APPENDIX II-B-1. DSP: EXAMPLE ASSIGNMENTS, LABS, AND STUDENT WORK

Appendix II-B-1 contains examples of DSP assignments and student work:

1 "Guidelines for DSP Application Presentations and Written Reports." These guidelines provide a lucid road map for the student's presentation and written report. The goal is to help students produce clearly written reports (effective communication) that will be useful to other class members, some of whom might want to further explore a given application.

2 "DSP Final Project—Guidelines for Project Written Reports and Presentations." As with the "Application Guidelines", the Final Project Guidelines stipulate the sections comprising the written report. Many of these section headings already appear in the Final Project *Proposal*, typically completed 4-5 weeks before the project is due. The Final Project *Proposal* receives written and oral feedback from the professor in individual meetings, after which the student revises and re-submits (revision). The instructor again provides written and oral feedback on the revision, at which point the student is more calibrated to produce the Final Project Demo and Final Project Report (application). That report will include much of the same Background, Significance, and Bibliographical material already completed for the Project Proposal, as well as much of the Project Description. This strong correlation between the Project Proposal and Final Report helps **motivate** students to write cogent *Proposals* that capture their best thinking and research to date.

3 Example of a DSP Final Project Written Report: M-synth. This student developed an interest in electronic music, including synthesizers and synthesis methods, as a result of her independent study in music production. Prior to this project, she found herself restricted to preset sounds provided by commercial synthesizers so she decided to build her own—one that would allow her to design her own sound palette. In doing so, she gained greater intuition on pole-zero plots and how they translate to various filters, as well as how different synthesis methods work, e.g., additive and subtractive synthesis. She designed her M-synth system to function as a subtractive synthesizer, an additive synthesizer, or both. It comprises six main sections: the oscillator, filter, envelope, amplifier, save, and play. She built a clean, elegant, and smart user interface.

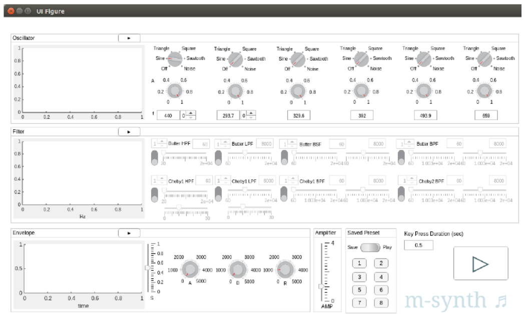


Figure 25. Graphical interface built for M-synth, a final project in DSP.

4 Examples of labs developed for the course:

- FIR Filters and Frequency Response
- Filter Implementation and Coefficient Quantization
- Filter Structures

Inserted PDFs start on the following page.

Guidelines for your DSP Application Presentations and Written Reports

Your audience will consist of students in our class. Your goals:

> **Inform** your peers about an Application involving Digital Signal Processing. Applications abound. Signal Processing pervades contemporary life.
>
> **Give context** for your App by providing succinct Description and Background sections.
>
> **Teach** one aspect of your App to yourself and then to the class. Present the App in a clear and engaging way. Clear explanations will save your classmates time and energy, making your researched Apps useful to them.

Your oral and written reports will consist of 5 sections, each of which should be clearly labeled in both your oral presentation and written report.

1. Description. Introduce your App in a clear and informative way. Digital Signal Processing Apps operate on one or more signals, transforming them in some way. What can the Application do? What kind of signal(s) is involved? List the strengths (e.g., cheap and fast) and any weaknesses (e.g., requires a lot of memory) of the App.

2. Background. Who created the App and why? How did it come to your attention? For example, "I read about the App in the January 2015 issue of IEEE's Spectrum Magazine." What engineering principles does it employ? (For example, "The App harnesses chaos theory to alter the content of a signal.")

3. Explain one (small) part/facet of the App. Narrow the topic of your App down to one small widget or self-contained concept. Specify the role your widget/concept plays in the App at large. Examine how your chosen widget or concept works. To encourage you to 'narrow your topic', you are allowed **only one figure and/or one math expression and/or math equation** to explain your chosen widget/concept. At all times, check to make sure any explanation does not skip from A to Z, but rather **progresses in a logical manner from point A to point B to C**, and so on. **Define all terms relevant to the widget/concept at hand** that have not yet been covered in our DSP vocabulary. Your figure must be **titled and captioned**. Provide a **detailed explanation of the figure in the text** of your written report and in your oral presentation. **Remember: you are teaching your chosen widget/concept in particular, and the App in general, to your classmates.** You are the teacher!

4. Significance of the App. Show the App's usefulness in any one or more fields such as medicine, transportation, logistics, business, entertainment, audio, etc. **How might a student in our class employ one or more aspects of the App for a final project?**

5. Bibliography. Provide a list of any references you consulted.

Rubric for your Written APP Report

You can allocate 2-3 pages (1.5 line spacing) to sections 1-4 described above (Description, Background, Explanation, Significance). Any figure should appear on a separate page (with a figure number, title, and caption), along with section 5 (Bibliography).

Your written report will be graded according to the following criteria:

1. **PROFESSIONAL WITHIN THE CONTEXT OF A DISCIPLINE**
   - A. The report's content is tailored to address the needs and understanding of the audience, i.e., students specifically in our class (not the professor).
   - B. The report includes citations for text and figures. See the "Style Guidelines for DSP Apps" given below.

2. **CLEAR AND INFORMED**
   - A. Your goal: de-mystify the App for your fellow students. **Remember: you are teaching the App, and in particular an aspect/widget/concept of the App, to your classmates.**
   - B. The report includes any background research, if needed, as a springboard to understanding.

3. **GOAL-DRIVEN AND STRUCTURED**
   - A. All sentences and body paragraphs are coherent and follow each other logically.

4. **SUPPORTED AND EFFECTIVELY ANALYZED you may include 1 figure and/or 1 math expression and/or 1 equation**
   - A. The figure is legible and has clearly annotated axes, legend (if applicable), and a title.
   - B. The figure has a caption which clearly explains what the figure represents.
   - C. The figure is clearly explained in the text of the written report.
   - D. The math expression and/or equation must be clearly explained in the text of the written report.
   - E. All variables, parameters, and terms in the math expression and/or equation must be defined.

5. **EXECUTED WITH CLEAR, ENGAGING, AND EFFICIENT MECHANICS**
   - A. The writer's control over grammar and punctuation facilitates clarity.
   - B. The writer makes clear and appropriate word choices, e.g., avoids contractions.
   - C. The writer uses clear sentence constructions and phrases. (*Awkward sentences and phrases hinder the reader's progress and understanding.*)

Style Guidelines for DSP Apps

For all App reports, please use the following settings:

- 12 point font. 1.5 line spacing. Margins 1.25" all around.

- Page numbers.
- Name at top of first page only.
- Professional font (e.g., Times, Times New Roman, etc.)
- In-text citations. Simply listing a source in your Bibliography is not adequate acknowledgment of that source. Therefore, use in-text citations to distinguish your own ideas from those of another. You need only include in brackets the author's last name and date of publication, e.g., (Strogatz 2010), for the work you are citing. You can then give the full citation in your Bibliography. Place the parenthetical citation in, or directly following, the sentence where you conclude the paraphrase, summary, or information.

- In the Bibliography, cite all sources that you use. Please use the Modern Library Association (MLA) guidelines for citations. You will find an excellent and clear explanation of the MLA Citation Style at https://owl.english.purdue.edu/owl/resource/747/01/

Please submit a hard copy of your report in class on the due date.

A Note on Plagiarism

Plagiarism can often result from confusion about how to summarize, paraphrase, and/or excerpt from another person's work. Princeton University has a very user-friendly set of resources to help you diagnose and avoid plagiarism. See, in particular, "When to Cite Sources"[1] and "Examples of Plagiarism."[2]

Most of us know to include a citation when we quote directly. Also remember to include citations when you paraphrase, summarize, or want to give credit for figures, facts, or originating ideas. If in doubt – cite!

Rubric for your oral presentation

The effective App presentation is …

1. **PROFESSIONAL WITHIN THE CONTEXT OF A DISCIPLINE**
   A. The presentation's content is tailored to address the needs and understanding of the audience.
   B. The presentation includes **citations** for text and figures **on each slide**. See above "Style Guidelines for DSP Apps".

[1] http://www.princeton.edu/pr/pub/integrity/pages/cite/
[2] http://www.princeton.edu/pr/pub/integrity/pages/plagiarism/

2. **CLEAR AND INFORMED**
    A. Your goal: de-mystify the App for your fellow students. **Remember: you are teaching the App to your classmates.**
    B. The slides include any background material, if needed, as a springboard to understanding.

3. **GOAL-DRIVEN AND STRUCTURED**
    A. Each slide has a main point that is identifiable in its heading.
    B. Each slide adds to the reader's understanding of the App and chosen widget or concept.
    C. All slides are coherent and follow each other logically.

4. **SUPPORTED AND EFFECTIVELY ANALYZED**
    A. The speaker's choice of 1 figure and/or 1 math expression and/or 1 equation conveys the concept/widget at hand and contextualizes it within the larger application topic.
    B. Each figure is legible and has clearly annotated axes, legend (if applicable), and title.
    C. Each figure has a caption which clearly explains what the figure represents.

5. **EXECUTED WITH CLEAR, ENGAGING, AND EFFICIENT MECHANICS**
    A. The speaker explains any figure, graph, equation, etc., that he or she shows.
    B. The speaker's choice of presentation format and style (*e.g., number of graphics, type of visuals, amount of text, etc.*) supports the goal of the presentation, and engages the audience.
    C. The speaker's presence evokes and maintains the listener's interest: *e.g., good eye contact, strong posture, confident gestures.*
    D. The speaker's voice evokes and maintains the listener's interest: *e.g., sufficient volume, varied inflection, effective pacing.*
    E. All sentences stay on task and cover ground efficiently (*e.g., avoid tangents and repetition*).
    F. The speaker allows an appropriate amount of time for listeners to absorb visual aids, media, etc., that are incorporated into the presentation.
    G. The speaker provides sufficient time for questions and comments from the audience.

2016 DSP Final Project—Guidelines for Project Written Reports and Presentations

Written Report Guidelines

At all times, make sure any **explanation progresses in a logical manner from point A to point B to C, and so on**. The report provides a level of analysis that elevates the reader's understanding of the topic, i.e., the reader learns something.

Please **include a small example as part of any explanation**. Keep in mind that you are teaching your Final Project to an audience. You are the professor! Remember how helpful it is when you are given examples in class to illustrate the concepts and material.

Define all terms relevant to the application at hand, e.g., define all variables and parameters.

All figures must be **titled, captioned and explained in the text**. All graphs must be legible with **clearly marked axes**.

Citations. Include all citations within the body of your written report, as you have done with your written App reports and in your Project Proposal submitted earlier.

Regarding equations:

- **Each equation must follow logically** from what preceded it. Do not skip steps.
- **Clearly explain each equation** so that you can show the reader that you understand its significance.
- **Define** all parameters and terms.
- Clearly explain **each step of any derivation**. Not only will it ensure you truly comprehend each step of the derivation, it will also lead your audience to full comprehension as well. Imagine you are writing a short book chapter on your particular final project that offers lucid explanations and takes the reader through the steps necessary to realize your project.

In sum, your Final Project Report should be written so clearly that your project could be duplicated according to what is explained in the report.

Check to ensure that:

- Each paragraph has a main point that is identifiable in its opening sentence(s).
- Each paragraph adds to the reader's understanding of the project.
- All sentences and body paragraphs are coherent and follow each other logically.
- Paragraphs stay on task and cover ground efficiently (*e.g., avoid tangents and repetition*).

Your report should be executed with the same clear and logical writing that we have discussed with respect to your written App Reports and Final Project Proposal.

Please include the following sections in your written report:

1. **Abstract**. A brief one-paragraph summary of your final project.

2. **Background**. This section sets the context for your project. Include a summary of your reference material. Definitely make use of the expanded Background section (Literature Search) that you researched for your project proposal. Remember to define terms so that your thinking can be creative *and* rigorous.

3. **Significance**. What distinguishes your idea from other similar ideas? How did you come up with it? How did you develop it? This section will use the "Significance" section you wrote for the Project Proposal, plus additional updates.

4. **Learning Objectives**. What did you learn as you worked on your project? State your learning objectives and specify how your project met them.

5. **Project Description**. Include a Block diagram of your project. Define each block and explain clearly how each block works and why. For any code you may write, give a clear flow diagram of your algorithmic processes. Include your code. Explain all relevant "small examples" that you have worked on during the course of this project, and include a block diagram of each. Show how the "small examples" relate to your larger project.

6. **Diagnosis**. What problems occurred during the project and how did you solve them?

7. **Improvement**. Describe ways in which your project could be improved.

8. **Bibliography**. Include the references from the Literature Search you did in preparing your Project Proposal, and include any updates.

Oral Presentation Guidelines

Allow 10 minutes for your oral presentation, and 3 minutes for Q&A.

Please follow the following guidelines:

1. **PROFESSIONAL WITHIN THE CONTEXT OF A DISCIPLINE**
   A. The presentation's content is tailored to address the needs and understanding of the audience.

B. The speaker uses vocabulary appropriate to the discipline and audience.
C. The presentation includes citations for text and figures **on each slide**.
D. The speaker is well-prepared to answer questions.

2. SIGNIFICANT AND ELEVATES UNDERSTANDING

A. Your goal: de-mystify your project for your audience. **Remember: you are teaching your project to the class.**
B. The presentation provides a level of analysis that elevates the listener's understanding of the topic. (*The audience learns something*).
C. The slides include any background research as a springboard to understanding.

3. GOAL-DRIVEN AND STRUCTURED

A. Each slide has a main point that is identifiable in its heading.
B. Each slide adds to the listener's understanding of your project.
C. All sentences and slides are coherent and follow each other logically.

4. SUPPORTED AND EFFECTIVELY ANALYZED

A. The speaker's choice and application of figures, graphics, equations, etc., supports, clarifies, and deepens the understanding of the audience.
B. Each figure is legible and has clearly annotated axes, legend, and title, wherever necessary.
C. Each figure has a caption which clearly explains what the figure represents.
D. All slides stay on task and cover ground clearly and efficiently

5. EXECUTED WITH CLEAR, ENGAGING, AND EFFICIENT MECHANICS

A. The speaker's language is clear (*awkward or ungrammatical words and phrases hinder the listener's course through the presentation*).
B. The speaker explains any figure, graph, equation, etc., that s/he shows.
C. The speaker's choice of presentation format and style (*e.g., number of graphics, type of visuals, amount of text, etc.*) supports the goal of the presentation, and engages the audience.
D. The speaker's presence evokes and maintains the listener's interest: *e.g., good eye contact, strong posture, confident gestures.*
E. The speaker's voice evokes and maintains the listener's interest: *e.g., appropriate volume, varied inflection, effective pacing.*
F. All sentences stay on task and cover ground efficiently (*e.g., avoid tangents and unnecessary repetition*).
G. The speaker allows an appropriate amount of time for listeners to absorb visual aids, media, etc., that are incorporated into the presentation.
H. The speaker provides sufficient time for questions and comments from the audience.

m-synth

ENGR 3415 Digital Signal Processing Final Project Report

Jee Kim, Olin College Class of 2018

Abstract

M-synth is an open source MATLAB synthesizer application built using MATLAB tools. Currently implemented as a single chain of four synthesis modules, m-synth has potential to scale to meet various needs, supported by the powerful MATLAB tools available. It also provides intuitive Graphical User Interface (GUI) that is designed for users to listen to the sound produced at the end of each module. The first version will be released on GitHub in early 2017.

Introduction

M-synth is an open source MATLAB synthesizer application built on the MathWorks Digital Signal Processing (DSP) System Toolbox and the MATLAB App Designer. The current initial version was built for the final project of Digital Signal Processing course. M-synth implements four synthesis modules - the oscillator, the filter, the amplifier and the envelope - connected in series.

This report is organized in the order of background of synthesizers, significance of the project, learning objectives, project description, problems and diagnosis, potential improvements, bibliography and appendix. The background section covers the basics of synthesizers and readers familiar with synthesizer should skip to the significance section.

[1] MathWorks, “DSP System Toolbox”, Internet: https://www.mathworks.com/products/dsp-system/ [November.1, 2016]

[2] MathWorks, “MATLAB App Designer”, Internet: https://www.mathworks.com/products/matlab/app-designer/ [November.1, 2016]

Background

A synthesizer is an electronic instrument that consists of various synthesis modules. Each synthesis module either generates or processes signals. These modules can be connected in various ways, enabling synthesizers to produce sounds that are impossible to create from the traditional instruments. The four modules implemented in m-synth are the oscillator, the filter, the amplifier and the envelope module.

Synthesis Modules

Oscillator

The oscillator is the main signal generator. It generates and outputs a basic waveform signal based on the frequency, the amplitude (A) and the waveform. The frequency determines the pitch and the amplitude determine the volume, which is a little different from the perceived volume based on the waveform. Common waveforms include periodic waveforms such as a sine wave, a square wave, a triangular wave, and a sawtooth wave.

Sine Wave

A sine wave is a periodic waveform that smoothly oscillates between the negative of the amplitude to the positive of the amplitude. It is made up of only the fundamental frequency, producing a clear sound.

Square Wave

A square wave is a periodic waveform that alternates between the positive of the amplitude and the negative of the amplitude. Square wave contains the fundamental frequency and odd harmonics, sounding higher and richer than the sound produced by a sine wave.

Triangular Wave

A triangular wave is a periodic, triangular shaped waveform contains the fundamental frequency and odd harmonics like a square wave. However, the higher harmonics roll off faster compared to those of a square wave, sounding higher and richer than the sound produced by a sine wave but having less timbre than the sound produced by a square wave.

Sawtooth Wave

A sawtooth wave ramps upward to the positive of the amplitude and drops sharply to the negative of the amplitude. It is the most common waveform used to create sound with a subtractive synthesis method described below. A sawtooth wave contains both odd and even harmonics, sounding harsh and clear.

Noise

White noise includes all frequencies at equal level, sounding thin and bright.

Filter

The filter module is used to modify the timbre of the produced sound. The filter module consists of various filters and each filter selectively amplifies or attenuates the specified frequency range. The filters are commonly used to tone down the brightness of the sound by removing the overtones.

Butterworth Filter

Butterworth filter has two main parameters: the cutoff frequency and the filter order. The cutoff frequency is the frequency characterizing a boundary between the passband and the stopband, where the output is 3dB below the nominal passband value. The filter order determines the sharpness of the transition from the passband to the stopband.

The passband of the Butterworth filter is designed to have a flat frequency response, at the expense of a wide transition band and a poor phase characteristics.

Chebyshev Filter

Chebyshev filter achieves a faster roll off by allowing ripple in the frequency response. The Chevyshev filters have poles that are closer to the imaginary axis, resulting in a faster roll off (narrower transition area). There are two types of Chebyshev filters. The Chebyshev type 1 filter have ripple in the passband while the Chebyshev type 2 filter have ripple in the stopband.

[12] A.Oppenheim, “Signals and Systems Butterworth Filters”, Internet: https://ocw.mit.edu/resources/res-6-007-signals-and-systems-spring-2011/lecture-notes/MITRES_6_007S11_lec24.pdf [December. 12, 2016]

[13] Analog Devices, “Basic Linear Design ”, Internet: http://www.analog.com/media/en/training-seminars/design-handbooks/Basic-Linear-Design/Chapter8.pdf?doc=ADA4661-2.pdf [December. 8, 2016]

[14] Analog Devices, “DSP Book”, Internet: http://www.analog.com/media/en/technical-documentation/dsp-book/dsp_book_Ch20.pdf [December. 8, 2016]

[15] MikroElectronica, “Reference Analog Prototype Filter”, Internet: http://learn.mikroe.com/ebooks/digitalfilterdesign/chapter/reference-analog-prototype-filter/ [December. 8, 2016]

Amplifier

The amplifier is the volume control of the synthesizer. The amplifier increases or decreases the amplitude of the waveform by raising and lowering the sound volume respectively.

Envelope

The envelope integrates the signal generated by the synthesizer with inputs from the Musical Instrument Digital Interface (MIDI). MIDI is the standard protocol for communication between musical devices. The envelope is used to drive either the filter or the amplifier modules. The amplifier envelope determines how quickly a sound fades in and fades out, and the level at which the sound is maintained.

