Experiment 2

Filters I – Study of System Response

The objective of this experiment is to learn different tools for analyzing filters in MATLAB. You will implement room effects on sound signals using the impulse response of various environments and familiarize yourself with different MATLAB commands for filter analysis.

Prelab

P1. Review of Filters – Time Domain

When a stimulus is applied to a system, the response of the system usually is not the same as the stimulus, i.e. the output of the system is not the same as its input. As an example consider the shock absorber in a car or bicycle. A simple model for this shock absorber would be a damper. The stimulus in this case is the displacement caused by the bumps on the road. The output of the damper is the displacement of the seat where you are sitting on and is proportional to the integral of the input stimulus. As a result the output would be *smoother*, since as a result of integration, fast displacements caused by the bumps of the road result in small changes in the output (Fig. P1)

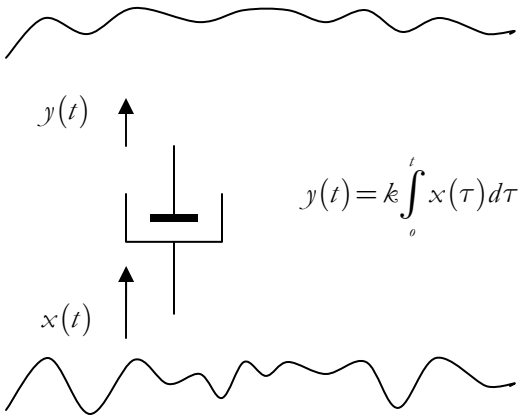


Figure P1 – A simplified model for the shock absorber: A damper, and its input and output signals

The output of a linear system may be computed in various ways. The time domain methods will be reviewed here. If the input and output are sampled signals the model for the system would also be discrete in time. The first form of derivation of the output of a system is by using the system's impulse response and convolution:

$$y(k) = h(k) * x(k) = \sum_{l=-\infty}^{+\infty} h(k-l)x(l)$$

where $h(k)$ is the response of the system to the unit impulse.

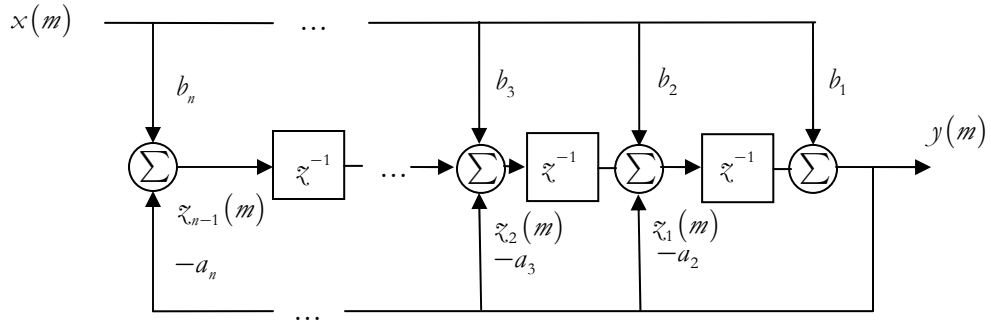
Another way is by using filter function:

$$y(k) = b_1 x(k) + b_2 x(k-1) + \ldots + b_{n+1} x(k-n) - a_2 y(k-1) - \ldots - a_{m+1} y(k-m)$$

From this equation the output at each instant can be found recursively. Starting from $k = 1$, the first few samples will be:

$$\begin{aligned} y(1) &= b_1 x(1) \\ y(2) &= b_1 x(2) + b_2 x(1) - a_2 y(1) \\ y(3) &= b_1 x(3) + b_2 x(2) + b_3 x(1) - a_2 y(1) - a_3 y(2) \\ &\vdots \end{aligned}$$

This can be implemented via the following diagram (all samples are at instant m):



This implementation, called the transposed direct-form II structure is the method used in the implementation of the MATLAB command `filter` which you will be using in this experiment.

Another way is by using the transfer function of the system.

$$Y(z) = H(z) X(z)$$

where $Y(z)$, $H(z)$ and $X(z)$ are the z-transforms of the output, impulse response and input, respectively. In this experiment you will mainly use the first two methods.

