MAT-INF2360
Project 2

Jonas van den Brink

j.v.d.brink@fys.uio.no

April 3, 2014

We will be looking at a multiresolution analysis with a scaling function

$$\phi_{0,n} = \frac{1}{\sqrt{2}}\left(\frac{1}{8}\phi_{1,2n-2} + \frac{1}{2}\phi_{1,2n-1} + \frac{3}{4}\phi_{1,2n} + \frac{1}{2}\phi_{1,2n+1} + \frac{1}{8}\phi_{1,2n+2}\right).$$

We then use the notation $\phi_{m,n} = 2^{m/2}\phi(2^m t - n)$, and let $\boldsymbol{\phi}_m = \{\phi_{m,n}\}_{n=0}^{2^m N-1}$ be a basis for the resolution spaces V_m.

We also define the mother wavelet as

$$\psi(t) = \frac{1}{\sqrt{2}}\phi_{1,1}(t).$$

and use the notation $\psi_{m,n}(t) = 2^{m/2}\psi(2^m t - n), \quad \text{for } n = 0, 1, \ldots, 2^m N - 1,$ letting $\boldsymbol{\psi}_m = \{\phi_{m,n}\}_{n=0}^{2^m N-1}$ be a basis for the detail spaces W_m.

Exercise 1)

We will now find the change of coordinate matrix $P_{\boldsymbol{\phi}_1 \leftarrow \mathscr{C}_1}$, where

$$\mathscr{C}_1 = \{\phi_{0,0}, \psi_{0,0}, \phi_{0,1}, \psi_{0,1}, \ldots, \phi_{0,N-1}, \psi_{0,N-1}\}.$$

This is the inverse discrete wavelet transform (IDWT)—and we will find the matrix for $N = 8$.

We have

$$\phi = \frac{1}{\sqrt{2}}\left(\frac{1}{8}\phi_{1,-2} + \frac{1}{2}\phi_{1,-1} + \frac{3}{4}\phi_{1,0} + \frac{1}{2}\phi_{1,1} + \frac{1}{8}\phi_{1,2}\right).$$

and

$$\psi(t) = \frac{1}{\sqrt{2}}\phi_{1,1}(t).$$

From these we can find the first two columns of $P_{\boldsymbol{\phi}_1} \leftarrow \mathscr{C}_1$, as $N = 8$, the final matrix will contain 16 rows.

$$P_{\boldsymbol{\phi}_1 \leftarrow \mathscr{C}_1} = \frac{1}{\sqrt{2}} \begin{pmatrix} 3/4 & 0 \\ 1/2 & 1 \\ 1/8 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 1/8 & 0 \\ 1/2 & 0 \end{pmatrix}.$$

To find the complete matrix we shift the first two columns, so that the final matrix is an MRA-matrix.

From these two columns, we can read of the IDWT filter components G_0 and G_1, expressed in compact filter notation, these are

$$G_0 = \{1/8, 1/2, \underline{3/4}, 1/2, 1/8\},$$
$$G_1 = \{\underline{0}, 1\}.$$

We now define

$$\phi^{(1)}(t) = \begin{cases} 1+t & \text{if } -1 \leq t < 0, \\ 1-t & \text{if } 0 \leq t < 1, \\ 0 & \text{otherwise} \end{cases}.$$

It can then be shown that the scaling function we defined earlier is the convolution of this function with itself, i.e.,

$$\phi(t) = \phi^{(1)}(t) * \phi^{(1)}(t) = \int_{-\infty}^{\infty} \phi^{(1)}(x)\phi^{(1)}(t-x)\ \mathrm{d}x.$$

It can also be shown that

1. ϕ is twice differentiable for alle t.
2. ϕ is a third-degree polynomial on each subinterval [n, n+1].
3. ϕ is 0 outside (-2,2).
4. The functions $\{\phi(t-n)\}_{n=0}^{N-1}$ are linearily independant.

