ECE 45 Synthesizer Project

Source Code: https://github.com/ZhenmanShen/ECE45ECProject.git

TABLE OF CONTENTS

CONTRIBUTORS

Names and PIDs:

1. Christopher Inzunza: A17445642
2. Kenneth Casimiro: A17132441
3. Max Shen: A17443771
4. Joel Jijo: A17441654
5. Ethan Kook: A18092777
6. Yanhua Liu : A18059861
7. Collin Walmsley: A18003913
8. Yash Joshi: A16788107
9. Alain Zhang: A17419638
10. Kyle Lou: A17475053
11. Eric Palafox: A18133064
12. Madhav Baghla: A17283136
13. Ashley Thai: A18023446
14. Jacob Zhang: A18033013
15. May Zhang: A17452487
16. Kevin Lebed: A18045149
17. Arihant Jain: A17317535
18. Kendra Chen: A18013421
19. Jasmine Le: A18072854
20. Hamza Ahmed: A17146423

Contributions:

1. **Kenneth Casimiro:** I authored the following files: “gen_delay.m”, “gen_highPitch.m”, “gen_offset.m”, “gen_reverbforSlow.m”, “gen_slowReverb.m”. Additionally, I co-authored the “app1.mlapp” file for the ui. The “gen_delay.m” allows the synthesizer to have delay functionality. The delayed effect is plotted into the ui to be compared to the original signal plot. This feature creates an echo-like effect where the original audio is heard followed by the delayed audio. The delayed effect can be played by the user. The delation is controlled by a knob in the ui which allows the user to delay at a certain amount of time. The “gen_highPitch.m” was created to make a chipmunk style filter that changes an audio file to be more high pitch and increased speed. This allows the user to use a “preset” to manipulate the pitch and speed of an audio signal. The “gen_offset.m” allows the user to change the phase and frequency offset of a signal. This allows the user to see what happens to the signal if they create an offset to the signal. The offset is shown in real time as it plots itself to the ui, allowing the user to compare the offset to the original signal plot. The “gen_reverforSlow.m” and “gen_slowReberb.m” were made to create a slow and reverb preset filter for the user. If the user selects this filer, it changes the signal to have a slowing playback speed and reverberation to the audio signal. Additionally, all of the ui components associated with these features were made by me. Each of these features plots their modified signal and allows the user to play the modified signal’s sound to see what has changed. I also helped the following people to implement/debug their code for the project: Madhav Baghla, Christopher Inzunza, Joel Jijo, May Zhang, Alain Zhang, Kevin Lebed, and Max Shen.
2. **Madhav Baghla:** I authored the file “amplitude-envelope.m” and co-authored the “app1.mlapp” file for the UI.

It generates an amplitude envelope based on the input parameters and applies it to the input signal. It provides a way to shape the amplitude of an input signal over time, allowing for effects such as fading in, sustaining, and fading out. All the UI components associated with this feature were made by me.

3. **Christopher Inzunza:** I authored the volume slider function in the MATLAB GUI. What my code does is take the song's signal and modify it by changing the amplitude, depending on the slider's value. It plots the modified signal in the time domain. Playing the modified signal would result in the song being quieter, louder, or even muted depending on the slider's value.
4. **Jacob Zhang:** I authored the file "plot_freqRes.m" and helped work on "app1.mlapp." The file I authored takes an input signal and the filtered signal and calculates the frequency response of the two signals. It then plots the frequency response as two graphs, one for the magnitude and another for the phase. On the "app1.mlapp," which controls the UI of the app, I made it such that the two plots will be updated with the new frequency response filter whenever the filter drop down menu is changed. I also helped smooth some code out relating to the filters.
5. **Arihant Jain**: I authored the file "pitch_change.m" and worked on "app1.mlapp". My file takes an input ".wav" signal, and uses the Audio Toolbox function "shiftPitch" to change the pitch. In the app file, I implemented a slider UI to change the pitch of the file, which outputs a graph of the new pitch. I also implemented the code in a way that you can change multiple parts of the sound file, and they will all play together.
6. **Kyle Lou:** I created the "audio_fade_in.m" and "audio_fade_out.m" file. They take ".wav" file inputs along with a fade duration and apply a fade-in and fade-out to the beginning and end of the signals.
7. **Collin Walmsley:** I authored the files "gen_sine.m", "gen_cosine.m", "gen_sawtooth.m", "gen_square.m", "plot_wave.m", and "plot_wave_snip.m". The gen files generate a digitized, sampled matrix of the wave (sin, cosine, sawtooth, square) that can be used for further processing or to play as a raw wave. I added features to adjust Amplitude, Frequency, Phase, Sampling Frequency, Duration (in seconds), and Duty Cycle (if applicable, like in Square Waves). The Plot functions serve to simplify plotting functions for testing, and the snip plot function shows only 2.5 cycles of the function to give a better picture of high frequency waves that look like smudges on the graph rather than a wave. I also added the functionality in the main application for the generator dropdowns to be selectable, and their waves to be shown on the graphs in time and frequency domain.
8. **Ashley Thai:** I authored the file "gen_amMod.n" and implemented my function in "app1.mlapp". My file takes an input signal and generates an amplitude modulated signal by calculating the carrier signal and modulated signal based on the input parameters. It then normalizes the signal to create the final amplitude modulated signal. In "app1.mlapp", I helped implement the function by allowing for a sine wave to be generated, then an amplitude modified sine wave to be generated to see the difference.
9. **Kendra Chen:** I wrote the file, "gen_song_examples.m." My method takes an input parameter and generates two song files based off of the inputs, such as sampling frequency, phase, and duration. These parameters are then used to generate two melodies: Ode to Joy and Mary Had a Little Lamb, which are then saved to .wav files for the program to use.
10. **Joel Jijo:** I authored the file "bandpass1.m", "bandstop1.m", and "enhance1.m". My filters take in an audio file as well as a range of frequencies, and for enhance a multiplication factor. Bandstop1, takes in a range of frequencies and then rejects these frequencies. Bandpass1 works in a similar fashion, but instead allows only the frequencies in the set range to show up. Enhance1, also uses a range of frequencies and then takes these range of frequencies and its amplitudes and multiplies the amplitude by the value on the volume slider.

11. **May Zhang:** I authored the files “gen_white_noise.m” and “flanger.m”. “gen_white_noise.m” generates a white noise signal that takes the input of the frequency and duration of the signal, and it can be plotted. “flanger.m” creates a flanging effect to the input signal file and the oscillation frequency can be changed to modify the rate the sound signal fluctuates.

12. **Alain Zhang:** I authored the file “chorus.m” and added its ui components to the main app. “Chorus.m” modulates, generating multiple copies of it with slight variations in timing and pitch. Through the calculation of delay times for each "voice" and the generation of a modulation signal, the function introduces subtle variations in the delay times of the individual voices, resulting in a characteristic "wobble" or "warble" effect associated with chorus in the audio sampling, and plotting it in the time graph and in frequency domain.

13. **Ethan Kook:** I authored the “SpeedAdjuster.m” file. This function takes in an audio file and changes the speed and pitch of the file using the value given by the Speed slider. My function uses the equation $freqChange\ =\ 2^{(SliderValue/12)}$, because there are 12 semitones, so a slider value of 2 (2 semitone increase) equates to a 11.25% change in the frequency. The speed of the audio would increase proportionally to match the change in pitch of the audio. I also co-authored the app1.mlapp file. I integrated my function with this gui to increase the speed with a slider and play at the push of the ‘play modified signal’ button. After the slider is adjusted, the speed-adjusted signal is also plotted in the time domain.

14. **Kevin Lebed:** I authored the “apply_echo.m” file. It takes the audio input signal with an echo delay amount to create an echo then overlay it with the original signal to create an echo.

15. **Max Shen**: I Co-Authored “app1.mlapp”. I implemented some functions into the file and created the UI for it and added the necessary code for it to run. I also authored the “convolution.m” file. It takes two input signals and uses convolution and outputs the resulting signal. You have different options to choose from the input signal such as rect, sin, cos, and triangle. The outputting signal would be displayed on the graph.

16. **Yanhua Liu:** I authored the file “gen_triangleWaveform” that function was written to generate triangular waveform signals

17. **Eric Palafox:** I authored the file “dist_filt.m”. It takes an audio input signal and applies a distortion filter by cutting the low frequencies at a constant amount. Also, I authored “gen_muffled.m” that takes an audio input signal and applies a muffled effect to the input signal by removing stereophonic property.

18. **Yash Joshi:** I authored the file “gen_reverb.m”. It simulates a reverberation effect on an audio signal, converting it to mono if necessary, and employing four delay lines with prime-numbered delay times for minimal periodicity. It dynamically adjusts decay factors based on input `reverb_time` and `room_size`, converts delay times to samples, generates delayed signals with decay, and mixes these with the original signal after scaling by an initial gain. The function aims to mimic the sound reflections of varying room sizes and reverberation durations, enhancing audio with a spatial quality by adjusting parameters such as decay based on the environment being simulated, and ensures the output matches the original signal's length.

19. **Hamza Ahmed:** I authored the file “additiveSynthesis3.m”. This takes an array of input sine waves (allowing the user to specify frequencies, amplitudes, phase shifts, and numerous other options) and combines them to produce audio. It essentially adds sine waves, and outputs audio, while also plotting the graph of the waveform.

20. **Jasmine Le:** I authorized the file “gen_ecg_waveform.m“ and “gen_heartbeat_sound.m”. It generates a simulated heartbeat sound using a simple sin wave with a decaying envelope to mimic a fading effect over time. The ECG waveform uses sinusoidal components to mimic the P wave, QRS complex, and T wave observed by the signals of a real ECG waveform. This plots an ECG wave combined with a heartbeat sound to create a

single signal to represent the combined output of both components.

