Name:	
ID number:	LiU-ID:
Passed:	Date:

TSBB06 Computer Exercise F
Over-sampling and the Discrete Wavelet Transform

Developed by Klas Nordberg

Computer Vision Laboratory, Linköping University, Sweden

October 31, 2018

Introduction

In the first part of this exercise you will work with various aspects of over-sampling and observe that over-sampling is a technique that, when properly implemented, can help reduce the reconstruction noise that originates from the sampling noise (quantization noise). In the second part you will work with the discrete wavelet transform, and in the end see how it can be used for a simple methods that compress a signal.

Extra

Sections marked with an "Extra" box may be temporarily skipped and completed when the other exercises are finished and if there is time left.

1 Preparatory exercises

Preparatory exercises

Before coming to the computer exercise it is necessary that you have a clear picture of the material related to frames (lecture 2F) and of the continuous and discrete wavelet transform (lecture 2F to 2H). Below, you will find a number of preparatory exercises to be answered **before** the session in order to save time for you, the assisting teacher, and your fellow students. You need to read and understand the specific tasks of the rest of the guide in order to answer some of the preparatory exercises.

1. What is the expected appearance of the Fourier transform of the down-sampled signal in relation to the Fourier transform of the original signal `sbl` in Section 3.1?
ANSWER:

2. What is the expected appearance of the reconstructing filter in this case, in the signal domain and in the frequency domain?
ANSWER:

3. How close do you expect the reconstructed signal in Section 3.1 to be relative to the original signal? Why
ANSWER:

4. In order to predict the results of adding noise to the samples, you need to assume two critical properties of this noise. Which two properties? *Hint:* se lecture 2F!
ANSWER:

5. What mean and standard deviation do you expect for the error in the reconstructed signal `sblrec8n` in Section 3.2?

ANSWER:

6. What is the expected appearance of the reconstructing filter in Section 3.3, in the signal domain and in the frequency domain? How does it differ from the previous one?

ANSWER:

7. What mean and standard deviation do you expect to get when you reconstruct the signal in Section 3.3? Why?

ANSWER:

8. In Section 3.3, the reconstructing functions form a basis of some subspace of the signal space. What subspace of the signal space does this basis span? *Hint:* What is their bandwidth in the frequency domain?

ANSWER:

9. How can you use SVD to verify if a matrix that holds a set of vectors in its column corresponds to a subspace basis or to a subspace frame?

ANSWER:

10. What mean and standard deviation of the reconstruction error do you expect in Section 3.4? Why?

ANSWER:

11. What is the difference between variance and standard deviation? Are you sure about how you formulated the answer of the previous question?

ANSWER:

12. In Section 3.4, the reconstructing functions form a frame of some subspace of the signal space. What subspace of the signal space does this frame span? *Hint:* What is their bandwidth in the frequency domain?

ANSWER:

13. If you get a finite number, k, of samples from a noise source that has zero mean, does this automatically imply that the average of the k samples is equal to zero?

ANSWER:

14. Given one of the filters in Section 4, e.g., h_1, how are the other three filters related to h_1 in an orthogonal filter bank? Assume that the filters are real and give the relation between the filters *in the signal domain.*

ANSWER:

15. Make an illustration of the complete processing structure that is used in Section 4.2 similar to the figure of two-channel filter bank above. The figure should specify what filters are used, and where you find the input and output signals as well as the signals a_1, d_1, a_2, d_2 and $a_{1\text{rec}}$.

ANSWER:

16. Examine the code marked (A) in Section 4.2 and describe what information you will find in the vector `ad` for each iteration of the for-loop.

ANSWER:

17. Examine the code marked (B) in Section 4.2 and describe what information you will find in the vector `ad` for each iteration of the for-loop.

