1}{8} H\left(\left(\Delta^2 v^j\right)_{k+1}, \left(\Delta^2 v^j\right)_k\right) \tag{40}$$

$$H(x,y) = \frac{xy}{x+y}\left(sgn(xy) + 1\right) \tag{41}$$

Then suppose there exists a matrix S s.t. $Sv^j = v^{j+1}$ as well. First observe that it is reasonable to assume that S should obey the same locality property as S_{PPH}, namely since v_{2k+1}^j only depends on the four neighboring values $v_{k+2}^j, v_{k+1}^j, v_k^j$ and v_{k-1}^j the same locality should hold for S as well. Since the scheme is interpolatory and $v_{2k}^{j+1} = v_k^j$ the first requirement on S is:

$$s_{2l,k} = 1 \qquad \forall l \in \mathbb{Z} \tag{42}$$

The second requirement, arising from equation (40), concerns the odd-indexed rows $2l+1$. To simplify the notation we only consider the case for v_1^{j+1} here, which depends on the neighboring points v_{-1}^j, v_0^j, v_1^j and v_2^j and we will also drop the superscript j, then one requires:

$$\sum_{k=-1}^{2} s_{1,k} v_k = \frac{v_0 + v_1}{2} + \frac{1}{8} H\left(\left(\Delta^2 v\right)_1, \left(\Delta^2 v\right)_0\right) \tag{43}$$

$$= \frac{v_0 + v_1}{2} + \frac{1}{8}\frac{(v_2 - 2v_1 + v_0)(v_1 - 2v_0 + v_{-1})}{v_2 - v_1 - v_0 + v_{-1}}\left(sgn\left((v_2 - 2v_1 + v_0)(v_1 - 2v_0 + v_{-1})\right) + 1\right) \tag{44}$$

If $sgn\left((v_2 - 2v_1 + v_0)(v_1 - 2v_0 + v_{-1})\right) = -1$, the equation is easily solved to yield $s_1 = s_0 = 1/2$ and $s_{-1} = s_2 = 0$ and the resulting matrix S would be just the matrix of a linear scheme originating from taking averages of two neighboring points, so the interesting case occurs when $sgn\left((v_2 - 2v_1 + v_0)(v_1 - 2v_0 + v_{-1})\right) = 1$, this yields:

$$\sum_{k=-1}^{2} s_{1,k} v_k = \frac{v_0 + v_1}{2} + \frac{1}{4}\frac{(v_2 - 2v_1 + v_0)(v_1 - 2v_0 + v_{-1})}{v_2 - v_1 - v_0 + v_{-1}} \tag{45}$$

One can now simply attempt to solve this equation algebraically for one of the unknown elements of the matrix $s_{1,k}$, this gives the following result for s_{-1}:

$$\begin{aligned} s_{-1} = \frac{-1}{4v_{-1}\left(v_{-1}-v_0-v_1+v_2\right)}(&-3v_{-1}v_0+4s_0v_{-1}v_0+4v_0^2-4s_0v_0^2+4s_1v_{-1}v_1-v_0v_1 \\ &-4s_0v_0v_1-4s_1v_0v_1+4v_1^2-4s_1v_1^2-v_{-1}v_2+4s_2v_{-1}v_2 \\ &+4s_0v_0v_2-4s_2v_0v_2-3v_1v_2+4s_1v_1v_2-4s_2v_1v_2+4s_2v_2^2) \end{aligned} \tag{46}$$

Similar expressions are obtained for s_0, s_1 and s_2. Suppose that all s_i are continuous functions of the unknowns v_{-1}, v_0, v_1 and v_2. Then set

$$v_{-1}=1 \qquad v_0=0 \qquad v_1=1 \tag{47}$$

$$\Rightarrow s_{-1}\left(1,0,1,v_2\right) = \frac{-1+v_2-s_1v_2-s_2v_2^2}{v_2} \tag{48}$$

$$= -\frac{1}{v_2}+1-s_1-s_2v_2 \tag{49}$$

Since s_1 and s_2 are continuous they have (bounded) limits for $v_2 \to 0$. Hence one has for the limit of s_{-1}:

$$\lim_{v_2\to 0} s_2\left(1,0,1,v_2\right) = \lim_{v_2\to 0}\left(-\frac{1}{v_2}+1-s_1-s_2v_2\right) \tag{50}$$

$$\lim_{v_2\to 0} -\frac{1}{v_2}+C \tag{51}$$

where C is some constant. This contradicts the assumption that s_{-1} is continuous. In particular it can obviously be not Lipschitz continuous either. Hence one cannot say anything about the stability properties for the PPH scheme just using the quasilinear theory as one needs Lipschitz dependence on data, i.e. $\forall v, w \in l_\infty(\mathbb{Z})$ the subdivision operators $S(v)$ and $S(w)$ should satisfy

$$\|S(v)-S(w)\| \le C\|v-w\| \tag{52}$$

where $C>0$ depends in a non-increasing way on $max\{\|v\|,\|w\|\}$. This is impossible for the function given in (46) as it is not Lipschitz for all v. This shows the need to find different ways to check for the stability of a nonlinear subdivision scheme and in the next section we prove a result, which can be used to check whether a nonlinear scheme is stable.

7 A Sufficient Condition for Stability

The following result is a generalization of a technique used by Amat and Liandrat in [2] to prove that PPH subdivision is stable. Let S be a (nonlinear) subdivision scheme, which acts on the space l_∞. Let $v^j \in l_\infty(\mathbb{Z})$, then $v^{j+1} := Sv^j$. Decompose S into the 'sum' of a linear subdivision scheme S_l and a nonlinear subdivision scheme S_{nl}, in particular $v^{j+1} = Sv^j = S_lv^j + S_{nl}v^j$. Note that this is always possible as we can take $S_l = 0$, but it is desirable to treat the 'easy' linear part seperately if possible. We remind the reader that all norms, which are used in the following are norms in $l_\infty(\mathbb{Z})$, i.e. $\|.\| = \|.\|_{l_\infty}$. Let $d\left(v^j\right) = d^j$ denote the details at level j obtained from the multiresolution transform.

Theorem 7.1. *Let $\left\{v^0, d^1, \ldots, d^{L-1}\right\}$ and $\left\{\tilde{v}^0, \tilde{d}^1, \ldots, \tilde{d}^{L-1}\right\}$ be two multiresolution decompositions for $v^L, \tilde{v}^L \in l_\infty(\mathbb{Z})$. Let M denote the reconstruction operator for the multiresolution defined by $v^{j+1} = Mv^j = Sv^j + d^j$. Assume further that $\|S_l\| \le 1$. Also for some constant C_0 and a finite k and all j:*

$$\|S_{nl}v^j - S_{nl}\tilde{v}^j\| \le C_0\|\Delta^k\left(v^j-\tilde{v}^j\right)\| \tag{53}$$

In addition, suppose there exists an n-step contraction property for the k-th differences of the subdivision scheme

$$\|\Delta^k\left(v^{j+n}-\tilde{v}^{j+n}\right)\| \le p\|\Delta^k\left(v^j-\tilde{v}^j\right)\| + C_1\sum_{i=1}^{n-1}\|\Delta^k\left(d^{j+i}-\tilde{d}^{j+i}\right)\| \tag{54}$$

where $p < 1$ and C_1 is a fixed constant. Suppose furthermore there exists also a 1-step bound on the k-th differences:

$$\|\Delta^k\left(v^{j+1} - \tilde{v}^{j+1}\right)\| \leq K\left(\|\Delta^k\left(v^j - \tilde{v}^j\right)\| + \|\Delta^k\left(d^j - \tilde{d}^j\right)\|\right) \tag{55}$$

where K is a constant. Then one has the following inequality

$$\|v^L - \tilde{v}^L\| \leq C_2\|v^0 - \tilde{v}^0\| + C_3\sum_{j=0}^{L-1}\|d^j - \tilde{d}^j\| \tag{56}$$

with constants C_2, C_3. In particular the multiresolution M associated to the subdivision scheme S is stable.