Exercise 2)

While the convolution integral that defines $\phi(t)$ is a bit tedious, it is possible to do analytically. On the next page, we show a figure that results from solving the integral numerically. The figure also shows how $\phi_{m-1,0}$ can be built up from five smaller similar functions at the level m.

From the figure we see that while $\phi(t)$ is not a third-degree polynomial on the whole $t \in [-2, 2]$, it is equal to a third degree polynomial on each subinterval, as stated above. This means that we are looking at an MRA with piecewise third-degree polynomials.

The first resolution space V_0 is thus the space that is spanned by having a third-degree polyinomial on each subinterval $[n, n+1]$, we can write this as

$$f \in V_0 \Leftrightarrow f = \sum_{n=0}^{N-1} c_n \phi_{0,n}.$$

When we move to higher order resolution spaces, we then see that the space V_m is the subspace of continuous functions where each subinterval

$$[n2^{-m}, (n+1)2^{-m}],$$

are third degree polynomials.

We know that the functions ϕ_n are linearily independant, we have also seen that they have compact supports on $t \in [-2+n, 2+n]$, meaning they have support size of four. So any sub-interval $[n, n+1]$ will have contributions from 4 different ϕ_n. As these functions are linearily independant, they therefore effectively span the space of all third-degree polynomials, and so we see that V_0 contains all third-degree polynomials. From this fact alone, I would suspect our MRA to be of better accuracy when approximating functions, compared to the piecewise linear and piecewise constant functions we have looked at earlier.

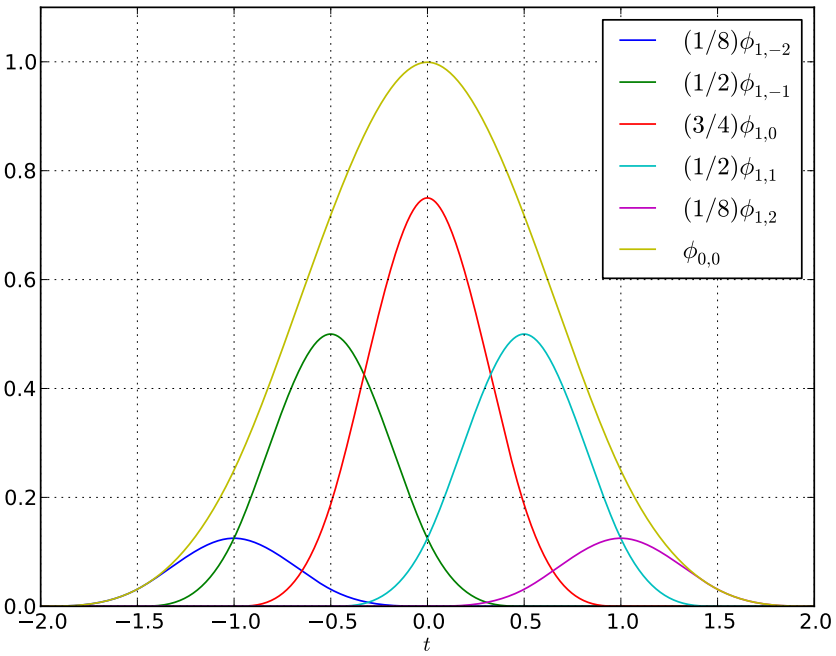

Figure 1: Plotting the basic scaling function, $\phi(t)$, for $t \in [-2, 2]$, and showing how it can be given as the sum of $\phi_{1,k}$ for $k = -2, -1, 0, 1, 2$.

However, the current mother wavelet ψ, (which depends on ϕ) has no vanishing moments. Having many vanishing moments is desirable for a mother wavelet to accuractly reproduce functions. The Haar wavelet has one vanishing moment, so it might seem better than our current MRA. However, as we will see in the coming exercises, we can find a new mother wavelet that has several vanishing moments—without redefining the scaling function ϕ. My conclusion is therefore that our MRA with the given mother wavelet might in fact be worse than the Haar wavelet. But after redefining the mother wavelet as we will do in the coming exercises, that our MRA will be better at approximating functions.

