# Experiment 3: Audio Spectrum Visualization

Laplace, Transfer Functions, Frequency Response

ECE 202: Circuits II

# I. Introduction

In this experiment, you will design and implement an audio spectrum visualizer using various filters on supplied audio files. More information on the frequency content of audible signals is available from

https://www.teachmeaudio.com/mixing/techniques/audio-spectrum/

Generally, the human ear can hear frequencies in the range 20 Hz to 20 kHz, though the extents of this range vary by individual and with age-related degeneration. Additionally, the human ear acts as a filter for incoming sound pressure waves which is tuned to perceive sounds in the low kilohertz range with the highest gain i.e. a small amplitude sound wave at 4 kHz will be perceived as equally loud to a much higher amplitude wave at 50 Hz. Most distinguishable features of normal speech occur in the low kilohertz range.

To make the spectrum generated in this experiment more meaningful, you will normalize the displayed amplitude of various frequency content according to the spectrum of the human ear. Figure 1 gives experimentally measured equal "loudness" curves for various frequency ranges.

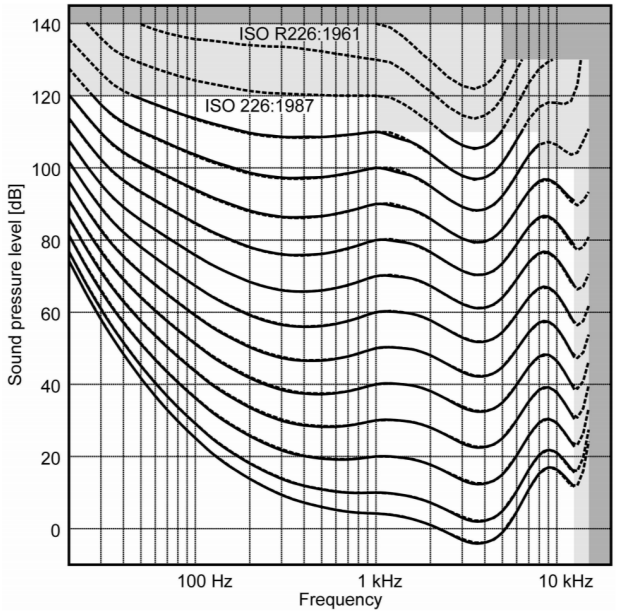



Figure 1. ISO-266 equal loudness contours. Each contour represents the audio signal amplitude at which humans perceive roughly equal "loudness" for varying frequencies.

Figure 2 gives an example snapshot of the spectrum of an audio signal generated at a single time instant. This spectrum shows, with normalized amplitude, the relative frequency content of the audio signal, with a peak amplitude near 800 Hz, in the "midrange". This frequency content will change over time in a non-periodic audio signal, in response to the type and amplitude of the sounds included.

A typical approach for generating such a spectrum is the fourier transform, which will be discussed later in the course. Instead, we will use various filters to attenuate all frequencies outside of specified passbands for each bar of the spectrum shown in Figure 2.

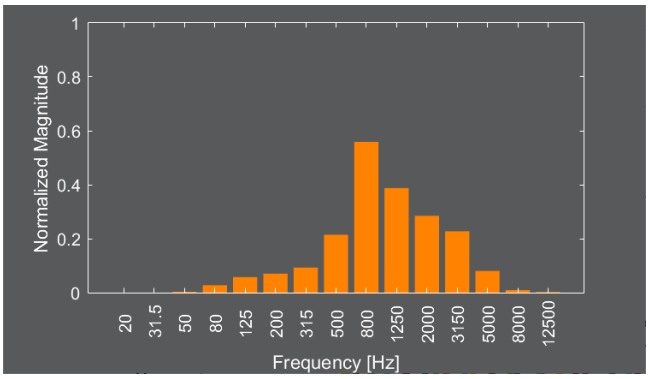



Figure 2. Example music spectrum generated in this experiment

# II. Laboratory Experiment

In this experiment, we will be designing a spectrum visualizer for audio files using MATLAB and implementing filters in LTSpice. Your task is to design the transfer function and circuit implementation of low pass, high pass, and bandpass filters necessary to isolate certain frequency ranges in an audio signal.