Proof. By splitting S into a linear and nonlinear part and using the assumptions that the operator norm of the linear part is bounded by 1 and that equation (53) holds, it follows that:

$$\begin{aligned}\|v^{j+1} - \tilde{v}^{j+1}\| &= \|Mv^j - M\tilde{v}^j\| \\ &\leq \|S_l v^j - S_l\tilde{v}^j\| + \|S_{nl}v^j - S_{nl}\tilde{v}^j\| + \|d^j - \tilde{d}^j\| \\ &\leq \|S_l\|\|v^j - \tilde{v}^j\| + C_0\|\Delta^k\left(v^j - \tilde{v}^j\right)\| + \|d^j - \tilde{d}^j\| \\ &\leq \|v^j - \tilde{v}^j\| + C_0\|\Delta^k\left(v^j - \tilde{v}^j\right)\| + \|d^j - \tilde{d}^j\|\end{aligned} \tag{57}$$

Now one simply recurses the procedure given above for $\|v^L - \tilde{v}^L\|$ to obtain:

$$\begin{aligned}\|v^L - \tilde{v}^L\| &\leq \|v^{L-1} - \tilde{v}^{L-1}\| + C_0\|\Delta^k\left(v^{L-1} - \tilde{v}^{L-1}\right)\| + \|d^{L-1} - \tilde{d}^{L-1}\| \\ &\leq \|v^0 - \tilde{v}^0\| + C_0\underbrace{\sum_{i=1}^{L-1}\|\Delta^k\left(v^i - \tilde{v}^i\right)\|}_{:=A} + \sum_{i=1}^{L-1}\|d^i - \tilde{d}^i\|\end{aligned} \tag{58}$$

Note that this estimate is already in the right form, except that the terms denoted by A have to be estimated. They originate from the nonlinear part of the subdivision scheme. Hence one has to consider the expression A for each value i, in particular:

$$\|\Delta^k\left(v^i - \tilde{v}^i\right)\| \leq p\|\Delta^k\left(v^{i-n} - \tilde{v}^{i-n}\right)\| + C_1\sum_{t=1}^{n}\|\Delta^k\left(d\left(v^{i-t}\right) - d\left(\tilde{v}^{i-t}\right)\right)\| \tag{59}$$

Now we can use the inequality above recursively $\lfloor i/n \rfloor := s$ times.

$$\|\Delta^k\left(v^i - \tilde{v}^i\right)\| \leq p^s\|\Delta^k\left(v^{i-sn} - \tilde{v}^{i-sn}\right)\| + C_1\sum_{r=1}^{s}p^{r-1}\sum_{t=(r-1)n}^{rn}\|\Delta^k\left(d^{i-t} - \tilde{d}^{i-t}\right)\| \tag{60}$$

The remaining steps to reach $\|d^0 - \tilde{d}^0\|$ will then be performed by a one-step estimate on the k-th differences as given in equation (55). Assume without loss of generality that $K = 1$; otherwise one just has to keep track of another fixed constant, which can be absorbed into C_3 anyway. In particular this means

$$\|\Delta^k\left(v^{i-sn} - \tilde{v}^{i-sn}\right)\| \leq \|\Delta^k\left(v^0 - \tilde{v}^0\right)\| + \underbrace{\sum_{t=0}^{i-sn-1}\|\Delta^k\left(d^t - \tilde{d}^t\right)\|}_{R_i} \tag{61}$$

where R_i simply denotes the 'remainder' with respect to the index i. Thus we have:

$$\|\Delta^k\left(v^i - \tilde{v}^i\right)\| \leq p^s\|\Delta^k\left(v^0 - \tilde{v}^0\right)\| + C_1\sum_{r=1}^{s}p^{r-1}\sum_{t=(r-1)n}^{rn}\|\Delta^k\left(d^{i-t} - \tilde{d}^{i-t}\right)\| + p^s R_i \tag{62}$$

Hence we obtain an estimate for the sum A:

$$\begin{aligned} A \leq & \sum_{i=1}^{L-1} \left[p^s \|\Delta^k \left(v^0 - \tilde{v}^0\right)\| + C_1 \sum_{r=1}^{s} p^{r-1} \sum_{t=(r-1)n}^{rn} \|\Delta^k \left(d^{i-t} - \tilde{d}^{i-t}\right)\| + p^s R_i \right] \\ = & \underbrace{\sum_{i=1}^{L-1} \left[p^s \|\Delta^k \left(v^0 - \tilde{v}^0\right)\| \right]}_{B_1} + C_1 \underbrace{\sum_{i=1}^{L-1} \sum_{r=1}^{s} p^{r-1} \sum_{t=(r-1)n}^{rn} \|\Delta^k \left(d^{i-t} - \tilde{d}^{i-t}\right)\|}_{B_2} + \underbrace{\sum_{i=1}^{L-1} p^s R_i}_{:=B_3} \end{aligned} \tag{63}$$

The last term B_3 is clear and just comes from the fact that the recursive use of an n-step property might not terminate at 0, but somewhere between 1 and $n-1$, but since it is a finite sum of details it can be incorporated in the second term on the right hand side in equation (56) as claimed in the proposition. This leaves B_1 and B_2; for B_1 we have:

$$B_1 \leq \sum_{i=1}^{\infty} p^i \|\Delta^k \left(v^0 - \tilde{v}^0\right)\| = \|\Delta^k \left(v^0 - \tilde{v}^0\right)\| \frac{1}{1-p} \leq \|\left(v^0 - \tilde{v}^0\right)\| \frac{1}{1-p} \left(\sum_{m=0}^{k} \begin{pmatrix} k \\ m \end{pmatrix} \right) \tag{64}$$

as $p < 1$ and one gets a convergent geometric series. The last multiplicative term in the result stems from a crude estimate of the finite difference operator:

$$|\Delta^k (\dots)| = |\sum_{m=0}^{k} (-1)^m \begin{pmatrix} k \\ m \end{pmatrix} (\dots)| =\leq \sum_{m=0}^{k} \begin{pmatrix} k \\ m \end{pmatrix} |(\dots)| \tag{65}$$

So the only remaining term is B_2, which has a triple sum, but rearrangement of finite sums is always possible, so after rearranging we again use the fact that one gets a geometric series and then use the estimate for the finite difference operator. This concludes the proof. □