Exercise 3)

We say that ψ has k vanishing moments if the integral

$$\int_0^N t^l \psi(t)\ \mathrm{d}t,$$

vanishes for $0 \leq l \leq k-1$. In our case, we have seen that $\psi \geq 0$ (see figure 1), and so $\psi \geq 0$, which means that our mother wavelet has no vanishing moments.

Having a mother wavelet with many vanishing moments is very desirable, so we will now try to introduce a new mother wavelets, with four vanishing moments—we therefore define a $\hat{\psi}$ with four degrees of freedom:

$$\hat{\psi} = \psi - \alpha\phi_{0,0} - \beta\phi_{0,1} - \gamma\phi_{0,-1} - \delta\phi_{0,2}.$$

The challenge is then to decide the four real constants α, β, γ, δ so that $\hat{\psi}$ indeed has four vanishing moments.

If $\hat{\psi}$ has four vanishing moments, we know that

$$\int_0^N t^k \hat{\psi}\,\mathrm{d}t = \underbrace{\int_0^N t^k \psi\ \mathrm{d}t}_{e_k} - \alpha \underbrace{\int_0^N t^k \phi_{0,0}\ \mathrm{d}t}_{a_k} - \beta \underbrace{\int_0^N t^k \phi_{0,1}\ \mathrm{d}t}_{b_k} - \gamma \underbrace{\int_0^N t^k \phi_{0,-1}\ \mathrm{d}t}_{g_k} - \delta \underbrace{\int_0^N t^k \phi_{0,2}\ \mathrm{d}t}_{d_k},$$

for $k \in 0, 1, 2, 3$. This means that we have the four equations

$$a_k\alpha + b_k\beta + g_k\gamma + d_k\delta = e_k, \text{ for } k = 0, 1, 2, 3.$$

We can write this system of equations as the matrix equation

$$\begin{pmatrix} a_0 & b_0 & g_0 & d_0 \\ a_1 & b_1 & g_1 & d_1 \\ a_2 & b_2 & g_2 & d_2 \\ a_3 & b_3 & g_3 & d_3 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} = \begin{pmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \end{pmatrix}.$$

4.)

Let us now assume that we run an IDWT using the G_0 and G_1 filters defined in exercise 1 on the vector

$$(-\alpha, -\beta, -\delta, 0, \ldots, 0, -\gamma, 1, 0 \ldots, 0),$$

this vector is defined in the basis $(\boldsymbol{\phi}_0, \boldsymbol{\psi}_0)$—and so performing and IDWT gives the coordinate vector in the basis $\boldsymbol{\phi}_1$.

If we write out the vector, using the fact that it is given in the basis $(\boldsymbol{\phi}_0, \boldsymbol{\psi}_0)$, we see that it is

$$-\alpha\phi_{0,0} - \beta\phi_{0,1} - \delta\phi_{0,2} - \gamma\phi_{0,N-1} + \psi_{0,0}.$$

As ϕ is periodic, we can use the equality $\phi_{0,N-1} = \phi_{0,-1}$, and then it is apparent that this coordinate vector corresponds to the function $\hat{\psi}$ in the basis $(\boldsymbol{\phi}_0, \boldsymbol{\psi}_0)$.

We can calculate both $\phi(t)$ and the coefficients α, β, γ, δ analytically, giving the MRA which uses $\hat{\psi}$ as the mother wavelet and ϕ as the scaling function. We will not do these calculations here, as they are tedious. But the resulting digital filters are as follows

$$
\begin{aligned}
G_0 &= \frac{1}{16}\{1, 4, \underline{6}, 4, 1\}, \\
G_1 &= \frac{1}{128}\{5, 20, 1, -96, -70, \underline{280}, -70, -96, 1, 20, 5\}, \\
H_0 &= \frac{1}{128}\{-5, 20, -1, -96, 70, \underline{280}, 70, -96, -1, 20, -5\}, \\
H_1 &= \frac{1}{16}\{1, -4, \underline{6}, -4, 1\}.
\end{aligned}
$$

Exercise 5)

The digital filters H_0, H_1, G_0, G_1 are especially suited for lossless compression for what I assume are two reasons. First of, the filter coefficients are all dyadic fractions, which can be handled exactly in computer arithmetic—this means we get no round-off errors, which would effectively ruin lossless coding.