The laboratory files include five code files for use in MATLAB for Exercise 1

1. `iso266.m` – function which returns sampled points on the ISO-266 equal loudness curves of Figure 1. You should not have to edit this file during this experiment.
2. `generateAudioVisualization.m` – function implementing the spectrum visualization. The input arguments are:
   i. `fn` – filename of the audio file to play
   ii. `H` – a matrix of transfer functions for the spectrum filters
   iii. `fbins` – a vector of frequency values giving the limits for each channel of the visualization

   You should not have to edit this file during this experiment
3. `main.m` – main file for the spectrum visualizer. You will need to edit this file. In the "`%% Filter Design`" section of the code, you will need to define `H(1,1)` through `H(15,15)`. These are the 15 transfer functions implementing each of the spectrum filters.
4. `chirp.wav` – An audio file for use in testing. This file implements a chirp: a sinusoid whose frequency gradually ramps from 20 Hz to 10kHz over 10 seconds.
5. `Tennessee__Rocky_Top__singing.mp3` – An audio file for use in testing. Contains a more complex audio signal with wide frequency content.

## LE1 Lab Exercise 1: Filter Design

In `main.m`, define the 15 transfer functions listed in Table I. The finite, non-zero corner frequencies are stored in the variable `fbins`, which is a 1x14 vector. The passband gains are stored in the 1 x 15 vector `mags`.

TABLE I: FILTER FORMER AND PASSBAND GAIN CHARACTERISTICS

| Variable | Passband Frequency | | Passband Gain |
|---|---|---|---|
| | Low Frequency | High Frequency | |
| H(1,1) | 0 Hz | 32 Hz | 1.00 |
| H(2,2) | 32 Hz | 50 Hz | 2.93 |
| H(3,3) | 50 Hz | 80 Hz | 7.30 |
| H(4,4) | 80 Hz | 125 Hz | 15.52 |
| H(5,5) | 125 Hz | 200 Hz | 27.86 |
| H(6,6) | 200 Hz | 315 Hz | 45.05 |
| H(7,7) | 315 Hz | 500 Hz | 63.80 |
| H(8,8) | 500 Hz | 800 Hz | 80.60 |
| H(9,9) | 800 Hz | 1.25 kHz | 92.48 |
| H(10,10) | 1.25 kHz | 2 kHz | 66.88 |
| H(11,11) | 2 kHz | 3.15 kHz | 83.21 |
| H(12,12) | 3.15 kHz | 5 kHz | 124.67 |
| H(13,13) | 5 kHz | 8 kHz | 73.89 |
| H(14,14) | 8 kHz | 12.5 kHz | 23.94 |
| H(15,15) | 12.5 kHz | ∞ Hz | 47.77 |

Each filter must exhibit the following characteristics

i. The filter must have the passband gain (to within +0/-6dB) shown in Table I.
ii. The filter must have corner frequencies at the limits of the passband frequency
iii. In the stopband(s), the filter must exhibit 40 dB/dec attenuation
iv. Each filter must have zero magnitude ripple in the passband

Figure 3 gives an example figure showing the Bode magnitude diagram of 15 filters similar (though with different magnitudes) as those you should construct in this exercise.

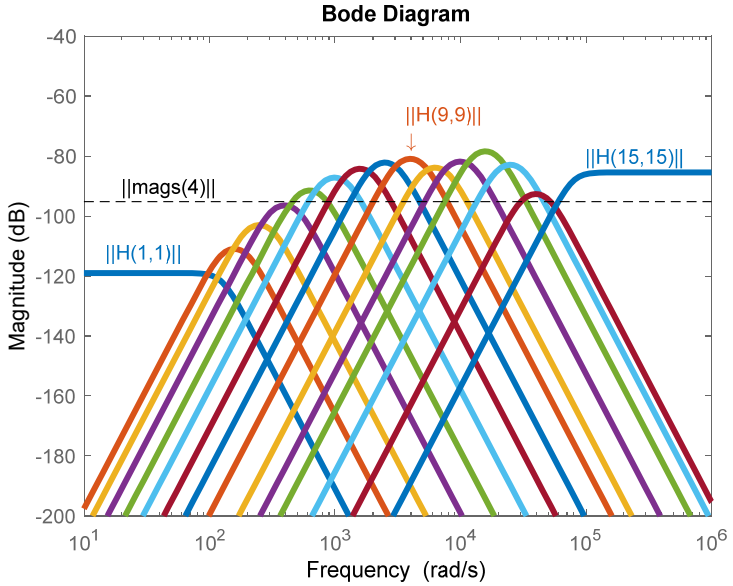



Figure 3. Example bode magnitude plot of 15 filters.