8 Applications of the Theorem

The result proven in section 7 is a generalization of techniques, which have been used to show that PPH subdivision is stable. Recall that there is an explicit formula for each subdivision step (see equation (40)):

$$v_{2k}^{j+1} = v_k^j \tag{66}$$

$$v_{2k+1}^{j+1} = \frac{v_k^j + v_{k+1}^j}{2} + \frac{1}{8} H\left(\left(\Delta^2 v^j\right)_{k+1}, \left(\Delta^2 v^j\right)_k \right) \tag{67}$$

$$H(x,y) = \frac{xy}{x+y} \left(sgn(xy) + 1\right)$$

To prove a property of the type as required by equation (54) one needs to investigate inequalities of the form $\|\Delta^k v^{j+n} - \Delta^k \tilde{v}^{j+n}\| \leq p \|\Delta^k v^j - \Delta^k \tilde{v}^j\|$ with the obvious goal to show for some number of steps n and some difference order k that $p < 1$. For the PPH scheme this can be accomplished for $k = 2$ and $n = 2$ as demonstrated by Amat and Liandrat in [2]. We give all the main steps here for convenience. First define a function $Z(x,y,z) = \frac{x}{2} + \frac{1}{8}(H(x,y) + H(x,z))$, then we obtain:

Lemma 8.1. *The following properties for the functions H and Z hold for all x, y, z, x', y', z':*

1. $|H(x,y)| \leq max\{|x|, |y|\}$
2. $|H(x,y) - H(x',y')| \leq 2 \cdot max\{|x - x'|, |y - y'|\}$
3. $|Z(x,y,z)| \leq \frac{|x|}{2}$
4. $|Z(x,y,z) - Z(x',y',z')| \leq \frac{1}{2}|x - x'| + \frac{1}{2} max\{|y - y'|, |z - z'|\}$

The properties can be verified by direct calculation and it should be remarked that the proof uses explicit formulas, so one cannot hope to obtain a generalized method for other subdivision schemes, but one indeed obtains:

Theorem 8.2 (2-step contraction property for PPH). *Let v and w denote two elements of $l_\infty(\mathbb{Z})$ then the following holds (we set $\|.\|_\infty = \|.\|$ for simplicity of notation):*

1. $\|\Delta^2(Sv)\| \leq \frac{1}{2}\|\Delta^2 v\|$
2. $|\Delta^2((Sv)_j - (Sw)_j)| \leq \frac{1}{2}\|\Delta^2(v-w)\|$ *j odd*
3. $|\Delta^2((Sv)_j - (Sw)_j)| \leq \frac{1}{2}\|\Delta^2(v-w)\|$ *j even*
4. $|\Delta^2((S^2v)_j - (S^2w)_j)| \leq \frac{3}{4}\|\Delta^2(v-w)\|$

Proof. For simplicity denote $Sv = v^{(1)}$ and $S^2v = v^{(2)}$.

1. The goal is to show $|(\Delta^2 v^{(1)})_j| \leq \frac{1}{2}\|\Delta^2 v\|$ for all j. So assume j is odd, say $j = 2n+1$. Then since PPH is interpolatory $\Delta^2 v_j^{(1)} = v_{j+1}^{(1)} - 2v_j^{(1)} + v_{j-1}^{(1)} = v_{n+1} - 2v_j^{(1)} + v_n$. And property 1 of the previous Lemma 8.1 implies $|v_{n+1} - 2v_j^{(1)} + v_n| \leq \frac{1}{4}max\{|\Delta^2 v_{n+1}|, |\Delta^2 v_n|\} \leq \frac{1}{4}\|\Delta^2 v\|$. For $j = 2n$ we obtain instead $v_{j+1}^{(1)} - 2v_j^{(1)} + v_{j-1}^{(1)} = v_{j+1}^{(1)} - 2v_n + v_{j-1}^{(1)}$ and from the definitions of H and Z it follows $v_{j+1}^{(1)} - 2v_n + v_{j-1}^{(1)} = Z(\Delta^2 v_n, \Delta^2 v_{n+1}, \Delta^2 v_{n-1})$. Now apply property 3 of Lemma 8.1.

2. Set $j = 2n+1$. Then $|v_{n+1} - 2v_j^{(1)} + v_n - w_{n+1} + 2w_j^{(1)} - w_n| = \frac{1}{4}|H(\Delta^2 v_{n+1}, \Delta^2 v_n) - H(\Delta^2 w_{n+1}, \Delta^2 w_n)|$ and property 2 of Lemma 8.1 gives the result.

3. Set $j = 2n$. Perform calculation as for the even case in *1.* and apply property 4 of Lemma 8.1.

4. The odd case $j = 2n+1$ is done already due to part *2.* of this theorem so set $j = 2n$. This gives the equation:

$$|\Delta^2(v_j^{(2)} - w_j^{(2)})| = |v_{j+1}^{(2)} - 2v_{2n}^{(2)} + v_{j-1}^{(2)} - (w_{j+1}^{(2)} - 2w_{2n}^{(2)} + w_{j-1}^{(2)})| \tag{68}$$

$$= |Z(\Delta^2 v_n^{(1)}, \Delta^2 v_{n+1}^{(1)}, \Delta^2 v_{n-1}^{(1)}) - Z(\Delta^2 w_n^{(1)}, \Delta^2 w_{n+1}^{(1)}, \Delta^2 w_{n-1}^{(1)})| \tag{69}$$

Obviously either n, $n-1$ or $n+1$ must be odd and hence *2.* of this theorem and property 4 of Lemma 8.1 imply the result. □

Although this proof by Amat and Liandrat (see [2]) is quite short it needs as an essential requirement the special properties of the nonlinear perturbation of the PPH scheme and there is no hope of generalizing this technique to other subdivision algorithms. Since Median-Subdivision is one of the main cases where no stability analysis is known the idea is to check whether there is a possibility to apply Theorem 7.1 to this case as well.

9 Numerical Simulation

The goal is to verify a property of the type given in equation (54) for triadic median subdivision in the easiest with $D = 2$. In particular an inequality of the form:

$$\|\Delta^k v^{j+n} - \Delta^k w^{j+n}\| \leq p\|\Delta^k v^j - \Delta^k w^j\| \tag{70}$$

for $p < 1$ is needed. The first question is obviously, which order of differences k and how many subdivision steps n will suffice to achieve this inequality. Obviously the analytical checking gets quite complicated, which is apparent by looking at equations (27) and (28). Hence we investigated the inequality (70) first numerically. In particular the problem can be formalized as follows for the case:

- Find the smallest possible p such that (70) holds for all initial sequence v and w

Hence we obtain an optimization problem with the number of variables equal to at most twice the size of the (k,n)**-invariant neighborhood for differences** of the subdivision operator, where by an (k,n)-invariant neighborhood of differences we mean the number of elements in v and w, which are required to determine the norm $\|\Delta^k v^{j+n} - \Delta^{k+n} w^j\|$ uniquely. For $k = 2$ and $n = 1$ it is easy to verify that one indeed needs 5 medians for v and w at most. Geometrically this can be related to the fact that 3 medians of v and 3 medians of w on

the same initial intervals, e.g. $[0,1]$,$[1,2]$ and $[2,3]$ cover all differences after one step on $[1,2]$. And 2 additional medians of v and w each defined on $[-1,0]$ and $[3,4]$ respectively cover differences across the intervals $[0,1]$ to $[1,2]$ and $[1,2]$ to $[2,3]$. Note that this is not the minimum number of variables since we have affine invariance (see section 5.1) we can set two values constantly to 0. This gives 8 variables and a well-defined optimization problem, which has been investigated.