Exercise 6)

We will now find the continiuous frequency response of the filters G_0 and G_1.

From theorem 3.15 we know that continiuous frequency response of a filter is given by

$$\lambda_s(\omega) = \sum_k t_k e^{-ik\omega},$$

where t_k are the filter coefficients.

Simply inserting the filter coefficients gives

$$\lambda_{G_0}(\omega) = \frac{1}{8}\Big(3 + 4\cos(\omega) + \cos(2\omega)\Big).$$

$$\lambda_{G_1}(\omega) = \frac{1}{64}\Big(5\cos(5\omega) + 20\cos(4\omega) + \cos(3\omega) - 96\cos(2\omega) - 70\cos(\omega) + 140\Big).$$

On the next page we have plotted these functions. We see that G_0 is a low-pass filter. As ω near 0 and 2π correspond to low frequencies, while frequencies near π correspond to high frequencies. Similarily we see that G_1 is a high-pass filter. This fits well with the examples of G_0 and G_1 given in the book, where they have always been a lowpass and highpass filter respectively.

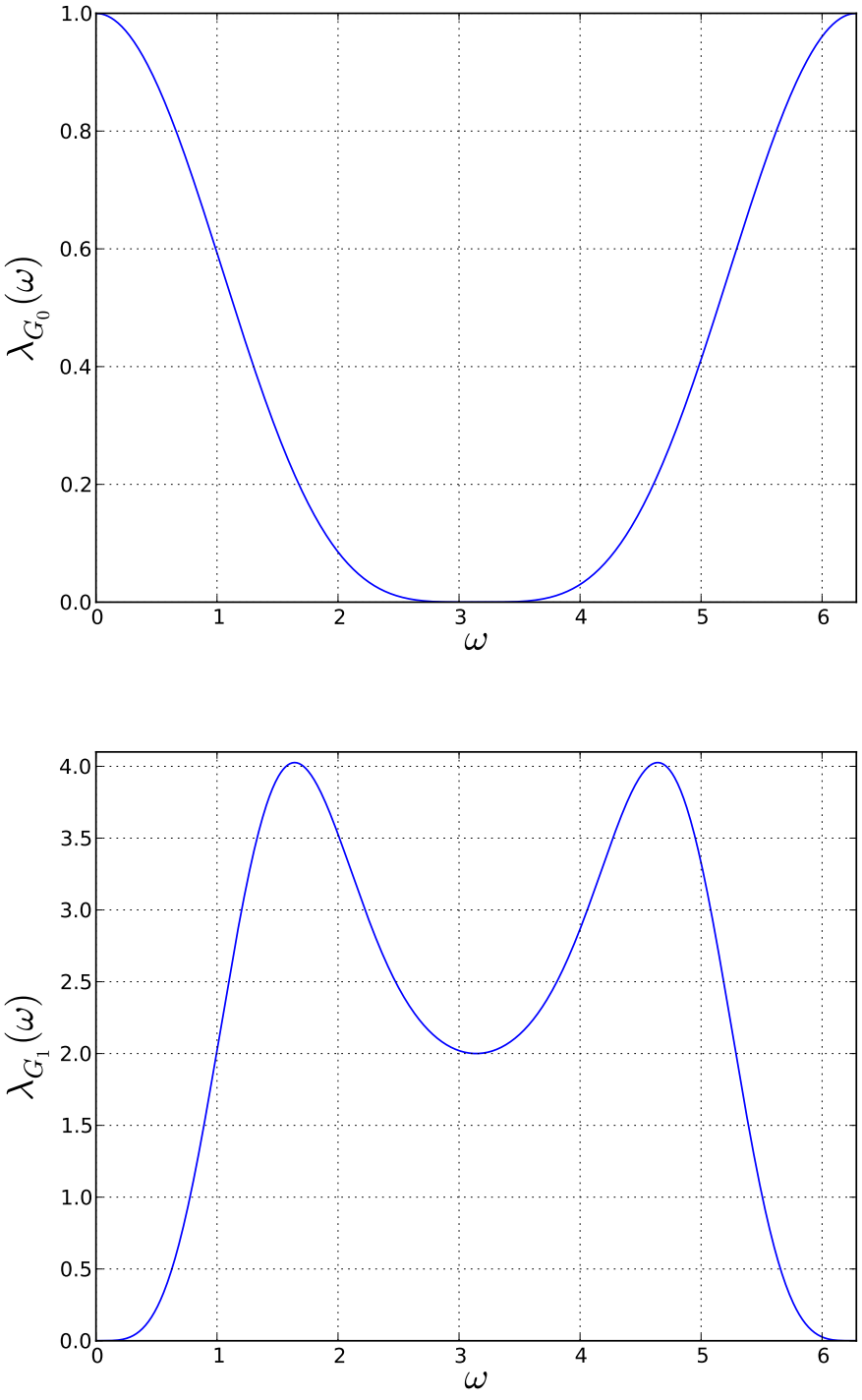

Figure 2: Continiuous frequency response of the two digital filters G_0 and G_1.

Exercise 7)

I opt not to do exercise 7 as it requries me to run code in matlab, while I prefer to stick with python—as decribed in the introduction to the assignment, it is OK to opt out of this exercise.

Exercise 8)

We will now run some sound experiments on our wavelet. We use the functions `playDWTfilterslower` and `playDWTfilterslowerdifference`. Here, the first function performs an m-level DWT on the first 2^{17} sound samples of the file `castanets.wav`, it then sets all the coefficients in the detail spaces to zero, and transforms the signal back into a sound using an IDWT. Meaning the first function performs lossy compression on the audio by truncating the detail. The second function takes only the truncated detail, and converts that back into a sound signal using an IDWT.