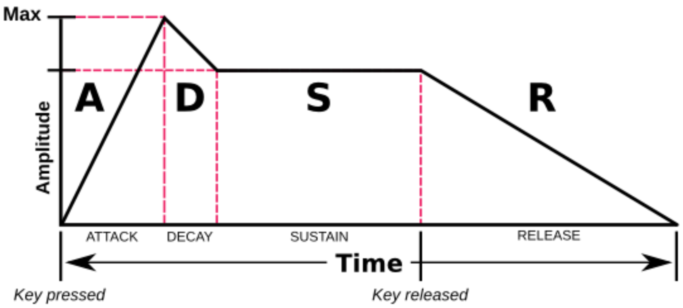


Figure 1. ADSR Envelope (Libre Music Production, Creating a simple synthesizer)

As shown in Figure 1, the envelope has four stages: Attack (A), Decay (D), Sustain (S) and Release (R). The attack time is the time taken for the input signal to reach the greatest amplitude from when a key is first pressed. The decay time is the time taken to reach the sustained amplitude after the key has reached the greatest amplitude (after attack time). The sustain level is the level at which the sound is held at while the key is still pressed. The release time is the time taken for a signal to fade out once the key is released.

Synthesis Methods

Synthesizers can generate signals using various synthesis methods. Fundamental synthesis methods include the subtractive and additive methods.

[4] T.Abdullah, “Subtractive Synthesis”, Internet: http://www.angelfire.com/in2/yala/2ansynth.htm [November.1, 2016]

[5] J.Krug, “Introduction to Sound Recording Technology”, Internet: https://public.wsu.edu/~jkrug/MUS364/audio.htm [November.1, 2016]

[10] Libre Music Production, “Creating a simple synthesizer in pure data Part III”, Internet: http://libremusicproduction.com/tutorials/creating-simple-synthesizer-pure-data-%E2%80%93-part-ii [November.1, 2016]

Subtractive synthesis

The idea behind the subtractive synthesis method is to generate harmonics and attenuate unwanted frequencies using various filters. It was first developed to commercial success in the 1960s by Bob Moog

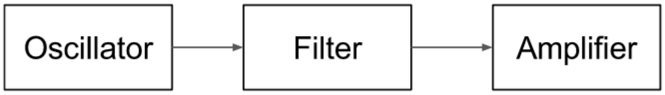


Figure 2. Block Diagram of Subtractive Synthesis (author)

The subtractive synthesis method is implemented as a signal chain of an oscillator, a filter and an amplifier as shown in Figure 2.

Additive Synthesis

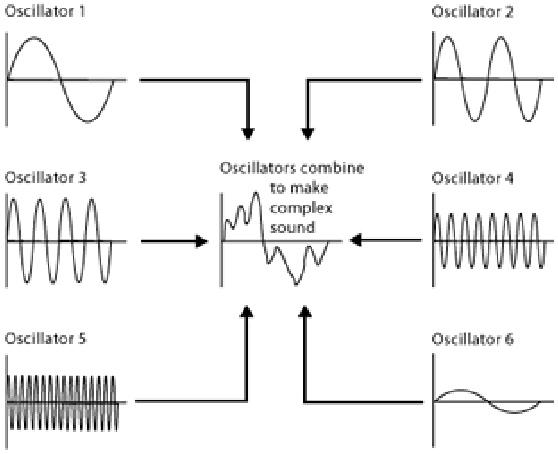


Figure 3. Additive Synthesis (Matt Ottewill, Synthesis Types)

The additive synthesis method combines multiple sine waves with varying amplitude and frequencies to build the desired sound as shown in Figure 3.

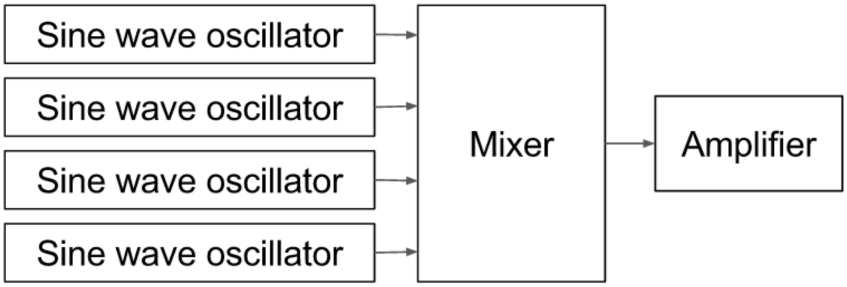


Figure 4. Block Diagram of Additive Synthesis (author)

[4] T.Abdullah, “Subtractive Synthesis”, Internet: http://www.angelfire.com/in2/yala/2ansynth.htm [November.1, 2016]

[9] Vintage synth explorer, “Moog Minimoog”, Internet: http://www.vintagesynth.com/moog/moog.php [November.1, 2016]

It is implemented as a multiple sine wave oscillators, a mixer and an amplifier as shown in Figure 4. The mixer combined the output signal from each of the oscillator into one signal.

Significance

What distinguishes your idea from other similar ideas?
Open source synthesizer, which allows the users to modify the code to suit their needs, can serve as a powerful addition to the sound library, or as a great learning tool to understand synthesizers better. The synthesizer functions are no longer bounded by the default settings imposed by the software; the users can create any sounds.

Given the powerful DSP System Toolbox, I initially thought it is highly probable that an open source synthesizer created using MATLAB exist. From research, however, I realized there are no proper synthesizers built using MATLAB with GUI.

M-synth aims to kick start a MATLAB synthesizer, that is easily configurable and modifiable for everyone's use. Filter design is easier in MATLAB than in other languages due to the support from the powerful DSP tool provided by the MathWorks. I hope m-synth could grow with contributions from the open source community.

How did you come up with it?
I have recently developed interest in synthesizers and various synthesis methods through my independent study in music production. As a programmer, it was only natural to have an interest in implementing a simple synthesizer and to look up programmable synthesizers. I am also familiar with the capabilities of the DSP toolbox from my research in Visible Light Communication. These backgrounds, with added motivation from having a DSP final project, have led to the decision of implementing m-synth.

[6] R.McGee, "Simple Music in MATLAB", Internet: http://www.lifeorange.com/MATLAB/MATLAB_music.htm [November.1, 2016]

[7] P.Leon, "Computer Music in Undergraduate Digital Signal Processing", Internet: http://www.ece.nmsu.edu/~pdeleon/Research/Publications/ASEE_GSW_2000.pdf [November.1, 2016]

[8] M.Petersen, "Musical Analysis and Synthesis in Matlab, MAA's College Mathematics Journal", Volume 35 No 5, 396-401, Nov 2004 [November.1, 2016]

How did you develop it?

The development process can be broken down to four main phases. First was the planning and researching phase. I researched into various synthesizers and their implementation methods to gain more understanding of how synthesizers work. I also researched on the tools and functionality provided by the DSP toolbox and the various MATLAB GUIs to know the limitations. The timeline was made in this phase to keep me on track throughout the project. Second was the decision phase. The first draft of the GUI layout was completed and I made decisions to use the App Designer GUI for the GUI and Butterworth and Chebyshev filters in the filter module. The next phase was the coding and debugging phase, which took up the most part of the project. The final phase involved testing and improving the m-synth. In this phase, I adjusted the ranges of the filters, increased the number of oscillators, set the default values of various control knobs, wrote functions to include harmonics for the first two oscillators, and added save and playback function.

Learning Objectives

Project m-synth covers two main topics, music and signal processing; these two topics align with my passion. Going into the project, I have aimed to gain deeper understanding on these topics, especially on filters and synthesizers. I have also aimed to apply what was learned in theory and pick up intuitions on how different signal processing techniques changes the signals and alters the sound.

The research phase made me to better understand how synthesizers worked and learn the advantages and limitation of different kinds of filters. I gained more intuition on how the pole zero diagrams translate to respective filters and better understand how each synthesis module functions and contribute to the overall signal. Preparing for the presentation definitely contributed to my understanding of filters.

Before the start of the project, I was limited by the preset sounds provided by the software synthesizers, only able to slightly modify the sound using filters. Now, I can distinguish between different types of synthesizers and build sound up from the oscillator, designing it to sound closer to what I am imagining.

Project Description

M-synth combines the idea of additive and subtractive synthesis methods. It consists of four modules connected in series in the order of oscillators, a filter, an amplifier and an envelope as shown in Figure 5.

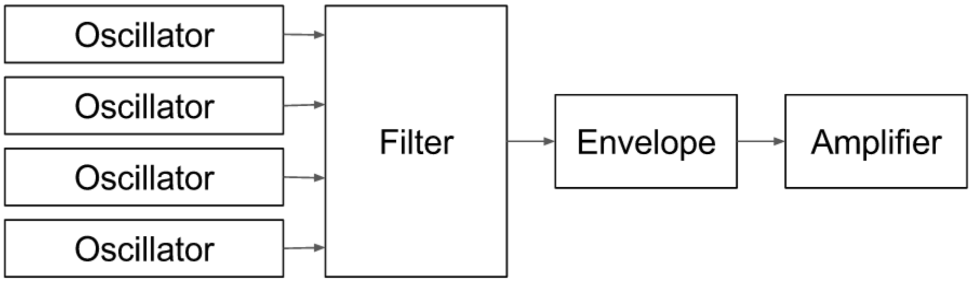


Figure 5. M-synth System Block Diagram (author)

A synthesizer that incorporates many synthesis methods provides more flexibility in producing a more varied sound. M-synth synthesizer system was designed this way to give more flexibility to the users. It can be a subtractive synthesizer, an additive synthesizer or both.

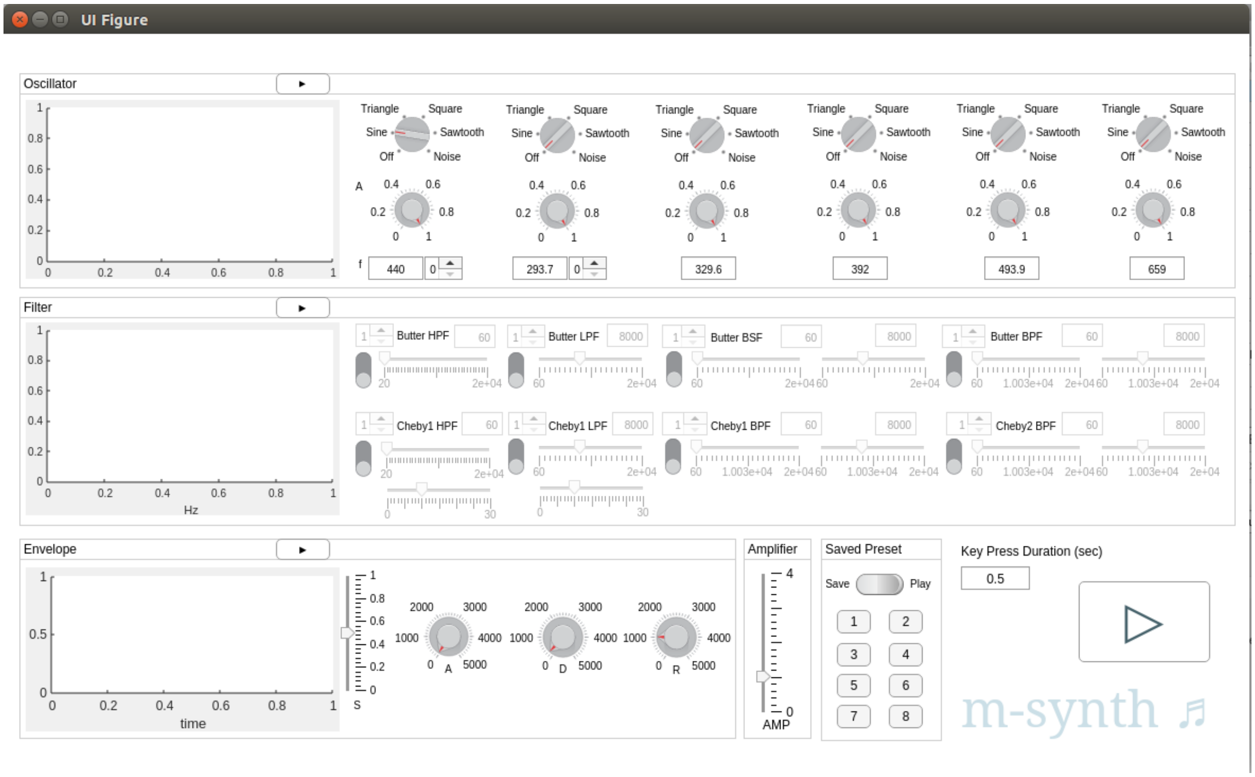


Figure 6. M-synth GUI layout (author)

The M-synth consist of 6 main sections, the oscillator, the filter, the envelope, the amplifier, the save section and the play section.

Oscillator Section

The oscillator section provides a play button, an axes and six oscillators. The play button plays the combination of output signals generated by the six oscillators and plots the signal onto the axes. Each oscillator consist of a waveform selector, an amplitude knob and a frequency edit field. The user can choose the basic waveform to be a sine wave, a triangle wave, a square wave, a sawtooth wave or to add white noise, or to switch it off. The amplitude can be adjusted using the amplitude knob, in the case the user wants a signal from one oscillator to be higher than the signal generated by another oscillator. The frequency edit field allows the user to write specific frequency value. The first two oscillators are designed a little differently from the other oscillators to give users option to add selected number of harmonics. The first oscillator adds user specified number of harmonics with equal amplitude as the fundamental frequency. The second oscillator adds user specified number of harmonics with decreasing amplitude. The second oscillator starts to look similar to reversed sawtooth wave with high number of harmonics.

Filter Section

The filter section consists of a play button, an axes and eight filters. The eight filters are Butterworth high pass filter (HPF), Butterworth low pass filter (LPF), Butterworth bandstop filter (BSF), Butterworth bandpass filter (BPF), Chebyshev 1 HPF, Chebyshev 1 LPF, Chebyshev BPS and Chebyshev 2 BPS. The input before the name of the filter sets in the order of the filter. The toggle bar enables and disables the filter, allowing users implement only the filters they choose to. The number edit field and the scale bar are both inputs for the cutoff frequency. Changing one value will also change the other; they mirror each other. The BSF and BPF have two inputs for the frequency which are for the lower cutoff frequency and the higher cutoff frequency. Similar to the oscillator module, the play button play the filtered sound and plots the Power Spectral Density (PSD) graph of unfiltered signal in blue and filtered signal in black with respect to the left axis, and filter shapes of enabled filter in red with respect to the right axis.

Envelope Section

The envelope section consists of a play button, an axes, a slider for sustain and three knobs for attack, delay and release. The sustain slider sets the amplitude value a note is sustained at after the decay. The attack, the delay and the release determine the attack time, the delay time and the release time respectively. The values for each knobs range from 0 to 5000 milliseconds, which maps to 0 to 5 seconds. The play button plays the signal reflecting the play length and the ADSR, and plots the ADSR graph with respect to the left axis and the processed signal with respect to the right axis.

Amplifier Section

The amplifier section has a slider which can increase or decrease the volume of the overall signal. The value on the slider is the actual value multiplied to the signal; slider with value 1 will output the same signal. Initially, I wanted to implement additional bars which display the original and amplified signal level as the sound was being played. However, there was no appropriate display option available and updating the bar could possibly introduce delays to the system. Hence, the amplifier module was simplified to a slider. Unlike the oscillator, the filter and the envelope sections, the amplifier does not have a play button because it would repeat the same function as the overall play button described in the play section.

Save Section

The save as introduced after testing the first completed draft of m-synth. It was added to make m-synth playable. In the saving mode, the user can create any sound and choose to save it in sound 1 to 8 by pressing the respective buttons. Switching to the play mode, the user can play saved sound by pressing the respective buttons. The save section can be improved by having an indicator next to each button to show if a sound is stored in each button. This will prevent the user from accidentally overwriting the signal.

Play Section

The play section consists of key press duration edit field and a play button. The key press duration input determines how long a key is pressed for, which is crucial in implementing the ADSR. The play button updates all three plots and plays the final output signal from the amplifier.

The play function for the oscillator and the filter was designed to reflect only the key press duration while the play function for the envelope and the play reflect both the key press duration and the release time.

Diagnosis

Diagnosis from proposal

Prior to the project, possible problems that might arise throughout the project were identified. These problems included time management and integration of the various modules. With the help of a timeline to keep me on track, the project could be completed on time. Integration did not pose as much of a problem as I initially thought it would.

Technical Problems

Throughout the project, I encountered some unexpected problems in key input, playing the sound and making a GUI with precise controls.

Initially, the user interaction was more interactive, involving the user to press the spacebar to mark the start of the note and release the spacebar to mark the end of the note. However, the current version of App Designer GUI does not support key recognition. Therefore, the user interaction was changed to the user playing a note through a play button and specifying the note duration in the key press duration input.

Due the duration in the saved sound could be as long as the user wanted, the user could play another note before the previous sound ended. Playing multiple notes gave rise to a problem and delay in the software. Therefore, play blocking was implemented in the code to prevent being able to play another sound while a sound was being played.

Also, I initially wanted to implement a sweep function in the filter module. Sweep module was aimed to allow the user to hear sounds at different frequency spectrum. However, I am not sure if it is practical to implement the function because play, playblock, and stop functions have to be called many times in a loop while updating the graph which has a noticeable delay time. Therefore, this stretch goal feature was not implemented for this version of m-synth.
At first, the filter sliders did not have the edit field inputs (where the user can type in specific frequency); it consisted of just the sliders. This resulted in needing to tradeoff between the flexibility (range of frequencies of the filter) and the accuracy of the value selection (selecting specific frequency value). However, I wanted to provide both the choice of choosing from a large range of frequencies and precise control. Therefore, the edit field was added as additional input for the frequency value and linked to the sliders so their values mirrored each other (changing the input in the slider will change the value in the number input and vice versa).

Improvement

Possible improvement areas include the GUI and the functions. For the functions, improvements can be made by allowing users to connect up the synthesis modules in any ways and by implementing sweep. For the GUI, the user experience can be enhanced by having a zoom in button for the filter graph.

One of the great strengths synthesizers possess lies in being able to connect the modules up in different manner. M-synth can become a more powerful and flexible synthesizer if the user can add or remove synthesis modules and connect them up in any ways they would like. This will require editing the entire code since the functions needs to act on the given inputs, and know how to interact with the other modules. Also, more variables would have to be kept track of to make this possible.

The sweep function will use a narrow BPF and plays the sound as the filter sweep from the lowest to the highest frequency. This function will allow the user to easily determine which frequencies to filter and which to keep.

The user would have a better understanding of how the filter is working with zoom button. Currently, the graph is always plotted over frequency range of 60Hz to 20000Hz. This makes it challenging to tell what the filter shape is and where the filter is cutting the sounds at the lower frequencies below 1000Hz.

Conclusion

Project m-synth is meaningful to me because it is the first synthesizer I have coded. The project allowed me to gain deeper understanding of synthesizers and filters and intuition on how to design sounds. Also, I hope to spend more time for further developments and update the current version.

