International Journal of Wavelets, Multiresolution and Information Processing
© World Scientific Publishing Company

Polynomial functions are refinable

Henning Thielemann
Institut für Informatik
Martin-Luther-Universität Halle-Wittenberg
06110 Halle
Germany
henning.thielemann@informatik.uni-halle.de

Received 28.11.2010
Revised (Day Month Year)
Communicated by (xxxxxxxxxx)

The scope of refinable functions in wavelet theory is focused to localized functions. In our paper we like to widen that scope, in particular we show that all polynomial functions are refinable. This may yield an interesting notion of convolution of polynomials.

Keywords: Refinable Function; Wavelet Theory; Polynomial.

AMS Subject Classification: 42C40,

1. Introduction

Refinable functions are functions that are in a sense self-similar: If you add shrinked translates of a refinable function in a weighted way, then you obtain that refinable function, again. For instances, see Figure 1 for how a quadratic B-spline can be composed from four small B-splines and how the so called DAUBECHIES-2 generator function is composed from four small variants of itself. All B-splines with successive

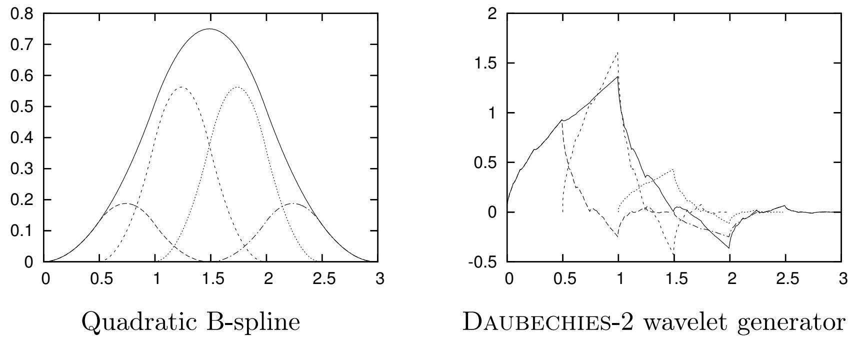

Figure 1. Refinement of a quadratic B-spline and the orthogonal DAUBECHIES-2 generator

integral nodes are refinable, but there are many more refinable functions that did not have names before the rise of the theory of refinable functions. In fact given some conditions we can derive a refinable function from the weights of the linear combination in the refinement.

Refinable functions were introduced in order to develop a theory of real wavelet functions that complements the discrete subband coding theory. [3] Following the requirements of wavelet applications, existing literature on wavelets focuses on refinable functions, that are $\mathcal{L}_2$-integrable and thus are accessible to FOURIER transform, are localized (finite variance) or even better of bounded support. In order to make the picture a bit more complete we will show in this paper, that polynomial functions are refinable, too.

Our results can be summarized as follows:

- Masks that sum up to a negative power of two refine polynomials that are uniquely defined up to constant factors. Other masks are not associated with a polynomial. (Theorem 2.1)
- For every polynomial there are infinitely many refinement masks and these refinement masks can be characterized in a simple form. (Theorem 2.2 and Theorem 2.3)
- There is a simple iterative algorithm for computing a polynomial that is associated with a mask. (Theorem 2.4)

2. Main Work

We start with a precise definition of a refinable function.

Definition 2.1 (Refinable function). *The vector m with $m \in \mathbb{R}^{\mathbb{Z}}$ and a finite number of non-zero entries ($m \in \ell_0(\mathbb{R})$) is called a* refinement mask *for the function φ if*

$$\varphi(t) = 2 \cdot \sum_{j\in\mathbb{Z}} m_j \cdot \varphi(2 \cdot t - j) \tag{2.1}$$

holds. Vice versa the function φ is called refinable *with respect to the mask m.*

The factor 2 before the sum is chosen, such that the following law (Lemma 2.1) about convolutions holds. Unfortunately this enforces adding or subtracting 1 here and there in some of the other theorems. There seems to be no convention that warrants overall simplicity.