Several functions in MatLab have been developed to analyse the behaviour for particular values of k and n. Those functions can be found in the Appendix A.1. First we have tracked the parameter p for random initial sequences; the results for this computation for $k = 1, 2$ and $n = 1, 2$ with 200 random sequences are displayed in figure 3.

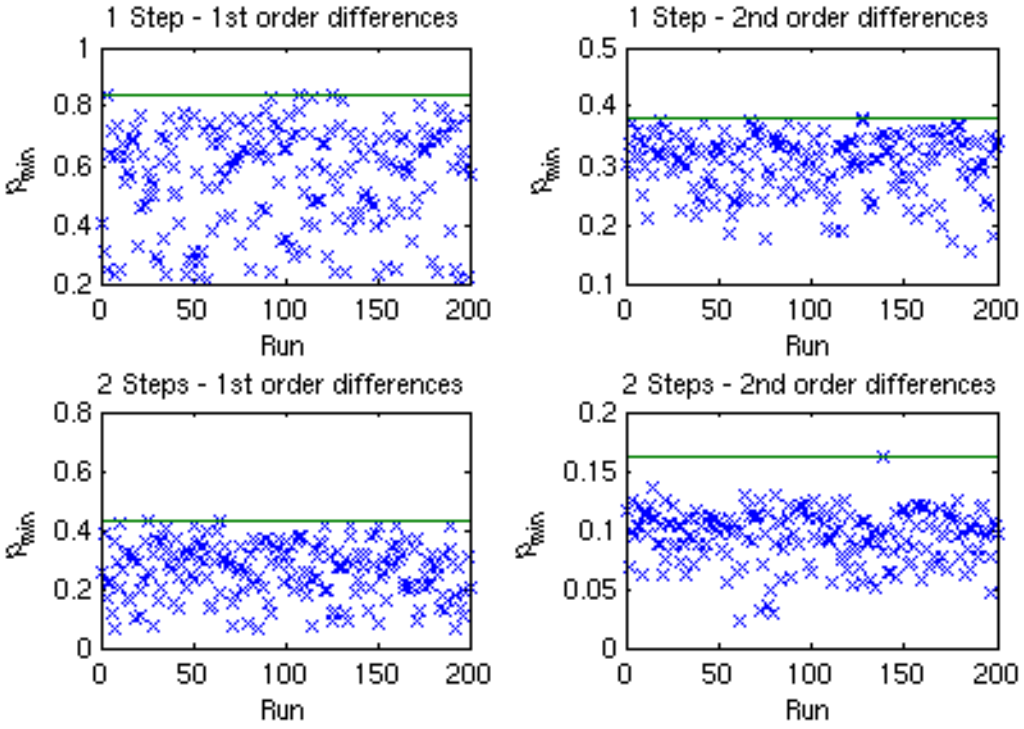

Figure 3: Random Simulation for the 'stability' parameter p

This simulation gives a good indication that it is not hopeless to attempt analyctical calculations. Furthermore a function has been developed to calculate the parameter p based on two distinct input sequences. Then this function has been converted into a suitable I/O format for the MatLab Optimization Toolbox. The toolbox documentation can be found online or is alternatively available in the MatLab Help. We chose to use a Nelder-Mead Simplex Algorithm (see [9]), which is very robust, but has the drawback that it does converge to local minima. Therefore we started at different points/regions in the search space and we also re-started the algorithm once a local minimum has been found several times. Although this method will not turn out a precise answer for the minimum value required for p, it provides some clear indications.

Contrary to the initial guess based on random simulation we have identified with the optimization algorithm two counter-examples, which show that only for one iteration step there is no chance to verify the required property namely for 1 step and first order differences we have:

- v=[0 0 0.25 -0.46 1.39] and w=[0 0 0.23 -0.46 1.37] $\Rightarrow p > 1.01$

Note that for one step and first order differences the $(1,1)$-invariant neighborhood for differences is indeed of size 4. Also for 1 step and second order differences we obtain a counterexample:

- v=[5.11 -1.14 -1.11 1.58 0.6] and w=[5 -1.87 -2.52 -0.44 -2.05] $\Rightarrow p > 1.5$

It is easy to obtain even more counterexamples with the programs provided in Appendix A.1. For $k = 1$ and $n = 2$ no counterexample could be obtained via optimization. So the conjecture is that median interpolation is indeed stable and one needs to take into account two iteration steps. For the smoothness analysis Xie and Yu (see [15]) calculated the dynamics associated to one sequence for two iterations explicitly. This approach is again very specialized and the numerical simulation together with the complicated formulas for two sequences show that for stability analysis new tools and ideas might be needed. Nevertheless, based on the simulation the goal must be to develop methods, which allow analysis of iterations of the subdivision process and yield explicit estimates on differences.

10 Local Linearization

One idea to accomplish the analysis is to consider a "linearized" median interpolation. Note that the direct approach is not always possible as demonstrated for the PPH scheme. Therefore we choose to employ a local linearization to the formulas presented in equations (27) and (28). Using linear interpolation for each of the functions with interpolation points on the boundary of definition we obtain a linear scheme. To demonstrate how this linearization process works recall the defintion of the subdivision operator for median interpolation with quadratic polynomials (we assume $m_2 - m_1 \neq 0$):

$$\begin{aligned} Q(m_1, m_2, m_3) &= (m_1, m_1, m_1) + (m_2 - m_1)Q(0, 1, 1 + \frac{m_3 - m_2}{m_2 - m_1}) \\ &:= (m_1, m_1, m_1) + (m_2 - m_1)(q_1(d), q_2(d), q_3(d)) \\ q_1(d) &= \begin{cases} \frac{59}{27} + \frac{7}{27}d - \frac{8}{27}\sqrt{16 + 16d + d^2} & \text{if } d \in [\frac{7}{3}, 5] \\ \frac{26}{27} + \frac{16}{27}d - \frac{4}{27}\sqrt{1 + 16d + 16d^2} & \text{if } d \in [\frac{1}{5}, \frac{3}{7}] \\ \frac{77}{135} + \frac{13}{135}d + \frac{8}{135}\sqrt{1 - 62d + d^2} & \text{if } d \in [-3, -\frac{1}{3}] \\ \frac{1}{288}\frac{323 - 214d + 35d^2}{1 - d} & \text{if } d \in [-11, -3] \\ \frac{7}{9} - \frac{d}{9} & \text{otherwise} \end{cases} \\ q_2(d) &= \begin{cases} -\frac{1}{270}\frac{1097 - 1174d + 17d^2 + (278 - 8d)\sqrt{1 - 62d + d^2}}{-4 + 4d - \sqrt{1 - 62d + d^2}} & \text{if } d \in [-\frac{10}{7}, -\frac{7}{10}] \\ \frac{23}{30} + \frac{7}{30}d + \frac{1}{15}\sqrt{1 - 62d + d^2} & \text{if } d \in [-3, -\frac{1}{3}] \setminus [-\frac{10}{7}, -\frac{7}{10}] \\ 1 & \text{otherwise} \end{cases} \\ q_3(d) &= \begin{cases} 1 + d - dq_1(\frac{1}{d}) & \text{if } d \neq 0 \\ \frac{10}{9} & \text{if } d = 0 \end{cases} \end{aligned} \tag{71, 72}$$