Bibliography

[1] MathWorks, “DSP System Toolbox”, Internet: https://www.mathworks.com/products/dsp-system/ [November.1, 2016]

[2] MathWorks, “MATLAB App Designer”, Internet: https://www.mathworks.com/products/matlab/app-designer/ [November.1, 2016]

[3] MathWorks, “soundsc”, Internet: https://www.mathworks.com/help/matlab/ref/soundsc.html [November.1, 2016]

[4] T.Abdullah, “Subtractive Synthesis”, Internet: http://www.angelfire.com/in2/yala/2ansynth.htm [November.1, 2016]

[5] J.Krug, “Introduction to Sound Recording Technology”, Internet: https://public.wsu.edu/~jkrug/MUS364/audio.htm [November.1, 2016]

[6] R.McGee, “Simple Music in MATLAB”, Internet: http://www.lifeorange.com/MATLAB/MATLAB_music.htm [November.1, 2016]

[7] P.Leon, “Computer Music in Undergraduate Digital Signal Processing”, Internet: http://www.ece.nmsu.edu/~pdeleon/Research/Publications/ASEE_GSW_2000.pdf [November.1, 2016]

[8] M.Petersen, “Musical Analysis and Synthesis in Matlab, MAA’s College Mathematics Journal”, Volume 35 No 5, 396-401, Nov 2004 [November.1, 2016]

[9] Vintage synth explorer, “Moog Minimoog”, Internet: http://www.vintagesynth.com/moog/moog.php [November.1, 2016]

[10] Libre Music Production, “Creating a simple synthesizer in pure data Part III”, Internet: http://libremusicproduction.com/tutorials/creating-simple-synthesizer-pure-data-%E2%80%93-part-ii [November.1, 2016]

[11] J.Kreidler, “Programming Electronic Music in Pd”, Internet: http://www.pd-tutorial.com/ [November.10, 2016]

[12] A.Oppenheim, “Signals and Systems Butterworth Filters”, Internet: https://ocw.mit.edu/resources/res-6-007-signals-and-systems-spring-2011/lecture-notes/MITRES_6_007S11_lec24.pdf [December. 12, 2016]

[13] Analog Devices, “Basic Linear Design ”, Internet: http://www.analog.com/media/en/training-seminars/design-handbooks/Basic-Linear-Design/Chapter8.pdf?doc=ADA4661-2.pdf [December. 8, 2016]

[14] Analog Devices, “DSP Book”, Internet: http://www.analog.com/media/en/technical-documentation/dsp-book/dsp_book_Ch20.pdf [December. 8, 2016]

[15] MikroElectronica, “Reference Analog Prototype Filter”, Internet: http://learn.mikroe.com/ebooks/digitalfilterdesign/chapter/reference-analog-prototype-filter/ [December. 8, 2016]

Appendix

```
% Properties that correspond to app components
    properties (Access = public)
        UIFigure                 matlab.ui.Figure
        OscillatorPanel          matlab.ui.container.Panel
        oscWaveKnob              matlab.ui.control.DiscreteKnob
        oscAmpKnob               matlab.ui.control.Knob
        oscFreqField             matlab.ui.control.NumericEditField
        oscWaveKnob_2            matlab.ui.control.DiscreteKnob
        oscAmpKnob_2             matlab.ui.control.Knob
        oscFreqField_2           matlab.ui.control.NumericEditField
        oscWaveKnob_3            matlab.ui.control.DiscreteKnob
        oscAmpKnob_3             matlab.ui.control.Knob
        oscFreqField_3           matlab.ui.control.NumericEditField
        oscAxes                  matlab.ui.control.UIAxes
        ALabel                   matlab.ui.control.Label
        fLabel                   matlab.ui.control.Label
        oscWaveKnob_4            matlab.ui.control.DiscreteKnob
        oscAmpKnob_4             matlab.ui.control.Knob
        oscFreqField_4           matlab.ui.control.NumericEditField
        oscWaveKnob_5            matlab.ui.control.DiscreteKnob
        oscAmpKnob_5             matlab.ui.control.Knob
        oscFreqField_5           matlab.ui.control.NumericEditField
        oscHarmSpinner           matlab.ui.control.Spinner
        oscHarmSpinner_2         matlab.ui.control.Spinner
        oscWaveKnob_6            matlab.ui.control.DiscreteKnob
        oscAmpKnob_6             matlab.ui.control.Knob
        oscFreqField_6           matlab.ui.control.NumericEditField
        KeyPressDurationsecEditFieldLabel  matlab.ui.control.Label
        keyDurationField         matlab.ui.control.NumericEditField
        Button                   matlab.ui.control.Button
        FilterPanel              matlab.ui.container.Panel
        ButterworthLPFSlider     matlab.ui.control.Slider
        filterAxes               matlab.ui.control.UIAxes
        ButterworthHPFSlider     matlab.ui.control.Slider
        ButterworthHPFSwitch     matlab.ui.control.Switch
        ButterHPFLabel           matlab.ui.control.Label
        ButterworthLPFSwitch     matlab.ui.control.Switch
        ButterLPFLabel           matlab.ui.control.Label
        ChebyshevHPFSwitch       matlab.ui.control.Switch
        Cheby1HPFLabel           matlab.ui.control.Label
        ChebyshevHPFSlider       matlab.ui.control.Slider
        ChebyshevHPFpbSlider     matlab.ui.control.Slider
        ChebyshevLPFSwitch       matlab.ui.control.Switch
        Cheby1LPFLabel           matlab.ui.control.Label
        ChebyshevLPFSlider       matlab.ui.control.Slider
        ChebyshevLPFpbSlider     matlab.ui.control.Slider
        ButterworthBSFfclSlider  matlab.ui.control.Slider
        ButterworthBSFSwitch     matlab.ui.control.Switch
        ButterBSFLabel           matlab.ui.control.Label
        ButterworthBSFfchSlider  matlab.ui.control.Slider
        ButterworthBPFfclSlider  matlab.ui.control.Slider
        ButterworthBPFSwitch     matlab.ui.control.Switch
        ButterBPFLabel           matlab.ui.control.Label
        ButterworthBPFfchSlider  matlab.ui.control.Slider
        ChebyshevBPFfclSlider_2  matlab.ui.control.Slider
        ChebyshevBPFSwitch_2     matlab.ui.control.Switch
        Cheby2BPFLabel           matlab.ui.control.Label
        ChebyshevBPFfchSlider_2  matlab.ui.control.Slider
        ChebyshevBPFfclSlider    matlab.ui.control.Slider
        ChebyshevBPFSwitch       matlab.ui.control.Switch
        Cheby1BPFLabel           matlab.ui.control.Label
        ChebyshevBPFfchSlider    matlab.ui.control.Slider
        ButterworthHPFField      matlab.ui.control.NumericEditField
        ButterworthLPFField      matlab.ui.control.NumericEditField
        ButterworthBSFfclField   matlab.ui.control.NumericEditField
        ButterworthBSFfchField   matlab.ui.control.NumericEditField
        ButterworthBPFfclField   matlab.ui.control.NumericEditField
        ButterworthBPFfchField   matlab.ui.control.NumericEditField
        ChebyshevBPFfclField_2   matlab.ui.control.NumericEditField
        ChebyshevBPFfchField_2   matlab.ui.control.NumericEditField
        ChebyshevBPFfclField     matlab.ui.control.NumericEditField
        ChebyshevBPFfchField     matlab.ui.control.NumericEditField
        ChebyshevLPFField        matlab.ui.control.NumericEditField
        ChebyshevHPFField        matlab.ui.control.NumericEditField
        ButterworthHPFSpinner    matlab.ui.control.Spinner
        ButterworthLPFSpinner    matlab.ui.control.Spinner
        ButterworthBSFSpinner    matlab.ui.control.Spinner
        ChebyshevHPFSpinner      matlab.ui.control.Spinner
        ChebyshevLPFSpinner      matlab.ui.control.Spinner
        ChebyshevBPFSpinner      matlab.ui.control.Spinner
        ChebyshevBPFSpinner_2    matlab.ui.control.Spinner
        ButterworthBPFSpinner    matlab.ui.control.Spinner
        EnvelopePanel            matlab.ui.container.Panel
        envAxes                  matlab.ui.control.UIAxes
        AKnobLabel               matlab.ui.control.Label
        AKnob                    matlab.ui.control.Knob
        DKnobLabel               matlab.ui.control.Label
        DKnob                    matlab.ui.control.Knob
        RKnobLabel               matlab.ui.control.Label
        RKnob                    matlab.ui.control.Knob
        SSliderLabel             matlab.ui.control.Label
        SSlider                  matlab.ui.control.Slider
        AmplifierPanel           matlab.ui.container.Panel
        AMPSliderLabel           matlab.ui.control.Label
        ampSlider                matlab.ui.control.Slider
        envPlayButton            matlab.ui.control.Button
        filterPlayButton         matlab.ui.control.Button
        oscPlayButton            matlab.ui.control.Button
        msynthLabel              matlab.ui.control.Label
        SavedPresetPanel         matlab.ui.container.Panel
        SaveSwitch               matlab.ui.control.RockerSwitch
        NoteButton               matlab.ui.control.Button
        NoteButton_2             matlab.ui.control.Button
        NoteButton_3             matlab.ui.control.Button
        NoteButton_4             matlab.ui.control.Button
        NoteButton_5             matlab.ui.control.Button
        NoteButton_6              matlab.ui.control.Button
        NoteButton_7              matlab.ui.control.Button
        NoteButton_8              matlab.ui.control.Button
    end


    properties (Access = private)
        osc1y % y of oscillator 1
        osc2y % y of oscillator 2
        osc3y % y of oscillator 3
        osc4y % y of oscillator 4
        osc5y % y of oscillator 5
        osc6y % y of oscillator 6
        oscy % y of all oscillators

        Fs = 44200 % sample rate
        t % total time (play time + release time)
        kp % key pressed length

        filtery % y after filters

        envy % y after envelope
        ADSR %ADSR

        ampy % y after amplifier

        y % final y (to be implemented after lfo)

        n1y = 0; % saved y for note button 1
        n2y = 0; % saved y for note button 2
        n3y = 0; % saved y for note button 3
        n4y = 0; % saved y for note button 4
        n5y = 0; % saved y for note button 5
        n6y = 0; % saved y for note button 6
        n7y = 0; % saved y for note button 7
        n8y = 0; % saved y for note button 8
    end

    methods (Access = private)

        %compute app.t
        function results = playtime(app)
            T = app.keyDurationField.Value;%*(1/f);
            dt = 1/app.Fs;
            app.t = 0:dt:T;
        end

        function results = totaltime(app)
            Tp = app.keyDurationField.Value;
            T = Tp + app.RKnob.Value/1000;
            dt = 1/app.Fs;
            app.t = 0:dt:T;
            app.kp = round(Tp*app.Fs);
        end


        %oscillator
        %generating wave for oscillator 1
        function results = osc1(app)

            A = app.oscAmpKnob.Value;
            f = app.oscFreqField.Value;
            harmonics = app.oscHarmSpinner.Value;
            app.osc1y = zeros(size(app.t));

            switch app.oscWaveKnob.Value
                case 'Sine'
                    for h = 1:(harmonics)
                        app.osc1y = app.osc1y + A*sin(2*pi*app.t*f*h);
                    end

                case 'Triangle'
                    for h = 1:(harmonics)
                        app.osc1y = app.osc1y + A*sawtooth(2*pi*app.t*f*h,
                            0.5);
                    end

                case 'Square'
                    for h = 1:(harmonics)
                        app.osc1y = app.osc1y + A*square(2*pi*app.t*f*h);
                    end

                case 'Sawtooth'
                    for h = 1:(harmonics)
                        app.osc1y = app.osc1y + A*sawtooth(2*pi*app.t*f*h);
                    end

                case 'Noise'
                    app.osc1y = awgn(A*app.t./app.t, 10);

                otherwise %off

            end

        end

        %generating wave for oscillator 2
        function results = osc2(app)

            A = app.oscAmpKnob_2.Value;
            f = app.oscFreqField_2.Value;
            harmonics = app.oscHarmSpinner_2.Value;
            app.osc2y = zeros(size(app.t));

            switch app.oscWaveKnob_2.Value
                case 'Sine'
                    for h = 1:(harmonics)
                        app.osc2y = app.osc2y + A/h*sin(2*pi*app.t*f*h);
                    end

                case 'Triangle'
                    for h = 1:(harmonics)
                        app.osc2y = app.osc2y + A/h*sawtooth(2*pi*app.t*f*h,
                            0.5);
                    end

                case 'Square'
                    for h = 1:(harmonics)
                        app.osc2y = app.osc2y + A/h*square(2*pi*app.t*f*h);
                    end

                case 'Sawtooth'
                    for h = 1:(harmonics)
                        app.osc2y = app.osc2y + A/h*sawtooth(2*pi*app.t*f*h);
                    end

                case 'Noise'
                    app.osc2y = awgn(A*app.t./app.t, 10);

                otherwise %off

            end
        end

        %generating wave for oscillator 3
        function results = osc3(app)

            A = app.oscAmpKnob_3.Value;
            f = app.oscFreqField_3.Value;

            switch app.oscWaveKnob_3.Value
                case 'Sine'
                    app.osc3y = A*sin(2*pi*app.t*f);

                case 'Triangle'
                    app.osc3y = A*sawtooth(2*pi*app.t*f, 0.5);

                case 'Square'
                    app.osc3y = A*square(2*pi*app.t*f);

                case 'Sawtooth'
                    app.osc3y = A*sawtooth(2*pi*app.t*f);

                case 'Noise'
                    app.osc3y = awgn(A*app.t./app.t, 10);

                otherwise %off
                    app.osc3y = zeros(size(app.t));
            end
        end

        %generating wave for oscillator 4
        function results = osc4(app)

            A = app.oscAmpKnob_4.Value;
            f = app.oscFreqField_4.Value;

            switch app.oscWaveKnob_4.Value
                case 'Sine'
                    app.osc4y = A*sin(2*pi*app.t*f);

                case 'Triangle'
                    app.osc4y = A*sawtooth(2*pi*app.t*f, 0.5);

                case 'Square'
                    app.osc4y = A*square(2*pi*app.t*f);

                case 'Sawtooth'
                    app.osc4y = A*sawtooth(2*pi*app.t*f);

                case 'Noise'
                    app.osc4y = awgn(A*app.t./app.t, 10);

                otherwise %off
                    app.osc4y = zeros(size(app.t));
            end
        end

        %generating wave for oscillator 5
        function results = osc5(app)

            A = app.oscAmpKnob_5.Value;
            f = app.oscFreqField_5.Value;

            switch app.oscWaveKnob_5.Value
                case 'Sine'
                    app.osc5y = A*sin(2*pi*app.t*f);

                case 'Triangle'
                    app.osc5y = A*sawtooth(2*pi*app.t*f, 0.5);

                case 'Square'
                    app.osc5y = A*square(2*pi*app.t*f);

                case 'Sawtooth'
                    app.osc5y = A*sawtooth(2*pi*app.t*f);

                case 'Noise'
                    app.osc5y = awgn(A*app.t./app.t, 10);

                otherwise %off
                    app.osc5y = zeros(size(app.t));
            end
        end

        %generating wave for oscillator 6
        function results = osc6(app)

            A = app.oscAmpKnob_6.Value;
            f = app.oscFreqField_6.Value;

            switch app.oscWaveKnob_6.Value
                case 'Sine'
                    app.osc6y = A*sin(2*pi*app.t*f);

                case 'Triangle'
                    app.osc6y = A*sawtooth(2*pi*app.t*f, 0.5);

                case 'Square'
                    app.osc6y = A*square(2*pi*app.t*f);

                case 'Sawtooth'
                    app.osc6y = A*sawtooth(2*pi*app.t*f);

                case 'Noise'
                    app.osc6y = awgn(A*app.t./app.t, 10);

                otherwise %off
                    app.osc6y = zeros(size(app.t));
            end
        end

        function results = osc(app)
            osc1(app);
            osc2(app);
            osc3(app);
            osc4(app);
            osc5(app);
            osc6(app);
            app.oscy = app.osc1y + app.osc2y + app.osc3y + app.osc4y +
               app.osc5y + app.osc6y;
        end

        function results = oscplot(app)
            plot(app.oscAxes, app.t(1:1000), app.oscy(1:1000));
%             ylim(app.oscAxes, [-5 5])
        end

        function results = oscplay(app)
            player = audioplayer(app.oscy,app.Fs);
            playblocking(player);
            stop(player);
        end



        % filter
        function results = butterhpf(app)
            fc = app.ButterworthHPFSlider.Value; % cutoff frequency
            order = app.ButterworthHPFSpinner.Value;
            [b, a] = butter(order, fc/(app.Fs/2), 'high'); %HP butterworth filter, fc/(Fs/2) rad/sample
            app.filtery = filter(b, a, app.filtery);

            [h, w] = freqz(b, a);
            yyaxis (app.filterAxes, 'right');
            plot(app.filterAxes, w, h, 'red');
        end

        function results = butterlpf(app)
            fc = app.ButterworthLPFSlider.Value; % cutoff frequency
            order = app.ButterworthLPFSpinner.Value;
            [b, a] = butter(order, fc/(app.Fs/2)); %LP butterworth filter
            app.filtery = filter(b, a, app.filtery);

            [h, w] = freqz(b, a);
            yyaxis (app.filterAxes, 'right');
            hold (app.filterAxes, 'on');
            plot(app.filterAxes, w, h, 'red');
        end

        function results = butterbsf(app)
            fcl = app.ButterworthBSFfclSlider.Value;
            fch = app.ButterworthBSFfchSlider.Value;
            order = app.ButterworthBSFSpinner.Value/2;
            [b,a] = butter(order,[fch/(app.Fs/2) fcl/(app.Fs/2)],'stop'); %butterworth BSF
            app.filtery = filter(b, a, app.filtery);

            [h, w] = freqz(b, a);
            yyaxis (app.filterAxes, 'right');
            hold (app.filterAxes, 'on');
            plot(app.filterAxes, w, h, 'red');
        end

        function results = butterbpf(app)
            fcl = app.ButterworthBPFfclSlider.Value;
            fch = app.ButterworthBPFfchSlider.Value;
            order = app.ButterworthBPFSpinner.Value/2;
            [b,a] = butter(order,[fch/(app.Fs/2) fcl/(app.Fs/2)],'bandpass'); %butterworth BPF
            app.filtery = filter(b, a, app.filtery);

            [h, w] = freqz(b, a);
            yyaxis (app.filterAxes, 'right');
            hold (app.filterAxes, 'on');
            plot(app.filterAxes, w, h, 'red');
        end

        function results = chebybpf(app)
            fcl = app.ChebyshevBPFfclSlider.Value;
            fch = app.ChebyshevBPFfchSlider.Value;
            order = app.ChebyshevBPFSpinner.Value;
            pbr = 3; %passband ripple
            [A,B,C,D] = cheby1(order/2, pbr, [fch/(app.Fs/2)
               fcl/(app.Fs/2)]); %chevyshev1 BSF
            d =
               designfilt('bandpassiir','FilterOrder',order,'PassbandFrequency1',fch,'Pass
            sos = ss2sos(A,B,C,D);
            app.filtery = filter2(sos, app.filtery);