Definition 2.2 (Convolution). *For sequences h and g the convolution is defined by*

$$(h * g)_k = \sum_{j\in\mathbb{Z}} h_j \cdot g_{k-j}$$

and for real functions the convolution is defined by

$$(\varphi * \psi)(t) = \int_{\mathbb{R}} \varphi(\tau) \cdot \psi(t - \tau) \,\mathrm{d}\,\tau \qquad .$$

Lemma 2.1. *If φ is refinable with respect to h and ψ is refinable with respect to g, then $\varphi * \psi$ is refinable with respect to $h * g$.*

For a proof see 5. For the proof of our theorems we need two further lemmas about differentiation and integration.

Lemma 2.2. *If the function φ is refinable with respect to mask m, then its derivative φ' is refinable with respect to mask $2 \cdot m$.*

Proof.

$$\begin{aligned}\varphi(t) &= 2 \cdot \sum_{j\in\mathbb{Z}} m_j \cdot \varphi(2\cdot t - j)\\ \varphi'(t) &= 2 \cdot \sum_{j\in\mathbb{Z}} m_j \cdot 2 \cdot \varphi'(2\cdot t - j)\end{aligned}$$

□

Lemma 2.3. *If the function φ is refinable with respect to mask m and $\sum_j m_j \neq 1$, then its antiderivative $\left(t \mapsto \Phi(t) + \frac{\sum_j m_j\cdot\Phi(-j)}{1-\sum_j m_j}\right)$ is refinable with respect to mask $\frac{1}{2}\cdot m$ and the additive constant is the only possible one. We define the antiderivative Φ by $\Phi(t) = \int_0^t \varphi(\tau)\,\mathrm{d}\,\tau$.*

Proof. We start with the necessary condition, that is: Given that the law of the anti-derivative holds, what are the possible integration constants?

$$\begin{aligned}\Phi(t) + c &= 2\cdot\sum_{j\in\mathbb{Z}} \frac{1}{2}\cdot m_j \cdot (\Phi(2\cdot t - j) + c)\\ \Phi(t) + c\cdot(1 - \sum_{j\in\mathbb{Z}} m_j) &= \sum_{j\in\mathbb{Z}} m_j \cdot \int_0^{2\cdot t - j} \varphi(\tau)\,\mathrm{d}\,\tau\\ &= \sum_{j\in\mathbb{Z}} m_j \cdot \left(\int_{-j}^{2\cdot t-j} \varphi(\tau)\,\mathrm{d}\,\tau + \Phi(-j)\right)\\ &= \sum_{j\in\mathbb{Z}} m_j \cdot \left(2\cdot\int_0^t \varphi(2\cdot\tau - j)\,\mathrm{d}\,\tau + \Phi(-j)\right)\\ &= \int_0^t \left(\sum_{j\in\mathbb{Z}} 2\cdot m_j\cdot\varphi(2\cdot\tau - j)\right)\mathrm{d}\,\tau + \sum_{j\in\mathbb{Z}} m_j\cdot\Phi(-j)\\ &= \int_0^t \varphi(\tau)\,\mathrm{d}\,\tau + \sum_{j\in\mathbb{Z}} m_j\cdot\Phi(-j)\\ c\cdot(1-\sum_{j\in\mathbb{Z}} m_j) &= \sum_{j\in\mathbb{Z}} m_j\cdot\Phi(-j)\end{aligned}$$

Proof of the sufficient condition: By substituting c by $\frac{\sum_j m_j\cdot\Phi(-j)}{1-\sum_j m_j}$ we verify that the anti-derivative with that offset is actually refined by $\frac{1}{2}\cdot m$. □

If we generalize refinable functions to refinable distributions, then the DIRAC impulse is refined by the mask δ with $\delta_j = \begin{cases} 1 & : j = 0 \\ 0 & : j \neq 0 \end{cases}$ and the k-th derivative of the DIRAC impulse is refined by $2^k \cdot \delta$. Vice versa the truncated power function $\left(t \mapsto t_+^k\right)$ with $k \in \mathbb{N}$ and $t_+ = \begin{cases} t & : t \geq 0 \\ 0 & : t < 0 \end{cases}$ is refined by $2^{-k-1} \cdot \delta$. Intuitively said, truncated power functions are antiderivatives of the DIRAC impulse.

