Homework Set # 9

Note: The matlab files for this homework can be found on the course webpage under "Additional Material".
You are required to hand in your Matlab plots, the Matlab code (namely the files "WT.m", "encode_block.m", "decode_block.m" and "check_file.m") as well as your written answers to **all the questions** asked in this document.

Problem 1 (Wavelet Transform)

(a) Consider the tree filter bank in Figure 1 with three levels. As you have seen in class, this filter bank can be converted into a single level filter bank such as the one in Figure 2. Let $h_0[n]$ and $h_1[n]$ be the Haar filters, i.e.,

$$h_0[n] = \frac{1}{\sqrt{2}} \begin{cases} 1 & n \in \{0, -1\} \\ 0 & \text{otherwise} \end{cases}$$

$$h_1[n] = \frac{1}{\sqrt{2}} \begin{cases} 1 & n = 0 \\ -1 & n = -1 \\ 0 & \text{otherwise.} \end{cases}$$

Compute the filters $h^{(i)}[n]$, $i = 0, \ldots, 3$, of the equivalent one-level filter bank. (*Hint:* Use the Noble identities.)

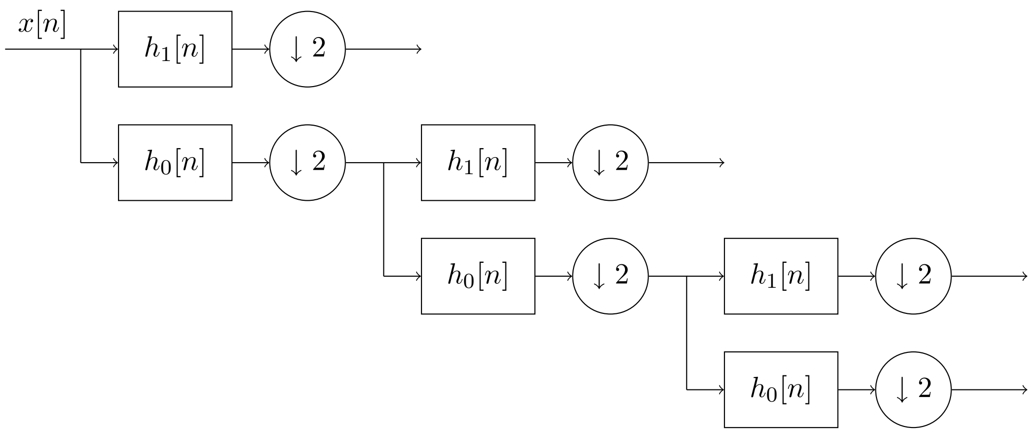


Figure 1: Analysis part of octave filter bank with $J = 3$ levels.

(b) In MATLAB, compute the 1024-point FFT of the filters $h^{(i)}[n]$, $i = 1, \ldots, 4$, and plot their magnitude on the same figure. How do the filters divide the spectrum?

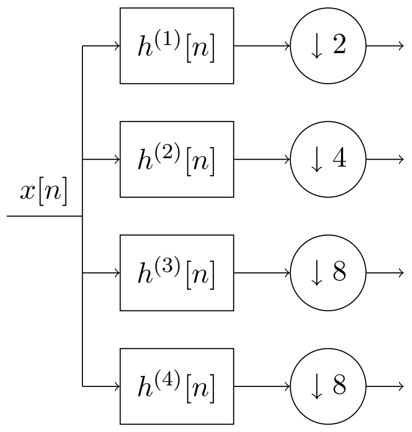


Figure 2: Equivalent single-level filter bank for the tree-structured filter bank of Figure 1.

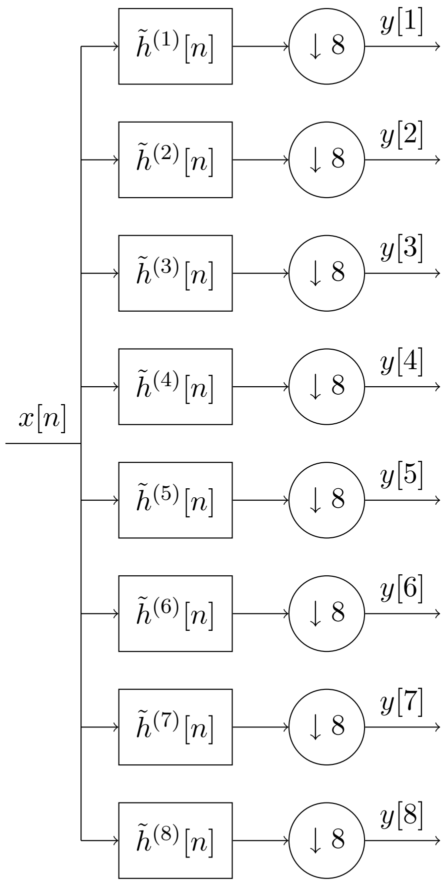


Figure 3: Equivalent filter bank with constant downsampling rate.