As an example suppose $d \in [-\frac{10}{7}, -\frac{7}{10}]$. Then evaluating at the endpoints of the interval for the functions q_i gives:

$$q_1(-3) = \frac{10}{9} \qquad q_1(-\frac{1}{3}) = \frac{22}{27} \tag{73}$$

$$q_2(-\frac{10}{7}) = \frac{15}{14} \qquad q_2(-\frac{7}{10}) = \frac{21}{20} \tag{74}$$

Now using Lagrange interpolation for the two endpoints of the interval for d this gives two new functions $r_1(d)$ and $r_2(d)$ defined on $[-3, -\frac{1}{3}]$ and $[-\frac{10}{7}, -\frac{7}{10}]$ namely:

$$r_1(d) = \frac{7}{9} - \frac{d}{9} \tag{75}$$

$$r_2(d) = \frac{35}{34} - \frac{d}{34} \tag{76}$$

Using the usual formula to express q_3 in terms of q_1 we immedeately get for $r_3(d)$:

$$r_3(d) = \frac{10}{9} + \frac{2}{9}d \tag{77}$$

Proceeding the same way for all the other cases and applying linear interpolation on the intervals $[-3, -\frac{10}{7}]$ and $[-\frac{7}{10}, -\frac{1}{3}]$ seperately we obtain a locally linearized version R of the median interpolation given by:

$$\begin{aligned} R(m_1, m_2, m_3) &= (m_1, m_1, m_1) + (m_2 - m_1)R(0, 1, 1 + \frac{m_3 - m_2}{m_2 - m_1}) \qquad (78) \\ &:= (m_1, m_1, m_1) + (m_2 - m_1)(r_1(d), r_2(d), r_3(d)) \\ r_1(d) &= \frac{7}{9} - \frac{d}{9} \qquad (79) \\ r_2(d) &= \begin{cases} \frac{25+d}{22} & \text{if } d \in [-3, -\frac{10}{7}] \\ \frac{35-d}{34} & \text{if } d \in [-\frac{10}{7}, -\frac{7}{10}] \\ \frac{3}{22}(7-d) & \text{if } d \in [-\frac{7}{10}, -\frac{1}{3}] \\ 1 & \text{otherwise} \end{cases} \qquad (80) \\ r_3(d) &= \frac{10}{9} + \frac{2}{9}d \qquad (81) \end{aligned}$$

Note that we now have indeed a linear scheme since the last term for each choice of function r_i is of the form:

$$(m_2 - m_1) \cdot (ad + b) = a(m_2 - m_1) \cdot \frac{m_3 - m_2}{m_2 - m_1} + b(m_2 - m_1) = a(m_3 - m_2) + b(m_2 - m_1) \qquad (82)$$

The most interesting observation of this calculation is that all the cases for the two functions q_1 and q_3 collapse into one linear scheme. Hence one expects the most complicated nonlinear behaviour of the subdivision algorithm in the middle interval of the evaluation of the quadratic function, which has the prescribed medians on 3 intervals. Note that we include in the Appendix A.2 a code in Mathematica 5 format, which makes all the algebraic calculations necessary for the local linearization reproducible. Furthermore this code contains a calculation for the maximum error, which occurs at each step for each interval of the parameter d if one replaces the functions q_i by r_i. This can be done as follows:

1. Choose an interval I for d on which a local linearzation has been applied
2. Calculate $\max_{d \in I} |r_i(d) - q_i(d)| := E_i(d)$

Note that this is the error comitted in the calculation for r_i instead of q_i and note that it gets multiplied by a factor $m_2 - m_1$ according to the definition of the operator R. The following table summarizes the results of the calculations of $E_i(d)$; for readability only the approximate values of the errors are given:

d	$[\frac{7}{3}, 5]$	$[\frac{1}{5}, \frac{3}{7}]$	$[-3, -\frac{10}{7}]$	$[-\frac{10}{7}, -\frac{7}{10}]$	$[-\frac{7}{10}, -\frac{1}{3}]$	$[-11, -3]$
$E_1(d)$	0.016	0.002	0.064	0.064	0.064	0.022
$E_2(d)$	0	0	0.012	0.001	0.006	0

where $E_3(d)$ follows immediately from E_1. Having linearized the median subdivision the hope is that it is much easier to verify by a careful error analysis that the contraction property for differences and two subdivision steps holds. The main reason for this is that one has basically obtained a quasilinear subdivision scheme, which means that an analysis of the joint spectral radius of the scheme and possibly its associated difference schemes can be attempted. The calculations have not been carried out yet.

Other options to verify the necessary contraction property for the differences for the median scheme are obviously direct estimates for the formulas or calculations involving dynamical systems theory as demonstrated by Xie and Yu in the convergence analysis in [15]. Note that both ways are not generic approaches, i.e. they take advantage of the special structure of a subdivision algorithm, whereas the way proposed here is an approach, which might possible turn out to be a more general analysis tool:

1. Linearize the scheme, i.e. replace the nonlinear operator by an appropiate number of linear operators
2. Use techniques from the analysis of quasilinear subdivision methods
3. Use an error analysis to conclude the properties also for the nonlinear scheme

11 Results and Conclusions

The overview of nonlinear subdivision schemes shows that convergence properties have been investigated, but stability analysis turns out to be a problem in many cases. Nevertheless, many ideas have been developed during the last ten years of research. In addition to several concrete examples for nonlinear schemes, especially the analysis of quasilinear schemes, methods for convergence analysis of nonlinear schemes and generalizations of subdivison to manifolds present interesting approaches.

Despite the fact that in many cases generalized versions of tools from linear subdivision can be used, it has been shown that not every scheme has a direct linearization by the example of PPH subdivision (see section 6). A special result for the stability analysis of PPH subdivision has been generalized to apply to any subdivision scheme once an inequality for the k-th order differences after n subdivision steps holds with a sufficiently small constant. In the special case of triadic median subdivision with quadratic polynomials, it has been demonstrated that 1 subdivision step is insufficient and it has been conjectured that two steps are sufficient to prove the desired inequality. To simplify the analysis a local linearization of the median subdivision has been carried out. The verification of properties for this local linearization still remains an open problem.

References

[1] Nira Dyn Albert Cohen and Basarab Matei. Quasilinear subdivision schemes with applications to eno interpolation. *Applied and Computational Harmonic Analysis*, 15:89–116, 2003.

[2] Sergio Amat and Jacques Liandrat. On the stability of the pph nonlinear multiresolution. *preprint*, 2004.

[3] W. Dahmen A.S. Cavaretta and C.A. Michelli. Stationary subdivision. *Memoirs of the American Mathematical Society*, 93:401–423, 1991.

[4] David L. Donoho and Thomas P.-Y. Yu. Nonlinear pyramid transforms based on median interpolation. *SIAM Journal on Mathematical Analysis*, 31(5):1030–1061, 2000.

[5] N. Dyn. *Subdivision schemes in computer aided geometric design*, pages 36–104. Oxford University Press, 1992.

[6] Tim N.T. Goodman and Thomas P.-Y. Yu. Interpolation of medians. *Advances in Computational Mathematics*, 11:1–10, 1999.

[7] Peter Schroeder et. al. Inam Ur Rahman. Multiscale representations for manifold-valued data. *preprint*, 49(1), 2005.