Once we are thinking about truncated power functions, we find that ordinary power functions with natural exponents are also refinable. Then the next step is to check, whether there are refinable polynomial functions. Indeed we find an example like $f(t) = 1 + 2 \cdot t + t^2$ that is refined by the mask $\frac{1}{8} \cdot (3, -3, 1)$.

$$\begin{aligned}
&2 \cdot \tfrac{1}{8} \cdot (3 \cdot f(2t) - 3 \cdot f(2t-1) + 1 \cdot f(2t-2)) \\
&\quad = \frac{1}{4} \cdot (3 \cdot (1 + 2 \cdot 2t + (2t)^2) - 3 \cdot (1 + 2 \cdot (2t-1) + (2t-1)^2) \\
&\qquad\qquad + (1 + 2 \cdot (2t-2) + (2t-2)^2)) \\
&\quad = \frac{1}{4} \cdot (3 \cdot (1 + 4t + 4t^2) - 3 \cdot 4t^2 + (1 - 4t + 4t^2)) \\
&\quad = 1 + 2 \cdot t + t^2
\end{aligned}$$

Now that we have an example of a refinable polynomial function we like to know, whether there are more examples, how we can characterize polynomial functions that are refinable, and how we can find a mask, that refines a polynomial function. Vice versa, we want to know what masks refine polynomial functions and what polynomial a mask refines.

Before we start answering these questions we like to stress the difference between a *polynomial* and a *polynomial function*.

Definition 2.3 (Polynomial and Polynomial function). *A polynomial p of degree n is a vector from $\mathbb{R}^{\{0,\dots,n\}}$. We need this for the actual computations and for performing linear algebra. A polynomial function $\widehat{p}$ is a real function. The refinement property is a property of real functions. The connection between polynomial and polynomial function is*

$$\widehat{p}(t) = \sum_{k=0}^{n} p_k \cdot t^k \qquad .$$

Our first theorem answers the question, what polynomial function a mask can refine.

Theorem 2.1. *Given a mask m that sums up to 2^{-n-1} for a given natural number n, there is a polynomial p of degree n such that m refines $\widehat{p}$. With the additional condition of the leading coefficient being 1, this polynomial is uniquely determined.*

Proof. We show this theorem by induction over n.

- Case $n = 0$
  We want to show, that a mask m with sum $\frac{1}{2}$ can only refine a constant polynomial. Thus we assume contrarily that m refines a polynomial with a degree d greater than zero. In the refinement relation
  $$\widehat{p}(t) = 2 \cdot \sum_{j\in\mathbb{Z}} m_j \cdot \widehat{p}(2 \cdot t - j)$$
  we only consider the leading coefficient, that is, the coefficient of t^d.
  $$\begin{aligned} p_d &= 2 \cdot \sum_{j\in\mathbb{Z}} m_j \cdot 2^d \cdot p_d \\ &= 2^d \cdot p_d \end{aligned}$$
  From $d > 0$ it follows that $p_d = 0$.
  Thus the degree d must be zero and by normalization it must be $p_0 = 1$. We can easily check that this constant polynomial is actually refined by a mask with sum $\frac{1}{2}$.
- Case $n > 0$: Induction step
  The induction hypothesis is, that for any mask with coefficients' sum 2^{-n-1} we can determine a refining polynomial, that is unique when normalized to leading coefficient being 1. The induction claim is, that this is also true for any mask m that sums up to 2^{-n-2}. We observe that $2 \cdot m$ satisfies the premise of the induction hypothesis and thus there is a polynomial q of degree n, that is refined by $2 \cdot m$, and that is unique when normalized. Since the coefficent sum of $2 \cdot m$ is at most $\frac{1}{2}$ we can apply Lemma 2.3, yielding a polynomial p that is refined by m in the following way: Let Q be the antiderivative polynomial of q where the absolute term is zero, then it is
  $$\begin{aligned} \forall k > 0 \qquad p_k &= \frac{Q_k}{Q_n} \\ p_0 &= \frac{2}{Q_n \cdot (1 - 2^{-n})} \cdot \sum_{j\in\mathbb{Z}} m_j \cdot \widehat{Q}(-j) \qquad . \end{aligned}$$
  Now we turn to the question, why p is uniquely determined. Assume, we have two normalized polynomial functions $\widehat{p_0}$ and $\widehat{p_1}$ that are both refined by mask m. Then their derivatives $\widehat{p_0}'$ and $\widehat{p_1}'$ are refined by mask $2 \cdot m$. Due to the induction hypothesis the normalized polynomial functions of $\widehat{p_0}'$ and $\widehat{p_1}'$ are equal. Lemma 2.3 implies that the antiderivatives with respect to $\widehat{p_0}'$ and $\widehat{p_1}'$ have the same integration constant, and thus $p_0 = p_1$.