            [h, w] = freqz(sos);
            yyaxis (app.filterAxes, 'right');
            hold (app.filterAxes, 'on');
            plot(app.filterAxes, w, h, 'red');
        end

        function results = cheby2bpf(app)
            fcl = app.ChebyshevBPFfclSlider_2.Value;
            fch = app.ChebyshevBPFfchSlider_2.Value;
            order = app.ChebyshevBPFSpinner_2.Value;
            pbr = 3; %passband ripple
            [A,B,C,D] = cheby2(order/2, pbr, [fch/(app.Fs/2)
               fcl/(app.Fs/2)]); %chevyshev2 BSF
            d =
               designfilt('bandpassiir','FilterOrder',order,'PassbandFrequency1',fch,'Pass
            sos = ss2sos(A,B,C,D);
            app.filtery = filter2(sos, app.filtery);

            [h, w] = freqz(sos);
            yyaxis (app.filterAxes, 'right');
            hold (app.filterAxes, 'on');
            plot(app.filterAxes, w, h, 'red');
        end

        function results = chebyhpf(app)
            fc = app.ChebyshevHPFSlider.Value; % cutoff frequency
            pb = app.ChebyshevHPFpbSlider.Value; % passband ripple
            order = app.ChebyshevHPFSpinner.Value;
            [b, a] = cheby1(order, pb, fc/(app.Fs/2), 'high'); %HP chebyshev
               type 1 filter, fc/(Fs/2) rad/sample
            app.filtery = filter(b, a, app.filtery);

            [h, w] = freqz(b, a);
            yyaxis (app.filterAxes, 'right');
            hold (app.filterAxes, 'on');
            plot(app.filterAxes, w, h, 'red');
        end

        function results = chebylpf(app)
            fc = app.ChebyshevLPFSlider.Value; % cutoff frequency
            pb = app.ChebyshevLPFpbSlider.Value; % passband ripple
            order = app.ChebyshevLPFSpinner.Value;
            [b, a] = cheby1(order, pb, fc/(app.Fs/2)); % order LP chebyshev
               type 1 filter
            app.filtery = filter(b, a, app.filtery);

            [h, w] = freqz(b, a);
            yyaxis (app.filterAxes, 'right');
            hold (app.filterAxes, 'on');
            plot(app.filterAxes, w, h, 'red');
        end

        function results = filt(app)
            osc(app);
            app.filtery = app.oscy;
            if strcmp(app.ButterworthHPFSwitch.Value, 'on')
                butterhpf(app);
            end

            if strcmp(app.ButterworthLPFSwitch.Value, 'on')
                butterlpf(app);
            end

            if strcmp(app.ButterworthBSFSwitch.Value, 'on')
                butterbsf(app);
            end

            if strcmp(app.ButterworthBPFSwitch.Value, 'on')
                butterbpf(app);
            end

            if strcmp(app.ChebyshevBPFSwitch.Value, 'on')
                chebybpf(app);
            end

            if strcmp(app.ChebyshevBPFSwitch_2.Value, 'on')
                cheby2bpf(app);
            end

            if strcmp(app.ChebyshevHPFSwitch.Value, 'on')
                chebyhpf(app);
            end

            if strcmp(app.ChebyshevLPFSwitch.Value, 'on')
                chebylpf(app);
            end
        end

        function results = filtpsdplot(app)
            %compute psd
            [pxx_osc, f_osc] = pwelch(app.oscy);
            [pxx, f] = pwelch(app.filtery);

            %plot
            hold (app.filterAxes, 'on');
            yyaxis (app.filterAxes, 'left');
            plot(app.filterAxes, f_osc, pxx_osc, 'blue');
            hold (app.filterAxes, 'on');
            plot(app.filterAxes, f, pxx, 'black');
        end

        function results = filtplay(app)
            player = audioplayer(app.filtery,app.Fs);
            playblocking(player);
            stop(player);
        end

        % ADSR envelope
        function results = attack(app, l)
            A = 0:1/(l-1):1;
            app.envy(1:l) = app.envy(1:l).*A;
        end

        function results = decay(app, l)
            S = app.SSlider.Value(); % Sustain level
            D = 1:-1/(l-1):S;
        end

        function results = envelope(app)
            filt(app);
            app.envy = app.filtery;
%             l = length(app.envy);

            lA = round(app.AKnob.Value()/1000*app.Fs); % length of Attack
            lD = round(app.DKnob.Value()/1000*app.Fs); % length of Decay
            SL = app.SSlider.Value(); % Sustain level
            lR = round(app.RKnob.Value()/1000*app.Fs); %length of Release

            if (SL == 1)
                SL = 0.999;
            end
            A = 0:1/(lA-1):1;
            D = 1:(SL-1)/(lD-1):SL;
            app.ADSR = [A D];
            lS = app.kp-lA-lD;
            if (lS > 0)
                S = SL*ones(1, lS);
                app.ADSR = [A D S];
            else
                app.ADSR = app.ADSR(1:app.kp);
            end

            app.envy(1:app.kp) = app.envy(1:app.kp).*app.ADSR(1:app.kp);
            if (lR > 0)
                R = app.ADSR(app.kp):-app.ADSR(app.kp)/(lR-1):0;
                app.envy(app.kp+1:length(R)+app.kp) =
                    app.envy(app.kp+1:length(R)+app.kp).*R;

                app.ADSR = [app.ADSR R];
            end
        end

        function results = envplay(app)
            player = audioplayer(app.envy,app.Fs);
            playblocking(player);
```

```matlab
            stop(player);
        end

        function results = envplot(app)
            yyaxis (app.envAxes, 'right');
            plot(app.envAxes, app.t(1:length(app.ADSR)), app.ADSR, 'LineWidth',3);
            ylim(app.envAxes, [0 1])

            yyaxis (app.envAxes, 'left');
            plot(app.envAxes, app.t, app.envy);
        end

        % amplifier
        function results = amplifier(app)
            envelope(app);
            ampfactor = app.ampSlider.Value();
            app.ampy = app.envy*ampfactor;
        end

        function results = ampplay(app)
            player = audioplayer(app.ampy,app.Fs);
            playblocking(player);
            stop(player);
        end

        function results = plotall(app)
            oscplot(app);
            filtpsdplot(app);
            try
                cla(app.envAxes,'reset')
                envplot(app);
            catch
            end
        end

        function results = presave(app)
            totaltime(app);
            amplifier(app);
            cla(app.filterAxes,'reset')
        end

        function results = play(app)
            player = audioplayer(app.y,app.Fs);
            playblocking(player);
            stop(player);
        end
    end


    methods (Access = private)

        % Button pushed function: Button
        function ButtonPushed(app, event)
```

```
            totaltime(app);
            cla(app.filterAxes,'reset')
            amplifier(app);
            plotall(app);
            ampplay(app);
        end

        % Button pushed function: oscPlayButton
        function oscPlayButtonPushed(app, event)
            playtime(app);
            osc(app);
            oscplot(app);
            oscplay(app);
        end

        % Button pushed function: filterPlayButton
        function filterPlayButtonPushed(app, event)
            cla(app.filterAxes,'reset')
            playtime(app);
            filt(app);
            filtpsdplot(app);
            filtplay(app);
        end

        % Value changed function: ButterworthHPFSwitch
        function ButterworthHPFSwitchValueChanged(app, event)
            if strcmp(app.ButterworthHPFSwitch.Value, 'on')
                app.ButterworthHPFSlider.Enable = 'on';
                app.ButterworthHPFField.Enable = 'on';
                app.ButterworthHPFSpinner.Enable = 'on';
            else
                app.ButterworthHPFSlider.Enable = 'off';
                app.ButterworthHPFField.Enable = 'off';
                app.ButterworthHPFSpinner.Enable = 'off';
            end
        end

        % Value changed function: ButterworthLPFSwitch
        function ButterworthLPFSwitchValueChanged(app, event)
            if strcmp(app.ButterworthLPFSwitch.Value, 'on')
                app.ButterworthLPFSlider.Enable = 'on';
                app.ButterworthLPFField.Enable = 'on';
                app.ButterworthLPFSpinner.Enable = 'on';
            else
                app.ButterworthLPFSlider.Enable = 'off';
                app.ButterworthLPFField.Enable = 'off';
                app.ButterworthLPFSpinner.Enable = 'off';
            end
        end

        % Value changed function: ChebyshevHPFSwitch
        function ChebyshevHPFSwitchValueChanged(app, event)
            if strcmp(app.ChebyshevHPFSwitch.Value, 'on')
                app.ChebyshevHPFSlider.Enable = 'on';
```

```matlab
                app.ChebyshevHPFpbSlider.Enable = 'on';
                app.ChebyshevHPFField.Enable = 'on';
                app.ChebyshevHPFSpinner.Enable = 'on';
            else
                app.ChebyshevHPFSlider.Enable = 'off';
                app.ChebyshevHPFpbSlider.Enable = 'off';
                app.ChebyshevHPFField.Enable = 'off';
                app.ChebyshevHPFSpinner.Enable = 'off';
            end
        end

        % Value changed function: ChebyshevLPFSwitch
        function ChebyshevLPFSwitchValueChanged(app, event)
            if strcmp(app.ChebyshevLPFSwitch.Value, 'on')
                app.ChebyshevLPFSlider.Enable = 'on';
                app.ChebyshevLPFpbSlider.Enable = 'on';
                app.ChebyshevLPFField.Enable = 'on';
                app.ChebyshevLPFSpinner.Enable = 'on';
            else
                app.ChebyshevLPFSlider.Enable = 'off';
                app.ChebyshevLPFpbSlider.Enable = 'off';
                app.ChebyshevLPFField.Enable = 'off';
                app.ChebyshevLPFSpinner.Enable = 'off';
            end
        end

        % Value changed function: ButterworthBSFSwitch
        function ButterworthBSFSwitchValueChanged(app, event)
            if strcmp(app.ButterworthBSFSwitch.Value, 'on')
                app.ButterworthBSFfclSlider.Enable = 'on';
                app.ButterworthBSFfchSlider.Enable = 'on';
                app.ButterworthBSFfclField.Enable = 'on';
                app.ButterworthBSFfchField.Enable = 'on';
                app.ButterworthBSFSpinner.Enable = 'on';
            else
                app.ButterworthBSFfclSlider.Enable = 'off';
                app.ButterworthBSFfchSlider.Enable = 'off';
                app.ButterworthBSFfclField.Enable = 'off';
                app.ButterworthBSFfchField.Enable = 'off';
                app.ButterworthBSFSpinner.Enable = 'off';
            end
        end

        % Value changed function: ButterworthBPFSwitch
        function ButterworthBPFSwitchValueChanged(app, event)
            if strcmp(app.ButterworthBPFSwitch.Value, 'on')
                app.ButterworthBPFfclSlider.Enable = 'on';
                app.ButterworthBPFfchSlider.Enable = 'on';
                app.ButterworthBPFfclField.Enable = 'on';
                app.ButterworthBPFfchField.Enable = 'on';
                app.ButterworthBPFSpinner.Enable = 'on';
            else
                app.ButterworthBPFfclSlider.Enable = 'off';
                app.ButterworthBPFfchSlider.Enable = 'off';
                app.ButterworthBPFfclField.Enable = 'off';
                app.ButterworthBPFfchField.Enable = 'off';
                app.ButterworthBPFSpinner.Enable = 'off';
            end
        end

        % Value changed function: ChebyshevBPFSwitch_2
        function ChebyshevBPFSwitch_2ValueChanged(app, event)
            if strcmp(app.ChebyshevBPFSwitch_2.Value, 'on')
                app.ChebyshevBPFfclSlider_2.Enable = 'on';
                app.ChebyshevBPFfchSlider_2.Enable = 'on';
                app.ChebyshevBPFfclField_2.Enable = 'on';
                app.ChebyshevBPFfchField_2.Enable = 'on';
                app.ChebyshevBPFSpinner_2.Enable = 'on';
            else
                app.ChebyshevBPFfclSlider_2.Enable = 'off';
                app.ChebyshevBPFfchSlider_2.Enable = 'off';
                app.ChebyshevBPFfclField_2.Enable = 'off';
                app.ChebyshevBPFfchField_2.Enable = 'off';
                app.ChebyshevBPFSpinner_2.Enable = 'off';
            end
        end

        % Value changed function: ChebyshevBPFSwitch
        function ChebyshevBPFSwitchValueChanged(app, event)
            if strcmp(app.ChebyshevBPFSwitch.Value, 'on')
                app.ChebyshevBPFfclSlider.Enable = 'on';
                app.ChebyshevBPFfchSlider.Enable = 'on';
                app.ChebyshevBPFfclField.Enable = 'on';
                app.ChebyshevBPFfchField.Enable = 'on';
                app.ChebyshevBPFSpinner.Enable = 'on';
            else
                app.ChebyshevBPFfclSlider.Enable = 'off';
                app.ChebyshevBPFfchSlider.Enable = 'off';
                app.ChebyshevBPFfclField.Enable = 'off';
                app.ChebyshevBPFfchField.Enable = 'off';
                app.ChebyshevBPFSpinner.Enable = 'off';
            end

        end

        % Button pushed function: envPlayButton
        function envPlayButtonPushed(app, event)
            totaltime(app);
            cla(app.envAxes,'reset')
            envelope(app);
            envplot(app);
            envplay(app);
        end

        % Value changed function: ButterworthLPFSlider
        function ButterworthLPFSliderValueChanged(app, event)
            app.ButterworthLPFField.Value = app.ButterworthLPFSlider.Value;
        end

        % Value changed function: ButterworthLPFField
        function ButterworthLPFFieldValueChanged(app, event)
            app.ButterworthLPFSlider.Value = app.ButterworthLPFField.Value;

        end

        % Value changed function: ButterworthHPFSlider
        function ButterworthHPFSliderValueChanged(app, event)
            app.ButterworthHPFField.Value = app.ButterworthHPFSlider.Value;

        end

        % Value changed function: ButterworthHPFField
        function ButterworthHPFFieldValueChanged(app, event)
            app.ButterworthHPFSlider.Value = app.ButterworthHPFField.Value;

        end

        % Value changed function: ChebyshevHPFSlider
        function ChebyshevHPFSliderValueChanged(app, event)
            app.ChebyshevHPFField.Value = app.ChebyshevHPFSlider.Value;

        end

        % Value changed function: ChebyshevHPFField
        function ChebyshevHPFFieldValueChanged(app, event)
            app.ChebyshevHPFSlider.Value = app.ChebyshevHPFField.Value;

        end

        % Value changed function: ChebyshevLPFSlider
        function ChebyshevLPFSliderValueChanged(app, event)
            app.ChebyshevLPFField.Value = app.ChebyshevLPFSlider.Value;

        end

        % Value changed function: ChebyshevLPFField
        function ChebyshevLPFFieldValueChanged(app, event)
            app.ChebyshevLPFSlider.Value = app.ChebyshevLPFField.Value;

        end

        % Value changed function: ButterworthBSFfclSlider
        function ButterworthBSFfclSliderValueChanged(app, event)
            app.ButterworthBSFfclField.Value =
                app.ButterworthBSFfclSlider.Value;

        end

        % Value changed function: ButterworthBSFfchSlider
        function ButterworthBSFfchSliderValueChanged(app, event)
            app.ButterworthBSFfchField.Value =
               app.ButterworthBSFfchSlider.Value;

        end

        % Value changed function: ButterworthBSFfclField
        function ButterworthBSFfclFieldValueChanged(app, event)
           app.ButterworthBSFfclSlider.Value =
              app.ButterworthBSFfclField.Value;

        end

        % Value changed function: ButterworthBSFfchField
        function ButterworthBSFfchFieldValueChanged(app, event)
            app.ButterworthBSFfchSlider.Value =
               app.ButterworthBSFfchField.Value;

        end

        % Value changed function: ButterworthBPFfclSlider
        function ButterworthBPFfclSliderValueChanged(app, event)
            app.ButterworthBPFfclField.Value =
               app.ButterworthBPFfclSlider.Value;

        end

        % Value changed function: ButterworthBPFfchSlider
        function ButterworthBPFfchSliderValueChanged(app, event)
            app.ButterworthBPFfchField.Value =
               app.ButterworthBPFfchSlider.Value;

        end

        % Value changed function: ButterworthBPFfclField
        function ButterworthBPFfclFieldValueChanged(app, event)
            app.ButterworthBPFfclSlider.Value =
               app.ButterworthBPFfclField.Value;

        end

        % Value changed function: ButterworthBPFfchField
        function ButterworthBPFfchFieldValueChanged(app, event)
            app.ButterworthBPFfchSlider.Value =
               app.ButterworthBPFfchField.Value;

        end

        % Value changed function: ChebyshevBPFfclSlider
        function ChebyshevBPFfclSliderValueChanged(app, event)
            app.ChebyshevBPFfclField.Value = app.ChebyshevBPFfclSlider.Value;

        end

        % Value changed function: ChebyshevBPFfchSlider
        function ChebyshevBPFfchSliderValueChanged(app, event)
```

```
            app.ChebyshevBPFfchField.Value = app.ChebyshevBPFfchSlider.Value;

        end

        % Value changed function: ChebyshevBPFfclField
        function ChebyshevBPFfclFieldValueChanged(app, event)
            app.ChebyshevBPFfclSlider.Value = app.ChebyshevBPFfclField.Value;

        end

        % Value changed function: ChebyshevBPFfchField
        function ChebyshevBPFfchFieldValueChanged(app, event)
            app.ChebyshevBPFfchSlider.Value = app.ChebyshevBPFfchField.Value;

        end

        % Value changed function: ChebyshevBPFfclSlider_2
        function ChebyshevBPFfclSlider_2ValueChanged(app, event)
            app.ChebyshevBPFfclField_2.Value =
               app.ChebyshevBPFfclSlider_2.Value;

        end

        % Value changed function: ChebyshevBPFfchSlider_2
        function ChebyshevBPFfchSlider_2ValueChanged(app, event)
            app.ChebyshevBPFfchField_2.Value =
               app.ChebyshevBPFfchSlider_2.Value;

        end

        % Value changed function: ChebyshevBPFfclField_2
        function ChebyshevBPFfclField_2ValueChanged(app, event)
            app.ChebyshevBPFfclSlider_2.Value =
               app.ChebyshevBPFfclField_2.Value;

        end

        % Value changed function: ChebyshevBPFfchField_2
        function ChebyshevBPFfchField_2ValueChanged(app, event)
            app.ChebyshevBPFfchSlider_2.Value =
               app.ChebyshevBPFfchField_2.Value;

        end

        % Button pushed function: NoteButton
        function NoteButtonPushed(app, event)
            if strcmp(app.SaveSwitch.Value, 'Save')
                presave(app);
                app.n1y = app.ampy;
            else
                app.y = app.n1y;
                play(app);
            end