P2. Room's Impulse Response

You have surely experienced the echoes of your voice when talking loud in an empty room or the humming accompanying your voice when taking a shower. These are everyday instances of convolution with a system's impulse response. In fact when you are in an empty room, your voice passes through a system which is the empty room and is thus convolved with its impulse response, which you hear as echoes.

The impulse response of a room can be recorded using special devices and by generating an impulse sound in the room (like firing an alarm pistol)[1]. Once recorded, these impulse responses may be used with arbitrary sounds to give them the feel as if they were played in that specific environment.

The impulse response of a room may also be generated using models for the room. These models usually require sophisticated computations and measurements of many parameters of room like the damping coefficients of the walls and other sound reflecting surfaces in the room[2].

[1] Cf. `http://www.xs4all.nl/~fokkie/IR.htm`

[2] Cf. *DAFX Digital Audio Effects*, U. Zölzer, John Wiley & Sons, 2002.

1. Filter Response Analysis

As a warm-up write a simple program to generate a random vector of size 100 and filter it with an averaging filter. To generate the random vector, use MATLAB command `rand`. This command generates random numbers in the interval $[0,1]$ with uniform probability distribution. To perform the filtering define the impulse response `h=[1 1 1 1]/4`. This is a length 4, digital averaging filter. Apply your filter to the random vector using MATLAB command `conv`. Plot both the input and output in the same figure, what is your deduction?

2. Room Simulation

In this section, you will use a simple GUI tool to generate room effects in sounds. The effects are simply generated by convolving the desired sound with the impulse response of the chosen environment.

The GUI requires a function that you will write in this section. This function should:

- Read the input sound file.
- Convert the sound data from stereo to mono.
- Downsample the input to reduce the processing load and the convolution time. This is also necessary to make the sampling rates of the input signal and the impulse response the same.
- Read the impulse response from file.
- Convert the impulse response data from stereo to mono.
- Downsample the impulse response.
- Perform the convolution and compute the output.
- Save the output as in a specified file.

Following is the header of this file:

Listing 1. Function convir

```
function [x, h, y, FS, NBITS, status] = convir(input, eir, outfile)

%-------------------------------------------------------------------------
%
%    Function to compute the convolution of an input sound signal with the
%    specified environment impulse response
%
%     Output:
%
%               x: The vector containing the samples of input signal
%               h: The vector containing the samples of impulse response
%               y: The vector containing the samples of the output
%              FS: The sampling frequency
%           NBITS: Number of bits used to store each sample value
%          status: 1: If both input files are read successfully
%                  0: If an invalid filename is given or the file format
%                     is incompatible
%
%     Input:
%
%           input: The name of the .wav file containing the input signal
%             eir: The name of the .wav file containing the impulse
%                  response
%         outfile: The name of the .wav file to store the output to
%
%-------------------------------------------------------------------------
```

Use MATLAB command `wavread` to read the specified files.

- The input file names are given as strings, e.g. `'tone.wav'`. Use the syntax `[Y,FS,NBITS]=wavread(FILE)`, to get the sampling frequency of the stored sound as well as the number of bits used to store each sample.
- You should check the sampling frequency, obtained from this command to see if downsampling is required. You should convert the sampling rate to 11025Hz in every case. This can be done very easily if the sampling rate is already a multiple of 11025, for instance if the sampling frequency is 44100Hz you just need to take every 4th sample and the resulting signal will have a sampling rate of 11025Hz. If the sampling rate is not a multiple of 11025, simply set `status = 0`, and exit the function.
- The signals may be stereo, in this case the `wavread` command returns a two column matrix where the 1st column contains the left channel of the signal and the 2nd column contains the right channel of the signal. To convert the signals to mono you may either take one of the right or left channel as the signal or take the average of the two channels.
- Use MATLAB `conv` command to convolve the impulse response with the input.
- Note that though you are checking if the returned arguments are out of range, when an invalid file name is specified, MATLAB still generates an error message.
- Check how many input arguments are provided to the function. You may just want to return the output signal and not save it in a file. Use `nargin` to see how many input arguments are present. Save the output signal in a file only if the name of the output file is specified, i.e. three input arguments are present.
- Use MATLAB command `wavwrite` to save the output signal as a .wav file with the name specified.