[8] Olof Runborg Ingrid Daubechies and Wim Sweldens. Normal multiresolution approximation of curves. Technical report, Bell Labs, 2002.

[9] J.A. Nelder and R. Mead. A simplex method for function minimization. *Computer J.*, 7:308–313, 1965.

[10] Peter Oswald. Smoothness of nonlinear median-interpolation subdivision. *Advances in Computational Mathematics*, 20:401–423, 2004.

[11] Jong-Shi Pang and Thomas P.-Y. Yu. Continuous m-estimators and their interpolation by polynomials. *SIAM Journal on Numerical Analysis*, 42(3):997–1017, 2004.

[12] Peter Schroeder and Denis Zorin. Course notes - subdivision for modelling and animation. SIGGRAPH, 2000.

[13] X.-D. Liu T. Chan and S. Osher. Weighted essentially non-oscillatory schemes. *Journal of Computational Physics*, pages 200–212, 1994.

[14] Joe Warren and Henrik Weimer. *Subdivision Methods for Geometric Design*. Morgan Kaufmann, 2002.

[15] Gang Xie and Thomas P.-Y. Yu. Smoothness analysis of nonlinear subdivision schemes of homogeneous and affine invariant type. *Constructive Approximation - to appear*, 2004.

A Appendix

A.1 MatLab Functions

The following MatLab functions have been developed to numerically investigate the properties of subdivision schemes and their stability. Especially the median subdivision for the case of quadratic polynomials has been investigated. Each function is documented by the comments to be found in the first few lines of the programming code. The functions have been tested using MatLab Version R14. For an optimization problem we used the MatLab Optimization Toolbox 3.

```
function P=poly_fit_median(intervals,medians,tol)

% Finds a polynomials of degree D for D given medians on given inervals
% Uses fixed-point iteration // adapted version of WaveLab Toolbox

% Input: intervals = the intervals where the medians are defined
%            medians = prescribed median values on the intervals
%            tol = tolerance for the fixed point iteration

% Output: Polynomial in MatLab format that fits medians
% e.g. intervals=[0:1:3] // medians = [0 1 0] // tol=10^(-8)

l=length(intervals);
m=length(medians);
if(m ~= l-1)
    disp('# of intervals and medians does not match!')
end

%Determine midpoins of intervals
for k=1:l-1
    mid(k)= (intervals(k)+intervals(k+1))/2;
end

residual = medians;         %initial residual
P = zeros(1,m);                 %intial guess for polynomial
counter=1;                        %Counter for iteration limit

while ( norm(residual)/norm(medians) > tol & counter < 1000)
% Suppose medians are values of P at midpoints
    predict_poly = polyfit(mid,residual,m-1);
    P = P + predict_poly;
    for i=1:m
        predict_medians(i) = BlockMedian(P,[intervals(i) intervals(i+1)]);
    end
    residual = medians - predict_medians;
    counter= counter+1;
end

if(counter>=1000)
    disp('Warning - limit of # iterations for poly_fit_median reached...')
end
```

```
function p=opt_prefactor(x)

%Suitable version for optimization toolbox of 'prefactor.m'

%Insert different data here:
v=x(1:5);
w=x(6:10);
k=2; % steps
l=1; % differences
method='sub_median2';

%Requires v,w to be big enough to compute the desired D^{l}

%Calculates the prefactor 'p' in the estimate necessary for stability
%  ||D^{l}v^{k}-D^{l}w^{k}|| <= p ||D^{l}v^{0}-D^{l}w^{0}||

%Iterate the given sequences for k steps
if(strcmp(method,'sub_pph'))
    v_k=sub_pph(v,k);
    w_k=sub_pph(w,k);
elseif(strcmp(method,'sub_median'))
    v_k=sub_median(v,k);
    w_k=sub_median(w,k);
elseif(strcmp(method,'sub_median1'))
    v_k=sub_median1(v,k);
    w_k=sub_median1(w,k);
elseif(strcmp(method,'sub_median2'))
    v_k=sub_median2(v,k);
    w_k=sub_median2(w,k);
else
    disp('Last input argument must be a valid string for a method');
end

%Apply the l-th order difference operator
Dv_k=v_k;
Dw_k=w_k;
Dv=v;
Dw=w;
for i=1:l
    Dv_k=diff(Dv_k);
    Dw_k=diff(Dw_k);
    Dv=diff(Dv);
    Dw=diff(Dw);
end

RHS=max(abs(Dv-Dw));
LHS=max(abs(Dv_k-Dw_k));
p=LHS/RHS;

%for maximizing give the negative
p=-p;
```

```
function p=prefactor(v,w,k,l,method)

%Calculates the prefactor 'p' in the estimate necessary for stability
%  ||D^{l}v^{k}-D^{l}w^{k}|| <= p ||D^{l}v^{0}-D^{l}w^{0}||

%Input: v = intital sequence
%           w= other intital sequence
%           k = number of steps for the subdivision
%           l = order of the differences
%           method = (string) giving the subdivision algorithm

%Requires v,w to be big enough to compute the desired D^{l}

%Iterate the given sequences for k steps
if(strcmp(method,'sub_pph'))
    v_k=sub_pph(v,k);
    w_k=sub_pph(w,k);
elseif(strcmp(method,'sub_median'))
    v_k=sub_median(v,k);
    w_k=sub_median(w,k);
elseif(strcmp(method,'sub_median1'))
    v_k=sub_median1(v,k);
    w_k=sub_median1(w,k);
elseif(strcmp(method,'sub_median2'))
    v_k=sub_median2(v,k);
    w_k=sub_median2(w,k);
else
    disp('Last input argument must be a valid string for a method');
end

%Apply the l-th order difference operator
Dv_k=v_k;
Dw_k=w_k;
Dv=v;
Dw=w;
for i=1:l
    Dv_k=diff(Dv_k);
    Dw_k=diff(Dw_k);
    Dv=diff(Dv);
    Dw=diff(Dw);
end

RHS=max(abs(Dv-Dw))
LHS=max(abs(Dv_k-Dw_k))
p=LHS/RHS;

%bad sequences are output (for control purposes)
%if(p>1)
%    v
%    w
%    p
%end
```

```
function res=sub_median(v,steps)

%Subdivision scheme - Median/Tryadic/D=2
% Using direct formulas

%Input: v = intital starting vector
%           steps = number of refinement steps
%Output: Longest sequence, which can be calculated from input

current_step=0; %Overall iteration steps
res=v;

while(current_step<steps) %Big outer loop
    v=res;

    for j=1:(length(v)-2)
        m1=v(j);
        m2=v(j+1);
        m3=v(j+2);

        %main decision process based on d
        if( (m2-m1) ==0 & (m3-m2)==0)
            new_med=[m1,m1,m1];
        else

            %distinguish reversal subcase
            if( (m2-m1)==0 & (m3-m2)~=0)
                %don't forget to reverse in the end
                temp=m1;
                m1=m3;
                m3=temp;
            else
                %just continue
            end

            d=(m3-m2)/(m2-m1);

            %Calculate q1(d)
            if(d>=7/3 & d<=5)
                q1=59/27+7/27*d-8/27*sqrt(16+16*d+d^2);
            elseif(d>=1/5 & d<3/7)
                q1=26/27+16/27*d-4/27*sqrt(1+16*d+16*d^2);
            elseif(d>=-3 & d<=-1/3)
                q1=77/135+13/135*d+8/135*sqrt(1-62*d+d^2);
            elseif(d>=-11 & d<-3)
                q1=-1/288*(323-214*d+35*d^2)/(-1+d);
            else
                q1=7/9-d/9;