ANSWER:

2 Starting and setting things up

In order to simplify the exercises, you should write the Matlab code that you use into a script file that can be executed in order to generate all results that are requested. This script file should be placed in your own home directory, so you need to add it to your path (if it is not already there). Start Matlab and change directory to the folder that is used in these exercises, and add your own local directory to the path:

```
cd /site/edu/bb/MultidimensionalSignalAnalysis/ExerciseF/
addpath ...
```

3 Noise reduction by over-sampling

In this task you will investigate how over-sampling can be used to reduce the effects of the noise that is inevitable in sampling. According to the sampling theorem you can sample a band-limited signal $s(t)$ of a continuous (time) variable t to a discrete signal $s[k]$ and reconstruct $s(t)$ from $s[k]$ by means of a linear combination with integer shifted sinc-functions. With $s(t)$ being 2π-band-limited and a sampling interval = 1, the reconstruction is done as:

$$s(t) = \sum_k s[k] \,\mathrm{sinc}(t-k) \tag{1}$$

This corresponds to convolution between the samples (multiplied with Dirac-impulses at the corresponding sample position) and the sinc-function.

In this exercise you will instead work only with discrete signals $s[k]$, which by definition have a 2π-periodic Fourier transform when the sample interval is = 1. This signal can then be assumed to be the samples of a time-continuous function $s(t)$ that you will never observe explicitly, but at least you will be able to observe its samples and a period of its Fourier transform.

This assumption also implies that $s(t)$ is 2π-bandlimited, i.e., its Fourier transform is = 0 outside the interval $[-\pi, \pi]$. This property includes the case that $s(t)$ is further band-limited to a smaller interval and, consequently, $s(t)$ can be sampled with a longer interval in this case. For example, if $s(t)$ is band-limited to $[-\frac{\pi}{2}, \frac{\pi}{2}]$ it can be sampled with the interval 2 rather than 1. This means that we can remove half of the samples in $s[k]$

$$s_2[k] = s[2k] \tag{2}$$

and still be able to reconstruct both $s(t)$, as well as $s[k]$ from $s_2[k]$. In this exercise you will reconstruct $s[k]$ from $s_2[k]$.

Furthermore, the discrete signals that you will work with in this exercise have to be of finite length. To make the convolutions with the reconstructing filters work properly, without any edge effects, you can assume that also $s(t)$ is periodic, leading to a periodic convolution. The perioic assumption is in practice often incorrect, but if the length of the signal is long, the effect of making a circular extension of a finite signal can often be neglected at some distance from the "edges".

3.1 Down-sampling and reconstruction of a discrete signal

To see how the down-sampling and reconstruction of discrete signals works, start by creating a discrete signal of 1024 random samples, which in addition is band-limited to $[-\frac{\pi}{8}, \frac{\pi}{8}]$:

```
s=randn(1024,1);
S=fftshift(fft(s));
u=(-512:511)'*pi/512;
Rect8=(abs(u)<pi/8);
Sbl=S.*Rect8; %Create a pi/4-band-limited transform
sbl=ifft(ifftshift(Sbl));
figure(1);subplot(2,1,1);plot(0:1023,sbl);
title('band-limited signal');
subplot(2,1,2);plot(u,abs(Sbl));
title('Fourier transform of band-limited signal');
```

The time discrete signal `sbl` has its origin ($k = 0$) at the first element, the Fourier transform has its origin ($u = 0$) at the center position which in this case is at element 513. It is `sbl` that is your **signal** in what follows, it lies in the **signal space** of bandlimited signals to the interval $[-\pi/8, \pi/8]$.