Files and Authors:

1. **Kenneth Casimiro:**
   a. app1.mlapp (Co-Author)
   b. gen_delay.m (Author)
   c. gen_highPitch.m (Author)
   d. gen_offset.m (Author)
   e. gen_reverbforSlow.m (Author)
   f. gen_slowReverb.m (Author)
2. **Madhav Baghla:**
   a. app1.mlapp (Co-Author)
   b. amplitude-envelope.m (Author)
3. **Christopher Inzunza:**
   a. Volume slider function (Author)
4. **Arihant Jain:**
   a. App1.mlapp (Co-Author)
   b. pitch_change.m (Author)
5. **Jacob Zhang:**
   a. plot_freqRes.m (Author)
6. **Kyle Lou**
   a. Audio_fade_in.m (Author)
   b. Audio_fade_out.m (Author)
7. **Collin Walmsley:**
   a. gen_sine.m (Author)
   b. gen_cosine.m (Author)
   c. gen_sawtooth (Author)
   d. gen_square (Author)
   e. plot_wave.m (Author)
   f. plot_wave_snip.m (Author)
8. **Ashley Thai:**
   a. gen.ampMod (Author)
   b. app1.mlapp (Co-Author)
9. **Kendra Chen:**
   a. gen_song_examples.m (Author)
10. **Joel Jijo**
    a. bandstop1.m (Author)
    b. bandpass1.m (Author)
    c. enhance1.m (Author)
11. **Alain Zhang:**
    a. Chorus.m (Author)
12. **May Zhang**
    a. gen_white_noise.m (Author)

    b. flanger.m (Author)

13. **Ethan Kook**
    a. SpeedAdjuster.m (Author)
    b. app1.mlapp (Co-Author)

14. **Kevin Lebed**
    a. Apply_echo.m (Author)
15. **Max Shen**
    a. Convolution.m (Author)
    b. app1.mlapp (Co-Author)
16. **Hamza Ahmed**
    a. additiveSynthesis3.m
17. **Yash Joshi**
    a. gen_reverb.m
18. **Eric Palafox**
    a. dist_filt.m (Author)
    b. gen_muffled.m (Author)
19. **Yanhua Liu**
    a. gen_grtangleWaveform.m(Author)
20. **Jasmine Le**
    a. gen_ecg_waveform.m
    b. gen_heartbeat_sound.m

POTENTIAL APPLICATIONS OF SYNTHESIZER

Kenneth Casimiro: This synthesizer application has several potential applications, both educational and practical in the field of signal processing and music production. This synthesizer and all of its features can be used as a valuable educational tool for students learning about digital signal processing and audio synthesized. We provided visual representation of various signal processing techniques, such as filtering, modulation, and time-domain manipulations, allowing the user to experiment and learn in a hands-on manner. Additionally the audio effects we created in our application, allows for sound engineers and producers to apply these effects to their individual tracks to shape the sound and create spatial depth and dimension to their music.

Madhav Baghla: Musicians and producers can use amplitude envelopes to shape the dynamics of musical performances, adding expressiveness and emotion to the music. Sound designers for films, video games, and other media use amplitude envelopes to create realistic or imaginative sound effects, such as footsteps, explosions, and environmental ambiances. In spatial audio systems, amplitude envelopes can be used to simulate the perceived distance and direction of sound sources, contributing to a more immersive listening experience.

Christopher Inzunza: Volume adjustment can be used almost anywhere for anything. In our daily lives, we could change the volume of a YouTube Video, our Spotify music, or a phone call to a comfortable volume. In the navy, the seamen might need to adjust their headsets’ volume to properly listen to commands from other ships. At your favorite music artist’s performance, they have to adjust the volume in their earpiece and in the speakers so nothing is too loud or too quiet. You may need to adjust the volume to your morning alarm to make sure it’s loud enough to wake you up for class or work. Almost anything that contains audio signals can change its volume to the user’s preference.

Arihant Jain: The ability to change the pitch of an audio has many applications, such as editing voice overs to sound funny or cartoonish, or to sample other music while making your own. This is used often by animation studios to make voices sound high pitched (like mickey mouse). Sampling songs by changing pitch is often used in the hip-hop industry to mix music.

Jacob Zhang: The frequency response graphs can give the user a quick observation of how the filter is modifying the original signal. A quick glance is all the user needs to gain an intuition of the magnitude and phase effect of the filter on the frequencies of the original signal.

Kyle Lou: Audio fade-in and fade-out can be used to transition between two segments without seeming clunky. If we had no fade-in and fade-out transitions, the changes would be sudden and be unpleasant. This provides a smooth transition and can be used in music, video editing, etc.

Collin Walmsley: Generating sampled waveforms is important in all matters of digital electronics and processing signals. It allows us to fluctuate existing signals and fine tune them. For instance, we can use a square wave to shift a note’s frequency up and down at intervals to create some funky beats, as heard in electronic dance music, and many songs today. Adding more signals together allow for more natural sounding notes like we know from physical musical instruments. These examples are mostly in audio format, but also apply for communications and any sort of signal we wish to generate and manipulate.

Ashley Thai: Amplitude Modulation is an important technique used in various circumstances, most popularly known for its use in Amplitude Modulation Radio, or AM radio that radio stations use. Radio stations are typically assigned a range of frequencies where users can “tune in” to that station and listen in. Amplitude modulation is used to essentially allow for any audio to transmit to this frequency range in order to not disrupt any other radio station, who is assigned a different range on the frequency spectrum.

Kendra Chen: The ability to introduce phase shifts and set the amplitude and sampling frequency is crucial when experimenting with music files. This allows the user to generate audio signals using sine waves and combining them to create melodies.

Joel Jijo: Utilizing a bandpass and bandstop filter is a way of filtering out unwanted frequency signals from a signal. This has many applications in the outside world, and in a synthesizer is utilized as a tone shaper, which helps in aiding the sound of an acoustic guitar. Additionally, the enhance filter can be used to tone down a range of frequencies or enhance them, rather than getting rid of them altogether.

May Zhang: The white noise generator can be applied to test and validate signal processing algorithms or systems since it contains all frequencies and does not favor any particular range. It can also be used in music synthesis for sound effect and in sleep aid devices to mask disturbing noise. The flanger effect can add a depth and “movement” to the sound, and it is commonly used in electronic music to make the sound more complex and interesting. It’s also used in video games and filming to give an alien and futuristic atmosphere.

Ethan Kook: Changing the speed of an audio is a common way for musicians to change up their music and experiment with different sounds. Many artists today sample songs from other artists, and speed it up or down to incorporate into their own music. Another potential application is to slow or speed up audios to teach musicians about tempo.

Kevin Lebed: Creating an echo of audio can be very useful in creating music for a movie or a show if you need to build tension or have audio in a space that would echo like a cave or building.

Alain Zhang: A real-world application of the chorus effect is in music production and audio engineering. By creating multiple slightly delayed and modulated copies of an audio signal, the chorus effect enriches the sound, adding depth and texture. This effect is commonly used to make vocals and instruments sound fuller and more vibrant, particularly in genres such as pop, rock, and jazz. It also finds applications in live performances and recording studios to create a sense of spaciousness and richness in the mix. Additionally, the chorus effect is utilized in electronic music production to enhance the complexity and interest of synthesized sounds, contributing to the overall atmosphere and mood of the composition.

Yanhua Liu: The gen_triangleWaveform function is used to generate a triangle waveform signal with a sampling frequency of fs. The user can specify parameters such as amplitude, frequency, phase and duration of the waveform.It can be applied to sound synthesis, music production, signal processing and other fields. Triangle waveform signals have a wide range of applications in synthesizers and audio processing.

Eric Palafox: The function ”dist_filt.m” alters the input sound signal and deforms the original waveform. This is used in music and when creating sound effects in many real life applications such as music, video, etc. The function “gen_muffled.m” is an important application and is also used in similar applications mentioned above. The muffled effect will make it seem that the signal is far, or even in the next room over, giving a muffled signal.

Hamza Ahmed: The applications of the addSynthesizer3 function are numerous. Because it allows you to modify and combine waves, you can create many beautiful musical sounds. This is useful to the music industry and musicians who wish to experiment with new sounds. In addition, there is a randomness factor built within the function if the user wants to enable that. This creates a level of randomness that hopefully creates unexpected but beautiful sounds. This may help musicians who wish to create new and beautiful sounds, but cannot think of how to do that.

Yash Joshi: A potential real-world application of the gen_reverb function is in music production and sound design, where it can be used to enhance recordings by simulating the acoustics of different environments. For example, music producers can apply this function to dry vocal tracks or instruments to give them in a big dance hall, a jazz club, or a music concert, without the need to physically record in these spaces. By adjusting parameters like reverb_time and room_size, producers can tailor the reverberation effect to suit various musical styles, enhancing the listener's experience by creating a sense of space and dimension.

Max Shen: There are various applications of convolution in different fields. It takes in two inputting signals and convolute them into a signal outputting signal. In terms of sound design, it can be used to blend different sounds together for creative purposes. It can also be used for sound shaping and filtering.

Jasmine Le: This function application stimulates an ECG heartwave waveform including a heartbeat sound to it. This application can be applied to medical education and software development. Medical professionals and students can utilize this to enhance their understanding of cardiac physiology and diagnostics. Researchers can utilize data to test and validate medical devices, algorithms, and applications for the ECG signals and monitoring.

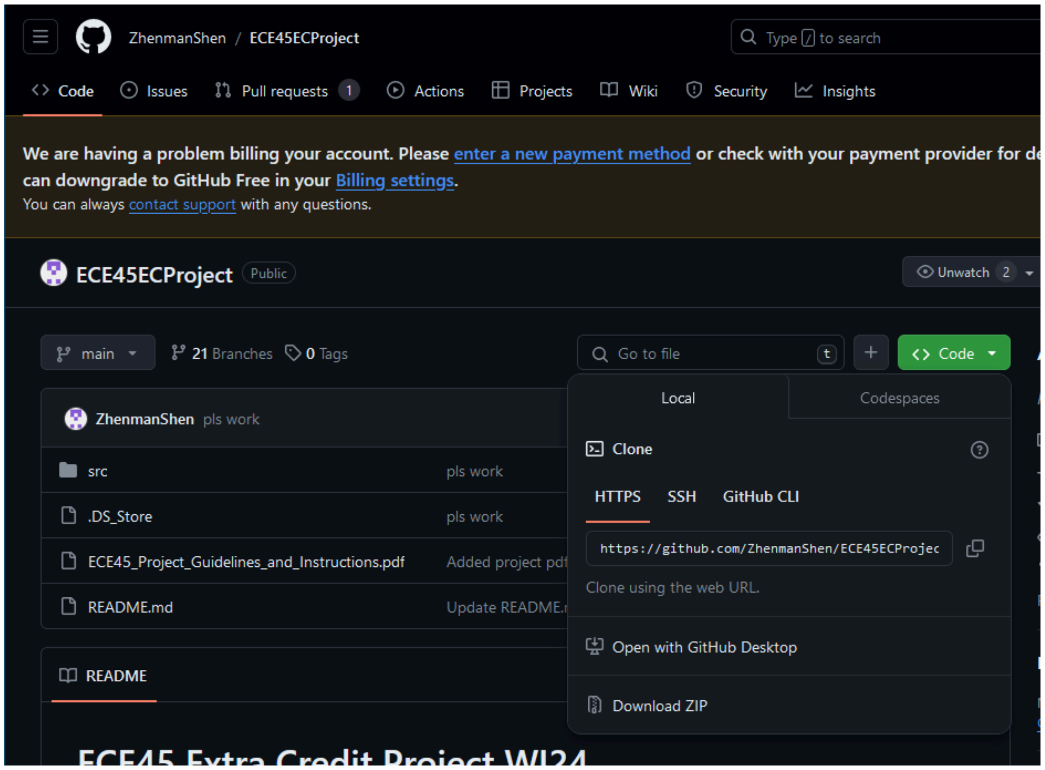
SETUP & USAGE GUIDE

a) How To Install The Synthesizer

i) Turn on the power button to your pc or device.

ii) Open the web browser, either chrome or fire fox or explorer or safari.

iii) Installing source code zip file from dropbox or github.



