Wavelet Assignment I

Fourier and Wavelet Analysis CTH & GU

September 12, 2000

1 Introduction

The purpose of this exercise is to get familiar with the WaveLab software package for filter banks and wavelet transforms. You will compute the wavelet transform of some simple signals which should lead to a better understanding of wavelets and filter banks. Also, we want to give a first glimpse of the compression potential of filter banks and wavelet transforms, something that we will exploit further in Assignment II.

1.1 The WaveLab Package

The WaveLab package is a collection of more than 800 Matlab-files developed by the Department of Statistics at Stanford for wavelet analysis computations. To get access to the WaveLab package execute the following command line in a Matlab session:

```
>> run /usr/pd/lib/WaveLab/WavePath.m
```

Execution of this command will enable Matlab to access the WaveLab toolbox for the duration of the session.

1.2 The Signals

The signals we will analyze are:

- `pulse` A Dirac impulse at $t = 1/2$.
- `step` A unit step function that jumps from 0 to 1 at $t = 1/2$.
- `triangle` A triangle or hat function.
- `wnoise` A white noise signal

- `sine` A pure harmonic sine wave with frequency 25 Hz.
- `realw` A real-world signal

All these signals are sampled on the unit interval $[0, 1]$ with $512 = 2^8$ samples. The signals can be downloaded in a Matlab session by executing

```
>> load ../../Lab1/signals.mat
```

In our discussion on filter banks we have always assumed the signals to be of infinite length. But in practice all signals are finite and this must be taken into account when implementing filter banks. One solution is to make a periodic extension of the signal. This is equivalent to using periodic or circular convolution. In this form the filter bank is fed with 2^J samples and it then splits the signal into two frequency components with 2^{J-1} samples each. A disadvantage with this approach is that it introduces a discontinuity at the boundaries of the signal if the boundary values do not match each other. This will affect the fine-scale wavelet coefficient close to the boundary, but the situation is not as bad as with the Fourier transform where this phenomenon is global. Another method is to extend the signal symmetrically around the boundaries, which is often used in image compression. It is also possible to modify the filters at the boundary, but this is rather complicated. We will only use the periodized wavelet transform here.

2 Filter Banks

2.1 Filter Banks in WaveLab

The first thing you have to do is to choose the filters you want to work with. To generate orthogonal filters you use the `MakeONFilter` command:

```
filter = MakeONFilter(Type,Par);
```

which gives you the synthesis lowpass filter coefficients in the vector `filter`. The `Type` argument is a text string that describes what kind of filter you want to use, e.g. `'Haar'` or `'Daubechies'`. The parameter `Par` controls the filter length. With the Daubechies filters `Par` is an even number equal to the filter length. The frequency response $H(\omega)$ has half as many zeros at $\omega = 0$. With the Haar filters, `Par` should be omitted.

Now, to compute the wavelet transform (iterated filter bank), type

```
wc = FWT_PO(signal,L,filter);
```

The `FWT_PO` (Forward Wavelet Transform, Periodized and Orthogonal) function has three arguments: the input signal `signal`, the lowpass filter `filter`, and an integer `L` that needs to be explained. We start with a signal containing 2^8 sample values. We split this signal into a high- and low-frequency part with 2^7 samples each. We then recursively split the low-frequency part and finally we end up with a low-frequency part containing 2^L samples. Usually one chooses L so that 2^L is greater than the filter length. The output `wc` is a vector containing the wavelet coefficients organized as in figure 1. In other words, the first 2^L elements of `wc` contains the scaling coefficients, then comes the wavelet coefficients at successively finer scales (higher frequencies).