Since the signal is $\frac{\pi}{4}$-band-limited it can be down-sampled by a factor 8, i.e., we can create a new signal that consists of every 8-th sample from `sbl`:

```
downsampl8=sbl(1:8:end);
```

This is a discrete signal of length 128 samples. To make it more comparable to the original signal, we can up-sample it to the same time scale by inserting 7 samples if value "0" between each sample in `downsample8`:

```
upsampl8=zeros(1024,1);
upsampl8(1:8:end)=downsampl8;
figure(2);subplot(2,1,1);plot(0:1023,upsampl8);
title('The signal down-sampled with a factor 8');
subplot(2,1,2);plot(u,abs(fftshift(fft(upsampl8))));
title('Fourier transform of down-sampled signal');
```

You are now going to reconstruct the discrete signal `sbl` from `downsampl8` or `upsampl8`. As seen from the Fourier transforms of these two signal, the former is given by taking the latter and multiply it by the previously defined rectangular function `Rect8` and then multiply by 8. Alternatively, this operation can be implemented in the signal domain by (circularly) convolving `upsampl8` with the reconstructing filter `rect8`. This operation, in turn, is equivalent to forming the linear combination of the samples in `downsampl8` and the functions generated from `rect8` with multiples-of-8-shifts:

```
rect8=ifft(fftshift(8*Rect8));
figure(3);
subplot(2,1,1);plot(-512:511,ifftshift(rect8));
title('Reconstructing filter rect8');
subplot(2,1,2);plot(u,abs(fftshift(fft(rect8))));
```

Compute the reconstructing functions and plot a few of them:

```
for ix=0:127,
  B8(:,ix+1)=circshift(rect8,8*ix);
end
figure(4);plot(0:1023,B8(:,[1 2 3]));
```

QUESTION: What do the reconstructing functions look like? Are they in accordance with your expectations described in preparatory exercise 2?
ANSWER:

Compute the reconstructed signal and compare it to the original signal:

```
sblrec8=B8*upsampl8(1:8:end);
figure(5);
subplot(2,1,1);plot(0:1023,sblrec8);
title('Reconstructed signal from factor 8 down-sampled signal');
subplot(2,1,2);plot(u,abs(fftshift(fft(sblrec8))));
title('Fourier transform of reconstructed signal');
fprintf('Reconstruction from factor 8 down-sampled signal (no noise)\n');
fprintf('Reconstruction error: %e\n',norm(sbl-sblrec8));
```

QUESTION: What amount of noise do you get (mean and standard deviation)? Is it consistent with your answer to preparatory exercise 3?
ANSWER:

Notice: the number of dimensions that are spanned by the basis in `B8` it is 127. This is because the frequencies $\pm\pi/8$ are excluded from `rect8`.

3.2 Sampling noise

You repeat the last exercise, but now each sample generated by the down-sampling process has some amount of noise added to it.

Add Gaussian noise with standard deviation $\sigma = 0.1$ to the down-sampled sequence, and then up-sample it in the same as previously:

```
sigma=0.1;
downsampl8n=sbl(1:8:end)+sigma*randn(128,1);
upsampl8n=zeros(1,1024);
upsampl8n(1:8:end)=downsampl8n;
figure(6);
subplot(2,1,1);plot(0:1023,upsampl8n);
title('The signal down-sampled with a factor 8 with noise');
subplot(2,1,2);plot(u,abs(fftshift(fft(upsampl8n))));
title('Fourier transform of down-sampled signal with noise');
```

Reconstruct the discrete signal in the same way as before, and compute the mean and standard deviation between the original discrete signal and the reconstructed one:

```
sblrec8n=B8*downsampl8n;
figure(7);subplot(2,1,1);plot(0:1023,sblrec8n);
title('Reconstructed signal from factor 8 down-sampled signal with noise');
subplot(2,1,2);plot(u,abs(fftshift(fft(sblrec8n))));
title('Fourier transform of reconstructed signal with noise');
err=sbl-sblrec8n;
fprintf('Reconstruction from factor 8 down-sampled signal with noise\n');
fprintf('Reconstruction error: mean %f std %f\n',mean(err),std(err));
```

Run this code snippet several times to see how the error varies!

