Technische Universität München
Lehrstuhl für Informatik V
Univ.-Prof. Dr. M. Bader
Dipl.-Inf. G. Buse, M.Sc. D. Jarema

SS 2013
Sheet 9
19.6.2013

Algorithms of Scientific Computing
(Algorithmen des Wissenschaftlichen Rechnens)
The Wavelet Scaling Function, Haar Wavelets

The wavelet families we look at (e.g. Haar wavelets) are constructed around a *mulitresolution analysis*, a nested sequence V_n of function spaces some of which properties are

$$\begin{aligned} V_j &\subset V_{j+1},\ j \in \mathbb{Z} && (1)\\ \bigcap_{j=-\infty}^{\infty} V_j &= \{0\} && (2)\\ f(t) \in V_l &\Leftrightarrow f(2^{-l}t) \in V_0 && (3)\\ V_l &= V_{l-1} \oplus W_{l-1} \\ &= V_{l-2} \oplus W_{l-2} \oplus W_{l-1} \\ &= V_0 \oplus W_0 \oplus W_1 \oplus \cdots \oplus W_{l-1}, && (4) \end{aligned}$$

with *orthogonal* functions $f \in V_j$ and $g \in W_j$, i.e. $< f, g >= 0$.
The theory of multiresolution analysis further states the existence of a unique function ϕ which satisfies a so-called *dilation equation* of the form

$$\phi(t) = \sum_{k \in \mathbb{Z}} c_k \cdot \phi(2t - k). \quad (5)$$

and which helps us define *orthonormal nodal bases* $\{ \phi_{l,k} \}$ for the V_l with

$$\begin{aligned} \phi_{l,k}(t) &= \phi(2^l\, t - k) \\ span\{ \phi_{l,k} \} &= V_l,\ < \phi_{l,k}, \phi_{l,m} >= \delta_{k,m}\ \ k, m \in \mathbb{Z}. && (6) \end{aligned}$$

The function ϕ is called *father wavelet* or *scaling function*, and together with a *mother wavelet* ψ it defines the wavelet family. It is not necessary to know a specific formula for ϕ, the dilation equation (5) with its coefficients c_k together with the theory of multiresolution analysis provide enough information to derive the mother wavelet ψ as well as *orthonormal wavelet bases* $\{ \psi_{l,m} \}$ for the W_l with

$$\begin{aligned} \psi_{l,k}(t) &= \psi(2^l\, t - k) \\ span\{ \psi_{l,k} \} &= W_l,\ < \psi_{l,k}, \psi_{l,m} >= \delta_{k,m}\ \ k, m \in \mathbb{Z}. && (7) \end{aligned}$$

1 Cranking The Machine

Typically the scaling function ϕ is not known explicitly, and sometimes a closed-form analytic formula does not even exist. However, for continuous ϕ we can approximate the function to arbitrarily high precision using the "Cascade Algorithm", a fixed-point method for functions.

In this exercise we want to implement this algorithm by iterating over the expression

$$F(\gamma)(t) = \sum_k c_k \cdot \gamma(2t - k) \tag{8}$$

in order to find the fixed point γ of F.

Our starting point γ_0 will be the hat function

$$\gamma_0(t) = \begin{cases} 1+t & \textit{for } -1 \leq t \leq 0 \\ 1-t & \textit{for } 0 < t \leq 1 \\ 0 & \textit{else} \end{cases} \tag{9}$$

(i) Over the interval $[-1; 3]$ plot the approximations of the scaling function ϕ for the Haar wavelet family obtained in the first 7 iterations of the cascade algorithm. Do so by plugging the refinements coefficients $c_k,\ k = 0, 1$ in (10) into (8) resp. (5).

$$c_0 = c_1 = 1 \tag{10}$$

(ii) Over the interval $[-1; 3]$ plot the approximations of the scaling function ϕ for the Daubechies wavelet family obtained in the first 7 iterations of the cascade algorithm. Do so by plugging the refinements coefficients $c_k,\ k = 0, \ldots, 3$ in (11) into (8) resp. (5).

$$c_0 = \frac{1+\sqrt{3}}{4} \qquad c_1 = \frac{3+\sqrt{3}}{4} \qquad c_2 = \frac{3-\sqrt{3}}{4} \qquad c_3 = \frac{1-\sqrt{3}}{4} \tag{11}$$

2 The Haar Wavelet Basis

In this exercise we want to compute the 1-d wavelet transform for the Haar wavelet family and apply it to a signal vector $\vec{s}$ of length $m = 2^n$. The transform can be implemented very efficiently as a "pyramidal algorithm" taking $\mathcal{O}(m)$ steps. For educational purpose we focus on the $\mathcal{O}(m^2)$ matrix-based algorithm.

(i) Write a function that constructs the transformation matrix M consisting of the basis vectors $\psi_{l,k},\ l \leq n,\ 0 \leq k \leq 2^n - 1$.

(ii) Use Python's package *numpy.linalg* to invert the matrix.

(iii) Use the program to compute the transform $M\vec{s} = \vec{d}$ as well as the reconstructed signal $M^{-1}\vec{d} = \vec{s'}$ of the vector

$$\vec{s} = [1, 2, 3, -1, 1, -4, -2, 4]^T$$

(iv) Verify the program's output tracing the steps by hand.