```

```matlab
        end

        % Button pushed function: NoteButton_2
        function NoteButton_2Pushed(app, event)
            if strcmp(app.SaveSwitch.Value, 'Save')
                presave(app);
                app.n2y = app.ampy;
            else
                app.y = app.n2y;
                play(app);
            end
        end

        % Button pushed function: NoteButton_3
        function NoteButton_3Pushed(app, event)
            if strcmp(app.SaveSwitch.Value, 'Save')
                presave(app);
                app.n3y = app.ampy;
            else
                app.y = app.n3y;
                play(app);
            end
        end

        % Button pushed function: NoteButton_4
        function NoteButton_4Pushed(app, event)
            if strcmp(app.SaveSwitch.Value, 'Save')
                presave(app);
                app.n4y = app.ampy;
            else
                app.y = app.n4y;
                play(app);
            end
        end

        % Button pushed function: NoteButton_5
        function NoteButton_5Pushed(app, event)
            if strcmp(app.SaveSwitch.Value, 'Save')
                presave(app);
                app.n5y = app.ampy;
            else
                app.y = app.n5y;
                play(app);
            end
        end

        % Button pushed function: NoteButton_6
        function NoteButton_6Pushed(app, event)
            if strcmp(app.SaveSwitch.Value, 'Save')
                presave(app);
                app.n6y = app.ampy;
            else
                app.y = app.n6y;
                play(app);
            end
        end

        % Button pushed function: NoteButton_7
        function NoteButton_7Pushed(app, event)
            if strcmp(app.SaveSwitch.Value, 'Save')
                presave(app);
                app.n7y = app.ampy;
            else
                app.y = app.n7y;
                play(app);
            end
        end

        % Button pushed function: NoteButton_8
        function NoteButton_8Pushed(app, event)
            if strcmp(app.SaveSwitch.Value, 'Save')
                presave(app);
                app.n8y = app.ampy;
            else
                app.y = app.n8y;
                play(app);
            end
        end
    end

    % App initialization and construction
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure
            app.UIFigure = uifigure;
            app.UIFigure.Color = [1 1 1];
            app.UIFigure.Position = [100 100 1175 694];
            app.UIFigure.Name = 'UI Figure';
            setAutoResize(app, app.UIFigure, true)

            % Create OscillatorPanel
            app.OscillatorPanel = uipanel(app.UIFigure);
            app.OscillatorPanel.Title = 'Oscillator';
            app.OscillatorPanel.BackgroundColor = [1 1 1];
            app.OscillatorPanel.Position = [16 456 1147 202];

            % Create oscWaveKnob
            app.oscWaveKnob = uiknob(app.OscillatorPanel, 'discrete');
            app.oscWaveKnob.Items = {'Off', 'Sine', 'Triangle', 'Square',
               'Sawtooth', 'Noise'};
            app.oscWaveKnob.FontSize = 10;
            app.oscWaveKnob.Position = [354 128 33 33];
            app.oscWaveKnob.Value = 'Sine';

            % Create oscAmpKnob
            app.oscAmpKnob = uiknob(app.OscillatorPanel, 'continuous');
            app.oscAmpKnob.Limits = [0 1];
            app.oscAmpKnob.FontSize = 10;
            app.oscAmpKnob.Position = [354 57 33 33];
            app.oscAmpKnob.Value = 1;

            % Create oscFreqField
            app.oscFreqField = uieditfield(app.OscillatorPanel, 'numeric');
            app.oscFreqField.Limits = [60 22400];
            app.oscFreqField.HorizontalAlignment = 'center';
            app.oscFreqField.FontSize = 10;
            app.oscFreqField.Position = [329 8 52.015625 22];
            app.oscFreqField.Value = 440;

            % Create oscWaveKnob_2
            app.oscWaveKnob_2 = uiknob(app.OscillatorPanel, 'discrete');
            app.oscWaveKnob_2.Items = {'Off', 'Sine', 'Triangle', 'Square',
                'Sawtooth', 'Noise'};
            app.oscWaveKnob_2.FontSize = 10;
            app.oscWaveKnob_2.Position = [491 127 33 33];

            % Create oscAmpKnob_2
            app.oscAmpKnob_2 = uiknob(app.OscillatorPanel, 'continuous');
            app.oscAmpKnob_2.Limits = [0 1];
            app.oscAmpKnob_2.FontSize = 10;
            app.oscAmpKnob_2.Position = [491 56 33 33];
            app.oscAmpKnob_2.Value = 1;

            % Create oscFreqField_2
            app.oscFreqField_2 = uieditfield(app.OscillatorPanel, 'numeric');
            app.oscFreqField_2.Limits = [60 22400];
            app.oscFreqField_2.HorizontalAlignment = 'center';
            app.oscFreqField_2.FontSize = 10;
            app.oscFreqField_2.Position = [465 8 52.015625 22];
            app.oscFreqField_2.Value = 293.7;

            % Create oscWaveKnob_3
            app.oscWaveKnob_3 = uiknob(app.OscillatorPanel, 'discrete');
            app.oscWaveKnob_3.Items = {'Off', 'Sine', 'Triangle', 'Square',
                'Sawtooth', 'Noise'};
            app.oscWaveKnob_3.FontSize = 10;
            app.oscWaveKnob_3.Position = [632 127 33 33];

            % Create oscAmpKnob_3
            app.oscAmpKnob_3 = uiknob(app.OscillatorPanel, 'continuous');
            app.oscAmpKnob_3.Limits = [0 1];
            app.oscAmpKnob_3.FontSize = 10;
            app.oscAmpKnob_3.Position = [632 56 33 33];
            app.oscAmpKnob_3.Value = 1;

            % Create oscFreqField_3
            app.oscFreqField_3 = uieditfield(app.OscillatorPanel, 'numeric');
            app.oscFreqField_3.Limits = [60 22400];
            app.oscFreqField_3.HorizontalAlignment = 'center';
            app.oscFreqField_3.FontSize = 10;
            app.oscFreqField_3.Position = [624 8 52.015625 22];
            app.oscFreqField_3.Value = 329.628;

            % Create oscAxes
            app.oscAxes = uiaxes(app.OscillatorPanel);
            app.oscAxes.FontSize = 10;
            app.oscAxes.Position = [5 8 296 168];

            % Create ALabel
            app.ALabel = uilabel(app.OscillatorPanel);
            app.ALabel.FontSize = 10;
            app.ALabel.Position = [317 86 15 15];
            app.ALabel.Text = 'A';

            % Create fLabel
            app.fLabel = uilabel(app.OscillatorPanel);
            app.fLabel.FontSize = 10;
            app.fLabel.Position = [320 13 15 15];
            app.fLabel.Text = 'f';

            % Create oscWaveKnob_4
            app.oscWaveKnob_4 = uiknob(app.OscillatorPanel, 'discrete');
            app.oscWaveKnob_4.Items = {'Off', 'Sine', 'Triangle', 'Square',
                'Sawtooth', 'Noise'};
            app.oscWaveKnob_4.FontSize = 10;
            app.oscWaveKnob_4.Position = [775 128 33 33];

            % Create oscAmpKnob_4
            app.oscAmpKnob_4 = uiknob(app.OscillatorPanel, 'continuous');
            app.oscAmpKnob_4.Limits = [0 1];
            app.oscAmpKnob_4.FontSize = 10;
            app.oscAmpKnob_4.Position = [775 57 33 33];
            app.oscAmpKnob_4.Value = 1;

            % Create oscFreqField_4
            app.oscFreqField_4 = uieditfield(app.OscillatorPanel, 'numeric');
            app.oscFreqField_4.Limits = [60 22400];
            app.oscFreqField_4.HorizontalAlignment = 'center';
            app.oscFreqField_4.FontSize = 10;
            app.oscFreqField_4.Position = [767 8 52.015625 22];
            app.oscFreqField_4.Value = 391.995;

            % Create oscWaveKnob_5
            app.oscWaveKnob_5 = uiknob(app.OscillatorPanel, 'discrete');
            app.oscWaveKnob_5.Items = {'Off', 'Sine', 'Triangle', 'Square',
                'Sawtooth', 'Noise'};
            app.oscWaveKnob_5.FontSize = 10;
            app.oscWaveKnob_5.Position = [916 128 33 33];

            % Create oscAmpKnob_5
            app.oscAmpKnob_5 = uiknob(app.OscillatorPanel, 'continuous');
            app.oscAmpKnob_5.Limits = [0 1];
            app.oscAmpKnob_5.FontSize = 10;
            app.oscAmpKnob_5.Position = [916 57 33 33];
            app.oscAmpKnob_5.Value = 1;

            % Create oscFreqField_5
            app.oscFreqField_5 = uieditfield(app.OscillatorPanel, 'numeric');
            app.oscFreqField_5.Limits = [60 22400];
            app.oscFreqField_5.HorizontalAlignment = 'center';
            app.oscFreqField_5.FontSize = 10;
            app.oscFreqField_5.Position = [910 8 52.015625 22];
            app.oscFreqField_5.Value = 493.9;

            % Create oscHarmSpinner
            app.oscHarmSpinner = uispinner(app.OscillatorPanel);
            app.oscHarmSpinner.Limits = [1 9];
            app.oscHarmSpinner.ValueDisplayFormat = '%11.2g';
            app.oscHarmSpinner.FontSize = 10;
            app.oscHarmSpinner.Position = [381.8125 8 35.984375 22];
            app.oscHarmSpinner.Value = 1;

            % Create oscHarmSpinner_2
            app.oscHarmSpinner_2 = uispinner(app.OscillatorPanel);
            app.oscHarmSpinner_2.Limits = [1 9];
            app.oscHarmSpinner_2.ValueDisplayFormat = '%11.2g';
            app.oscHarmSpinner_2.FontSize = 10;
            app.oscHarmSpinner_2.Position = [517.8125 8 35.984375 22];
            app.oscHarmSpinner_2.Value = 1;

            % Create oscWaveKnob_6
            app.oscWaveKnob_6 = uiknob(app.OscillatorPanel, 'discrete');
            app.oscWaveKnob_6.Items = {'Off', 'Sine', 'Triangle', 'Square',
                'Sawtooth', 'Noise'};
            app.oscWaveKnob_6.FontSize = 10;
            app.oscWaveKnob_6.Position = [1053 128 33 33];

            % Create oscAmpKnob_6
            app.oscAmpKnob_6 = uiknob(app.OscillatorPanel, 'continuous');
            app.oscAmpKnob_6.Limits = [0 1];
            app.oscAmpKnob_6.FontSize = 10;
            app.oscAmpKnob_6.Position = [1053 57 33 33];
            app.oscAmpKnob_6.Value = 1;

            % Create oscFreqField_6
            app.oscFreqField_6 = uieditfield(app.OscillatorPanel, 'numeric');
            app.oscFreqField_6.Limits = [60 22400];
            app.oscFreqField_6.HorizontalAlignment = 'center';
            app.oscFreqField_6.FontSize = 10;
            app.oscFreqField_6.Position = [1047 8 52.015625 22];
            app.oscFreqField_6.Value = 659;

            % Create KeyPressDurationsecEditFieldLabel
            app.KeyPressDurationsecEditFieldLabel = uilabel(app.UIFigure);
            app.KeyPressDurationsecEditFieldLabel.VerticalAlignment =
                'center';
            app.KeyPressDurationsecEditFieldLabel.Position = [904 202 139 15];
            app.KeyPressDurationsecEditFieldLabel.Text = 'Key Press Duration (sec)';

            % Create keyDurationField
            app.keyDurationField = uieditfield(app.UIFigure, 'numeric');
            app.keyDurationField.HorizontalAlignment = 'center';
            app.keyDurationField.Position = [904 173 65 22];
            app.keyDurationField.Value = 0.5;

            % Create Button
            app.Button = uibutton(app.UIFigure, 'push');
            app.Button.ButtonPushedFcn = createCallbackFcn(app, @ButtonPushed, true);
            app.Button.VerticalAlignment = 'top';
            app.Button.BackgroundColor = [1 1 1];
            app.Button.FontSize = 48;
            app.Button.FontColor = [0.2745 0.3843 0.4196];
            app.Button.Position = [1015 105 124 76];
            app.Button.Text = 'âŰů';

            % Create FilterPanel
            app.FilterPanel = uipanel(app.UIFigure);
            app.FilterPanel.Title = 'Filter';
            app.FilterPanel.BackgroundColor = [1 1 1];
            app.FilterPanel.Position = [16 233 1147 215];

            % Create ButterworthLPFSlider
            app.ButterworthLPFSlider = uislider(app.FilterPanel);
            app.ButterworthLPFSlider.Limits = [60 20000];
            app.ButterworthLPFSlider.MajorTickLabels = {'60', '', '2e+04'};
            app.ButterworthLPFSlider.ValueChangedFcn = createCallbackFcn(app, @ButterworthLPFSliderValueChanged, true);
            app.ButterworthLPFSlider.Enable = 'off';
            app.ButterworthLPFSlider.FontSize = 10;
            app.ButterworthLPFSlider.Position = [489.6875 155 97 3];
            app.ButterworthLPFSlider.Value = 8000;

            % Create filterAxes
            app.filterAxes = uiaxes(app.FilterPanel);
            xlabel(app.filterAxes, 'Hz');
            app.filterAxes.FontSize = 10;
            app.filterAxes.Position = [5 9 296 181];

            % Create ButterworthHPFSlider
            app.ButterworthHPFSlider = uislider(app.FilterPanel);
            app.ButterworthHPFSlider.Limits = [20 20000];
            app.ButterworthHPFSlider.MajorTickLabels = {'20', '', '2e+04'};
            app.ButterworthHPFSlider.ValueChangedFcn = createCallbackFcn(app, @ButterworthHPFSliderValueChanged, true);
            app.ButterworthHPFSlider.Enable = 'off';
            app.ButterworthHPFSlider.FontSize = 10;
            app.ButterworthHPFSlider.Position = [343.6875 155 97 3];
            app.ButterworthHPFSlider.Value = 60;

            % Create ButterworthHPFSwitch
            app.ButterworthHPFSwitch = uiswitch(app.FilterPanel, 'slider');
            app.ButterworthHPFSwitch.Items = {'off', 'on'};
            app.ButterworthHPFSwitch.Orientation = 'vertical';
            app.ButterworthHPFSwitch.ValueChangedFcn = createCallbackFcn(app,
               @ButterworthHPFSwitchValueChanged, true);
            app.ButterworthHPFSwitch.FontColor = [1 1 1];
            app.ButterworthHPFSwitch.Position = [317 129 15 33.75];
            app.ButterworthHPFSwitch.Value = 'off';

            % Create ButterHPFLabel
            app.ButterHPFLabel = uilabel(app.FilterPanel);
            app.ButterHPFLabel.VerticalAlignment = 'center';
            app.ButterHPFLabel.FontSize = 10;
            app.ButterHPFLabel.Position = [356 170 55 15];
            app.ButterHPFLabel.Text = 'Butter HPF';

            % Create ButterworthLPFSwitch
            app.ButterworthLPFSwitch = uiswitch(app.FilterPanel, 'slider');
            app.ButterworthLPFSwitch.Items = {'off', 'on'};
            app.ButterworthLPFSwitch.Orientation = 'vertical';
            app.ButterworthLPFSwitch.ValueChangedFcn = createCallbackFcn(app,
               @ButterworthLPFSwitchValueChanged, true);
            app.ButterworthLPFSwitch.FontColor = [1 1 1];
            app.ButterworthLPFSwitch.Position = [461 129 15 33.75];
            app.ButterworthLPFSwitch.Value = 'off';

            % Create ButterLPFLabel
            app.ButterLPFLabel = uilabel(app.FilterPanel);
            app.ButterLPFLabel.VerticalAlignment = 'center';
            app.ButterLPFLabel.FontSize = 10;
            app.ButterLPFLabel.Position = [499 169 53 15];
            app.ButterLPFLabel.Text = 'Butter LPF';

            % Create ChebyshevHPFSwitch
            app.ChebyshevHPFSwitch = uiswitch(app.FilterPanel, 'slider');
            app.ChebyshevHPFSwitch.Items = {'off', 'on'};
            app.ChebyshevHPFSwitch.Orientation = 'vertical';
            app.ChebyshevHPFSwitch.ValueChangedFcn = createCallbackFcn(app,
               @ChebyshevHPFSwitchValueChanged, true);
            app.ChebyshevHPFSwitch.FontColor = [1 1 1];
            app.ChebyshevHPFSwitch.Position = [317 46 15 33.75];
            app.ChebyshevHPFSwitch.Value = 'off';

            % Create Cheby1HPFLabel
            app.Cheby1HPFLabel = uilabel(app.FilterPanel);
            app.Cheby1HPFLabel.VerticalAlignment = 'center';
            app.Cheby1HPFLabel.FontSize = 10;
            app.Cheby1HPFLabel.Position = [356 85 63 15];
            app.Cheby1HPFLabel.Text = 'Cheby1 HPF';

            % Create ChebyshevHPFSlider
            app.ChebyshevHPFSlider = uislider(app.FilterPanel);
            app.ChebyshevHPFSlider.Limits = [20 20000];
            app.ChebyshevHPFSlider.MajorTickLabels = {'20', '', '2e+04'};
            app.ChebyshevHPFSlider.ValueChangedFcn = createCallbackFcn(app,
              @ChebyshevHPFSliderValueChanged, true);
            app.ChebyshevHPFSlider.Enable = 'off';
            app.ChebyshevHPFSlider.FontSize = 10;
            app.ChebyshevHPFSlider.Position = [345.6875 70 97 3];
            app.ChebyshevHPFSlider.Value = 60;

            % Create ChebyshevHPFpbSlider
            app.ChebyshevHPFpbSlider = uislider(app.FilterPanel);
            app.ChebyshevHPFpbSlider.Limits = [1 30];
            app.ChebyshevHPFpbSlider.MajorTickLabels = {'1', '', '', '', '',
              '', '', '30'};
            app.ChebyshevHPFpbSlider.Enable = 'off';
            app.ChebyshevHPFpbSlider.FontSize = 10;
            app.ChebyshevHPFpbSlider.Position = [346.6875 32 97 3];
            app.ChebyshevHPFpbSlider.Value = 10;

            % Create ChebyshevLPFSwitch
            app.ChebyshevLPFSwitch = uiswitch(app.FilterPanel, 'slider');
            app.ChebyshevLPFSwitch.Items = {'off', 'on'};
            app.ChebyshevLPFSwitch.Orientation = 'vertical';
            app.ChebyshevLPFSwitch.ValueChangedFcn = createCallbackFcn(app,
              @ChebyshevLPFSwitchValueChanged, true);
            app.ChebyshevLPFSwitch.FontColor = [1 1 1];
            app.ChebyshevLPFSwitch.Position = [461 48 15 33.75];
            app.ChebyshevLPFSwitch.Value = 'off';

            % Create Cheby1LPFLabel
            app.Cheby1LPFLabel = uilabel(app.FilterPanel);
            app.Cheby1LPFLabel.VerticalAlignment = 'center';
            app.Cheby1LPFLabel.FontSize = 10;
            app.Cheby1LPFLabel.Position = [499 85 61 15];
            app.Cheby1LPFLabel.Text = 'Cheby1 LPF';

            % Create ChebyshevLPFSlider
            app.ChebyshevLPFSlider = uislider(app.FilterPanel);
            app.ChebyshevLPFSlider.Limits = [60 20000];
            app.ChebyshevLPFSlider.MajorTickLabels = {'60', '', '2e+04'};
            app.ChebyshevLPFSlider.ValueChangedFcn = createCallbackFcn(app,
              @ChebyshevLPFSliderValueChanged, true);
            app.ChebyshevLPFSlider.Enable = 'off';
            app.ChebyshevLPFSlider.FontSize = 10;
            app.ChebyshevLPFSlider.Position = [489.6875 72 97 3];
            app.ChebyshevLPFSlider.Value = 8000;

            % Create ChebyshevLPFpbSlider
            app.ChebyshevLPFpbSlider = uislider(app.FilterPanel);
            app.ChebyshevLPFpbSlider.Limits = [1 30];
            app.ChebyshevLPFpbSlider.MajorTickLabels = {'1', '', '', '', '',
              '', '', '30'};
            app.ChebyshevLPFpbSlider.Enable = 'off';
            app.ChebyshevLPFpbSlider.FontSize = 10;
            app.ChebyshevLPFpbSlider.Position = [490.6875 34 97 3];
            app.ChebyshevLPFpbSlider.Value = 10;

            % Create ButterworthBSFfclSlider
            app.ButterworthBSFfclSlider = uislider(app.FilterPanel);
            app.ButterworthBSFfclSlider.Limits = [60 20000];
            app.ButterworthBSFfclSlider.MajorTickLabels = {'60', '', '2e+04'};
            app.ButterworthBSFfclSlider.ValueChangedFcn =
                createCallbackFcn(app, @ButterworthBSFfclSliderValueChanged,
                true);
            app.ButterworthBSFfclSlider.Enable = 'off';
            app.ButterworthBSFfclSlider.FontSize = 10;
            app.ButterworthBSFfclSlider.Position = [756.6875 155 97 3];
            app.ButterworthBSFfclSlider.Value = 8000;

            % Create ButterworthBSFSwitch
            app.ButterworthBSFSwitch = uiswitch(app.FilterPanel, 'slider');
            app.ButterworthBSFSwitch.Items = {'off', 'on'};
            app.ButterworthBSFSwitch.Orientation = 'vertical';
            app.ButterworthBSFSwitch.ValueChangedFcn = createCallbackFcn(app,
                @ButterworthBSFSwitchValueChanged, true);
            app.ButterworthBSFSwitch.FontColor = [1 1 1];
            app.ButterworthBSFSwitch.Position = [610 130 15 33.75];
            app.ButterworthBSFSwitch.Value = 'off';

            % Create ButterBSFLabel
            app.ButterBSFLabel = uilabel(app.FilterPanel);
            app.ButterBSFLabel.VerticalAlignment = 'center';
            app.ButterBSFLabel.FontSize = 10;
            app.ButterBSFLabel.Position = [652 168 54 15];
            app.ButterBSFLabel.Text = 'Butter BSF';

            % Create ButterworthBSFfchSlider
            app.ButterworthBSFfchSlider = uislider(app.FilterPanel);
            app.ButterworthBSFfchSlider.Limits = [60 20000];
            app.ButterworthBSFfchSlider.MajorTickLabels = {'60', '', '2e+04'};
            app.ButterworthBSFfchSlider.ValueChangedFcn =
                createCallbackFcn(app, @ButterworthBSFfchSliderValueChanged,
                true);
            app.ButterworthBSFfchSlider.Enable = 'off';
            app.ButterworthBSFfchSlider.FontSize = 10;
            app.ButterworthBSFfchSlider.Position = [638.6875 155 97 3];
            app.ButterworthBSFfchSlider.Value = 60;

            % Create ButterworthBPFfclSlider
            app.ButterworthBPFfclSlider = uislider(app.FilterPanel);
            app.ButterworthBPFfclSlider.Limits = [60 20000];
            app.ButterworthBPFfclSlider.MajorTickLabels = {'60', '1.003e+04',
                '2e+04'};
            app.ButterworthBPFfclSlider.ValueChangedFcn =
                createCallbackFcn(app, @ButterworthBPFfclSliderValueChanged,
                true);
            app.ButterworthBPFfclSlider.Enable = 'off';
            app.ButterworthBPFfclSlider.FontSize = 10;
            app.ButterworthBPFfclSlider.Position = [1020.6875 155 97 3];
            app.ButterworthBPFfclSlider.Value = 8000;

            % Create ButterworthBPFSwitch
            app.ButterworthBPFSwitch = uiswitch(app.FilterPanel, 'slider');
            app.ButterworthBPFSwitch.Items = {'off', 'on'};
            app.ButterworthBPFSwitch.Orientation = 'vertical';
            app.ButterworthBPFSwitch.ValueChangedFcn = createCallbackFcn(app,
                @ButterworthBPFSwitchValueChanged, true);
            app.ButterworthBPFSwitch.FontColor = [1 1 1];
            app.ButterworthBPFSwitch.Position = [874 130 15 33.75];
            app.ButterworthBPFSwitch.Value = 'off';

            % Create ButterBPFLabel
            app.ButterBPFLabel = uilabel(app.FilterPanel);
            app.ButterBPFLabel.VerticalAlignment = 'center';
            app.ButterBPFLabel.FontSize = 10;
            app.ButterBPFLabel.Position = [916 169 54 15];
            app.ButterBPFLabel.Text = 'Butter BPF';

            % Create ButterworthBPFfchSlider
            app.ButterworthBPFfchSlider = uislider(app.FilterPanel);
            app.ButterworthBPFfchSlider.Limits = [60 20000];
            app.ButterworthBPFfchSlider.MajorTickLabels = {'60', '1.003e+04',
                '2e+04'};
            app.ButterworthBPFfchSlider.ValueChangedFcn =
                createCallbackFcn(app, @ButterworthBPFfchSliderValueChanged,
                true);
            app.ButterworthBPFfchSlider.Enable = 'off';
            app.ButterworthBPFfchSlider.FontSize = 10;
            app.ButterworthBPFfchSlider.Position = [902.6875 155 97 3];
            app.ButterworthBPFfchSlider.Value = 60;

            % Create ChebyshevBPFfclSlider_2
            app.ChebyshevBPFfclSlider_2 = uislider(app.FilterPanel);
            app.ChebyshevBPFfclSlider_2.Limits = [60 20000];
            app.ChebyshevBPFfclSlider_2.ValueChangedFcn =
                createCallbackFcn(app, @ChebyshevBPFfclSlider_2ValueChanged,
                true);
            app.ChebyshevBPFfclSlider_2.Enable = 'off';