QUESTION: What amount of noise do you get (mean and standard deviation)? Is it consistent with your answer to preparatory exercise 5?
ANSWER:

3.3 Over-sampling, reconstruction with a basis

You are now going to do the same thing again, but now down-sample the original signal with a factor 4 instead of 8. This means that you have twice as many samples to reconstruct from; you *over-sample* the signal with a factor 2. Since the sampling is made at every fourth sample of the original signal, the sampling theorem states that you can reconstruct the signal with a basis of sinc-functions that are scaled by a factor of 2 relative to the previous reconstructing functions.

Use the same type of noise on each sample as before:

```
downsampl4n=sbl(1:4:end)+sigma*randn(256,1);
upsampl4n=zeros(1,1024);
upsampl4n(1:4:end)=downsampl4n;
figure(8);
subplot(2,1,1);plot(0:1023,upsampl4n);
title('The signal down-sampled with a factor 4 with noise');
subplot(2,1,2);plot(u,abs(fftshift(fft(upsampl4n))));
title('Fourier transform of down-sampled signal with noise');
```

You are down-sampling the signal with a factor 4, and should then be able to reconstruct the signal by multiplying the Fourier transform of the signal with a rectangle that cuts out the frequency range $[-\frac{\pi}{4}, \frac{\pi}{4}]$:

```
Rect4=(abs(u)<=pi/4);
rect4=ifft(fftshift(4*Rect4));
figure(9);
subplot(2,1,1);plot(-512:511,ifftshift(rect4));
title('Reconstructing filter rect4');
subplot(2,1,2);plot(u,abs(fftshift(fft(rect4))));
```

Compute the reconstructing functions, they are multiples-of-4-shifts of `rect4`, and plot a few of them:

```
for ix=0:255,
  B4(:,ix+1)=circshift(rect4,4*ix);
end
figure(10);plot(0:1023,B4(:,[1 2 3])');
```

QUESTION: What do the reconstructing functions look like in this case? Are they as you expected?
ANSWER:

Reconstruct the signal and compute the reconstruction error:

```
sblrec4b=B4*downsampl4n;
figure(11);subplot(2,1,1);plot(0:1023,sblrec4b);
title('Reconstructed signal from factor 4 down-sampled signal (basis)');
subplot(2,1,2);plot(u,abs(fftshift(fft(sblrec4b))));
title('Fourier transform of reconstructed signal (basis)');
err=sbl-sblrec4b;
fprintf('Reconstruction from factor 4 down-sampled signal (basis)\n');
fprintf('Reconstruction error: mean %f std %f\n',mean(err),std(err));
```

QUESTION: What amount of noise do you get (mean and standard deviation)? Is it consistent with your answer to preparatory exercise 7?
ANSWER:

Use the Matlab code that you wrote in preparatory exercise 9 to check that $\mathbf{B}_4$ is a basis.

QUESTION: Does it give you the expected answer?
ANSWER:

3.4 Over-sampling, reconstruction with a frame

As an alternative to the reconstruction in the last exercise, you can make use of the fact that the original signal is $\frac{\pi}{8}$-bandlimited, i.e., it can be reconstructed from suitably shifted versions of the function `rect8`. More precisely, they should be shifted by multiples-of-4 in order to correspond to the fact that the original signal in this case is down-sampled with a factor 4. Because of the over-sampling, however, this will in the ideal case (no noise) produce the original times 2, which is why you need to divide these reconstructing functions by 2 in order to get the expected result. Start by computing the reconstructing functions, and plotting a few of them:

```
for ix=0:255,
  F8(:,ix+1)=circshift(rect8,4*ix);
end
figure(13);plot(0:1023,F8(:,[1 2 3])');
```

QUESTION: What do these functions look like? How do they differ from the reconstructing functions in the previous exercise? Compare!
ANSWER:

Reconstruct the signal:

```
sblrec4f=0.5*F8*downsampl4n;
figure(15);
subplot(2,1,1);plot(0:1023,sblrec4f);
title('Reconstructed signal from factor 4 down-sampled signal (frame)');
subplot(2,1,2);plot(u,abs(fftshift(fft(sblrec4f))));
title('Fourier transform of reconstructed signal (frame)');
err=sbl-sblrec4f;
fprintf('Reconstruction from factor 4 down-sampled signal (frame)\n');
fprintf('Reconstruction error: mean %f std %f\n',mean(err),std(err));
```

QUESTION: Did you get a noise level in accordance with preparatory exercise 10?
ANSWER:

In this case, the reconstructing functions form a frame of some subspace of the signal space. Do the same check as you did in the last exercise, now to confirm that `F8` holds a proper frame rather than a basis in its columns.

