Computer Assignment 1: EE2800 - Digital Signal Processing
January-May 2024

Instructor: Sundar Vanka

Guest Lecturer: N. R. Sanjeev

Due: Feb. 19, 2024

SUBMISSION OF STUDENT WORK:

1. Each student should create a tarball containing all the required deliverables from each question. The file shall be named `⟨ROLLNUMBER⟩.tar` (e.g., if the students has a roll number EE22RESCH01004, the expected tar file will be `EE22RESCH01004.tar`.
2. All filenames and variable names specified are case sensitive.
3. This tarball is to be uploaded to the Google Classroom page as an assignment, as is being done for homework.

DELIVERABLES AND THEIR EVALUATION:

1. The deliverable for each question is an m-file that meets the API spec at the end of each problem. Deliverables that do not meet the API spec will not be graded.
2. The name of each m-file should be same as the function name given in the API spec at the end of the question.
3. Evaluation of each m-file will be done by executing the submitted code for an arbitrarily chosen input that meets the API spec given in the assignment.
4. Using MATLAB calls/commands that are explicitly forbidden by the question will result in a 100% penalty of the marks for that question.
5. Each MATLAB file should bear the same name as the function it contains. So, e.g., the function `convolution_sum`(x, h) should be in the file `convolution_sum.m`.
6. **LATE SUBMISSIONS WILL NOT BE GRADED.**

Assume all the audio signals are sampled at 8 kHz unless stated otherwise.

1. (20 Marks) FIR Convolution Sum

Implement the following on MATLAB:

$$y[n] = \sum_{k=n-N+1}^{n} x[k]h[n-k],$$

where N is the filter length, $x[n]$ is the audio input given in the file `x_conv.txt`. Assume that $x[n] = 0$ for $n < 0$. $h[n]$ is the impulse response given in the file `h_conv.txt`. You should name the function `convolution_sum` and its behaviour should mimic the 'conv' command on MATLAB executed with the `'full'` option. The function should take two input arguments: the input signal x, the FIR filter coefficients h.

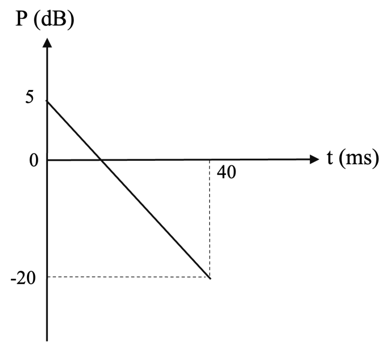
API Specification:

1. Function name : `convolution_sum`
2. Inputs/Arguments to the function should be: x : `inp.txt` (as a column vector) and h : `filter.txt` (as a row vector)
3. Output of the function should be y : convolved sum (as a column vector)
4. The function should be callable as: $y =$ `convolution_sum`(x, h)

The output of your code should be exactly the same as that of the MATLAB `'conv'` command (with `'full'` option) and any formatting (like transposing) as required.

2. (10 Marks) Room Reverberation

Create a room reverb impulse response using the following info: reverb power falls linearly in the dB domain from 5 dB to -20 dB in 40 ms. The impulse response samples equal the (positive) square root of the corresponding power samples. In other words, the impulse response at t, i.e., $h(t)$ and the reverb power are related as : $10\log_{10}|h(t)|^2 \triangleq P(t)$ in dB. Convolve the signal in the file "`solo.wav`" with the reverb impulse response generated by sampling $h(t)$ at 8 kHz.



Power decay with respect to time

API Specification:

1. Function name : `reverb`
2. Inputs/Arguments to the function should be: x : `inp.wav` (as a column vector)

Components of the echo

Time instant	Impulse response amplitude (linear scale)
20 ms	0.55
30 ms	-0.25
60 ms	0.15
70 ms	-0.05

3. Output of the function should be: y : reverb output (as a column vector) and h : impulse response (as a row vector)
4. The function should be callable as: $[y, h] =$ `reverb(`x`)`

Note that you could use the convolution code developed in the previous problem. Also, you must verify that the output agrees with the output of the `'conv'` function in MATLAB.

3. (10 Marks) Echo

Write down the impulse response of the echo in a room that has non-zero components as mentioned in table below: Note that the impulse response of the echo is zero everywhere else. Determine the echo generated in the room for the "`solo.wav`" file. As discussed in the class make use of the sparsity of the echo impulse response to achieve computational efficiency. Do NOT use MATLAB '`conv`' or the implementation developed in the first problem.

API Specification:

1. Function name : `echo_response`
2. Inputs/Arguments to the function should be: x : `inp.wav` (a column vector)
3. Output of the function should be a column vector y : echo output
4. The function should be callable as: $[y] =$ `echo_response(`x`)`

4. (25 Marks) Linear Time Variant System

The output of a Linear Time Variant System is given as

$$y(n) = \sum_{k=n-L+1}^{k=n} x(k)h_k(n)$$

Implement the above sum in MATLAB.

- N = length of the input signal x
- $h_k(\cdot)$ is defined as the k^{th} the impulse response of the system, i.e., the response to a unit impulse shifted by k samples to the right.
- The matrix $N \times L$ matrix H is formed by stacking these $h_k(\cdot)$ such that its k^{th} row is the k^{th} impulse response.
- Sample files `x_LTV.mat` and `h_LTV.mat` containing example input and time varying impulse responses are provided to you for verification of your code.

API Specification:

1. Function name : `LTV_impulse`
2. Inputs/Arguments to the function should be: x (column vector of $N \times 1$ size and H (matrix of $N \times L$ size)
3. Output of the function should be: y (column vector of $N + L - 1 \times 1$ size)
4. The function should be callable as: $[y]$ = `LTV_impulse(`x, H`)`

5. (5 Marks) Musical Notes

Musical notes can be modeled as sinusoids where the i^{th} note is given by

$$m_i(t) = \sin(2\pi f_i t).$$

An *octave* has twelve notes ranging from f_1 to $2f_1$ which are spaced equally on a *logarithmic* scale, i.e., $\log(f_1), \log(f_2), \ldots, \log(f_{12})$ form an arithmetic progression[1]. For the octave range 440 Hz to 880 Hz generate the 12 notes, i.e., in addition to the 440 Hz and 880 Hz tones you should generate ten more notes. Each note should be of two seconds duration.

API Specification:

1. Function name : `musical_notes`
2. The function will not take any inputs or arguments.
3. Output of the function should be a column vector y.
4. The function should be callable as $[y]$ = `musical_notes( )`

[1]One reason, among many, is that such an arrangement is pleasing to the ear.