            app.ChebyshevBPFfclSlider_2.FontSize = 10;
            app.ChebyshevBPFfclSlider_2.Position = [1020.6875 72 97 3];
            app.ChebyshevBPFfclSlider_2.Value = 8000;

            % Create ChebyshevBPFSwitch_2
            app.ChebyshevBPFSwitch_2 = uiswitch(app.FilterPanel, 'slider');
            app.ChebyshevBPFSwitch_2.Items = {'off', 'on'};
            app.ChebyshevBPFSwitch_2.Orientation = 'vertical';
            app.ChebyshevBPFSwitch_2.ValueChangedFcn = createCallbackFcn(app,
                @ChebyshevBPFSwitch_2ValueChanged, true);
            app.ChebyshevBPFSwitch_2.FontColor = [1 1 1];
            app.ChebyshevBPFSwitch_2.Position = [874 48 15 33.75];
            app.ChebyshevBPFSwitch_2.Value = 'off';

            % Create Cheby2BPFLabel
            app.Cheby2BPFLabel = uilabel(app.FilterPanel);
            app.Cheby2BPFLabel.VerticalAlignment = 'center';
            app.Cheby2BPFLabel.FontSize = 10;
            app.Cheby2BPFLabel.Position = [921 85 62 15];
            app.Cheby2BPFLabel.Text = 'Cheby2 BPF';

            % Create ChebyshevBPFfchSlider_2
            app.ChebyshevBPFfchSlider_2 = uislider(app.FilterPanel);
            app.ChebyshevBPFfchSlider_2.Limits = [60 20000];
            app.ChebyshevBPFfchSlider_2.MajorTickLabels = {'60', '1.003e+04',
                '2e+04'};
            app.ChebyshevBPFfchSlider_2.ValueChangedFcn =
                createCallbackFcn(app, @ChebyshevBPFfchSlider_2ValueChanged,
                true);
            app.ChebyshevBPFfchSlider_2.Enable = 'off';
            app.ChebyshevBPFfchSlider_2.FontSize = 10;
            app.ChebyshevBPFfchSlider_2.Position = [902.6875 72 97 3];
            app.ChebyshevBPFfchSlider_2.Value = 60;

            % Create ChebyshevBPFfclSlider
            app.ChebyshevBPFfclSlider = uislider(app.FilterPanel);
            app.ChebyshevBPFfclSlider.Limits = [60 20000];
            app.ChebyshevBPFfclSlider.MajorTickLabels = {'60', '1.003e+04',
                '2e+04'};
            app.ChebyshevBPFfclSlider.ValueChangedFcn =
                createCallbackFcn(app, @ChebyshevBPFfclSliderValueChanged,
                true);
            app.ChebyshevBPFfclSlider.Enable = 'off';
            app.ChebyshevBPFfclSlider.FontSize = 10;
            app.ChebyshevBPFfclSlider.Position = [755.6875 72 97 3];
            app.ChebyshevBPFfclSlider.Value = 8000;

            % Create ChebyshevBPFSwitch
            app.ChebyshevBPFSwitch = uiswitch(app.FilterPanel, 'slider');
            app.ChebyshevBPFSwitch.Items = {'off', 'on'};
            app.ChebyshevBPFSwitch.Orientation = 'vertical';
            app.ChebyshevBPFSwitch.ValueChangedFcn = createCallbackFcn(app,
                @ChebyshevBPFSwitchValueChanged, true);
            app.ChebyshevBPFSwitch.FontColor = [1 1 1];
            app.ChebyshevBPFSwitch.Position = [609 48 15 33.75];
            app.ChebyshevBPFSwitch.Value = 'off';

            % Create Cheby1BPFLabel
            app.Cheby1BPFLabel = uilabel(app.FilterPanel);
            app.Cheby1BPFLabel.VerticalAlignment = 'center';
            app.Cheby1BPFLabel.FontSize = 10;
            app.Cheby1BPFLabel.Position = [652 85 62 15];
            app.Cheby1BPFLabel.Text = 'Cheby1 BPF';

            % Create ChebyshevBPFfchSlider
            app.ChebyshevBPFfchSlider = uislider(app.FilterPanel);
            app.ChebyshevBPFfchSlider.Limits = [60 20000];
            app.ChebyshevBPFfchSlider.MajorTickLabels = {'60', '1.003e+04',
                '2e+04'};
            app.ChebyshevBPFfchSlider.ValueChangedFcn =
                createCallbackFcn(app, @ChebyshevBPFfchSliderValueChanged,
                true);
            app.ChebyshevBPFfchSlider.Enable = 'off';
            app.ChebyshevBPFfchSlider.FontSize = 10;
            app.ChebyshevBPFfchSlider.Position = [637.6875 72 97 3];
            app.ChebyshevBPFfchSlider.Value = 60;

            % Create ButterworthHPFField
            app.ButterworthHPFField = uieditfield(app.FilterPanel, 'numeric');
            app.ButterworthHPFField.ValueChangedFcn = createCallbackFcn(app,
                @ButterworthHPFFieldValueChanged, true);
            app.ButterworthHPFField.Limits = [20 20000];
            app.ButterworthHPFField.Enable = 'off';
            app.ButterworthHPFField.FontSize = 10;
            app.ButterworthHPFField.Position = [410.015625 167 39.015625 22];
            app.ButterworthHPFField.Value = 60;

            % Create ButterworthLPFField
            app.ButterworthLPFField = uieditfield(app.FilterPanel, 'numeric');
            app.ButterworthLPFField.ValueChangedFcn = createCallbackFcn(app,
                @ButterworthLPFFieldValueChanged, true);
            app.ButterworthLPFField.Limits = [60 20000];
            app.ButterworthLPFField.Enable = 'off';
            app.ButterworthLPFField.FontSize = 10;
            app.ButterworthLPFField.Position = [554.015625 168 39.015625 22];
            app.ButterworthLPFField.Value = 8000;

            % Create ButterworthBSFfclField
            app.ButterworthBSFfclField = uieditfield(app.FilterPanel,
                'numeric');
            app.ButterworthBSFfclField.ValueChangedFcn =
                createCallbackFcn(app, @ButterworthBSFfclFieldValueChanged,
                true);
            app.ButterworthBSFfclField.Limits = [60 20000];
            app.ButterworthBSFfclField.Enable = 'off';
            app.ButterworthBSFfclField.FontSize = 10;
            app.ButterworthBSFfclField.Position = [807.015625 168 39.015625
                22];
            app.ButterworthBSFfclField.Value = 8000;

            % Create ButterworthBSFfchField
            app.ButterworthBSFfchField = uieditfield(app.FilterPanel,
                'numeric');
            app.ButterworthBSFfchField.ValueChangedFcn =
                createCallbackFcn(app, @ButterworthBSFfchFieldValueChanged,
                true);
            app.ButterworthBSFfchField.Limits = [60 20000];
            app.ButterworthBSFfchField.Enable = 'off';
            app.ButterworthBSFfchField.FontSize = 10;
            app.ButterworthBSFfchField.Position = [718.015625 167 39.015625
                22];
            app.ButterworthBSFfchField.Value = 60;

            % Create ButterworthBPFfclField
            app.ButterworthBPFfclField = uieditfield(app.FilterPanel,
                'numeric');
            app.ButterworthBPFfclField.ValueChangedFcn =
                createCallbackFcn(app, @ButterworthBPFfclFieldValueChanged,
                true);
            app.ButterworthBPFfclField.Limits = [60 20000];
            app.ButterworthBPFfclField.Enable = 'off';
            app.ButterworthBPFfclField.FontSize = 10;
            app.ButterworthBPFfclField.Position = [1079.015625 168 39.015625
                22];
            app.ButterworthBPFfclField.Value = 8000;

            % Create ButterworthBPFfchField
            app.ButterworthBPFfchField = uieditfield(app.FilterPanel,
                'numeric');
            app.ButterworthBPFfchField.ValueChangedFcn =
                createCallbackFcn(app, @ButterworthBPFfchFieldValueChanged,
                true);
            app.ButterworthBPFfchField.Limits = [60 20000];
            app.ButterworthBPFfchField.Enable = 'off';
            app.ButterworthBPFfchField.FontSize = 10;
            app.ButterworthBPFfchField.Position = [983.015625 168 39.015625
                22];
            app.ButterworthBPFfchField.Value = 60;

            % Create ChebyshevBPFfclField_2
            app.ChebyshevBPFfclField_2 = uieditfield(app.FilterPanel,
                'numeric');
            app.ChebyshevBPFfclField_2.ValueChangedFcn =
                createCallbackFcn(app, @ChebyshevBPFfclField_2ValueChanged,
                true);
            app.ChebyshevBPFfclField_2.Limits = [60 20000];
            app.ChebyshevBPFfclField_2.Enable = 'off';
            app.ChebyshevBPFfclField_2.FontSize = 10;
            app.ChebyshevBPFfclField_2.Position = [1079.015625 85 39.015625
                22];
            app.ChebyshevBPFfclField_2.Value = 8000;

            % Create ChebyshevBPFfchField_2
            app.ChebyshevBPFfchField_2 = uieditfield(app.FilterPanel,
                'numeric');
            app.ChebyshevBPFfchField_2.ValueChangedFcn =
                createCallbackFcn(app, @ChebyshevBPFfchField_2ValueChanged,
                true);
            app.ChebyshevBPFfchField_2.Limits = [60 20000];
            app.ChebyshevBPFfchField_2.Enable = 'off';
            app.ChebyshevBPFfchField_2.FontSize = 10;
            app.ChebyshevBPFfchField_2.Position = [983.015625 85 39.015625
                22];
            app.ChebyshevBPFfchField_2.Value = 60;

            % Create ChebyshevBPFfclField
            app.ChebyshevBPFfclField = uieditfield(app.FilterPanel,
                'numeric');
            app.ChebyshevBPFfclField.ValueChangedFcn = createCallbackFcn(app,
                @ChebyshevBPFfclFieldValueChanged, true);
            app.ChebyshevBPFfclField.Limits = [60 20000];
            app.ChebyshevBPFfclField.Enable = 'off';
            app.ChebyshevBPFfclField.FontSize = 10;
            app.ChebyshevBPFfclField.Position = [807.015625 85 39.015625 22];
            app.ChebyshevBPFfclField.Value = 8000;

            % Create ChebyshevBPFfchField
            app.ChebyshevBPFfchField = uieditfield(app.FilterPanel,
                'numeric');
            app.ChebyshevBPFfchField.ValueChangedFcn = createCallbackFcn(app,
                @ChebyshevBPFfchFieldValueChanged, true);
            app.ChebyshevBPFfchField.Limits = [60 20000];
            app.ChebyshevBPFfchField.Enable = 'off';
            app.ChebyshevBPFfchField.FontSize = 10;
            app.ChebyshevBPFfchField.Position = [718.015625 85 39.015625 22];
            app.ChebyshevBPFfchField.Value = 60;

            % Create ChebyshevLPFField
            app.ChebyshevLPFField = uieditfield(app.FilterPanel, 'numeric');
            app.ChebyshevLPFField.ValueChangedFcn = createCallbackFcn(app,
                @ChebyshevLPFFieldValueChanged, true);
            app.ChebyshevLPFField.Limits = [60 20000];
            app.ChebyshevLPFField.Enable = 'off';
            app.ChebyshevLPFField.FontSize = 10;
            app.ChebyshevLPFField.Position = [559.015625 85 39.015625 22];
            app.ChebyshevLPFField.Value = 8000;

            % Create ChebyshevHPFField
            app.ChebyshevHPFField = uieditfield(app.FilterPanel, 'numeric');
            app.ChebyshevHPFField.ValueChangedFcn = createCallbackFcn(app,
                @ChebyshevHPFFieldValueChanged, true);
            app.ChebyshevHPFField.Limits = [20 20000];
            app.ChebyshevHPFField.Enable = 'off';
            app.ChebyshevHPFField.FontSize = 10;
            app.ChebyshevHPFField.Position = [417.015625 85 39.015625 22];
            app.ChebyshevHPFField.Value = 60;

            % Create ButterworthHPFSpinner
            app.ButterworthHPFSpinner = uispinner(app.FilterPanel);
            app.ButterworthHPFSpinner.Limits = [1 9];
            app.ButterworthHPFSpinner.ValueDisplayFormat = '%11.2g';
            app.ButterworthHPFSpinner.Enable = 'off';
            app.ButterworthHPFSpinner.FontSize = 10;
            app.ButterworthHPFSpinner.Position = [316.796875 168 36 22];
            app.ButterworthHPFSpinner.Value = 1;

            % Create ButterworthLPFSpinner
            app.ButterworthLPFSpinner = uispinner(app.FilterPanel);
            app.ButterworthLPFSpinner.Limits = [1 9];
            app.ButterworthLPFSpinner.ValueDisplayFormat = '%11.2g';
            app.ButterworthLPFSpinner.Enable = 'off';
            app.ButterworthLPFSpinner.FontSize = 10;
            app.ButterworthLPFSpinner.Position = [459.796875 167 36 22];
            app.ButterworthLPFSpinner.Value = 1;

            % Create ButterworthBSFSpinner
            app.ButterworthBSFSpinner = uispinner(app.FilterPanel);
            app.ButterworthBSFSpinner.Step = 2;
            app.ButterworthBSFSpinner.Limits = [2 20];
            app.ButterworthBSFSpinner.ValueDisplayFormat = '%11.2g';
            app.ButterworthBSFSpinner.Enable = 'off';
            app.ButterworthBSFSpinner.FontSize = 10;
            app.ButterworthBSFSpinner.Position = [605.796875 167 41 22];
            app.ButterworthBSFSpinner.Value = 2;

            % Create ChebyshevHPFSpinner
            app.ChebyshevHPFSpinner = uispinner(app.FilterPanel);
            app.ChebyshevHPFSpinner.Limits = [1 9];
            app.ChebyshevHPFSpinner.ValueDisplayFormat = '%11.2g';
            app.ChebyshevHPFSpinner.Enable = 'off';
            app.ChebyshevHPFSpinner.FontSize = 10;
            app.ChebyshevHPFSpinner.Position = [316.796875 85 36 22];
            app.ChebyshevHPFSpinner.Value = 1;

            % Create ChebyshevLPFSpinner
            app.ChebyshevLPFSpinner = uispinner(app.FilterPanel);
            app.ChebyshevLPFSpinner.Limits = [1 9];
            app.ChebyshevLPFSpinner.ValueDisplayFormat = '%11.2g';
            app.ChebyshevLPFSpinner.Enable = 'off';
            app.ChebyshevLPFSpinner.FontSize = 10;
            app.ChebyshevLPFSpinner.Position = [460.796875 85 36 22];
            app.ChebyshevLPFSpinner.Value = 1;

            % Create ChebyshevBPFSpinner
            app.ChebyshevBPFSpinner = uispinner(app.FilterPanel);
            app.ChebyshevBPFSpinner.Step = 2;
            app.ChebyshevBPFSpinner.Limits = [2 20];
            app.ChebyshevBPFSpinner.ValueDisplayFormat = '%11.2g';
            app.ChebyshevBPFSpinner.Enable = 'off';
            app.ChebyshevBPFSpinner.FontSize = 10;
            app.ChebyshevBPFSpinner.Position = [605.796875 85 43 22];
            app.ChebyshevBPFSpinner.Value = 2;

            % Create ChebyshevBPFSpinner_2
            app.ChebyshevBPFSpinner_2 = uispinner(app.FilterPanel);
            app.ChebyshevBPFSpinner_2.Step = 2;
            app.ChebyshevBPFSpinner_2.Limits = [2 20];
            app.ChebyshevBPFSpinner_2.ValueDisplayFormat = '%11.2g';
            app.ChebyshevBPFSpinner_2.Enable = 'off';
            app.ChebyshevBPFSpinner_2.FontSize = 10;
            app.ChebyshevBPFSpinner_2.Position = [873.796875 85 43 22];
            app.ChebyshevBPFSpinner_2.Value = 2;

            % Create ButterworthBPFSpinner
            app.ButterworthBPFSpinner = uispinner(app.FilterPanel);
            app.ButterworthBPFSpinner.Step = 2;
            app.ButterworthBPFSpinner.Limits = [2 20];
            app.ButterworthBPFSpinner.ValueDisplayFormat = '%11.2g';
            app.ButterworthBPFSpinner.Enable = 'off';
            app.ButterworthBPFSpinner.FontSize = 10;
            app.ButterworthBPFSpinner.Position = [869.796875 167 41 22];
            app.ButterworthBPFSpinner.Value = 2;

            % Create EnvelopePanel
            app.EnvelopePanel = uipanel(app.UIFigure);
            app.EnvelopePanel.Title = 'Envelope';
            app.EnvelopePanel.BackgroundColor = [1 1 1];
            app.EnvelopePanel.Position = [16 33 676 188];

            % Create envAxes
            app.envAxes = uiaxes(app.EnvelopePanel);
            xlabel(app.envAxes, 'time');
            app.envAxes.Position = [5 6 296 154];

            % Create AKnobLabel
            app.AKnobLabel = uilabel(app.EnvelopePanel);
            app.AKnobLabel.HorizontalAlignment = 'center';
            app.AKnobLabel.FontSize = 10;
            app.AKnobLabel.Position = [392 56 25 15];
            app.AKnobLabel.Text = 'A';

            % Create AKnob
            app.AKnob = uiknob(app.EnvelopePanel, 'continuous');
            app.AKnob.Limits = [0 5000];
            app.AKnob.FontSize = 10;
            app.AKnob.Position = [386 78 37 37];
            app.AKnob.Value = 10;

            % Create DKnobLabel
            app.DKnobLabel = uilabel(app.EnvelopePanel);
            app.DKnobLabel.HorizontalAlignment = 'center';
            app.DKnobLabel.FontSize = 10;
            app.DKnobLabel.Position = [500 56 25 15];
            app.DKnobLabel.Text = 'D';

            % Create DKnob
            app.DKnob = uiknob(app.EnvelopePanel, 'continuous');
            app.DKnob.Limits = [0 5000];
            app.DKnob.FontSize = 10;
            app.DKnob.Position = [494 77 38 38];
            app.DKnob.Value = 200;

            % Create RKnobLabel
            app.RKnobLabel = uilabel(app.EnvelopePanel);
            app.RKnobLabel.HorizontalAlignment = 'center';
            app.RKnobLabel.FontSize = 10;
            app.RKnobLabel.Position = [607 55 25 15];
            app.RKnobLabel.Text = 'R';
            % Create RKnob
            app.RKnob = uiknob(app.EnvelopePanel, 'continuous');
            app.RKnob.Limits = [0 5000];
            app.RKnob.FontSize = 10;
            app.RKnob.Position = [601 77 38 38];
            app.RKnob.Value = 1000;

            % Create SSliderLabel
            app.SSliderLabel = uilabel(app.EnvelopePanel);
            app.SSliderLabel.HorizontalAlignment = 'right';
            app.SSliderLabel.FontSize = 10;
            app.SSliderLabel.Position = [297 22 25 15];
            app.SSliderLabel.Text = 'S';

            % Create SSlider
            app.SSlider = uislider(app.EnvelopePanel);
            app.SSlider.Limits = [0 1];
            app.SSlider.Orientation = 'vertical';
            app.SSlider.FontSize = 10;
            app.SSlider.Position = [307.6875 45 3 108];
            app.SSlider.Value = 0.5;

            % Create AmplifierPanel
            app.AmplifierPanel = uipanel(app.UIFigure);
            app.AmplifierPanel.Title = 'Amplifier';
            app.AmplifierPanel.BackgroundColor = [1 1 1];
            app.AmplifierPanel.Position = [699 33 64 188];

            % Create AMPSliderLabel
            app.AMPSliderLabel = uilabel(app.AmplifierPanel);
            app.AMPSliderLabel.HorizontalAlignment = 'center';
            app.AMPSliderLabel.VerticalAlignment = 'center';
            app.AMPSliderLabel.Position = [15 6 32 15];
            app.AMPSliderLabel.Text = 'AMP';

            % Create ampSlider
            app.ampSlider = uislider(app.AmplifierPanel);
            app.ampSlider.Limits = [0 4];
            app.ampSlider.MajorTicks = [0 1 2 3 4];
            app.ampSlider.MajorTickLabels = {'0', '', '', '', '4'};
            app.ampSlider.Orientation = 'vertical';
            app.ampSlider.Position = [16.6875 25 3 130];
            app.ampSlider.Value = 1;

            % Create envPlayButton
            app.envPlayButton = uibutton(app.UIFigure, 'push');
            app.envPlayButton.ButtonPushedFcn = createCallbackFcn(app,
                @envPlayButtonPushed, true);
            app.envPlayButton.BackgroundColor = [1 1 1];
            app.envPlayButton.FontSize = 11;
            app.envPlayButton.Position = [258 201 51 20];
            app.envPlayButton.Text = 'âÚž';

            % Create filterPlayButton
            app.filterPlayButton = uibutton(app.UIFigure, 'push');
            app.filterPlayButton.ButtonPushedFcn = createCallbackFcn(app,
               @filterPlayButtonPushed, true);
            app.filterPlayButton.BackgroundColor = [1 1 1];
            app.filterPlayButton.FontSize = 11;
            app.filterPlayButton.FontWeight = 'bold';
            app.filterPlayButton.Position = [257.5 428 51 20];
            app.filterPlayButton.Text = 'âŰž';

            % Create oscPlayButton
            app.oscPlayButton = uibutton(app.UIFigure, 'push');
            app.oscPlayButton.ButtonPushedFcn = createCallbackFcn(app,
               @oscPlayButtonPushed, true);
            app.oscPlayButton.BackgroundColor = [1 1 1];
            app.oscPlayButton.FontSize = 10;
            app.oscPlayButton.FontWeight = 'bold';
            app.oscPlayButton.Position = [257.5 638 51 20];
            app.oscPlayButton.Text = 'âŰž';

            % Create msynthLabel
            app.msynthLabel = uilabel(app.UIFigure);
            app.msynthLabel.FontName = 'Droid Serif';
            app.msynthLabel.FontSize = 48;
            app.msynthLabel.FontColor = [0.7961 0.8745 0.9059];
            app.msynthLabel.Position = [904 31 243 63];
            app.msynthLabel.Text = 'm-synth âŹń';

            % Create SavedPresetPanel
            app.SavedPresetPanel = uipanel(app.UIFigure);
            app.SavedPresetPanel.Title = 'Saved Preset';
            app.SavedPresetPanel.BackgroundColor = [1 1 1];
            app.SavedPresetPanel.Position = [772 31 114 190];

            % Create SaveSwitch
            app.SaveSwitch = uiswitch(app.SavedPresetPanel, 'rocker');
            app.SaveSwitch.Items = {'Save', 'Play'};
            app.SaveSwitch.Orientation = 'horizontal';
            app.SaveSwitch.FontSize = 10;
            app.SaveSwitch.Position = [33 137 45 20];
            app.SaveSwitch.Value = 'Save';

            % Create NoteButton
            app.NoteButton = uibutton(app.SavedPresetPanel, 'push');
            app.NoteButton.ButtonPushedFcn = createCallbackFcn(app,
               @NoteButtonPushed, true);
            app.NoteButton.Position = [15 101 32 22];
            app.NoteButton.Text = '1';

            % Create NoteButton_2
            app.NoteButton_2 = uibutton(app.SavedPresetPanel, 'push');
            app.NoteButton_2.ButtonPushedFcn = createCallbackFcn(app,
               @NoteButton_2Pushed, true);
            app.NoteButton_2.Position = [64 101 32 22];
            app.NoteButton_2.Text = '2';

            % Create NoteButton_3
            app.NoteButton_3 = uibutton(app.SavedPresetPanel, 'push');
            app.NoteButton_3.ButtonPushedFcn = createCallbackFcn(app,
               @NoteButton_3Pushed, true);
            app.NoteButton_3.Position = [15 70 32 22];
            app.NoteButton_3.Text = '3';

            % Create NoteButton_4
            app.NoteButton_4 = uibutton(app.SavedPresetPanel, 'push');
            app.NoteButton_4.ButtonPushedFcn = createCallbackFcn(app,
               @NoteButton_4Pushed, true);
            app.NoteButton_4.Position = [64 70 32 22];
            app.NoteButton_4.Text = '4';

            % Create NoteButton_5
            app.NoteButton_5 = uibutton(app.SavedPresetPanel, 'push');
            app.NoteButton_5.ButtonPushedFcn = createCallbackFcn(app,
               @NoteButton_5Pushed, true);
            app.NoteButton_5.Position = [15 41 32 22];
            app.NoteButton_5.Text = '5';

            % Create NoteButton_6
            app.NoteButton_6 = uibutton(app.SavedPresetPanel, 'push');
            app.NoteButton_6.ButtonPushedFcn = createCallbackFcn(app,
               @NoteButton_6Pushed, true);
            app.NoteButton_6.Position = [64 41 32 22];
            app.NoteButton_6.Text = '6';

            % Create NoteButton_7
            app.NoteButton_7 = uibutton(app.SavedPresetPanel, 'push');
            app.NoteButton_7.ButtonPushedFcn = createCallbackFcn(app,
               @NoteButton_7Pushed, true);
            app.NoteButton_7.Position = [15 12 32 22];
            app.NoteButton_7.Text = '7';

            % Create NoteButton_8
            app.NoteButton_8 = uibutton(app.SavedPresetPanel, 'push');
            app.NoteButton_8.ButtonPushedFcn = createCallbackFcn(app,
               @NoteButton_8Pushed, true);
            app.NoteButton_8.Position = [64 12 32 22];
            app.NoteButton_8.Text = '8';
        end
    end

    methods (Access = public)

        % Construct app
        function app = msynth()

            % Create and configure components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end]]>
```