□

Theorem 2.2. *Given a polynomial p of degree n, there is a uniquely defined mask m of support $\{0, \dots, n\}$ that refines $\widehat{p}$.*

For giving the proof of that theorem we introduce some matrices.

Definition 2.4. *We express shrinking a polynomial by factor k by the matrix S_k.*

$$\begin{aligned} S_k &\in \mathbb{R}^{\{0,\dots,n\}\times\{0,\dots,n\}} \\ S_k &= \operatorname{diag}(1, k, \dots, k^n) \end{aligned}$$

We represent translation of a polynomial by 1 by the matrix T and translation of a distance i by the power T^i.

$$\begin{aligned} T &\in \mathbb{R}^{\{0,\dots,n\}\times\{0,\dots,n\}} \\ (T^i)_{j,k} &= \begin{cases} \binom{k}{j}\cdot(-i)^{k-j} & : j \le k \\ 0 & : j > k \end{cases} \end{aligned}$$

The proof of Theorem 2.2 follows.

Proof. We define the matrix P that consists of translated polynomials as columns.

$$P = (T^0 p, \dots, T^n p)$$

Now computing m is just a matter of solving the simultaneous linear equations

$$p = S_2 P m \qquad .$$

We only have to show that P is invertible. We demonstrate that by doing a kind of LU decomposition, that also yields an algorithm for actually computing m. Our goal is to transform P into triangular form by successive subtractions of adjacent columns. We define

$$\Delta p = Tp - p$$

what satisfies

$$\Delta(T^k p) = T^k \Delta p \qquad .$$

In the first step we replace all but the first columns of P by differences, yielding the matrix U_1.

$$\begin{aligned} U_1 &= (p, \Delta p, T\Delta p, \dots, T^{n-1}\Delta p) \\ &= P \cdot L_1^{-1} \end{aligned}$$

$$L_1^{-1} = \begin{pmatrix} 1 & -1 & 0 & \cdots & 0 \\ 0 & 1 & -1 & \cdots & 0 \\ 0 & 0 & 1 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix} \qquad L_1 = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 0 & 1 & 1 & \cdots & 1 \\ 0 & 0 & 1 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

In the second step we replace all but the first two columns of U_1 by differences (of the contained differences), yielding the matrix U_2.

$$\begin{aligned} U_2 &= (p, \Delta p, \Delta^2 p, T\Delta^2 p, \dots, T^{n-2}\Delta^2 p) \\ &= U_1 \cdot L_2^{-1} \end{aligned}$$

$$L_2^{-1} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -1 & \cdots & 0 \\ 0 & 0 & 1 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix} \qquad L_2 = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 1 & \cdots & 1 \\ 0 & 0 & 1 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

We repeat this procedure n times, until we get

$$\begin{aligned} U_n &= (p, \Delta p, \Delta^2 p, \ldots, \Delta^n p) \\ P &= U_n \cdot L_n \cdot \cdots \cdot L_2 \cdot L_1 \qquad . \end{aligned}$$

Since the k-th difference of a polynomial of degree n is a polynomial of degree $n-k$, the matrix U_n is triangular and invertible. Thus P is invertible. □

Theorem 2.3. *If p is a polynomial of degree n and m is a mask that refines $\widehat{p}$, then for every mask v the mask $m + v * (1, -1)^{n+1}$ refines $\widehat{p}$ as well. Only masks of this kind refine $\widehat{p}$. The expression $(1, -1)^{n+1}$ denotes the $(n+1)$-th convolution of the mask $\delta_0 - \delta_1$, that is $(1, -1)^0 = (1)$, $(1, -1)^1 = (1, -1)$, $(1, -1)^2 = (1, -2, 1)$ and so on.*

Proof. We denote the convolution of a mask m with a polynomial by the matrix C_m.