            end

            %Calculate q2(d)
            if(d>=-10/7 & d<=-7/10)
                q2=-1/270*(1097-1174*d+17*d^2+(278-8*d)*sqrt(1-62*d+d^2))
                    / (-4+4*d-sqrt(1-62*d+d^2)) ;
            elseif( (d>=-3 & d<-10/7) | (d>-7/10 & d<=-1/3) )
```

```
            q2=23/30+7/30*d+1/15*sqrt(1-62*d+d^2);
        else
            q2=1;
        end

        %Calculate q3(d)
        if(d~=0)
            d_inv=1/d;
            if(d_inv>=7/3 & d_inv<=5)
              q3_temp=59/27+7/27*d_inv-8/27*sqrt(16+16*d_inv+d_inv^2);
            elseif(d_inv>=1/5 & d_inv<3/7)
              q3_temp=26/27+16/27*d_inv-4/27*sqrt(1+16*d_inv+16*d_inv^2);
            elseif(d_inv>=-3 & d_inv<=-1/3)
              q3_temp=77/135+13/135*d_inv+8/135*sqrt(1-62*d_inv+d_inv^2);
            elseif(d_inv>=-11 & d_inv<-3)
              q3_temp=-1/288*(323-214*d_inv+35*d_inv^2)/(-1+d_inv);
            else
              q3_temp=7/9-d_inv/9;
            end
            q3=1+d-d*q3_temp;
        else
            q3=10/9;
        end

        %get the result vector new_med
        new_med=[m1,m1,m1]+(m2-m1)*[q1,q2,q3];

        %handle the reversal case
        if( (m2-m1)~=0 & (m3-m2)==0)
            new_med=[new_med(3),new_med(2),new_med(1)];
        else
            %just continue
        end

    end %ends the decision and calculation based on d
    res(j*3-2:j*3)=new_med;

end %ends the current step

current_step=current_step+1;
end %ends the whole procedure
```

```
function res=my_sgn(x)

%Version of the 'sgn' function for PPH-scheme

if(x>0)
    res=1;
else
    res=-1;
end
end
```

```
function res=sub_median2(v,steps)

%Subdivision scheme - Median/Tryadic/D=2
% Linearized Version !!

%Input: v = intital starting vector
%           steps = number of refinement steps
%Output: Longest sequence, which can be calculated from input

current_step=0; %Overall iteration steps
res=v;

while(current_step<steps) %Big outer loop
    v=res;

    for j=1:(length(v)-2)
        m1=v(j);
        m2=v(j+1);
        m3=v(j+2);

        %main decision process based on d
        if( (m2-m1) ==0 & (m3-m2)==0)
            new_med=[m1,m1,m1];
        else

            %distinguish reversal subcase
            if( (m2-m1)==0 & (m3-m2)~=0)
                %don't forget to reverse in the end
                temp=m1;
                m1=m3;
                m3=temp;
            else
                %just continue
            end

            d=(m3-m2)/(m2-m1);

            %Calculate q1(d)
            q1=7/9-d/9;

            %Calculate q2(d)
            if(d>=-10/7 & d<=-7/10)
                q2=(35-d)/34 ;
            elseif( d>=-3 & d<-10/7  )
                q2=(25+d)/22;
            elseif( d>-7/10 & d<=-1/3 )
                q2=-3/22*(-7+d);
            else
                q2=1;
            end

            %Calculate q3(d)
            q3=2/9*d+10/9;

            %get the result vector new_med
```

```
            new_med=[m1,m1,m1]+(m2-m1)*[q1,q2,q3];

            %handle the reversal case
            if( (m2-m1)~=0 & (m3-m2)==0)
                new_med=[new_med(3),new_med(2),new_med(1)];
            else
                %just continue
            end

        end %ends the decision and calculation based on d
        res(j*3-2:j*3)=new_med;

    end %ends the current step

    current_step=current_step+1;
end %ends the whole procedure
```

```
function res=pph(v,steps)

%Subdivision scheme - PPH

%Input: v = intital starting vector
%           steps = number of refinement steps
%Output: Longest sequence, which can be calculated from input

current_step=0; %Overall iteration steps
res=v;

while(current_step<steps) %Big outer loop
    v=res;
    %Calculate the maximum number of new elements (odd entries)
    for k=1:(length(v)-3)
        d1=v(k)-2*v(k+1)+v(k+2);
        d2=v(k+1)-2*v(k+2)+v(k+3);
        if(d1+d2==0)
            v_new(k)= (v(k+1)+v(k+2))/2; %Prevent division by zero
         else
            v_new(k)=( v(k+1)+v(k+2) )/2-1/8*(d1*d2/(d1+d2)*(my_sgn(d1*d2)+1));
        end
    end

    %Put old and new entries together in one vector
    new_count=1;
    old_count=2; %first entry must be deleted (see PPH scheme)

    for j=0:2*length(v_new)
        if(mod(j,2)==0)
            res(j+1)=v(old_count);
            old_count=old_count+1;
        else
            res(j+1)=v_new(new_count);
            new_count=new_count+1;
        end
    end
    current_step=current_step+1;
end
```

```
function [res,bad,bad_fval]=optimize(runs)

%Finds optimal value 'p' using MatLab optimization toolbox
%Input: runs = number of runs for optimization

counter=1;
%Note that stepsize, order of differences, etc. are directly coded in opt_prefactor

for i=1:runs
     input=(-1)*ones(1,10)+2*rand(1,10);
     options=optimset('MaxFunEvals',1500);
     [x,fval,exitflag,output]=fminsearch(@opt_prefactor,input,options);
     res(i)=fval;
     if (fval < -1) %bad sequence
        bad(counter,:)=x;
        bad_fval(counter)=fval;
        counter=counter+1;
     end
end

plot([1:length(res)],-res,'x')
```

```
function bad=simulate_rand(seq,seq_length,steps,diff_order,scheme)

%Random simulation for stability analysis ||...||<p||...||

%Checks 'p' for random intitial sequences between -1 and 1

%Input: seq = number of sequences to try
%           seq_length = length of those sequences
%           steps = number of steps for the subdivision method
%           diff_order = order of the differences to use
%           scheme = (string) subdivision method to use
counter=1;

for i=1:seq
    v=2*rand(1,seq_length)-ones(1,seq_length);
    w=2*rand(1,seq_length)-ones(1,seq_length);
    result(i)=prefactor(v,w,steps,diff_order,scheme);
    if(result(i)>1)
        bad(counter,:)=[v w];
        counter=counter+1;
    end
end
plot([1:seq],result,'x',[1:seq],max(result)*ones(1,length(result)))
```

```
function  medians = sub_median1(v,steps)

%Subdivision scheme - Median/Tryadic/D=2
% Uses WaveLab functions to cross-check

%Input: v = intital starting vector
%           steps = number of refinement steps
%Output: Longest sequence, which can be calculated from input

tol = 10^(-8);      % tolerance for fixed point iteration
medians=v;

for k=1:steps       %Big outer loop
    counter=1;
    for j=1:length(medians)-2
        P=poly_fit_median([0:1:3],medians(j:j+2),tol);
        %Find medians on new intervals
        for i=0:2
            medians_new(counter)=BlockMedian( P , [1+1/3*i 1+1/3*(i+1)] );
            counter=counter+1;
        end
    end
    medians=medians_new;
end
```