Lab 04: FIR Filters and Frequency Response

Lab Goals

- Build customized FIR filters
- Connect the following concepts:
  - filter coefficients
  - impulse response
  - frequency response
  - pole-zero plots
- Intuitively create high pass and low pass filters
- Cascade filters to create more complicated systems

Warm-up

You've been learning about FIR filters in class – now we get to apply that theory to creating real filters.

So first some warm-up – consider the generic three tap FIR filter:

$$y[n] = ax[n] + bx[n-1] + cx[n-2]$$

What's the impulse response, $h[n]$?

What's the frequency response, $H(e^{j\omega})$? Find the magnitude and phase.

What are the zeros of the frequency response (roots of the numerator)?

What are the poles of the frequency response? (roots of the denominator)?

What are the possible poles for any FIR filter?

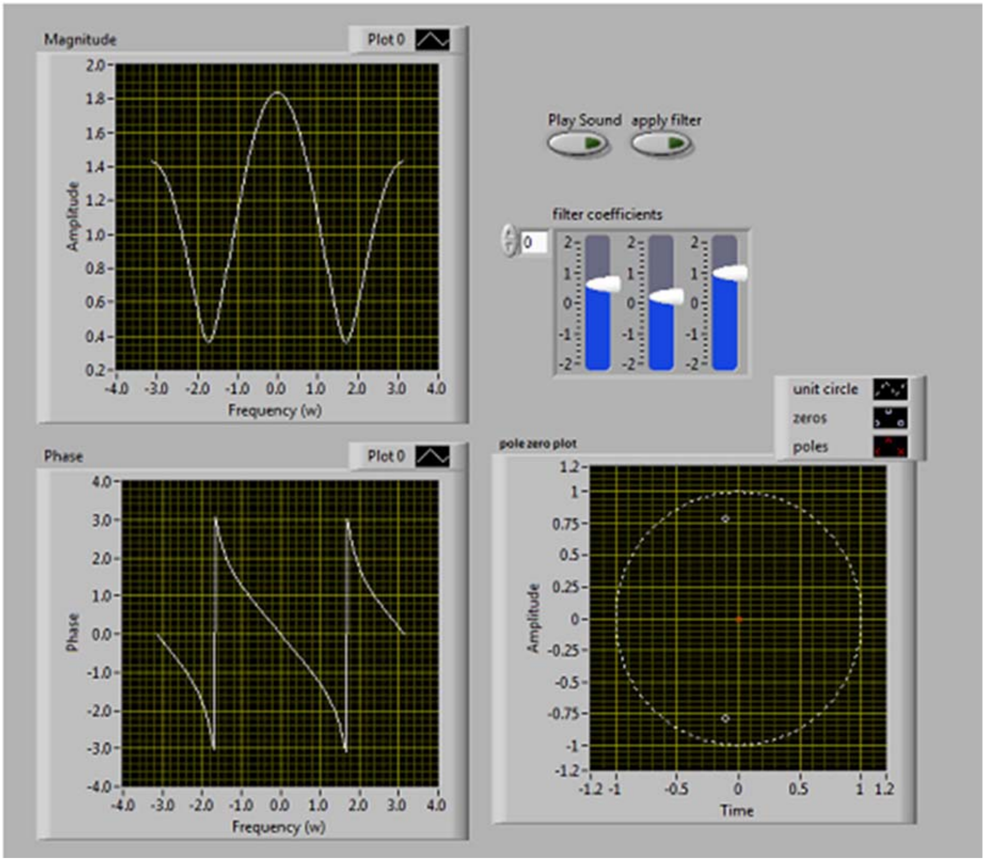
Please think about generalizing the results you've just arrived at for an arbitrary FIR filter of any order – these will be good things to keep in mind as we learn more about filter design.

Pre-lab

You'll be developing a VI that will help you visualize the effects of changing filter coefficients on an FIR filter. We'll be using three DSPFirst VI's in this lab: freqz, filter, and zplane. All three take in filter coefficients – we'll only be using the "b" coefficients (the "a" coefficients are for IIR filters). Freqz outputs a frequency response; filter applies your coefficients to an input waveform and outputs the result; and zplane outputs a pole-zero plot. Create an input array of filter coefficients on your front panel, and have your VI use these basic blocks to compute the frequency response (magnitude and phase) and pole zero plot. Try using only three filter coefficients to create high pass, low pass, notch, and band-pass filters.

Next, use the wavread, filter, and soundsc VI's to apply these tools to an audio file and observe the results of your filters. Use case structures and buttons to turn off the audio when you are creating your filters; only play the audio file when you are ready to try out your filter. You can use the sample.wav file provided in the course folder (a sample of Bohemian Rhapsody, by Queen), but make sure you create your own sample file to try out your filter. You can use Audacity (available at http://audacity.sourceforge.net/) if you want to convert an mp3 into a wav file.

Your front panel should look something like this:



Pre-lab Deliverable

Please write up your responses to the warm-up and hand them in at the beginning of lab to be checked off. The VI you created will also be checked off. If you have any questions, contact your ninja via email.

Lab Exercises

1) Design a low pass filter using the VI from the warm-up. A lot of filter design is done by intelligently changing parameters in order to achieve a specification that defines your pass and stop bands. Here is your specification:
    a. Pass band: 0 to 1.0 rad/s. Magnitude must be greater than 1.0
    b. Stop band: greater than 2.5 rad/s. Magnitude must be less than 0.5

    Hint: Multiplying all of your coefficients by a constant gain will make the specification easier to achieve once you have the right shape.

    Save these filter coefficients.

2) You'll notice this filter isn't very good because it has a very low order. A good way to create more complicated filters is by cascading two low order filters together. Cascading filters is quite simple – you connect the output of one filter into the input of the next one. You can also just multiply the transfer functions in order to compute the system function H(z) of the single cascaded filter. You can do this in several ways. If you want the frequency response, the Multiply Polynomial Labview block will let you multiply your coefficients together to create the polynomial coefficients for the cascaded filter. If you just want to filter a signal, you can just create two distinct filters and connect them in series. Cascade the filter you designed in step 1 with itself to make a better low pass filter. Take a screenshot showing the difference between the original filter and the filter cascaded with itself twice.

    Food for thought: What are the poles and zeros of this new cascaded filter?

3) Another method of designing filters is to specify where the zeros (and eventually poles) are in the imaginary plane. Test out your warm-up VI to explore how moving the zeros affects the frequency response. Pay particular attention to where the zeros need to be to create low or high pass filters. Then create a VI that generates filter coefficients from an array of desired zeros. Create Polynomial From Roots is a great block for this. Use this VI to make a high pass filter with the following specification
    a. Pass Band: Greater than 0.5 rad/s. Magnitude must be greater than 1.0
    b. Stop Band: 0 to 0.25 rad/s. Magnitude must be less than 0.5.

    Hint: Feel free to use your knowledge of cascaded filters to meet this specification if you're having trouble.

    Take a screenshot of this filter's frequency response and pole-zero plot.

4) Now you are hopefully gaining an intuition for FIR filter design. Finish off this lab by developing either a band pass or band stop (also known as notch) filter, and running an audio sample through it. Decide on your own frequency specifications (for the pass and stop bands), but your pass band must have magnitude greater than 1.0, and your stop band must have a magnitude of less than 0.1. Use the VI's you created to help design your filters; you will need to cascade them in order to meet your specification. Write up your specification in a comment, and take a

screenshot of the frequency response of your filter. Also, demonstrate the audio performance of your filter to your ninja.

Food for thought: What's the relationship between the frequency axis of the frequency response and the frequencies we hear? Is there a difference between the two? If so, why?

Lab 6: Filter Implementation and Coefficient Quantization

Lab Report: Please follow the provided lab report structure for Lab 6. Write up your answers with requested screenshots and have your Course Assistant sign off on your work.

Background Info

Fixed-Point Filter Design Process

Fixed-point signal processing platforms, such as fixed-point digital signal processors (DSPs) and field-programmable gate arrays (FPGAs), are typically more power-efficient and less expensive than floating-point alternatives. However, fixed-point systems are generally more difficult to design. For example, you must consider the effects of coarser quantizations in fixed-point systems.

To design a fixed-point filter, you first must design a floating-point filter, also known as a reference filter, that meets the target specifications. In some cases, you need to design a reference filter that exceeds the target specifications to get a fixed point filter with the same specifications. The excess margin ensures a smooth conversion from a floating-point representation to a fixed-point representation. You then must modify the floating-point filter to accommodate the finite-precision constraints of the target platform while still trying to meet the target specifications. Figure 1 illustrates the fixed-point filter design process. The grey boxes illustrate the floating-point filter design process, the dotted lines represent optional steps, and the arrows on the left indicate to which steps you can return if the filter design fails to meet the requirements in the current step.

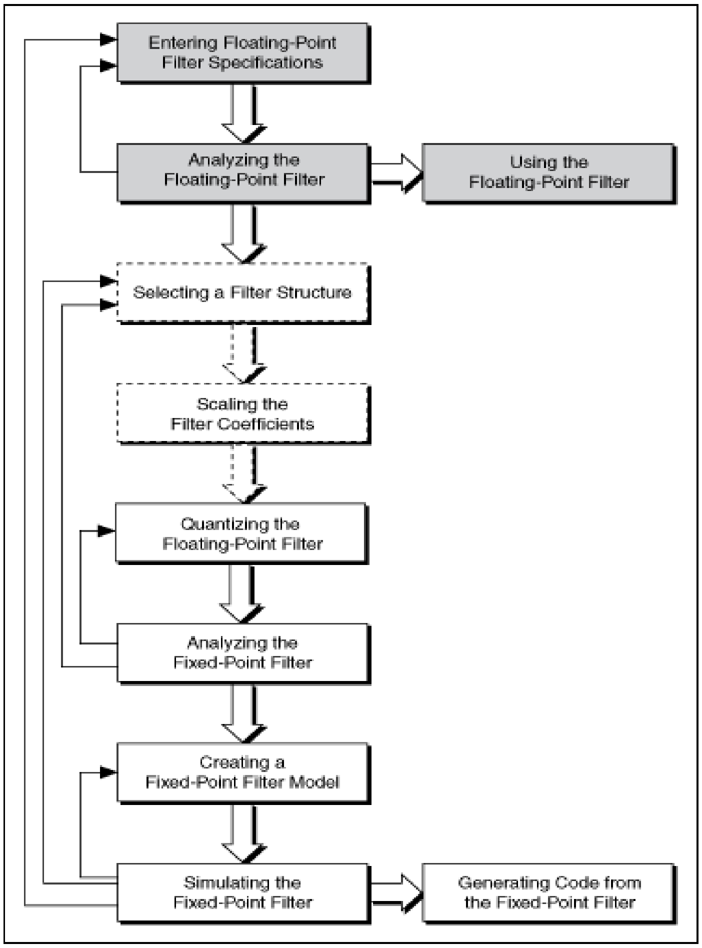

Entering Floating-Point Filter Specifications
Analyzing the Floating-Point Filter
Using the Floating-Point Filter
Selecting a Filter Structure
Scaling the Filter Coefficients
Quantizing the Floating-Point Filter
Analyzing the Fixed-Point Filter
Creating a Fixed-Point Filter Model
Simulating the Fixed-Point Filter
Generating Code from the Fixed-Point Filter

Figure 1: Fixed-point filter design process

Designing a fixed-point filter from a reference floating-point filter involves the following steps:

1. Selecting a filter structure. In floating-point filter design, after you select a design method, the LabVIEW Digital Filter Design Toolkit uses a default filter structure according to the specified design method. However, in fixed-point implementations, different filter structures can have different memory and multiplier requirements and might cause different finite word length effects. To obtain the best filtering results, you can convert the default filter structure to an appropriate structure. This step is optional.
2. Scaling the filter coefficients. Every filter structure contains many accumulators, each of which might use a different data range. You can scale the filter coefficients by using the DFD Scale Filter VI to ensure that all of the accumulators use the same data range. Scaling the filter coefficients can help you obtain a better filtering result, especially for IIR Cascaded Second-Order Sections Form structures. This step is optional.
3. Quantizing the floating-point filter. Quantization is the process of approximating a fixed-point value for each reference floating-point value. You then can use the fixed-point values in fixed-point mathematical computation or a hardware implementation. By

quantizing the coefficients of the reference floating-point filter, you convert a floating-point filter to a fixed-point filter.

4. Analyzing the fixed-point filter. To determine how the characteristics of the realized fixed-point filter deviate from the characteristics of the reference floating-point filter, you must analyze the fixed-point filter.
5. Creating a fixed-point filter model. To create the fixed-point filter model, you must configure the quantizers for the input and output signals and specify the settings for internal computation.
6. Simulating the fixed-point filter. Before applying the fixed-point filter model in real-world applications, you must simulate the behavior of the filter to verify if the fixed-point filter model works as you require in a simulation. If the fixed-point filter does not provide the required performance in the simulation, you can change the implementation structure, modify quantization settings, or redefine the filter specifications for the reference floating-point filter.
7. Generating code from the fixed-point filter. You can export filter coefficients and automatically generate integer LabVIEW code, LabVIEW FPGA code, and C code from the fixed-point filter for designated hardware targets.

Finite Word Length Effects

Converting a floating-point filter to fixed-point can alter the characteristics and performance of the filter significantly. You must analyze the filter and simulate the filtering process with expected input signals. Fixed-point arithmetic can have the following effects on filter performance:

- Degrade signal-to-noise ratio (SNR) due to the reduced precision of internal registers, adders, subtracters, and multipliers
- Distort frequency response from a limited word length representation of filter coefficients
- Overflow or clip signal information due to insufficient headroom in the signal paths
- Cause zero-input limit cycles (self-sustaining oscillations) of infinite impulse response (IIR) filters due to nonlinear quantizers in the feedback loop of IIR filters or to the overflow of the summation operations

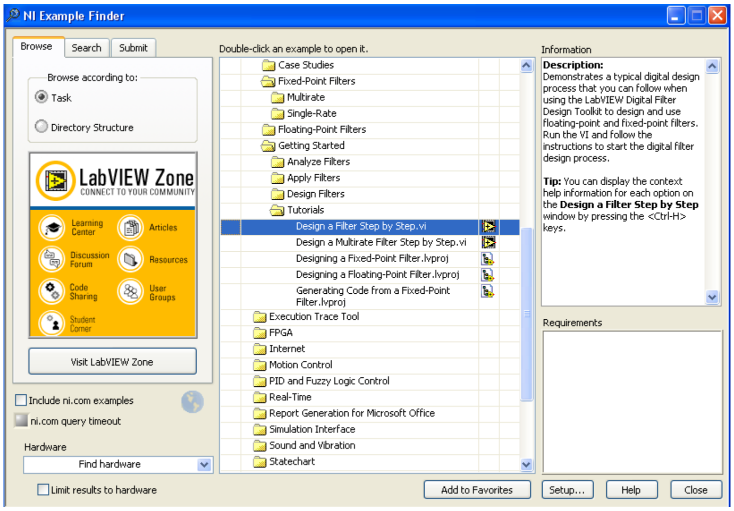
Specifying the Word Length and Integer Word Length

The word length indicates the number of bits you want to use in representing a fixed-point number. The integer word length specifies the number of bits, including the sign bit, you use in representing the integer part of a fixed-point number. The difference in bits between the word length and the integer word length determines the digits of precision.