QUESTION: Is it a proper frame?
ANSWER:

Extra

Try to reduce the reconstruction noise even further by using half of the original sampling points (over-sampling with a factor 4) or even all of them (over-sampling with a factor 8), with the same sampling noise added as before.

Extra

Add some noise that does not comply with the assumed properties and see what happens. Do you still get the same type of reduction of the reconstruction noise as before?

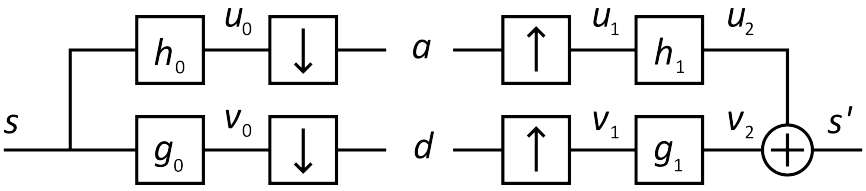

Figure 1: A two-channel filter bank

4 DWT of 1D signals

In this part of the exercise you are going to investigate properties of the discrete wavelet transform (DWT) applied to 1D signals. The basic computational steps related to the DWT are illustrated in Figure 1.

This is the two-channel filter bank, where on the left side a discrete input signal s is analysed by two filters, here denoted h_0 and g_0 and then down-sampled by a factor 2. This produces two signals, a and d, each containing half as many samples as the input signal per time unit. The new signals can then be process or compress in various way but we assume that they can more or less undistorted be fed into the right half of the filter bank, were they are up-sampled and then filtered by the reconstructing filters h_1 and g_1 and then added to form the output signal s'. The basic principle behind constructing a filter bank is that the signal s' should be equal to the signal s (possibly with some time delay) and that a and d possibly can be compressed or processed in different way.

The requirement that the input and output signals should be equal introduces some constraints on the four filters h_0, h_2, g_0, g_1. In this exercise we will not consider how to design these filters, and instead make use of established methods that can produce useful filters, e.g., based on wavelet theory. There is a family of well-known filters called *Daubechies* filters that you will use in this exercise, but there are also other filters described in the literature that can be used for these filter banks. The Daubechies filters define a *conjugate mirror filter bank*, an *orthogonal* filter bank.

4.1 The two-channel filter bank

You will now look at how to work with the two-channel filter bank in practice. Start by generating a signal in a similar way as in the previous task; it has random values and is band-limited:

```
l=256;
u=((-l/2):(l/2-1))*2*pi/l;
s0=rand(1,l);
S0=fftshift(fft(ifftshift(s0)));
S=S0.*(abs(u)<pi/4);
s=real(ifftshift(ifft(fftshift(S))));
figure(1);plot(0:255,s);
title('Input signal');
```

Matlab has a toolbox of functions that can be used for computing wavelet transforms. For example, there is a function `wfilters` that can produce the two analysing filters and the two reconstructing filters for a specific wavelet family. Try the family called `db3`:

```
[h0 g0 h1 g1]=wfilters('db3')
```

The coefficients of the filters are printed in the Matlab window, and you can also look at their Fourier transforms:

```
flength=length(h0);
u1=((-flength/2):(flength/2-1))*2*pi/flength;
figure(2);
subplot(4,1,1);plot(u1,abs(fftshift(fft(h0))));title('FT of h0');
subplot(4,1,2);plot(u1,abs(fftshift(fft(g0))));title('FT of g0');
subplot(4,1,3);plot(u1,abs(fftshift(fft(h1))));title('FT of h1');
subplot(4,1,4);plot(u1,abs(fftshift(fft(g1))));title('FT of g1');
```