(c) The filter bank on Figure 2 can further be converted into the one shown on Figure 3. Find the equivalent filters $\tilde{h}^{(i)}[n]$, $i = 1, \ldots, 8$.

The advantage of this form is that it can easily be implemented in MATLAB as a matrix multiplication.

(d) Let $\mathbf{x} = [x[1], x[2]. \ldots, x[8]]^T$ be a vector of length 8 which is fed to the system in Figure 3. The filter bank will output a single number on each branch. Let $D_8(\cdot)$ represent the downsampling operation, and let $y[i] = D_8(x[n] * \tilde{h}^{(i)}[n])|_{n=0}$, for $i = 1, \ldots, 8$, and $\mathbf{y} = [y[1], y[2], \ldots, y[8]]^T$. Based on the filters you have found in (c), find an 8×8 matrix $\mathbf{Q}$ which satisfies $\mathbf{y} = \mathbf{Q}\mathbf{x}$.

(e) Show that this transform is a projection of vectors of length 8 onto the new basis, with basis vectors in the rows of $\mathbf{Q}$. Show that the basis is orthonormal.

(f) Consider the matrix $\mathbf{Q}$ found in (d). Try to find its structure. Note that the first 2^2 rows have a simple structure. Write this sub-matrix as a Kronecker product of the identity matrix, and vectors $q_0 = \frac{1}{\sqrt{2}}[1\ 1]$ and $q_1 = \frac{1}{\sqrt{2}}[1\ \ -1]$. Consider the next 2^1 rows (rows 5 and 6), and 2^0 rows (row 7) of the matrix. Find the Kronecker product representation for each of them.

To learn about the Kronecker product, see the MATLAB help on the function `kron`.

(g) Download the MATLAB file `WT_skeleton.m`. Fill the blank parts such that the resulting function produces the transform matrix $\mathbf{Q}$ for a similar tree-structured filter bank with depth J. Rename the file to `WT.m`.
Hint: Extend the idea part (f) to arbitrary J.

Problem 2

In this problem, we will use the filterbank from Problem 1 to write an audio compression program (like MP3, but using different techniques). For doing this, we derive a very simple audio-codec, and a corresponding file-format. The file-format is called EMA (for "EPFL Media Audio"). The basic idea is the following: An audio sequence is read from a (MONO) wave-file. The sequence is then split into blocks of length 2^J. We use the filterbank from Problem 1 (with J levels) to decompose each block into subbands. We then use a different number of quantization bits for each subband. We use the fact that the human ear can not hear high frequencies very well. Hence, we use more quantization bits for lower frequencies than for higher ones.

(a) Assume that $J = 6$. Let M be the corresponding projection matrix from Problem 1. Let $\mathbf{c}$ be a block of $2^J = 64$ entries of the audio sequence. Let

$$\tilde{\mathbf{c}} = M\mathbf{c}.$$

Note that $\tilde{\mathbf{c}}$ is also of length 64. Each entry of $\tilde{\mathbf{c}}$ belongs to one of several frequency bands. How many frequency bands are there? For each frequency band, say which entries of $\tilde{\mathbf{c}}$ belong to it.

(b) Now, assume that $J = 10$. Again, let M be the corresponding projection matrix from Problem 1. In Matlab, use the function "WT.m" from Problem 1 to construct M. Use `wavread` to open the wave-file "conga.wav". Cut a block c of length $2^J = 1024$ from the audio sequence, for instance:

```
>> [conga,fs] = wavread('conga.wav');
>> c = conga(300001:301024);
```

Now, apply the matrix M to the block of length 1024. Use the function *plot_wt* to display the subbands of c:

```
>>plot_wt(M*c);
```

Explain the meaning of the regions of the figure.
Hint: See the appendix for some explanations.

(c) Note that the projection M corresponds to a perfect reconstruction filterbank. If we split the audio sequence into blocks of length 2^J and apply M to every block, by how much has the sequence been compressed? In other words, if we stored the resulting sequence to the harddisk, how much smaller would the file be?

(d) To (further) compress the sequence, we quantize the transformed blocks. The entries of a block that correspond to the same frequency subband are quantized using the same quantizer. We use a different quantizer for every subband. The aim is to use less bits for high frequencies and low bits for low frequencies. Which entries of the transformed sequence should be quantized using the fewest bits? Which entries using the most bits?

(e) In the zip file that you have downloaded, you find the files *encode_file.m*, *decode_file.m*, *encode_block_skeleton.m* and *decode_block_skeleton.m*. The first two files are already finished. They load the indicated wave-file and split it into blocks. In the files *encode_block_skeleton.m* and *decode_block_skeleton.m*, some of the code is missing (marked by "... to complete". Read the file headers and complete the files with Matlab code. When you are done, rename the files to *encode_block.m* and *decode_block.m*.
Note: For efficiency reasons, we decided to use the same number of quantization bits for the 6 lowest-frequency subbands. This is why the variable "qbits" contains less entries than the number of subbands.