We are having a problem billing your account. Please enter a new payment method or check with your payment provider for de
can downgrade to GitHub Free in your Billing settings.
You can always contact support with any questions.

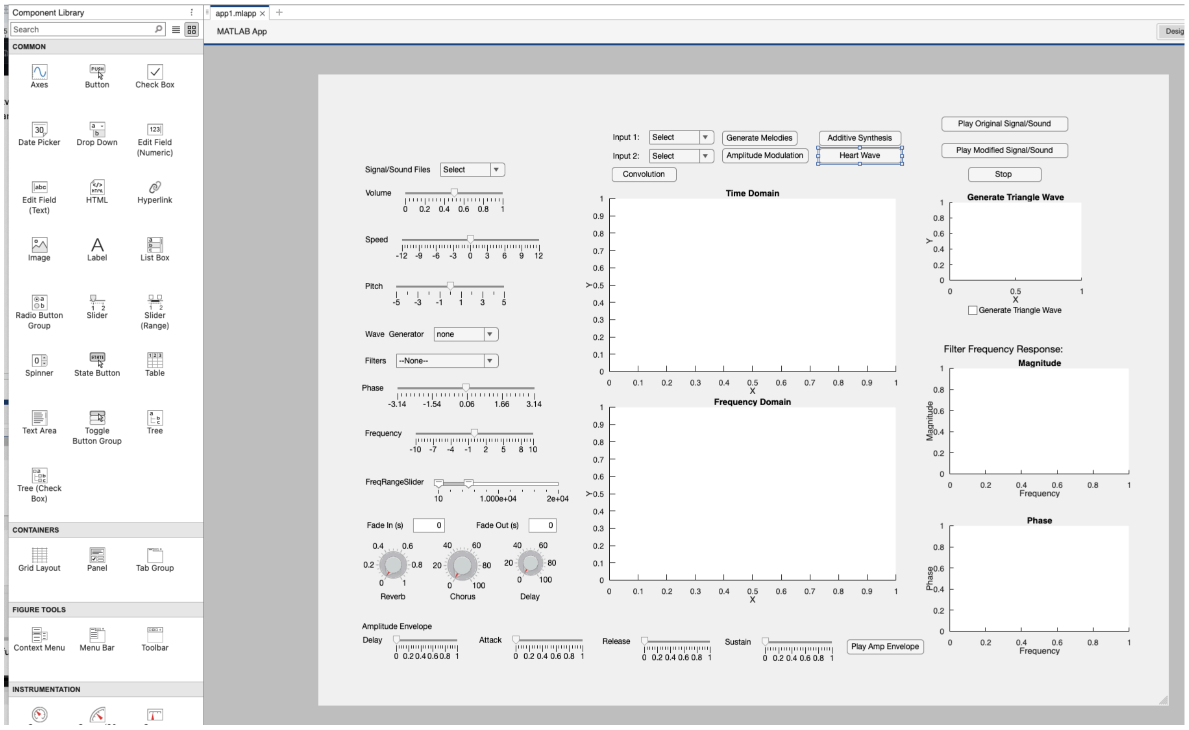
https://github.com/ZhenmanShen/ECE45ECProjec

Clone using the web URL.

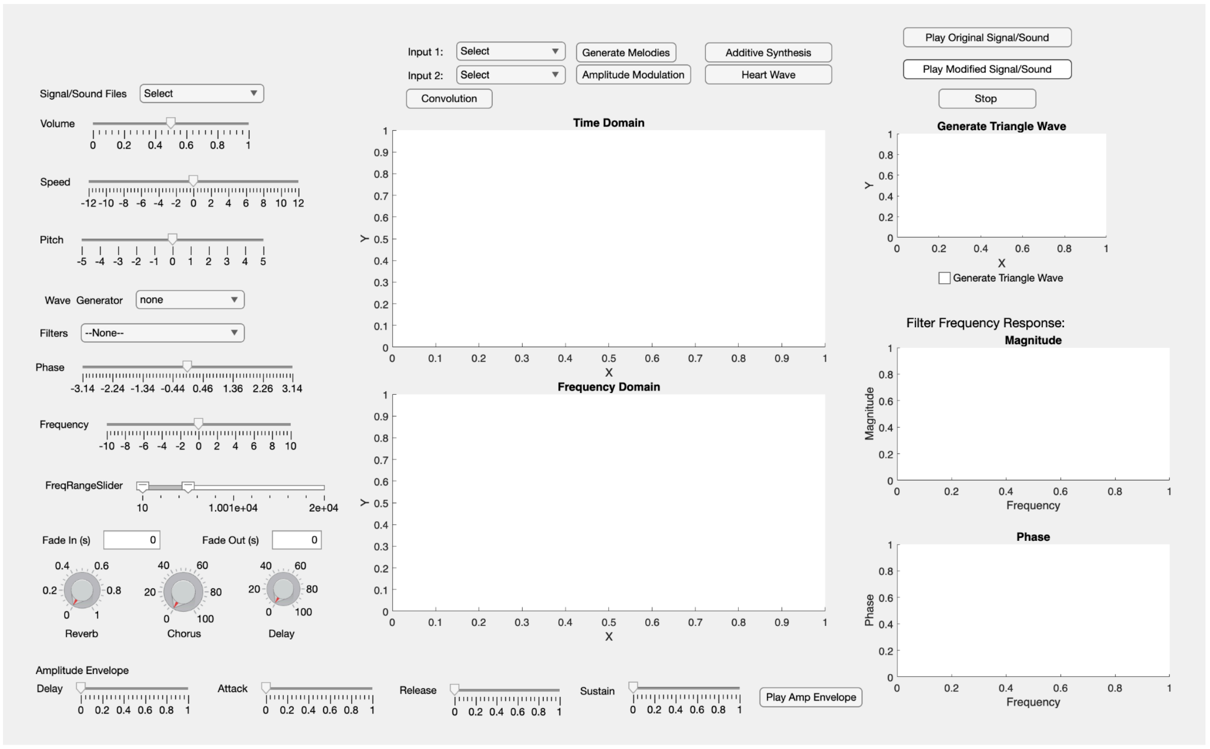
iv) Either open Matlab online or download Matlab software if you have a subscription and set it up

v) If asked, download Audio Toolbox from MATLAB.
Link can be found here: https://www.mathworks.com/products/audio.html

vi) In the application or webapp, unzip the source file and click on the app1.mlapp, and once a new window pops up select run



vii) Congrats! You’ve opened the app. I hope you have fun! Make sure to check out the basic usage guide for more info



b) Basic Usage

i) Select a Signal or Sound file.

(1) Sin Wave

(2) Sad Valentine by No Vacations

(3) White Noise

(4) Mary Had a Little Lamb

(5) Ode to Joy

ii) Apply filters and adjust audio.

(1) Sin Wave is only for graph demonstrations. Specified ui components will be enabled to allow you to change the sin wave. These ui components are the phase and frequency slider, amplitude envelope, and wave generators.

(2) White Noise is only a graph demonstration.

(3) For Sad Valentine by No Vacations. Use the ui components that will be enabled to modify the signal. Use the slider and knob components to modify the sound. Each modification will plot their respective changes to the graphs. You can play the modified signal through the “Play Modified Signal/Sound” button.

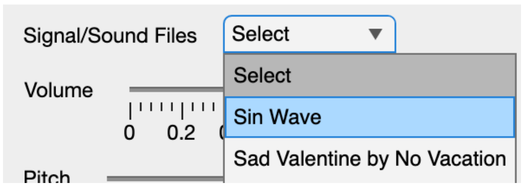
(4) Mary Had a Little Lamb and Ode to Joy are graph/sound demos. They will plot their signal and play their respective sound but no modifications will be applied.

iii) Play around and test all the synthesizer features!

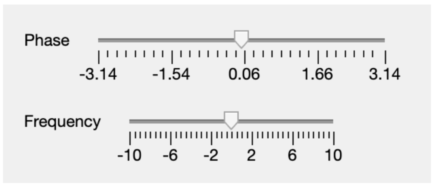
c) Advance Usage

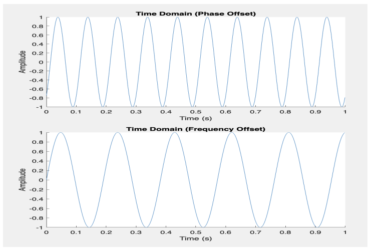
i) Phase and Frequency Offset

To use the Phase and Frequency Offset, click on the Signal/Sound File and Select “Sin Wave”.



Once doing so, you will see that the graphs will be populated in time-domain, a basic sinusoidal signal.From there, please use the sliders below to modify the offset that will be applied to the signal.

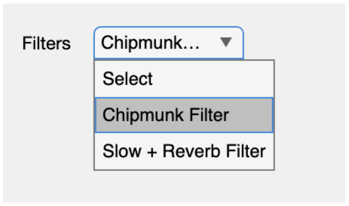


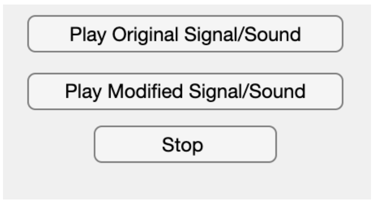


By changing the phase offset, we can see that the waveform changes relative to its original starting point. With a frequency offset, we can see how it stretches and compresses the waveform along the time axis, determining how much it oscillates.

ii) Filter: Chipmunk

Select “Sad Valentine by No Vacation” in the Signal/Sound Files.
From there, select the “Chipmunk Filter” from the Filters dropdown menu. The graphs in the middle will be repopulated with the original signal at the top and the modified signal at the bottom. Play the original and or modified sound on the right.