A.2 Calculations for Local Linearization

The following code can be executed in Mathematica. It has been tested on version 5.0 and is used for the local linearization of the quadratic median interpolation. It employs linear Lagrange interpolation; furthermore the error for each interpolation polynomial is calculated.

```
f1[d_]:=59/27 + 7/27 * d - 8/27 * Sqrt[16 + 16 * d + d^2];
f2[d_]:=26/27 + 16/27 * d - 4/27 * Sqrt[1 + 16 * d + 16 * d^2];
f3[d_]:=77/135 + 13/135 * d + 8/135 * Sqrt[1 - 62 * d + d^2];
f4[d_]:= - 1/288 * (323 - 214 * d + 35d^2)/(-1 + d);
f5[d_]:= - 1/270 * (1097 - 1174 * d + 17 * d^2 + (278 - 8 * d) *
Sqrt[1 - 62 * d + d^2])/(-4 + 4 * d - Sqrt[1 - 62 * d + d^2]);
f6[d_]:=23/30 + 7/30 * d + 1/15 * Sqrt[1 - 62 * d + d^2];
f7[d_]:=23/30 + 7/30 * d + 1/15 * Sqrt[1 - 62 * d + d^2];

f1r = f1[7/3];
f1l = f1[5];
f2r = f2[1/5];
f2l = f2[3/7];
f3r = f3[-3];
f3l = f3[-1/3];
f4r = f4[-11];
f4l = f4[-3];
f5r = f5[-10/7];
f5l = f5[-7/10];
f6r = f6[-3];
f6l = f6[-10/7];
f7r = f7[-7/10];
f7l = f7[-1/3];

g1[d_]:=(d - 7/3)/(5 - 7/3) * f1l + (d - 5)/(7/3 - 5) * f1r;
g2[d_]:=(d - 1/5)/(3/7 - 1/5) * f2l + (d - 3/7)/(1/5 - 3/7) * f2r;
g3[d_]:=(d + 3)/(-1/3 + 3) * f3l + (d + 1/3)/(-3 + 1/3) * f3r;
g4[d_]:=(d + 11)/(-3 + 11) * f4l + (d + 3)/(-11 + 3) * f4r;
g5[d_]:=(d + 10/7)/(-7/10 + 10/7) * f5l + (d + 7/10)/(-10/7 + 7/10) * f5r;
g6[d_]:=(d + 3)/(-10/7 + 3) * f6l + (d + 10/7)/(-3 + 10/7) * f6r;
g7[d_]:=(d + 7/10)/(-1/3 + 7/10) * f7l + (d + 1/3)/(-7/10 + 1/3) * f7r;

Plot[{59/27 + 7/27 * d - 8/27 * Sqrt[16 + 16 * d + d^2], g1[d]}, {d, 7/3, 5}];
Plot[{26/27 + 16/27 * d - 4/27 * Sqrt[1 + 16 * d + 16 * d^2], g2[d]}, {d, 1/5, 3/7}];
Plot[{77/135 + 13/135 * d + 8/135 * Sqrt[1 - 62 * d + d^2], g3[d]}, {d, -3, -1/3}];
Plot[{-1/288 * (323 - 214 * d + 35d^2)/(-1 + d), g4[d]}, {d, -11, -3}];
Plot[{-1/270 * (1097 - 1174 * d + 17 * d^2 + (278 - 8 * d) *
Sqrt[1 - 62 * d + d^2])/(-4 + 4 * d - Sqrt[1 - 62 * d + d^2]), g5[d]}, {d, -10/7, -7/10}];
Plot[{23/30 + 7/30 * d + 1/15 * Sqrt[1 - 62 * d + d^2], g6[d]}, {d, -3, -10/7}];
Plot[{23/30 + 7/30 * d + 1/15 * Sqrt[1 - 62 * d + d^2], g7[d]}, {d, -7/10, -1/3}];

e1 = Flatten[Solve[D[f1[d] - g1[d], d] == 0, d]]
e2 = Flatten[Solve[D[f2[d] - g2[d], d] == 0, d]]
e3 = Flatten[Solve[D[f3[d] - g3[d], d] == 0, d]]
e4 = Flatten[Solve[D[f4[d] - g4[d], d] == 0, d]]
e5 = Flatten[Solve[D[f5[d] - g5[d], d] == 0, d]]
e6 = Flatten[Solve[D[f6[d] - g6[d], d] == 0, d]]
e7 = Flatten[Solve[D[f7[d] - g7[d], d] == 0, d]]
```

$\{d \to \frac{4}{3}(-6 + 5\sqrt{3})\}$

$\{d \to \frac{1}{140}(-70 + 19\sqrt{35})\}$
$\{d \to \frac{1}{3}(93 - 56\sqrt{3})\}$
$\{d \to 1 - 4\sqrt{3}, d \to 1 + 4\sqrt{3}\}$
$\{d \to \frac{1}{35}(1085 - 134\sqrt{70})\}$
$\{d \to \frac{31}{7}(7 - 2\sqrt{14})\}$
$\{d \to \frac{1}{15}(465 - 122\sqrt{15})\}$

$\text{err1} = \text{Abs}[\text{f1}[\frac{4}{3}(-6 + 5\sqrt{3})] - \text{g1}[\frac{4}{3}(-6 + 5\sqrt{3})]]$
$\text{err2} = \text{Abs}[\text{f2}[\frac{1}{140}(-70 + 19\sqrt{35})] - \text{g2}[\frac{1}{140}(-70 + 19\sqrt{35})]]$
$\text{err3} = \text{Abs}[\text{f3}[\frac{1}{3}(93 - 56\sqrt{3})] - \text{g3}[\frac{1}{3}(93 - 56\sqrt{3})]]$
$\text{err4} = \text{Abs}[\text{f4}[1 - 4\sqrt{3}] - \text{g4}[1 - 4\sqrt{3}]]$
$\text{err5} = \text{Abs}[\text{f5}[\frac{1}{35}(1085 - 134\sqrt{70})] - \text{g5}[\frac{1}{35}(1085 - 134\sqrt{70})]]$
$\text{err6} = \text{Abs}[\text{f6}[\frac{31}{7}(7 - 2\sqrt{14})] - \text{g6}[\frac{31}{7}(7 - 2\sqrt{14})]]$
$\text{err7} = \text{Abs}[\text{f7}[\frac{1}{15}(465 - 122\sqrt{15})] - \text{g7}[\frac{1}{15}(465 - 122\sqrt{15})]]$

$N[\text{err1}]$
$N[\text{err2}]$
$N[\text{err3}]$
$N[\text{err4}]$
$N[\text{err5}]$
$N[\text{err6}]$
$N[\text{err7}]$

$\text{Simplify}[\text{g1}[d]]$
$\text{Simplify}[\text{g2}[d]]$
$\text{Simplify}[\text{g3}[d]]$
$\text{Simplify}[\text{g4}[d]]$
$\text{Simplify}[\text{g5}[d]]$
$\text{Simplify}[\text{g6}[d]]$
$\text{Simplify}[\text{g7}[d]]$