$$\begin{aligned} C_m &\in \mathbb{R}^{\{0,\ldots,n\} \times \{0,\ldots,n\}} \\ C_m &= \sum_{i=\nu}^{\kappa} m_i \cdot T^i \end{aligned}$$

The refinement equation can be written

$$p = 2 \cdot S_2 C_m \cdot p \qquad .$$

Since C_m is a Laurent matrix polynomial expression with respect to T, it holds

$$\begin{aligned} C_{h+g} &= C_h + C_g \\ C_{h*g} &= C_h \cdot C_g \qquad . \end{aligned}$$

$$\begin{aligned} & 2 \cdot S_2 C_{m+v*(1,-1)^{n+1}} \cdot p \\ & \qquad = S_2(2 \cdot C_m \cdot p) + S_2(2 \cdot C_{v*(1,-1)^{n+1}} \cdot p) \\ & \qquad = p + S_2(2 \cdot C_v \cdot (C_{(1,-1)^{n+1}} \cdot p)) \\ & (n+1)\text{-th difference of an } n\text{-degree polynomial vanishes: } C_{(1,-1)^{n+1}} \cdot p = 0 \\ & \qquad = p \end{aligned}$$

We still have to show, that refining masks of $\widehat{p}$ always have the form $m + v * (1, -1)^{n+1}$. Consider a mask h_1 that refines $\widehat{p}$. By computing the Laurent polynomial division remainder we can reduce h_1 to a mask h_0 that has support $\{0,\ldots,n\}$ and we can reduce m to a mask g with the same support.

$$\begin{aligned} m &= g + v_0 * (1,-1)^{n+1} \\ h_1 &= h_0 + v_1 * (1,-1)^{n+1} \end{aligned}$$

From the above considerations we derive that both g and h_0 refine $\widehat{p}$ and the uniqueness property in Theorem 2.2 eventually gives us $g = h_0$, thus

$$h_1 = m + (v_1 - v_0) * (1,-1)^{n+1} \qquad .$$

□

Remark 2.1. By adding terms of the form $v * (1,-1)^{n+1}$ we can shift the support of a mask, while the refined polynomials remain the same ones.

2.1. *The cascade algorithm*

There is also an iterative algorithm for the approximate computation of a polynomial that is associated with a refinement mask. This is analogous to the *cascade algorithm* known for refinable functions of bounded support. [2] The refinement relation

$$p = 2 \cdot S_2 C_m \cdot p$$

is interpreted as recursively defined function sequence with

$$p_{j+1} = 2 \cdot S_2 C_m \cdot p_j \quad .$$

This iteration is in fact the vector iteration method for computing the eigenvector that corresponds to the largest eigenvalue.

Theorem 2.4. *Given a mask m that sums up to 2^{-n-1} for a given natural number n, and a starting polynomial p_0 of degree n that is not orthogonal to the refined polynomial, the recursion*

$$p_{j+1} = 2 \cdot S_2 C_m \cdot p_j$$

converges to a polynomial $\lim_{j\to\infty} p_j$ that is refined by m.

An appropriate choice for p_0 is $(0,\ldots,0,1)$.

Proof. The matrix $S_2 C_m$ expands to

$$S_2 C_m = \begin{cases} 2^j \cdot \binom{k}{j} \cdot \sum_{i=\nu}^{\kappa} (-i)^{k-j} \cdot m_i & : j \le k \\ 0 & : j > k \end{cases}$$

and thus is of upper triangular shape. This implies that the diagonal elements $2^j \cdot \sum_{i=\nu}^{\kappa} m_i$ for $j \in \{0,\ldots,n\}$ are the eigenvalues. Because the mask sums up to 2^{-n-1} the eigenvalues of $2 \cdot S_2 C_m$ are $\{1,\ldots,2^{-n}\}$. That is the largest eigenvalue is 1 and it is isolated. These are the conditions for the vector iteration method, consequently the iteration converges to a vector that is a fixpoint of the refinement operation. □

Remark 2.2. The eigenvectors of the eigenvalues are the respective derivatives of the main refinable polynomial.