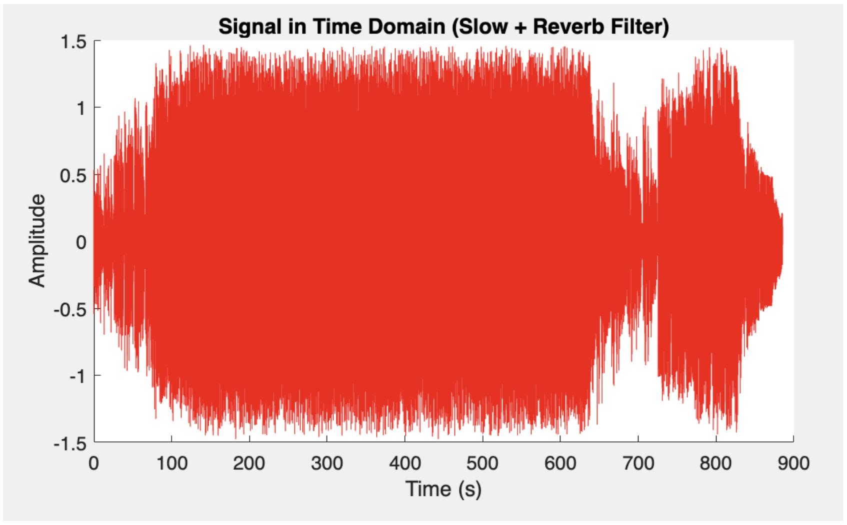




iii) Filter: Slow + Reverb

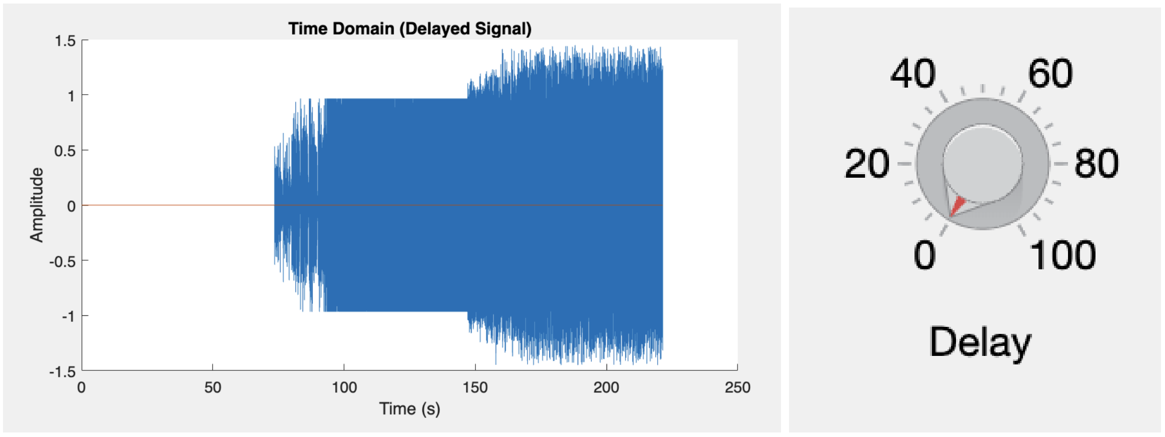
For the “Slow + Reverb” filter, you would do the same as the Chipmunk filter but you would select the Slow + Reverb option in the filters dropdown menu. From there, you will see the graphs being re-plotted with the original signal on top and the modified signal at the bottom.

You are able to play the original and or modified sound on the right.



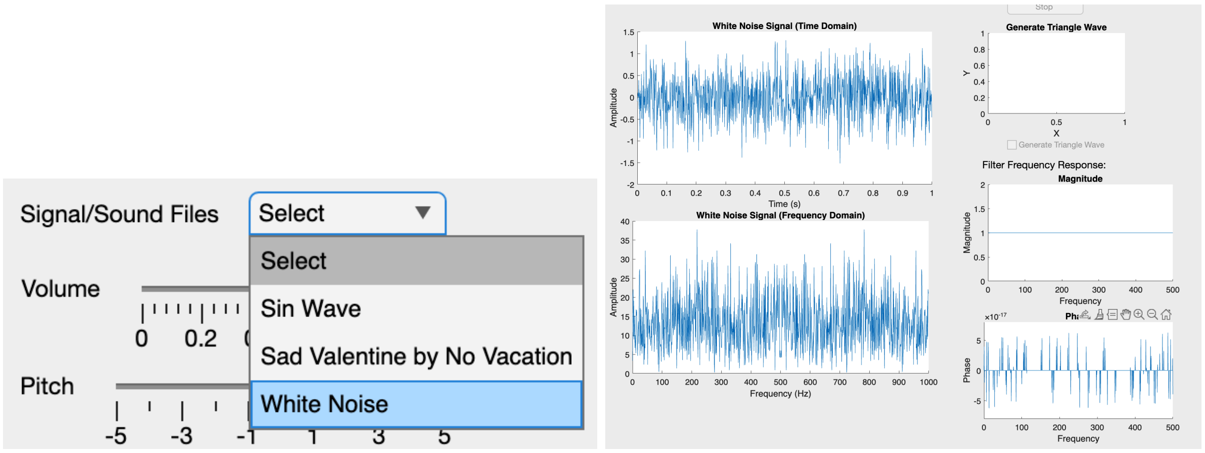
iv) Delay

Select “Sad Valentine by No Vacation” in the Signal/Sound Files. With delay, there will be a knob ui at the bottom left labeled “Delay” where you are able to change the values to. When doing so, the bottom graph will be re-plotted with the delay that you chose from the knob. When playing the modified signal, it will play the original sound and with some delay, will echo the delayed sound under the original sound.



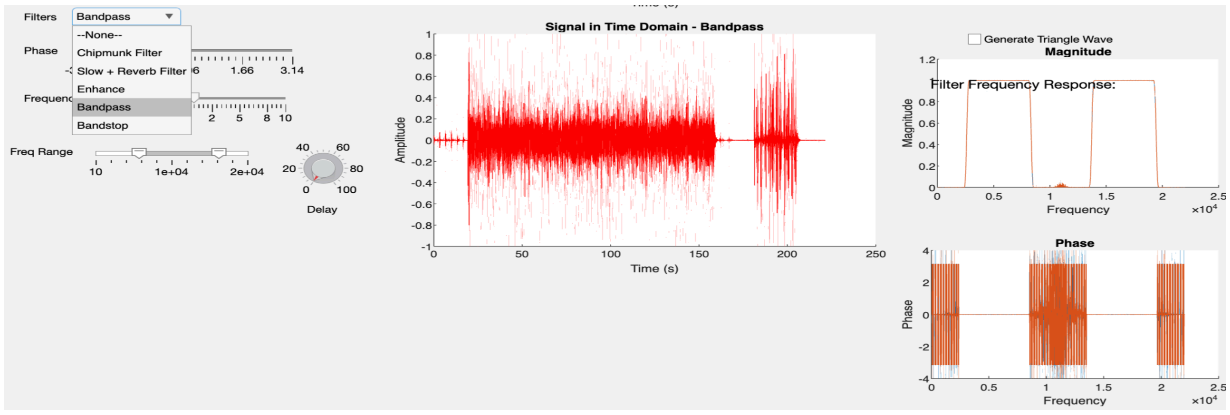
v) White Noise

Select “White Noise” in the Signal/Sound Files. A random white noise will be plotted in its time and frequency domain. It will cover frequencies of all ranges.



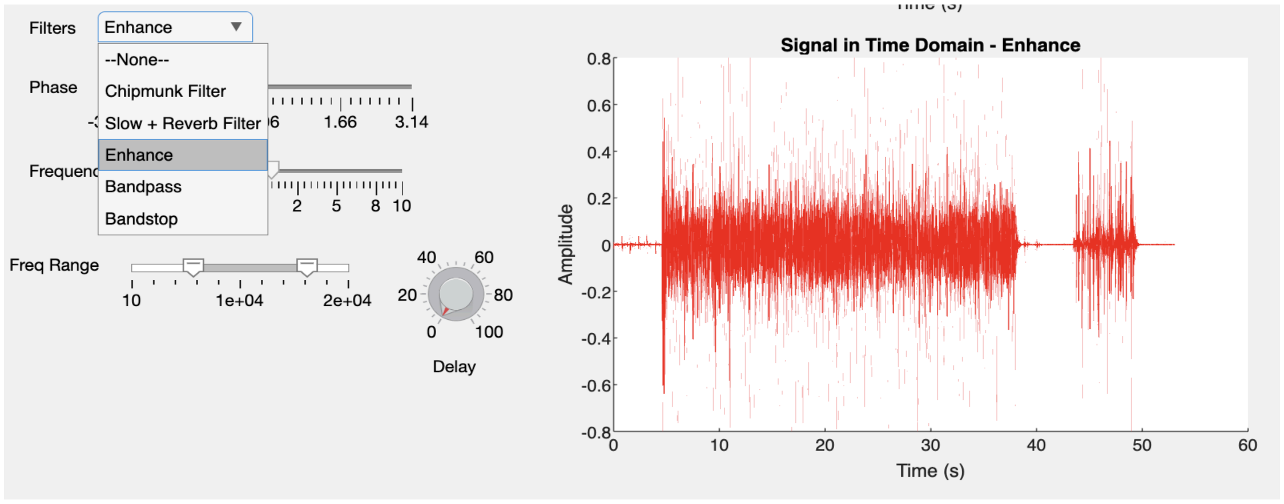
vi) **Bandpass/Bandstop**

Select “Sad Valentine by No Vacation” in the Signal/Sound Files. Then use the frequency range slider to get the range of frequencies to be applied. Then select either bandpass or bandstop, using the filters drop down menu. Due note that depending on the amount of frequencies you cut in the case of bandstop, or stay in the case of bandpass it would take longer for the graph in the time domain to load, as well as for the song to load. Utilizing the buttons on the right you can play the modified or unmodified file.



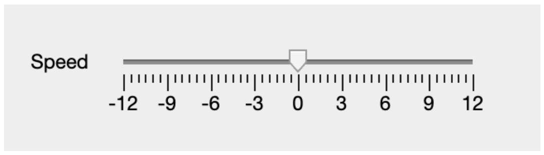
vii) **Enhance**

Select “Sad Valentine by No Vacation” in the Signal/Sound Files. Then use the frequency range slider to get the range of frequencies to be applied. Then select enhance, using the filters drop down menu, then use the volume knob to adjust how much you want to enhance the selected frequencies. Due note that depending on the amount of frequencies you choose, it will take a long time to load in whether it be the graph or the song. Utilizing the buttons on the right you can play the modified or unmodified file.