(f) Test the encoder:

```
>>encode_file('conga.wav','conga.ema');
```

Compare the sizes of the files "conga.wav" and "conga.ema". (You can use *ls -l* in Matlab to do this, or you can check the file properties in your operating system.) By how much has "conga.wav" been compressed?

(g) Test the decoder:

```
>>decode_file('conga.ema','reconstruction.wav');
```

Listen to the files "conga.wav" and "reconstruction.wav" (either in Matlab using *wavread* and *soundsc* or using your favorite music player). Are you satisfied with the quality? If not, you can try out different bitrates and different block-sizes (variables "J" and "qbits" in *encode_file.m* and *decode_file.m*).

Problem 3

Since peer-to-peer software has become popular, music piracy on the internet is a great concern for the music industry. This is why in today's audio compression formats, digital rights management (DRM) plays an important role. Some DRM techniques are very restrictive, limiting the number of times a compressed file can be copied, or how many times it can be burned to an audio-CD. Other techniques are much softer: Copying of files is not restricted; however, the files are tagged with an undetectable watermark. Using this watermark, the origin of a file can be determined by certain authorities, and music pirates can be tracked.

In this problem, we try to "crack" an easy watermarking technique.

(a) The file "conga_tagged.ema" has been encoded with a blocksize of 1024 and a quantization-bit vector $(3, 4, 5, 6, 7)$. In addition, the file has been tagged with a watermark. Listen to the file. Can you tell that a tag is present?

(b) Without knowing the algorithm used for tagging the file, it is virtually impossible to find and remove the tag. This is why we explain now how the tag has been placed:

1. A string of characters containing a copyright-message is written as a stream of bits.
2. During the encoding of the wave-file, one bit of the stream is hidden in each block of the audio file.
3. To hide an information bit in a block, a key is required, which is an integer number between 1 and 64. The bit is hidden in the last subblock of the current block. The last subblock contains 64 entries. The key indicates which entry to choose.
4. To hide the bit inside that entry, we use the last bit of that entry (bit number 7, because the entries of the last subblock are encoded using 7 bits).

Using this information, and assuming that you know the key, it is easy to extract the copyright message from the ema-file. The file "check_file.m" does part of this job. It sends the key to the function "decode_block.m" and expects the bit hidden in that block to be returned as a second return value. Modify the function "decode_block.m" to do this job. You first need to replace the function definition by the one given as a comment. Then, you just need to insert (at the right place) a line of the form:

```
bit = % some function of "key"
```

(c) Test your program "check_file.m":

```
>>check_file('conga_tagged.ema','reconstruction.wav');
```

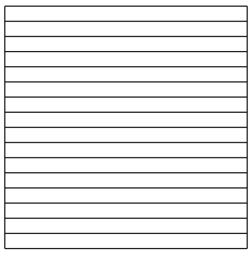
What is the copyright message? Hint: the value of the key that is set in "check_file.m" is correct (key = 1).

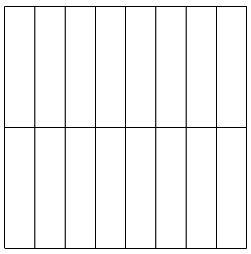
(d) The file "yourname_xmas.ema" in the folder "emas" also contains a watermark tag (use the file that corresponds to your name). The key is unknown. Can you find the tag?
Hint: Instead of trying keys by hand, it is more useful to program a loop inside "check_file.m". What is the message and what is the key that worked?

A Time-Frequency Representation of a Filterbank

In Homework 8 we have encountered the Block Discrete Fourier Transform, where a signal of length L is divided into Q subblocks of lenght N. We have seen that for $Q = 1$, $N = L$, this is equivalent to the DFT, and if $Q = L/2$, $N = 2$, this is equivalent to the Haar Transform. In fact, if we take $Q = L$ and $N = 1$, then we get another "special case", namely the time domain representation we started with. These three cases are represented graphically for a length-16 signal in Figures 4 (a)–(c).

In Problem 2 of this homework, you are asked to analyze a similar time-frequency plot, but for the filterbank used in this homework.


(a) DFT: fine resolution in frequency, no distinction in time.


(b) Haar Transform: medium resolution in time, coarse resolution in frequency.

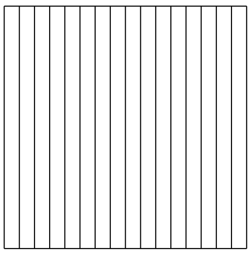

(c) Time domain representation: fine resolution in time, no distinction in frequency.

Figure 4: Block DFT and as special cases the DFT, the Haar Transform, and the time domain representation. The horizontal axis represents the time range, the vertical axis represents the frequency range.