Once you have defined the transfer function of all 15 filters, run the file `main.m` using the first audio file. The final non-commented line in the script should be

```
generateAudioVisualization('chirp.wav', H, fbins);
```

Make sure to have your speakers at a low volume before running the script; the sound produced by the chirp can be painful at high volumes. You should see the spectrum of the signal plotted and varying as the sound clip plays. This chirp is a pure sinusoid with ramping frequency, increasing from 20 Hz to 20 kHz. If your filters are properly designed, you should see a spectrum plot that corresponds to this behavior.

Next, comment the above line of code and uncomment the final line, so that the final line of code executed is

```
generateAudioVisualization('Tennessee__Rocky_Top__singing.mp3', H, fbins);
```

The code should display the frequency spectrum of the sound of this song. Feel free to visualize any additional sound files that you have by modifying the filename in the above function call.

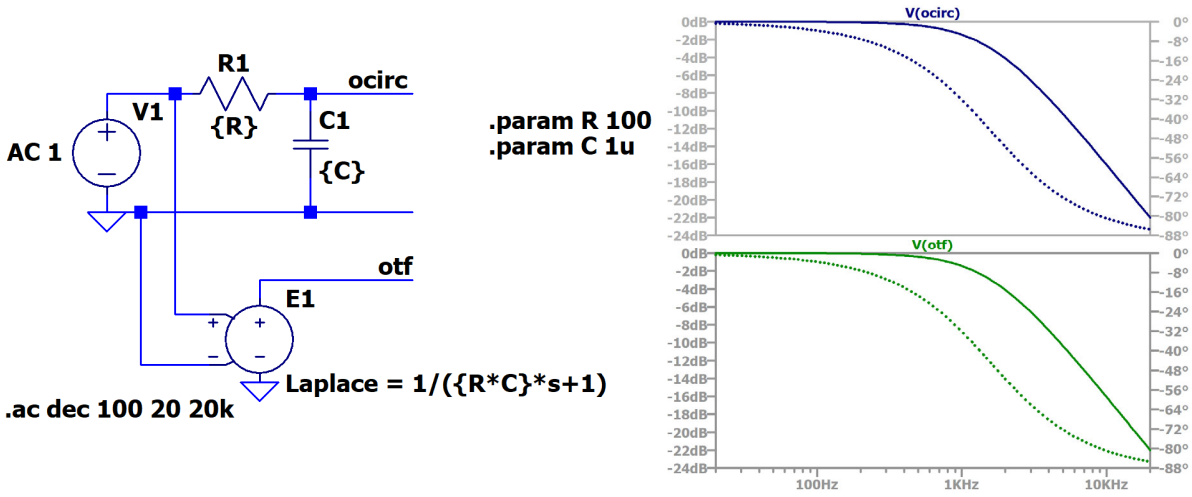
## LE2 Lab Exercise 2: Filter Circuit Implementation

In LTSpice implement the following three of the filter designed in LE1. Each filter must be designed using only the specified number of op-amps and as many resistors and capacitors as needed.

Table II: Filters implemented in LTspice

| Filter | Max number of op-amps |
|---|---|
| H(1,1) | 2 |
| H(9,9) | 3 |
| H(15,15) | 2 |

For a comparison, implement your *s*-domain transfer function alongside each filter using an E source, as shown in Figure 4 for a simple *RC* circuit.




(a) (b)

Figure 4. Example LTSpice circuit (a) and simulation result (b) comparing transfer function and circuit implementation

Using an ac simulation, verify that your circuit implementation achieves the same frequency response as your design over the frequency range from 20 Hz to 20 kHz.

# III. Deliverables

For this experiment, you do not need to turn in a report. Instead, turn in your `main.m` Matlab file, and your LTspice simulation (.asc) file for the three filters. Also, turn in a single pdf document with a screenshot of the bode plot generated in LE1 and the simulated plot(s) comparing frequency responses in LE2. No additional narrative is needed beyond these items in your submission.