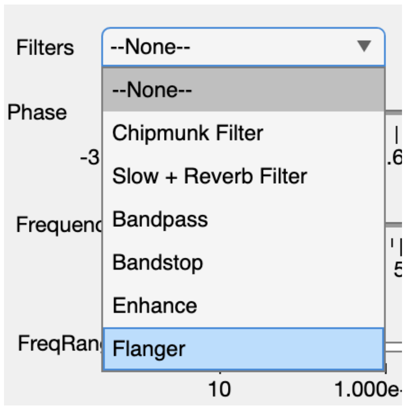
viii) **Speed**

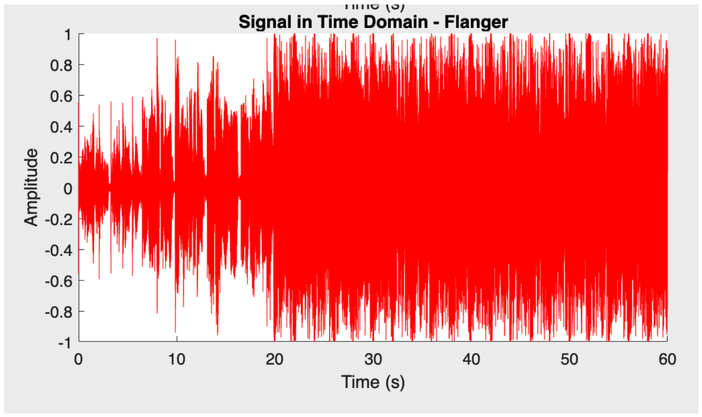
Select an audio of your choice from the drop down menu. Then use the speed slider to either increase or decrease the speed and pitch of the function. A value of 2 would increase the frequency of the audio by 11.25%, while conversely -2 would decrease it by 11.25%



ix) **Flanger**

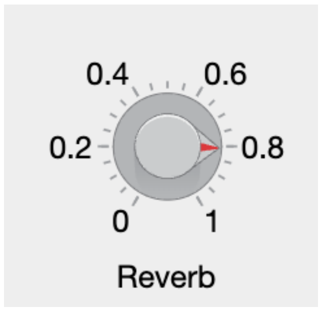
Select “Sad Valentine by No Vacation” in the Signal/Sound Files. A graph will then be populated in time-domain. Then select “Flanger” in the Filters. It will create a flanging effect to the sound file, as shown in the graph below.
You are able to play the original and or modified sound on the right.

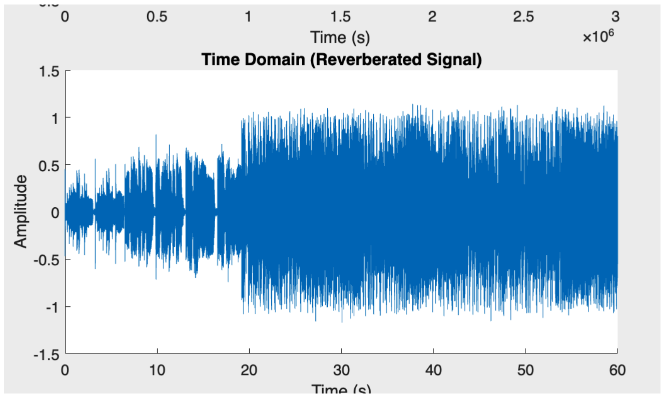




x) Reverb (Yash Joshi)

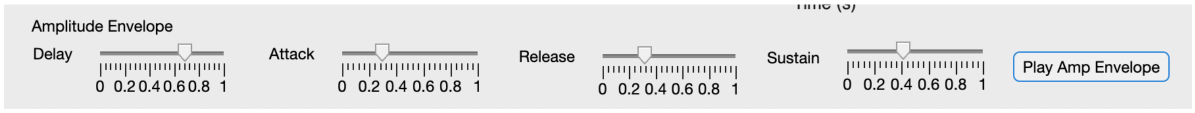
Select an audio of your choice from the drop down menu. A graph will then be populated in time-domain. Then select “Reverb” in the Filters. It will create a reverb effect to the sound file, as shown in the graph below. You can adjust the amount of reverb using the knob below.

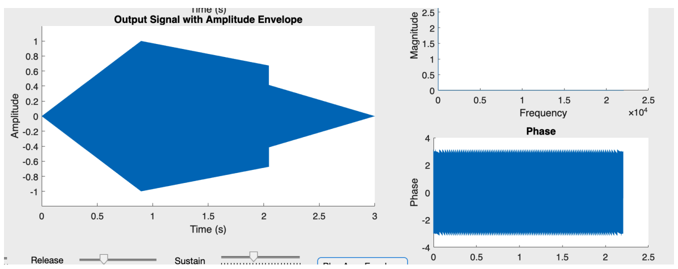




xi) Amplitude Envelope

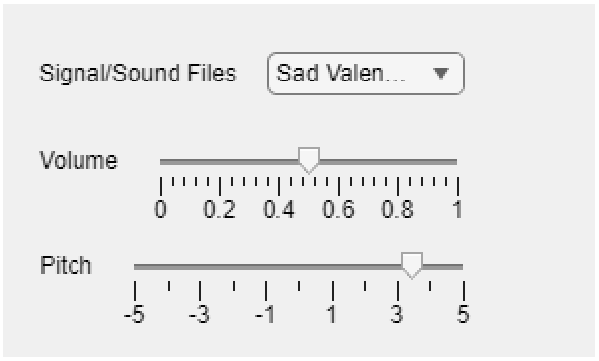
Select Sin Wave and change the following sliders to modify the Amplitude Envelope. Click on “Play Amp Envelope” to plot the graphs and play the modified sound.

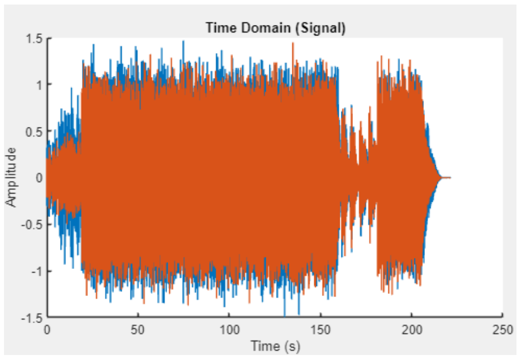




xii) Pitch

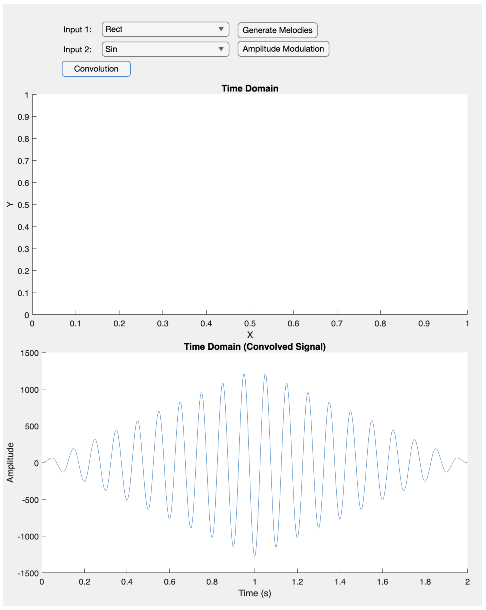
Select “Sad Valentine by No Vacation” in the Signal/Sound Files. Once doing so, you will see that the graphs will be populated in time-domain, a complicated wave which is about 200 seconds long. From there, please use the sliders below to modify the pitch that will be applied to the signal. You can then play the modified signal.





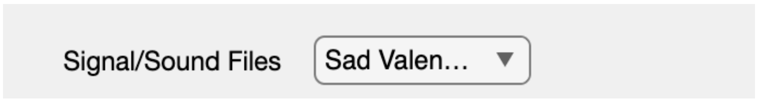
xiii) Convolution

Select the two options from the drop down menu that you want as your inputting signal (input 1 and input 2). Then press “Convolution” and the output signal will be displayed in the time domain.

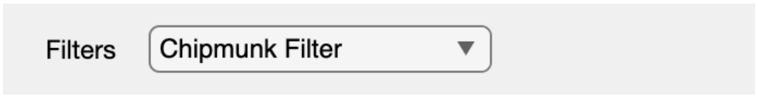


xiv) Frequency Response Graph

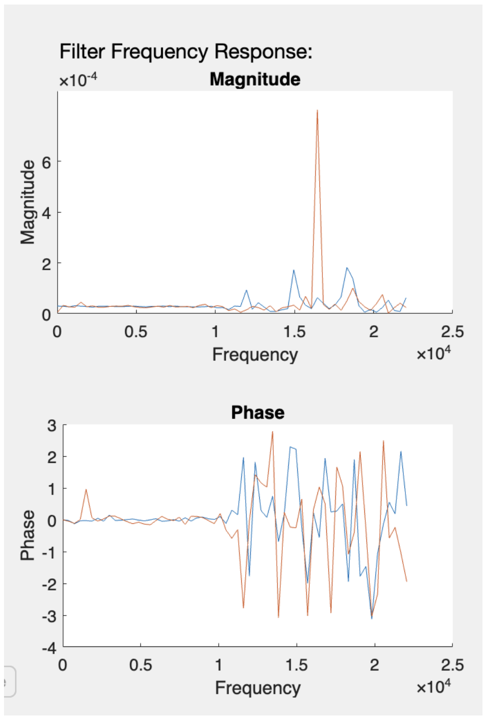
Select a signal/ sound file.



Then, select a filter.

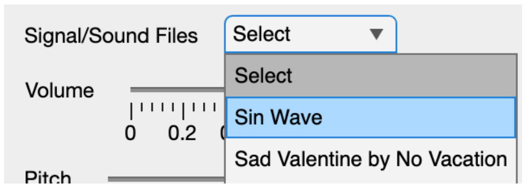


Wait a moment, the frequency response graphs should be updated!

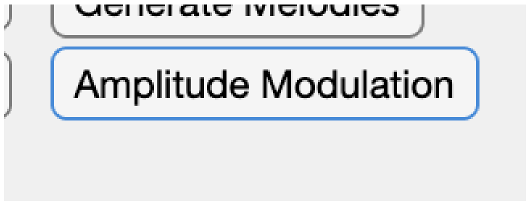


xv) AM Modulation

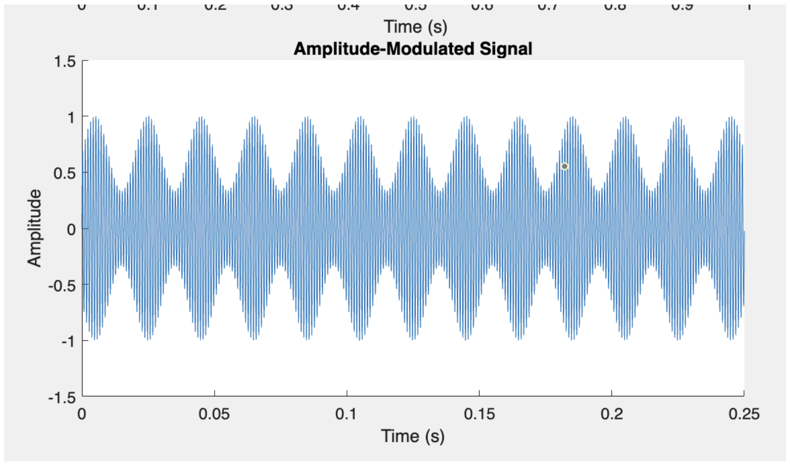
Select “Sin Wave” from the Signal Files Drop down menu.



