Acoustic modem project

Session 6: OFDM over the acoustic channel: transmitting an image

Paschalis Tsiaflakis, Hanne Deprez[1]

Goal: Using the OFDM modem to transmit an image over the acoustic channel, based on block-training or pilot tones.

Requirements: Matlab/Simulink in Windows, a sound card, a loudspeaker and a microphone.

Required files from DSP-CIS website: *recplay.mdl* (see session 1), *imagetobitstream.m*, *bitstreamtoimage.m*, *image.bmp* (see session 4), *simulate_channel*

Required files from previous sessions: *initparams.m*, *alignIO.m*, *qam_mod.m*, *qam_demod.m*, *ofdm_mod.m*, *ofdm_demod.m*, *ber.m*, *main.m*

Outcome: 2 m-files (+1 optional): *transmit_pic.m*, *visualize_demod.m*, *transmit_pic_pilots.m* (optional)

Deliverables: 1 m-file (+1 optional): *milestone3.m*, *transmit_pic_pilots.m* (optional)

1 Exercise 6-1 (Core): Using packet-based training to transmit an image

In this exercise, we will alternately transmit training packets and data packets over the acoustic channel. The training packets will be used to estimate and re-estimate the channel frequency response (see previous session). Each intermediate channel estimate will then be used to construct an equalizer to demodulate the subsequent data packet. For this, we assume that the channel does not change much during the transmission of a data packet.

1. Create an m-file *transmit_pic.m* in which the following steps are implemented (reuse your code from *main.m* in session 4):
    - Create a vector `trainblock` that contains the training symbols that will be used by the OFDM modem (see exercise 5-1). Furthermore, create a QAM symbol sequence `qamStream` that represents the data obtained from the image *image.bmp* (see session 4).
    - Modify *ofdm_mod.m*, such that the OFDM modem alternately transmits a training packet and a data packet. A training packet contains L_t training frames, where L_t can be set by the user. Each training

[1]Post your questions and remarks on the Discussion Board of toledo (English course) or look at question that are already there.

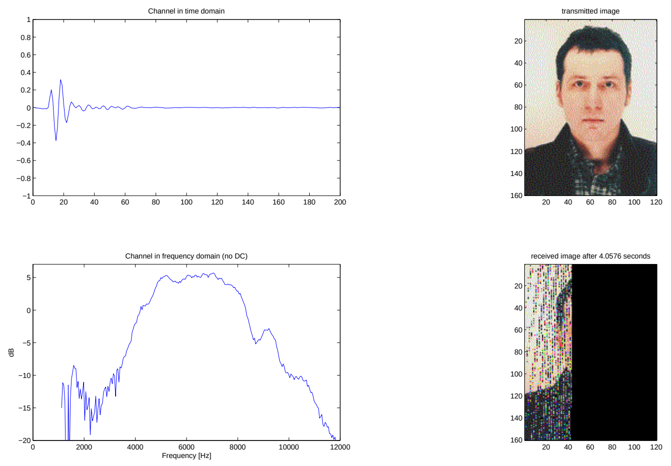

Figure 1: Snapshot of the visualization of the demodulation process

frame is identical and contains the QAM training symbols from the vector `trainblock`. A data packet contains L_d data frames, where L_d can be set by the user. Each data frame contains part of the data from the `qamStream` vector.

- Use the supplied function *simulate_channel* as `[Rx] = simulate_channel(ofdmstream, Lt, Ld, N, L)` to simulate a channel that changes over different data packets.
- Modify *ofdm_demod.m*, such that the OFDM modem uses a training packet to estimate the channel (see session 5), and compute an equalizer to demodulate the subsequent data packet. The function *ofdm_demod.m* should return a demodulated QAM symbol sequence `rxQamStream` and a matrix `channel_est` that contains the different channel estimates in its columns (one for each training packet).
- Use the demodulated data in `rxQamStream` to reconstruct the transmitted image, and compute the BER. *Only continue if the BER is exactly 0.*
- In stead of the function *simulate_channel*, use now the simulink model *recplay.mdl* to play the OFDM signal at the loudspeakers, and record it with the microphone. Align input and output using the function *alignIO.m* (see session 5).
- Find a good value for L_t and L_d. What is the trade-off here (in terms of noise, bitrate and channel tracking performance)?