To test if your program works correctly, run the GUI `roomsim`. If you receive an error message, don't close `roomsim`, use the editor to debug your code and simply select an input file or an impulse response from the list boxes in the GUI.

Note: Some impulse responses are very long (like the huge cavern), so it may take a couple of minutes till you get the output. In order to test your program, use the input 'tone' and the impulse response 'echo' which are shorter and take less time to process.

Now that your program works, try testing it with some different sounds and impulse responses. The controls of the `roomsim` GUI should be clear enough, so I skip their description here. Note that the guitar piece is very long so you probably won't have enough time in the lab to hear all different environment effects on it. The conic long echo hall, is among the interesting results you may get from this sound file. With the French speaker sound the echo, bath and large metallic pipe are among the interesting effects. It is also good to here the piano tone in all the environments.

In every case you try, first listen to the impulse response and look at its shape, the shape of the impulse response for echo is quite insightful. To compare the sounds, first play the input and them play the output to see the effect of the environment.

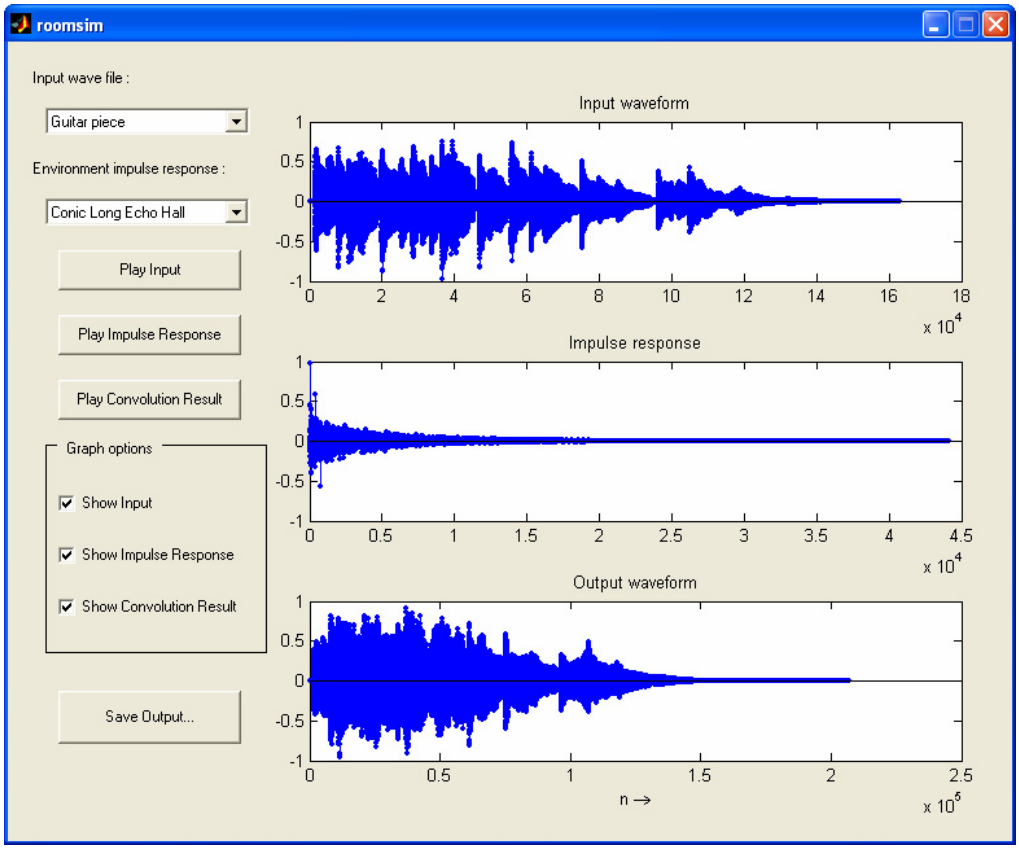


Figure 1 – The `roomsim` GUI

Make sure you have checked one of the graph options' checkboxes so the plot will be updated and you would know when the processing is completed. MATLAB does not respond to anything when processing your function.