Push the “Amplitude Modulation” button near the top of the graph to generate an amplitude-modified wave.

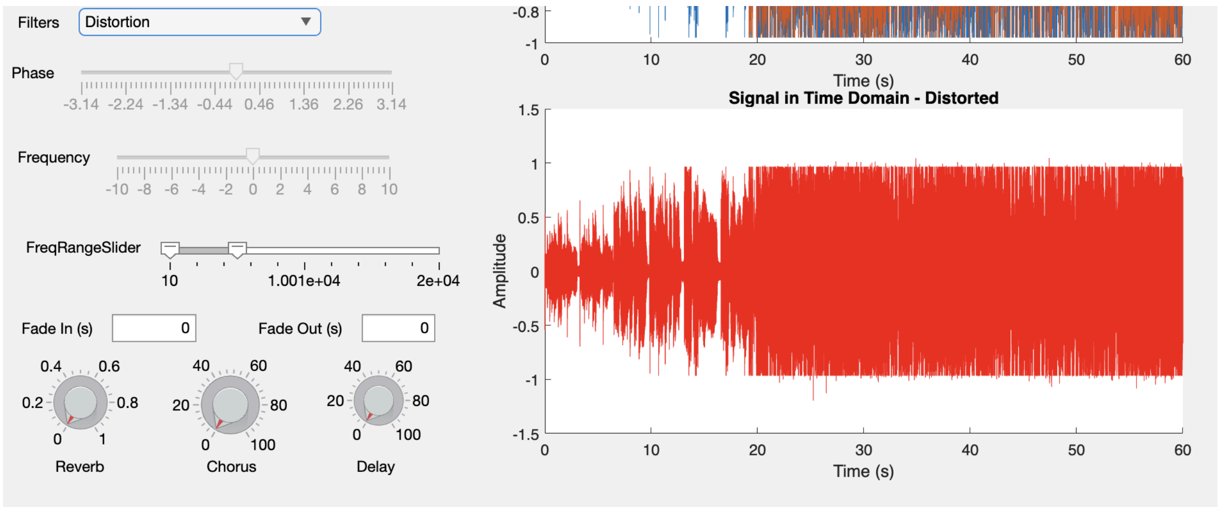


The Amplitude-Modulated Signal will be generated below.



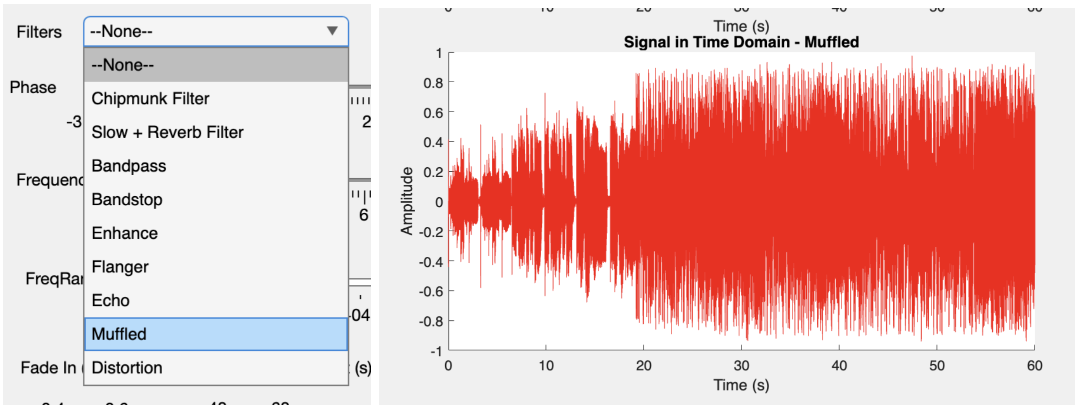
xvi) Distortion Filter

Select “Sad Valentine by No Vacation” in the Signal/Sound Files. After that click the filter drop-down menu and select the “Distortion” filter. Then the filter will be automatically applied and you can hear the distortion if you click play modified sound.



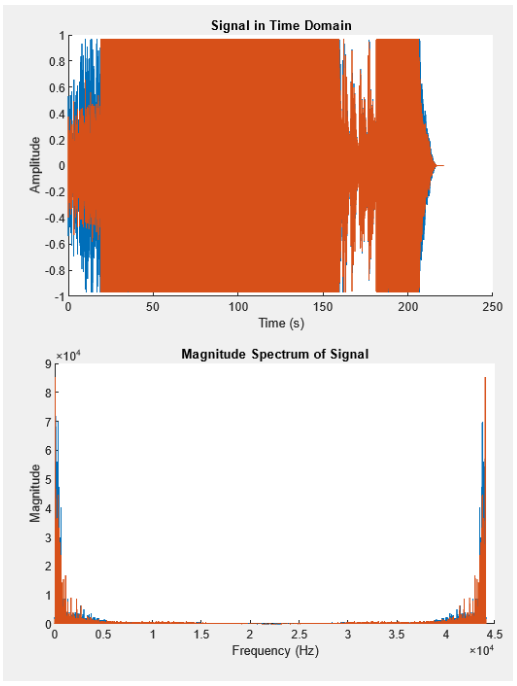
xvii) Muffled Filter

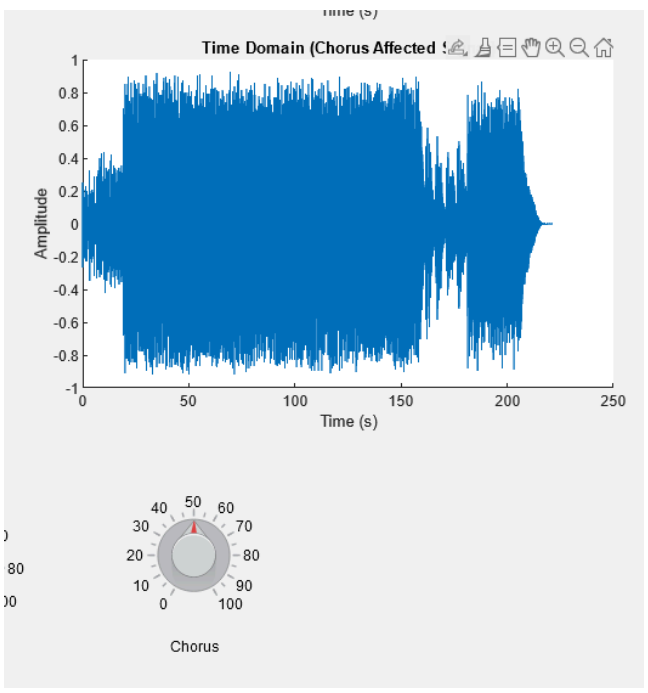
Select “Sad Valentine by No Vacation” in the Signal/Sound Files. After that click the filter drop-down menu and select the “Muffled” filter. Then the filter will be automatically applied and you can hear the muffled signal if you click play modified sound.



xviii) Chorus

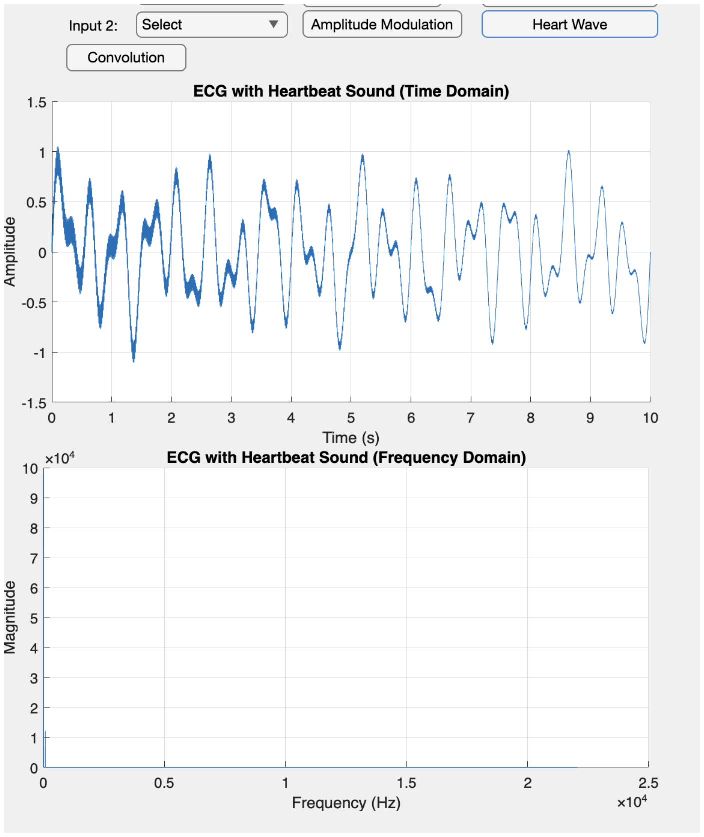
For the chorus functionality, if we choose a sample signal or sound file like “Sad Valentine”, the method will copy a specific segment and create multiple delayed and modulated copies of it (with each turn of the knob being 10 times more), then mixing these copies with the original signal. This process introduces variations in timing and pitch, simulating the effect of multiple performers or instruments playing the same part with slight timing and pitch differences.