QUESTION: How would you characterise the filters in terms of their frequency response functions?
ANSWER:

QUESTION: Are the filter coefficients related to each other in the way you expected?
ANSWER:

Due to the properties of the filters and the fact that they are designed from wavelets, we call the signal a an *approximation* of the input signal and d is referred to as the *details* of the input signal. Signal a is an approximation of the input signal at half the scale of the input signal, and d represent the details that are missing in this approximation.

The Matlab function `dwt` can compute the two down-sampled signals a and d given the filters h_0 and g_0. To simplify the computation and the analysis of the results, we will use the same type of periodic extension of the signal as in the previous task:

```
dwtmode('per');  %Set periodic mode of filtering operations
[a d]=dwt(s,h0,g0);
figure(3);
subplot(2,1,1);plot(a);title('a');
subplot(2,1,2);plot(d);title('d');
```

QUESTION: How would you characterise the two signals a and d?
ANSWER:

You can now feed these two signals into the reconstruction step of the right hand side of the filter bank. This is done by means of Matlab function `idwt`:

```
srec1=idwt(a,d,h1,g1);
figure(4);plot(0:255,srec1);
title('Reconstructed signal');
```

QUESTION: The reconstructed signal should be equal to the input signal. Verify this. Are they equal?
ANSWER:

Notice: the signals a and d each have half as many samples as the input signal s, and together they represent the same amount of data as s. Instead of plotting a and d one on top of the other, as in figure 3, they can be represented as the concatenation $[a\ d]$ which is a signal of the same length as the input signal. This form will be used in the following task.

4.2 Multi-level filter bank

In the previous task you analysed a signal using a wavelet based filter bank. The result are two signals: an approximation of the input signal at a coarser scale (a), and the details needed to reconstruct the input signal from this approximation (d). This analysis can be repeated, now on the approximation a; we compute an approximation of a at the next coarser level, producing a new approximation and the corresponding details. The approximation and the details can then be fed into the proper reconstruction processing (right

hand side of a filter bank) to reconstruct the approximation a.

To simplify this discussion, let a_1 and d_1 denote the approximation and details of the processing made in the previous task. We are now discussing the possibility of using a_1 as input signal to a new filter bank that decomposes it into a new approximation a_2 and details d_2. The reconstructing part of the filter bank can then produce a_1 again which, together with d_1, can reconstruct the input signal s. The analysis can be done like this:

```
[a1 d1]=dwt(s,h0,g0);
[a2 d2]=dwt(a1,h0,g0);
figure(5);
subplot(3,1,1);plot(a2);title('a2');
subplot(3,1,2);plot(d2);title('d2');
subplot(3,1,3);plot(d1);title('d1');
```

Here is the reconstruction:

```
a1rec=idwt(a2,d2,h1,g1);
srec2=idwt(a1rec,d1,h1,g1);
figure(6);plot(srec2);title('Reconstructed signal');
```

QUESTION: Verify that the reconstructed signal equals the input signal. Are they equal?
ANSWER:

The idea of applying a new two-channel filter bank onto the approximation component can be iterated as many times as is practically possible. Let N be the number of iterations. The analysis/decompositon of the input signal can be done as:

```
% Code snippet (A)
N=5;ad=s;p=length(s);figure(7);
for cnt=1:N,
  [a d]=dwt(ad(1:p),h0,g0);
  ad(1:p)=[a d];
  subplot(N+1,1,N+2-cnt);
  plot(d);title(sprintf('details level %d',cnt));
  p=p/2;
end
subplot(N+1,1,1);plot(a);
title(sprintf('approximation level %d',cnt));
figure(8);plot(ad);
title('Concatenated approximation and details');
```

Similarly, the reconstruction can be done as:

```
% Code snippet (B)
for cnt=1:N,
  ad(1:(2*p))=idwt(ad(1:p),ad((p+1):(2*p)),h1,g1);
  p=2*p;
end
figure(9);plot(ad);title('Reconstructed signal');
```

QUESTION: Is the reconstructed signal in figure 9 equal to the input signal, figure 1?
ANSWER:

Try different values for N and look at the results!