2. Create an m-file *visualize_demod.m* that visualizes aspects of the OFDM demodulation function during its operation. This file should show a single

figure with 4 subplots[2] (see Fig. 1) that is refreshed each time a new data packet is demodulated. The refresh-rate should match the transmission time of the packets, i.e., as if you analyze the demodulation in real-time. For example, if the transmission of $(L_t + L_d)$ OFDM frames takes 0.2 seconds, the refresh rate of the figure should be approximately 5 times/second. The subplots should contain the following:

   - Upper left: the estimated time-domain channel impulse response, based on the most recent training packet. Make sure the scale on both axes is kept fixed when refreshing the figure.
   - Bottom left: the estimated channel frequency response, based on the most recent training packet (in dB). Make sure the scale on both axes is kept fixed when refreshing the figure.
   - Upper right: the transmitted image (this is fixed, i.e., it does not change when the figure is refreshed).
   - Bottom right: the received image (all pixels starting from the first data frame until the most recent data frame).

3. When running *visualize_demod.m*, look at the time variation, and the shape of the channel. What can you conclude?

2 Exercise 6-2 (Core): Using bitloading to improve BER

1. Use the ON-OFF bitloading of exercise 4-3 to improve the BER for the transmission of the image. For this, modify *transmit_pic.m* as follows:
   - In the beginning of the file, define a scalar parameter `BWusage` that defines the percentage of frequency bins (tones) that will be used for data transmission. E.g. if you set `BWusage=70;`, then 70% of the tones will be used for data transmission. The other tones will not be used.
   - Perform a single dummy transmission of max. 2 seconds to estimate the channel (this is basically what you did in exercise 5-1 and 5-2).
   - Now transmit the image. Based on the channel estimate from the dummy transmission, only use the [`BWusage`] percent best tones for data transmission (what is meant by 'a good tone'?). The other tones are not used. Use your ON-OFF bitloadling code from exercise 4-3 to implement this.
2. Can you observe that the quality of the received image has improved? What is the drawback of using ON-OFF bitloading?

[2]Note that the content of your plots may look very different then those of Fig. 1, i.e., it is just an example of how the demodulation should be visualized.

3. Modify the file *visualize_demod.m* such that the channel response for the unused frequencies is set to zero in the subplot showing the frequency response of the channel (bottom left).

4. (Elite): If you have implemented adaptive bit loading in session 4, you can also use this code instead of ON-OFF bitloading for this exercise. Develop a method to determine the QAM constellation for each tone, based on a dummy transmission (this should happen automatically). A hybrid implementation is also possible (both adaptive and ON-OFF bitloading).

3 Exercise 6-3 (Elite): Using pilot tone training to transmit an image

The Elite part(s) are good for 20% of the grades of the upcoming milestone. However, it does not need to be implemented for the core parts of the exercise sessions that will follow. In case you run out of time, you should focus on understanding and implementing the core part(s) (80% of the grade for the milestone) and spend less time on the Elite part(s).

Instead of using a training packet, one can also use continuous training with pilot tones, and concurrently use the non-pilot tones to transmit actual data. This removes the need for training packets. Try to implement the transmission of an image, based on pilot tones, without using training packets. Save your code in the file *transmit_pic_pilots.m*.

What is the advantage of such an approach? If you would move your microphone around while transmitting the image (assuming the channel impulse response remains shorter than the CP length), what is the best training scheme: training packets or pilot tones? Why?

4 Milestone demo

The third milestone demo takes place at the start of the seventh exercise session. All team members need to be present at the start of this session to be allowed to show the demo. You will be asked to show the following demo(s). Before the start of the seventh exercise session, upload the Matlab code of your demo(s) in one zip-package to the correct milestone on toledo. Make sure to mention your group number as well as the names of all group members in the name of the zip-file. Do not send additional files (only the files that are required to execute the demo).

1. Demo 1: A single m-file named *milestone3.m* should essentially run your files *transmit_pic.m* and *visualize_demod.m* subsequently. It should do this twice: once with `BWusage=100;` (no bitloading) and once with `BWusage=50;` (add a `pause` command between both experiments). Also show the BER in each experiment. If you have also implemented adaptive bitloading in exercise 6-2, you can also demonstrate this.

2. Demo 2 (*Elite*): Demonstrate the transmission of an image by using pilot tones (using *transmit_pic_pilots.m*). Visualize the demodulation with *visualize_demod.m*, showing the channel estimate for each OFDM frame.