We make a simple script that plays both the compressed, truncated sound and the missing detail for $m = 1, 2, 3, 4$ for both the Haar wavelet, and the mother wavelet we have defined in this assignment. The code is as follows

```
g0 = (1./16)*array([1,4,6,4,1])
g1 = (1./128)*array([5,20,1,-96,-70,280,-70,-96,1,20,5])
h0 = (1./128)*array([-5,20,-1,-96,70,280,70,-96,-1,20,-5])
h1 = (1./16)*array([1,-4,6,-4,1])

for m in range(1, 5):
    print "Haar wavelet, m=%i, compressed sound" % m
    playDWTlower(m)
    print "Our mother wavelet, m=%i, compressed sound" % m
    playDWTfilterslower(m,h0,h1,g0,g1)
    print "Haar wavelet, m=%i, detail information" % m
    playDWTlowerdifference(m)
    print "Our mother wavelet, m=%i, detail information" % m
    playDWTfilterslowerdifference(m,h0,h1,g0,g1)
```

When listening to the sound, we hear that for $m = 1$, both wavelets make little difference to the sound, and the missing detail is of very low volume. Due to the small difference, it is hard to discern any real difference between the wavelets. For $m = 2$ however, the Haar wavelet still plays a sound that sounds a lot like the original, but our wavelet has truncated a lot more of the signal. This is even more evident when listening to the missing detail, while the detail for the Haar wavelet has increased somewhat from $m = 1$, the detail for our wavelet has grown a lot more, and we hear that when trucating this much detail the sound gets distored a whole lot. For $m = 3$ the trucated sound of the Haar wavelet is getting quite poor, but it is still comparably better than our wavelet gave for $m = 2$. The truncated sound for our wavelet for $m = 3$ has very little resemblence to the original sound. We hear that the missing details of both wavelets start to sound like the original audio, which makes sense as we are effectively truncating out more and more of the original sound. For $m = 4$ both truncated sounds are pretty rubbish.