3. Implementation

The presented conversions from masks to polynomials and back are implemented in the functional programming language Haskell as module `MathObj.RefinementMask2` of the NumericPrelude project [6]. However, note that the definition of the Haskell functions slightly differ from this paper, since the factor 2 must be part of the mask.

4. Related work

So far, refinable functions were mostly explored in the context of wavelet theory. In this context an important problem was to design refinement masks that lead to smooth finitely supported refinable functions. [7,8] It was shown that smoothness can be estimated the following way: Decompose the refinement mask m into the form $(1,1)^n * v$, where n is chosen maximally. According to Lemma 2.1 this corresponds to a convolution of functions. However, strictly speaking, v corresponds to a distribution. The exponent n represents the order of a B-spline and is responsible for the smoothness of the refinable function, whereas for v there is an eigenvalue problem, where the largest eigenvalue determines how much the smoothness of the B-spline is reduced.

The cascade algorithm 2 was developed in order to compute numerical approximations to refinable functions. A combination of the cascade algorithm and Lemma 2.1 was used by 1 for computing scalar products and other integrals of products of refinable functions. This is required for solving partial differential equations using a wavelet Галёркин approach.

Usually discrete wavelet functions are defined in terms of refinable functions but were not considered refinable functions at first. However in 4 it is shown, that wavelets are refinable with respect to infinite masks. The trick is to use polynomial division for dividing the wavelet masks of adjacent scales: If φ is refinable with respect to mask h, and ψ is a wavelet with respect to mask g and generator φ, then ψ is refinable with respect to $\frac{g\uparrow 2}{g} \cdot h$, $g \uparrow 2$ is g upsampled by a factor of 2.

5. Future work

There are some obvious generalizations to be explored: Refinement with respect to factors different from 2, separable multidimensional refinement and most general multidimensional refinement with respect to arbitrary matrices.

Another interesting question is the following one: By Lemma 2.1 we know, that convolution of functions maps to convolution of their refinement masks. We can use this for defining a kind of convolution. In order to convolve two functions φ_0 and φ_1, we compute refining masks m_0 and m_1, respectively, convolve the masks and then find a function that is refined by $m_0 * m_1$. In case of polynomial functions there is no notion of convolution, because the involved integrals diverge. We can however define a convolution based on refinement. Unfortunately, the mapping from a polynomial function to a refinement mask is not unique, consequently the defined convolution is not unique as well – not to speak of the arbitrary constant factor. If we choose arbitrary masks from the admissible ones, then the convolution is not distributive with addition, i.e. $\psi * (\varphi_0 + \varphi_1) = \psi * \varphi_0 + \psi * \varphi_1$ is not generally satisfied. The open question is, whether it is possible to choose masks for polynomials, such that the polynomial convolution via refinement is commutative, associative and distributive.

Bibliography

1. Johan M. De Villiers, Charles A. Micchelli, and Tomas Sauer. Building refinable functions from their values at integers. *Calcolo*, 37:139–158, 2000.
2. Gilbert Strang. Eigenvalues of $(\downarrow 2)H$ and convergence of the cascade algorithm. *IEEE Transactions on Signal Processing*, 44:233–238, 1996.
3. Gilbert Strang and Truong Nguyen. *Wavelets and Filter Banks*. Wellesley-Cambridge Press, 1997.
4. Gilbert Strang, Vasily Strela, and Ding-Xuan Zhou. Compactly supported refinable functions with infinite masks. In L. Baggett and D. Larson, editors, *The Functional and Harmonic Analysis of Wavelets and Frames*, volume 247 of *Contemporary Mathematics*, pages 285–296. American Mathematical Society, 1999.
5. Henning Thielemann. *Optimally matched wavelets*. PhD thesis, Universität Bremen, March 2006.
6. Dylan Thurston, Henning Thielemann, and Mikael Johansson. numeric-prelude: An experimental alternative hierarchy of numeric type classes. `http://hackage.haskell.org/package/numeric-prelude-0.2`, September 2010.
7. Lars F. Villemoes. Sobolev regularity of wavelets and stability of iterated filter banks. *Progress in wavelet analysis and applications*, pages 243–251, 1993.
8. Lars F. Villemoes. Wavelet analysis of refinement equations. *SIAM Journal on Mathematical Analysis*, 25(5):1433–1460, 1994.