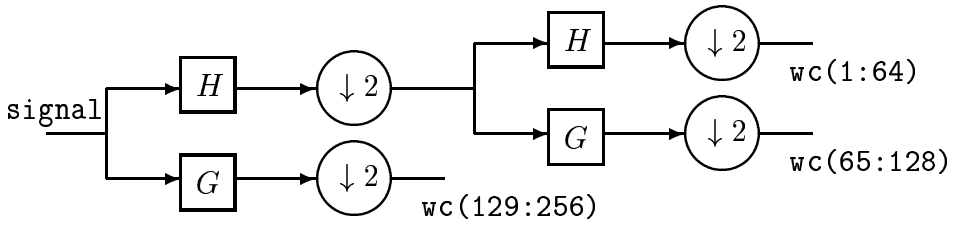

Figure 1: Organization of wavelet coefficients in WaveLab.

2.2 Displaying Wavelet Coefficients

There are several ways to display the wavelet transform, one is to use the `PlotWaveCoeff` command:

```
PlotWaveCoeff(wc,L,scal);
```

This gives a "spike plot" of the wavelet coefficients. Wavelet coefficients at the finest scale will appear at the bottom of the figure and the coarse-scale coefficients at the top. Note that he scaling coefficients are not plotted in the figure. The argument `scal` scales the height of the spikes. Setting it to zero will force the `PlotWaveCoeff(wc,L,scal)` function to automatically choose the scaling.

2.3 Exercise 1

For the signals `pulse` and `wnoise`, compute the wavelet transform and display the results using `PlotWaveCoeff`. Use the Haar filter and the Daubechies filters with filter length 4 and 6. Then answer the following questions:

1. Describe the behaviour of the wavelet coefficients of the Dirac pulse. Is there a difference depending on which filters you are using? If so, explain this difference.

2. Describe the wavelet coefficients of the white noise signal.

3. Plot the frequency responses of the different filters you have used. What is the difference?

3 Wavelet Transforms

In this part we will compute wavelet transforms in the light of wavelet theory. In particular, we will see how vanishing moments is connected to the compression ability of the wavelet transform.

3.1 Multiresolution Plots

To display the multiresolution decomposition itself, rather then just the wavelet coefficients, use the `PlotMultiRes` command:

```
PlotMultiRes(wc,L,scal,filter);
```

The result is a plot of the functions d_j and f_L in the multiresolution decomposition

$$f_J = d_{J-1} + d_{J-2} + \cdots + d_L + f_L,$$

in a single figure. The finest scale d_{J-1} is at the bottom of the figure and f_L is at the top.

3.2 Exercise 2

For each of the signals `step`, `triangle`, `sine`, and `realw`, compute the wavelet transform using the same filters as above. Use the `PlotMultiRes` command to plot the multiresolution decomposition in each case. Please answer the following questions:

1. Describe the behaviour of the signals d_j for the `step` and `triangle` signal. Is there a difference depending on what filters you are using. If so, explain this difference.

2. Do the same thing for the `sine` signal.

3. For the `realw` signal, plot a histogram showing the distribution of the absolute value of the sample values. Here the `hist` command is useful. Do the same thing for the wavelet coefficients. What is the difference? Is there a difference depending on what signal or filter you are using? If so, explain this difference.

4. Finally, we want you to plot the scaling functions and wavelets corresponding to the Daubechies-4 and Daubechies-6 filters. To do this, you use the *cascade algorithm*. To plot a scaling function $\varphi_{j,k}$ in the interval $[0, 1]$, choose the coarsest scale $L = j$. Then define a vector

   ```
   wc␣=␣zeros(1,2^J);
   ```

   for some number $J > L$. Now set the coefficient in `wc` corresponding to $s_{j,k}$ equal to one. Then compute the inverse wavelet transform of `wc`

   ```
   signal␣=␣IWT_PO(wc,L,filter);
   ```

   The scaling coefficients $s_{J,k}$ in `signal` will then be good approximations to the sample values of $\varphi_{j,k}$, provided that J is large enough. In the same way, setting the coefficient in `wc` corresponding to $w_{j,k}$ to one will give a approximation to $\psi_{j,k}$. You will probably have to do some trial-and-error before the plots look nice. For instance, you have to choose j large enough and choose k properly so that the scaling function and wavelet are supported in the unit interval [0,1].