4.3 Simple signal compression

So far you have only seen that a filter bank can decompose a signal into an approximation and a set of detail signals which in turn can be used to reconstruct the input signal again. No real benefit from using the filter bank has been demonstrated. In this task you are going to apply a very simple type of compression onto the approximation and detail signals by quantising them into a number of bits that can be different for different components of the signal.

QUESTION: If you use some reasonable value for N and consider the approximation and detail signals, how would you implement such a quantisation? Which components seem to need more bits than others?
ANSWER:

You are now going to repeat the multi-level analysis and reconstruction as in the previous task, but now quantise the approximation and detail signals. This will introduce some quantisation noise that will propagate to the reconstructed signal. However, if the quantisation is made with some care, the reconstruction noise will be reasonably small. Define a matrix `q` of size

$(N+1) \times 2$. Each row of `q` describes a quantisation of a specific channel such that the first element b is the number of bits and the second element is the range r of the values that are quantised, i.e., each channel is quantised in the range $[-r, r]$ in steps of $(2r)2^{-b}$. For a specific N there is one approximation channel and N detail channels, each with its individual quantisation parameters. The first row of `q` are the parameters of the approximation channels, the second for the details at level N, the third for the details at level $N-1$, etc.

Run the previous multi-level decomposition for $N = 3$ but only the analysis part that produces the vector `ad`. Set up the quantisation parameters in `q` with many quantisation levels, e.g., 16 bits for all channels and ranges that are adapted to what you see in the plots. Finally, use the function `quantisead` for performing the quantisation according to `q`. For example:

```
...
q = [16 3;16 2;16 1;16 0.1];  %Change this if needed
[ad bps] = quantisead(ad,q);  %Replace the channels with quantised values
```

`quantisead` produces also an average number of bits per sample, that here ends up in `bps`.

Run the reconstruction part of the filter bank as before, and plot the original signal, the reconstructed signal, and the difference between the two, and compute the signal to noise ratio (SNR):

```
...
figure(100);
subplot(3,1,1);plot(s);title('Original signal');
subplot(3,1,2);plot(ad);title('Reconstructed signal');
subplot(3,1,3);plot(s-ad);title(Difference');
SNR=log10(max(s)/std(ad-s))*20
bps
```

QUESTION: What SNR do you get? What is the average number of bits per sample?
ANSWER:

You are responsible to developing an application where SNR=30 is considered acceptable. You have a number of free parameters: the number of levels N, the number of bits that each of the $N+1$ channels as specified by the matrix `q`, and you can also change from the 'db3' filter set to some other set if you want to. Try to find a good choice of parameters that gives you a filter bank with a quantization that meets the specification, while at the same time has as low average number of bits (≈ 2) per sample.

IMPORTANT NOTE: You should not tune your parameters to a specific random input signal. Make sure that it works for any random input signal that can be generated by your code.

IMPORTANT NOTE: Do not use more than 7 levels. Set number of bits to an integer value > 0.

IMPORTANT NOTE: It not sufficient to demonstrate that you achieve the stipulated SNR and bitrate for a single signal. Therefore, run the script several times for each setting of your parameters and observe the mean values of SNR and bitrate!

QUESTION: What parameter do you use, what is the resulting SNR and average bits per sample?
ANSWER:

Extra

Try to implement the same type of compression scheme on 2D images. Choose some image that has equal height and width and which is an even multiple of 2, e.g., 256×256 pixels. How many bits per pixel do you need without introducing too much noise? How does this compression scheme differ from the one you worked with in the PCA exercise?