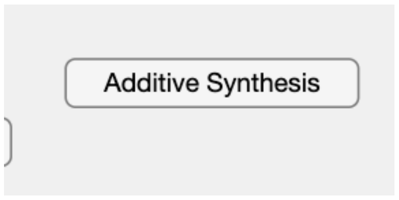
xix) Heartbeat Wave

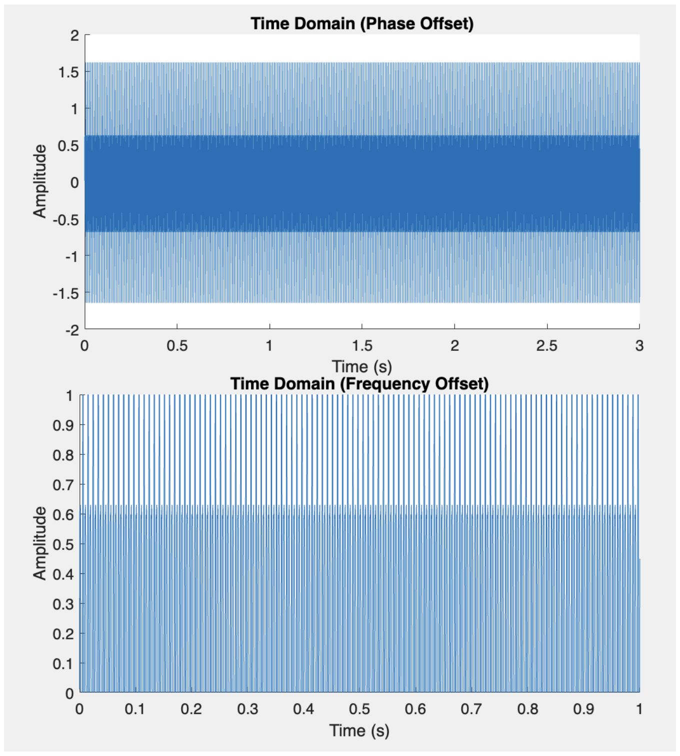
For the heart wave functionality when you select the heart wave dropdown section, there will be an ECG graph and waveform that will show a pop-up followed by a heartbeat sound playing in the background. This mimics an ECG wave and sounds similar/closely related to the ECG machines seen in medical settings.



xx) Additive Synthesis

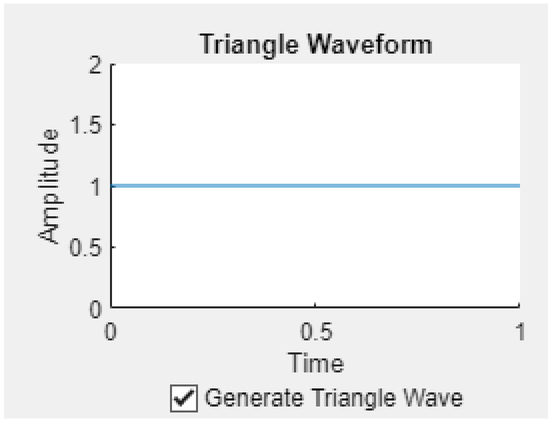
The additive synthesis functionality allows the user to combine sound waves. Click on additive synthesis, and adjust the frequency and phase sliders to vary the sound produced. This has the potential to create beautiful sounds by combination. Play around with it, and have fun!





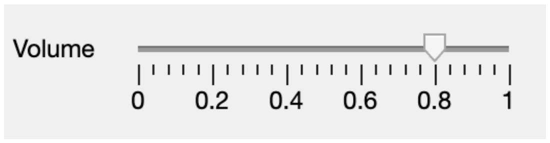
xxi) Triangle Waveform

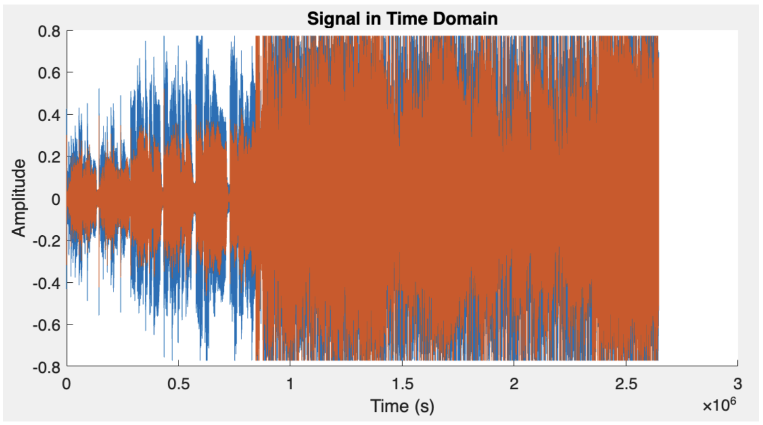
Click on the "Generate triangle wave" will show up the wave from the graph.Note that not all audio will appear, this test only limits the amp to 150-220 hz.



xxii) Volume

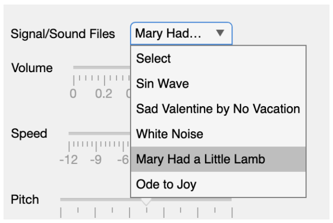
After selecting the sound file, use the volume slider to change the volume of the signal. The number you select will be the new amplitude of the signal. The higher the amplitude, the higher the volume. After choosing your desired volume, hit the "Play Modified Signal/Sound" button and the volume will be different. You can also look at the signal in the time domain graph which reflects the new amplitude.

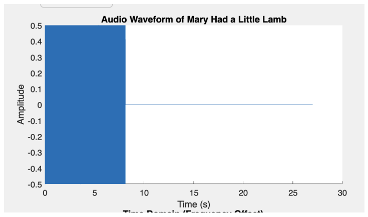




xxiii) Generate Melodies to "Mary Had a Little Lamb" and "Ode to Joy"

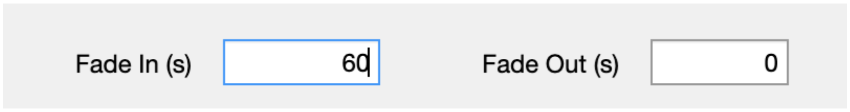
Click on the "Generate Melodies'' button in the top middle of the application. This will generate melodies to Mary Had a Little Lamb and Ode to Joy. Both of which will play.
In addition, you can check the plot for these melodies by selecting the dropdown menu for Signal/Sound Files to the selected melodies.

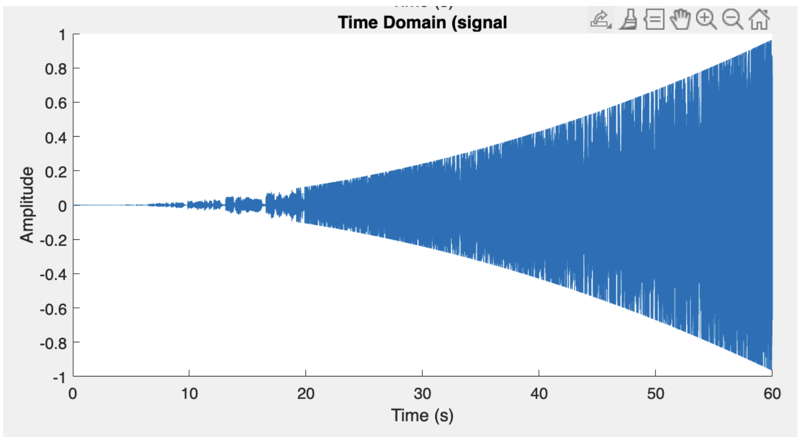




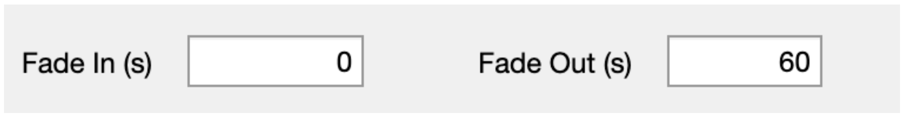
xxiv) Audio Fade In/Out

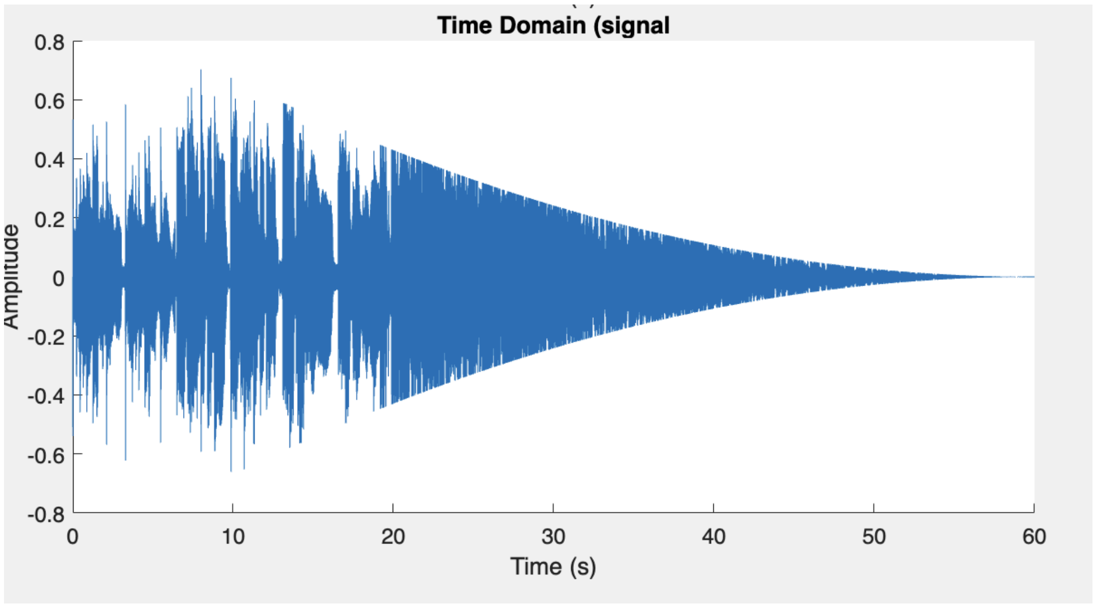
For the fade-in functionality, we can input a number (in seconds) into the fade-in textbox. The program will apply a quadratic fade-in to the signal upon pressing "enter." The edited audio will play in the background, and a time domain graph showing the audio with fade-in applied to it will be displayed. As we can see here, for an extreme example with audio wave of 60 seconds, the quadratic scaling of the wave amplitude is stretched for the entire duration.


Fade In (s) 60
Fade Out (s) 0



Using the fade-out functionality is the same as using fade-in. It just applies a quadratic fade-out Effect to the audio signal. An extreme example where the fade-out effect is applied for the entire duration is showed below.

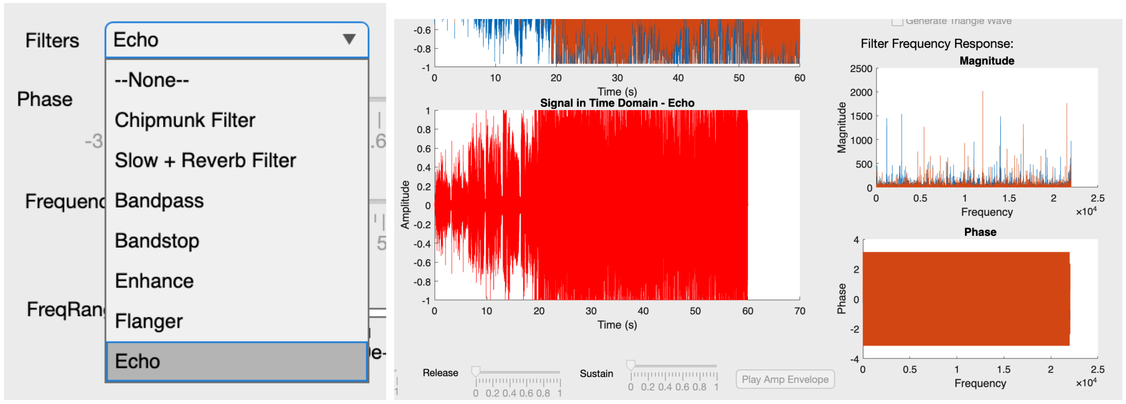

Fade In (s) 0 Fade Out (s) 60



In the event that the user inputs a value larger than the duration of the audio signal, we simply set the fade length to the duration of the audio. If the event inputs a negative value, we do not edit the signal at all.

xxv) Echo

Select "Sad Valentine by No Vacation" in the Signal/Sound Files. After that click the filter drop-down menu and select the "echo" filter. Then the filter will be automatically applied and you can hear the echo if you click play modified sound.


Filters: Echo

APPLICATION OF CLASS MATERIAL

Kenneth Casimiro: For this project, I was able to apply numerous concepts learned in class, from phasors, to Fourier series and transforms to time-domain analysis, sampling, and impulse response. For instance, I generated sinusoidal waveforms to utilize the phasor method to represent the signal's amplitude and phase angle. The Fourier Transform is employed to analyze the frequency content of my audio input signals to be analyzed in the frequency domain. Time-domain analysis was utilized to visualize the waveform's behavior over time, allowing me to understand signal characteristics and effects of signal processing techniques. When I developed delay and reverb features for my filters, I utilized my knowledge of impulse response and convolution techniques, where the response of a system to an impulse/sample is convolved with my input to produce a modified output. Additionally, sampling concepts were employed in manipulating signals, altering their properties, and generating new sounds.

Madhav Baghla: Employing time-domain analysis, I visualized the behavior of the waveform over time, gaining insights into signal characteristics and the effects of signal processing techniques. The sampling frequency is crucial for determining the time durations of each envelope phase and ensuring accurate temporal manipulation of the input signal. It allows for precise control over the shape of the amplitude envelope relative to the time domain of the input signal.

Christopher Inzunza: In this class, I understood what each aspect of a signal meant and what would happen if they were modified. The amplitude of a signal is the volume. A large amplitude results in high volume, and a lower amplitude results in low volume. I applied that information to this project by multiplying the signal's data by the value that the volume slider pointed at.

Arihant Jain: I primarily used the effects of changing pitch on the speed of the audio, and gained insight into how to change the period of the wave so I can change pitch without affecting the speed of the entire audio file.

Jacob Zhang: To plot the frequency response graphs, I had to understand fourier transforms and frequency response. Understanding bode plots was also essential to my additions to this project.

Kyle Lou: In class, I learned that the key characteristic of a signal is its amplitude, which defines the intensity of the sound. By adjusting the amplitudes, I was able to create a linear fade-in and fade-out effect in the audio.

Collin Walmsley: I used the concept of sampling to generate the wave generation functions. I create an analog function, then apply a sampling frequency (a trail of delta functions) to create a digital representation of the signal. This can then be used to do whatever digital manipulation desired. In the graphing domain, I used the concept of Fourier Transform to convert the time representation of the signal into the frequency representation.

Kendra Chen: I used the concept of signal generation in the time domain to design this function, where each note is represented as a sine wave with a specific frequency. I used the Nyquist sampling theorem in my program, using the sampling frequency to convert continuous sine waves into discrete signals. My program's parameters, amplitude and phase, demonstrate how amplitude can alter the perceived loudness of a sound file, and how phase shifts can alter the signal's alignment in time.

Joel Jijo: We learned the importance of a bandpass and a bandstop filter in class, and to implement a function that would work with this concept requires understanding how it works and the inputs it needs. The enhance filter utilizes the same principles of understanding fourier transforms, but instead of removing these frequencies altogether, rather amplifying them or diminishing them. This is done by multiplying the amplitude by a factor, which corresponds to the fact that it would then affect the waveform by linearity.

Ethan Kook: Using time-domain analysis, similar to what we learned in class, I was able to plot the modified signal in the time domain after an increase in frequency and pitch.

Kevin Lebed: This is an application of modulation as we are creating a duplicate of the signal at a lower "volume" and then moving it back on top of the original signal.

Yanhua Liu: In my project, I applied concepts such as sampling frequency, signal synthesis, time-domain and frequency-domain representation, and filters to manipulate audio signals. These concepts helped me generate, visualize, and modify audio signals effectively, improving my understanding of signal processing principles.

Yash Joshi:
In this project, I integrated audio processing techniques learned from our class to create a reverb effect. By generating sinusoidal waveforms and employing the phasor method, I represented signal amplitude and phase angle. Utilizing Fourier Transforms allowed me to examine the frequency content of audio inputs, transitioning between time-domain visualization to understand waveform behaviors and the effects of signal processing modifications. The development of delay and reverb functionalities used my understanding of impulse responses and convolution techniques.

Eric Palafox: I applied concepts from the ECE 45 class such as sampling frequency, time domain, frequency domain, and manipulating audio signals. Also, Fourier transforms were used when obtaining the input of the sound signal. This certainly gives a good idea of how electrical engineering works and how to apply each of the principles learned.

Ashley Thai: Through this project, I was able to resolidify the concepts of amplitude modulation we went over in class. By integrating the concepts I learned about how a cosine function can modulate another signal to fit within certain frequencies helped me understand how audio signals work and transmit in our everyday items, like our car radios.

Hamza Ahmed: This function uses the idea of systems that take in an input signal, modify it, and produce an output signal. My function in particular adds different signals and may phase shift, adjust amplitude, etc. This function uses time domain, frequency domain, signal synthesis, and many other concepts learned in class. This is useful to electrical engineers and musicians alike.

May Zhang: Signal generation in the time domain is used to develop the filter function for flanger. It applies the modulation techniques like the amplitude modulation we learnt in class. The depth and amplitude of the sound is modulated by adding a delay and oscillating frequency to it and creates a whooshing sound. The project reinforces my understanding of signal modulation in real life applications.

Jasmine Le: By making the heart wave function, I was able to apply the principles of signal processing concepts we learned in this class. In the ECG waveform, I was able to utilize a greater understanding of sinusoidal components to mimic the complex patterns observed in the signals. The visualization of the waveform demonstrates the understanding of basic signal characteristics that we have come across in this class.

Max Shen: Convolution is a big part of the class. We took a few weeks to learn how to calculate convolution and what it means in terms of the time domain and the frequency domain. We learned a lot of properties about it that would help us solve convolution problems. Therefore, with MATLAB, it would be way easier to see what the convolution looks like on a graph and help us solve some of the convolution problems. It also allows us to understand more visually by looking at what the convolution is doing with the two inputting signals.

Alain Zhang: By making the chorus function, I was able to reinforce my understanding of sampling frequency and amplitude modulation, as well as integrate concepts of setting a specific filter/range to sample from using Matlab. This helped me apply my understanding of the class topics in real-world applications, and learn how to generally plot things in Matlab.

CITATIONS

Kenneth Casimiro: Used the resampling built-in function from MATLAB to help create "gen_highPitch". Used the following documentation to help me implement it. Used the following MATLAB forum to implement reverb using the reverberator object in the Audio Toolbox. Used the following MATLAB forum to implement delaying signals.
https://www.mathworks.com/help/signal/ug/resampling.html

https://www.mathworks.com/matlabcentral/answers/712278-reverb-effect-in-matlab
https://www.mathworks.com/matlabcentral/answers/15109-apply-a-variable-delay-to-an-audio-signal

Madhav Baghla: I looked at multiple MATLAB sources online to assist me in writing my code.
Envelope Extraction - MATLAB & Simulink (mathworks.com)
Matlab Examples: Amplitude Modulation and Envelope Detection (youtube.com)
envelope - Obtain the envelop of a signal using MATLAB - Stack Overflow

Christopher Inzunza: I used MATLAB's documentation to understand relevant functions.
https://www.mathworks.com/help/matlab/ref/audioread.html

Arihant Jain: I used the MATLAB Audio Toolbox to find the shiftPitch function, and implemented it using existing knowledge.
https://www.mathworks.com/help/audio/ref/shiftpitch.html

Jacob Zhang: I used in-built MATLAB functions to generate the fourier transform, create a linspace, and to plot the graphs. I had to refer to Matlab's documentation, at:
https://www.mathworks.com/help/matlab/

Joel Jijo: I used Matlab functions and their documentation to help creating my filters, as well as documentation on how to use them
https://www.mathworks.com/help/matlab/ref/fft.html
https://www.mathworks.com/help/signal/ref/bandpass.html
https://www.mathworks.com/help/signal/ref/bandstop.html

Collin Walmsley: I used Matlab functions and documentation to better help with creating the signal generator functions, as well as figuring out a way to plot only a few cycles of a function by mapping all the maximas of a function and then plotting from 0 to a few maximas later:
https://www.mathworks.com/help/signal/ref/findpeaks.html

Kendra Chen: I used the following website to help create the frequencies to make the melodies.
https://mixbutton.com/mixing-articles/music-note-to-frequency-chart/
https://cdn3.virtualsheetmusic.com/images/first_pages/BIG/Beethoven/OdeJoySQuartetFirst_BIG.gif
https://www.musicallthetime.com/images-songs/mary-had/mary-had-a-little-lamb-for-piano-simple.png

Eric Palafox: I used the following website to help create the frequencies to create the effects.
https://www.mathworks.com/help/matlab/ref/fftshift.html

Hamza Ahmed: I used MATLAB documentation for this project:
https://www.mathworks.com/help/matlab/

Yash Joshi: I used MATLAB documentation for this project:
https://www.mathworks.com/help/matlab/

Alain Zhang: I used matlab docs for this project
https://www.mathworks.com/help/signal/ref/findpeaks.html

Max Shen: I used MATLAB documentation for this project:
https://www.mathworks.com/help/fixedpoint/ref/conv.html

Jasmine Le: I used MATLAB documentation for this project:
https://www.mathworks.com/matlabcentral/fileexchange/10858-ecg-simulation-using-matlab