The finite word length of a quantizer can affect the frequency response of the resulting fixed-point filter. The larger word length value you specify, the less the fixed-point representation distorts the frequency response. However, a larger word length value also requires more hardware resources, so you must specify a word length that provides an acceptable tradeoff between distortion and hardware resource consumption.

Part I: Quantization of IIR filter coefficients

1. Go to the Example Finder of LabVIEW (Help >> Find Examples). See Figure 2. Browse to Toolkits and Modules >> Digital Filter Design >> Getting Started >> Tutorials >> Design a Filter Step by Step.vi. Open the file and run it.

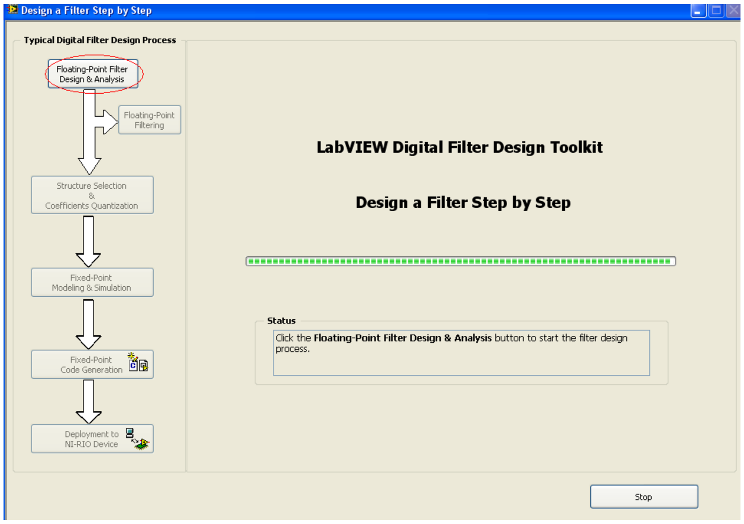


Description: Demonstrates a typical digital design process that you can follow when using the LabVIEW Digital Filter Design Toolkit to design and use floating-point and fixed-point filters. Run the VI and follow the instructions to start the digital filter design process.

Tip: You can display the context help information for each option on the **Design a Filter Step by Step** window by pressing the <Ctrl-H> keys.

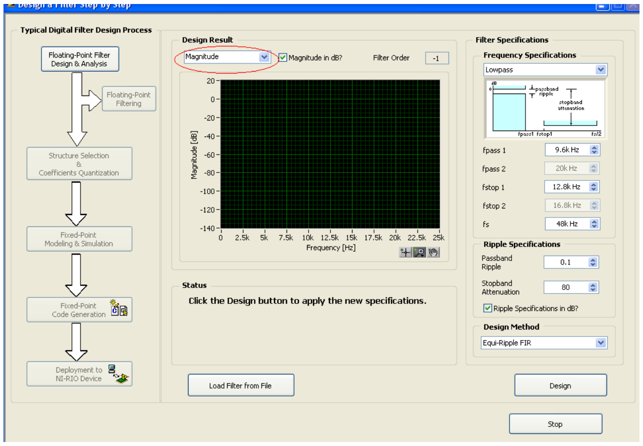
Figure 2: Example Finder

2. This demonstration will walk you through the series of steps involved in filter design (flow chart on left). Click on "Floating-Point Filter Design and Analysis". You should see the code shown in Figure 4.


Design a Filter Step by Step
Typical Digital Filter Design Process
Floating-Point Filter Design & Analysis
Floating-Point Filtering
Structure Selection & Coefficients Quantization
Fixed-Point Modeling & Simulation
Fixed-Point Code Generation
Deployment to NI-RIO Device
LabVIEW Digital Filter Design Toolkit
Design a Filter Step by Step
Status
Stop

Click the **Floating-Point Filter Design & Analysis** button to start the filter design process.

Figure 3: Design a Filter Step by Step VI



Magnitude
☑ Magnitude in dB?
Filter Order -1

Click the Design button to apply the new specifications.

Lowpass
fpass 1 9.6k Hz
fpass 2 20k Hz
fstop 1 12.8k Hz
fstop 2 16.8k Hz
fs 48k Hz
Passband Ripple 0.1
Stopband Attenuation 80
☑ Ripple Specifications in dB?
Equi-Ripple FIR

Figure 4: Floating-Point Filter Design

3. Create a filter with the following parameters:

   Filter Type: Lowpass Filter
   fpass1: 500 Hz
   fstop1: 600 Hz
   fs = 8kHz
   Passband Ripple: 0.1dB
   Stopband Attenuation: 80dB
   Design Method: Chebyshev

When you click "Design", your filter will be updated and you'll see an analysis of your filter. If you click on the drop box under "Design Result", you'll be able to look at a different analysis of the filter (e.g., a pole-zero plot). Take a few minutes to go through them and see if these results agree with what you've learned so far. For example, is the filter stable? How do you know that? What is the order of the filter? Is it Chebyshev I or Chebyshev II? How can you more clearly see the ripple? Answer these questions and include a screen shot of the filter in your lab report.

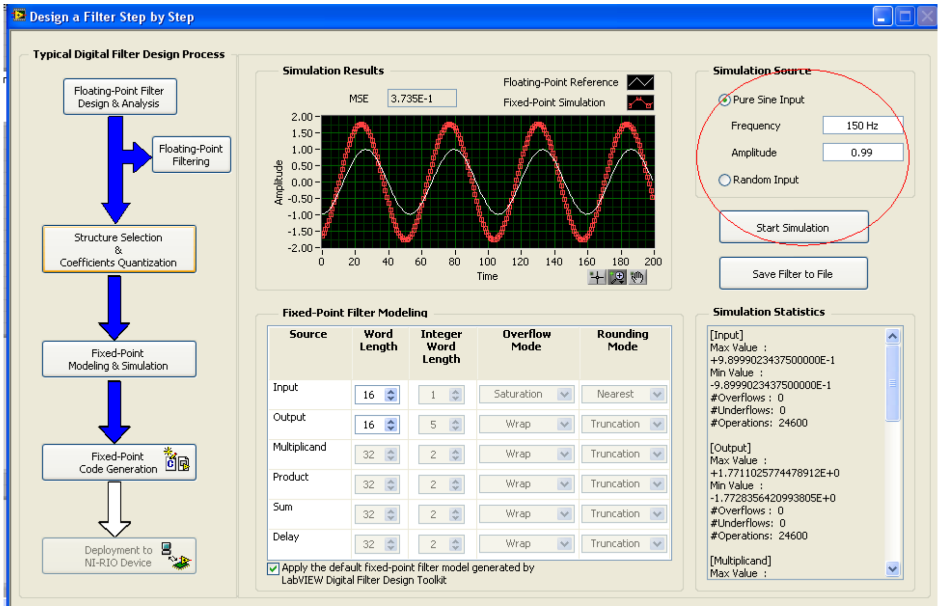
4. We will bypass the "Floating Point Filtering" step on the flow diagram (left hand side of the code), which simulates the floating point filter (we will see the same results later). Click on the "Structure Selection and Coefficients Quantization" of the flow diagram on the left hand side of the code.
5. The a/k and b/v are filter coefficients applied to the inputs and past outputs of the filter. Change the a/k and b/v coefficient quantization length to 8. Leave everything else as default. In your lab report, include the quantized feed forward and feedback filter coefficients. How many significant digits are allocated for each filter coefficient?

$$H(z) = \frac{b(1) + b(2)z^{-1} + \ldots + b(n+1)z^{-n}}{1 + a(2)z^{-1} + \ldots + a(n+1)z^{-n}}$$

Figure 5: Example of IIR system function

6. Look at the magnitude and phase graphs again of the both the 8-bit and 16-bit filters. For the superimposed magnitude graphs, provide a screen shot and explain how and why the 8-bit fixed point filter differs from the floating point filter. What happens to the passband ripple in the 8-bit realization? Now contrast the phase graphs of the two filters. Unclick 'unwrap the phase'. At what frequencies does the phase break up? Does it matter? Explain your answer.

7. Go to the "Fixed-Point Modeling and Simulation" step. The "Simulation Source" section allows you to apply your filter on a simulated input signal.


Design a Filter Step by Step
Typical Digital Filter Design Process
Floating-Point Filter Design & Analysis
Floating-Point Filtering
Structure Selection & Coefficients Quantization
Fixed-Point Modeling & Simulation
Fixed-Point Code Generation
Deployment to NI-RIO Device
Simulation Results
Floating-Point Reference
Fixed-Point Simulation
Amplitude
2.00
1.50
1.00
0.50
0.00
-0.50
-1.00
-1.50
-2.00
0
20
40
60
80
100
120
140
160
180
200
Time
Simulation Source
Start Simulation
Save Filter to File
Fixed-Point Filter Modeling
Simulation Statistics

MSE 3.735E-1

● Pure Sine Input
Frequency 150 Hz
Amplitude 0.99
○ Random Input

Source	Word Length	Integer Word Length	Overflow Mode	Rounding Mode
Input	16	1	Saturation	Nearest
Output	16	5	Wrap	Truncation
Multiplicand	32	2	Wrap	Truncation
Product	32	2	Wrap	Truncation
Sum	32	2	Wrap	Truncation
Delay	32	2	Wrap	Truncation

☑ Apply the default fixed-point filter model generated by LabVIEW Digital Filter Design Toolkit

```
[Input]
Max Value :
+9.8999023437500000E-1
Min Value :
-9.8999023437500000E-1
#Overflows : 0
#Underflows: 0
#Operations: 24600

[Output]
Max Value :
+1.7711025774478912E+0
Min Value :
-1.7728356420993805E+0
#Overflows : 0
#Underflows: 0
#Operations: 24600

[Multiplicand]
Max Value :
```

Figure 6: Fixed point modeling and simulation

Click "Start Simulation" and run through the following frequencies:

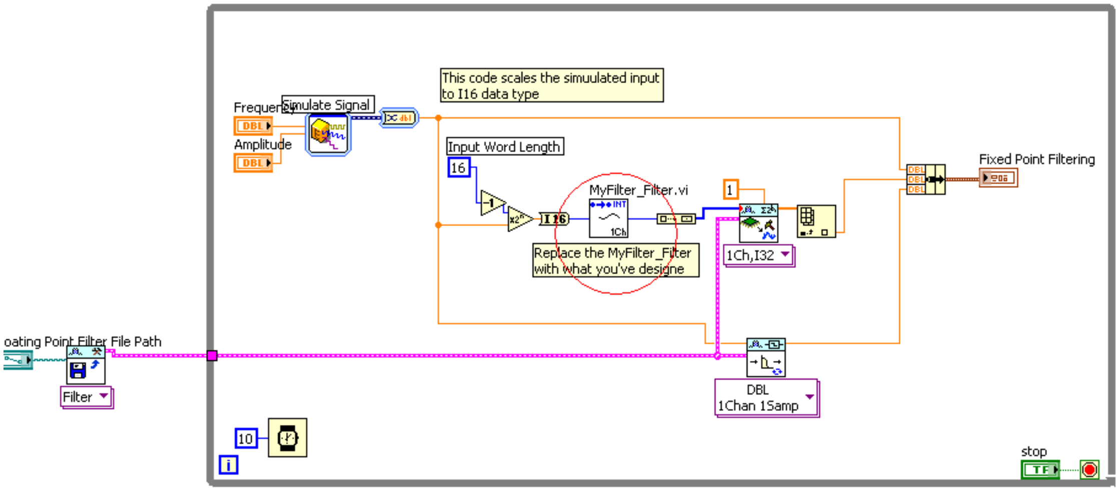
Frequency (Hz)	Signal Attenuated or Amplified?
100	

150	
500	
600	

In your lab report, fill in the above table and include screen shots of the output due to (1) a sinusoidal input of 150 Hz and (2) a sinusoidal input of 500 Hz. What do the screen shots show? Explain.

8. Sometimes a filter designer is limited to an 8-bit realization. Can you find a way to still realize the desired Low Pass Filter specifications while not attenuating frequencies at 500Hz? Take a screen shot of your solution and explain the reasoning behind your solution. What sacrifice did you have to make? Include in your lab report.

9. Set the input and output word length back to 16. Click "Save Filter to File" and save it as "Chebyshev" in a known directory.
10. Go to "Fixed-Point Code Generation" step. Select "Integer LabVIEW Code". Select a destination folder. Call the filter Chebyshev. This step will create the filter in a LabVIEW project containing the filter VI and the subVIs (sub-functions). This project will pop open. You can close it.
11. Open up "Deploy Filter" VI. We will use the fixed-point filter that you've generated here. Go to the block diagram.


This code scales the simuulated input to I16 data type

Replace the MyFilter_Filter with what you've designe

Figure 5: Deploy filter VI

12. Replace the current placeholder filter with what you've designed. The easiest way to do this is to right click on the current filter VI (the placeholder), Replace >> Select a VI. Navigate to where you've saved the fixed code filter. LabVIEW should've placed a suffix "_filter" after the name of your filter.
13. On the front panel, under "Floating Point Filter File Path", navigate to where you've saved the floating point filter. This should have the extension ".fds".
14. Run the VI. Vary the sine wave frequency to make sure the filter corresponds with your expected results. You should see a sine wave with the fundamental frequency you specified, superposed with white noise.
15. Run the code. Include a screen shot and explain the result.

Part II. Coefficient Quantization in FIR Filters

16. Click on "Floating-Point Filter Design and Analysis". Repeat Step 3, but set the Filter Design Method as "Equi-Ripple FIR". What is the filter order compared to Chebyshev filter? You can also compare it to other IIR filters.
17. Go on to the Coefficients Quantization part. Can you change the a/k word length? Why or why not?
18. Repeat Steps 6 to 15. You can skip redesign if the filter fits the requirements. Since no specifics for the passband are given, the primary goal is not to attenuate signals in the 0 – 500Hz range. Also, you want to save this filter as an FIR filter in step 9. When you're testing out your filter in the steps 11 and on, use the same "Deploy Filter" VI and replace the filter with the FIR one.

    Do you think quantization affects FIR or IIR filters more? Why?

Lab 7—Filter Structures

Background Info

With infinite precision data, coefficients, and arithmetic, all filter structures with the same transfer functions will have the same results. Once a filter is quantized, different structures may produce very different errors. Different structures are used because the memory usage and implementation time can vary.

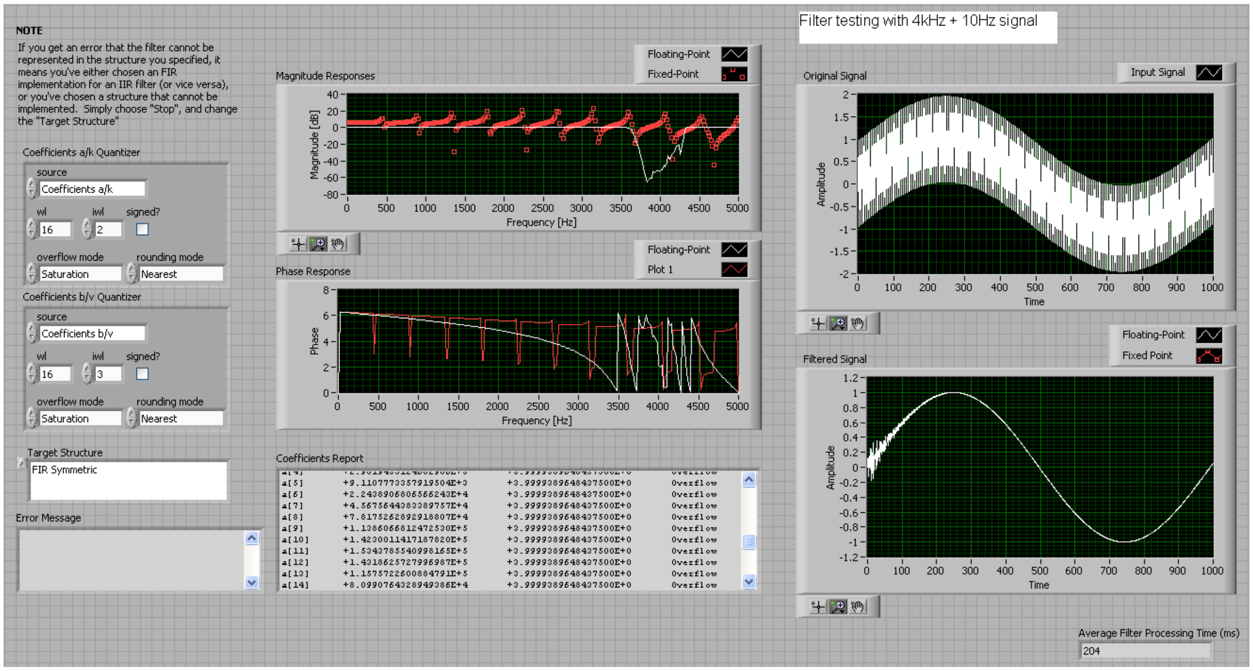
A filter structure specifies how you arithmetically use a set of filter coefficients to process an input signal. For a specified digital filter, dozens of mathematically equivalent implementation structures are available. For a floating-point digital filter, the effects of different implementation structures on the filter behavior are negligible in most cases. For a fixed-point digital filter, different implementation structures can result in different signal outputs.

When you select a filter structure, you must balance a number of factors, including the filter type, implementation resources, and computational complexity. For IIR filters, you also need to consider the sensitivity to coefficient quantization of each structure.

FIR Filter

You're designing an audio filter to cut out an annoying noise introduced during recording (recall Lab 5). Design a filter that will minimize this noise as well as possible. You're free to pick the sampling frequency, passband ripple, etc. as needed. This first filter should be a FIR filter.

1. Open the program "Lab 3 Structure".



NOTE
If you get an error that the filter cannot be represented in the structure you specified, it means you've either chosen an FIR implementation for an IIR filter (or vice versa), or you've chosen a structure that cannot be implemented. Simply choose "Stop", and change the "Target Structure"

Coefficients a/k Quantizer
- source: Coefficients a/k
- wl: 16, iwl: 2, signed? □
- overflow mode: Saturation, rounding mode: Nearest

Coefficients b/v Quantizer
- source: Coefficients b/v
- wl: 16, iwl: 3, signed? □
- overflow mode: Saturation, rounding mode: Nearest

Target Structure: FIR Symmetric

Error Message:

Coefficients Report

```
a[5]     +9.1107773357919504E+3     +3.9999389648437500E+0     Overflow
a[6]     +2.2438906806566243E+4     +3.9999389648437500E+0     Overflow
a[7]     +4.5675644383389757E+4     +3.9999389648437500E+0     Overflow
a[8]     +7.8175262892918807E+4     +3.9999389648437500E+0     Overflow
a[9]     +1.1386066812472530E+5     +3.9999389648437500E+0     Overflow
a[10]    +1.4230011417187820E+5     +3.9999389648437500E+0     Overflow
a[11]    +1.5343785540998165E+5     +3.9999389648437500E+0     Overflow
a[12]    +1.4318625727996987E+5     +3.9999389648437500E+0     Overflow
a[13]    +1.1575722600884791E+5     +3.9999389648437500E+0     Overflow
a[14]    +8.0990764328949386E+4     +3.9999389648437500E+0     Overflow
```

Filter testing with 4kHz + 10Hz signal

Average Filter Processing Time (ms): 204

Figure 1: Front Panel of Lab 3 Structures VI

2. Go to the block diagram and double click on the Filter Design VI to design a filter with the following specifications:
    - Get rid of 4kHz noise
    - Passband ripple < 0.1dB
    - Stopband Attenuation > 60dB
    - FIR Filter

    Include a pole-zero plot in your write up and explain its significance to the design problem at hand.
3. Implement the filter with the structures in the table. Observe if the fixed point filter meets the requirements and what the average processing time is.

Filter Structure	**Meet Requirements?**	**Average Processing Time (ms)**
FIR Direct Form		
FIR Direct Form Transposed		
FIR Symmetric		

Table 1: FIR Filter Structure Performance

Can you explain why the FIR Symmetric takes significantly longer than the FIR Direct Form? What are the trade-offs you need to consider when choosing one of these filter structures?

There are two rounding modes available for coefficient quantization: nearest and truncation. Compare these two modes and state which is better. Change the rounding mode to truncation from the current value of “nearest”.

4. In the Coefficients b/v Quantizer box, uncheck the “Signed” box. What happens to the coefficients in the Coefficients Report?
   Why do the negative values become zero?
   .
   If wrap around mode is selected, what happens to the filter coefficients?

   Check the “Signed” box. Change the Word Length (wl) to 8. Do you notice any overflows or underflows in the Coefficient Report? What changes do you see in the Coefficient Report?

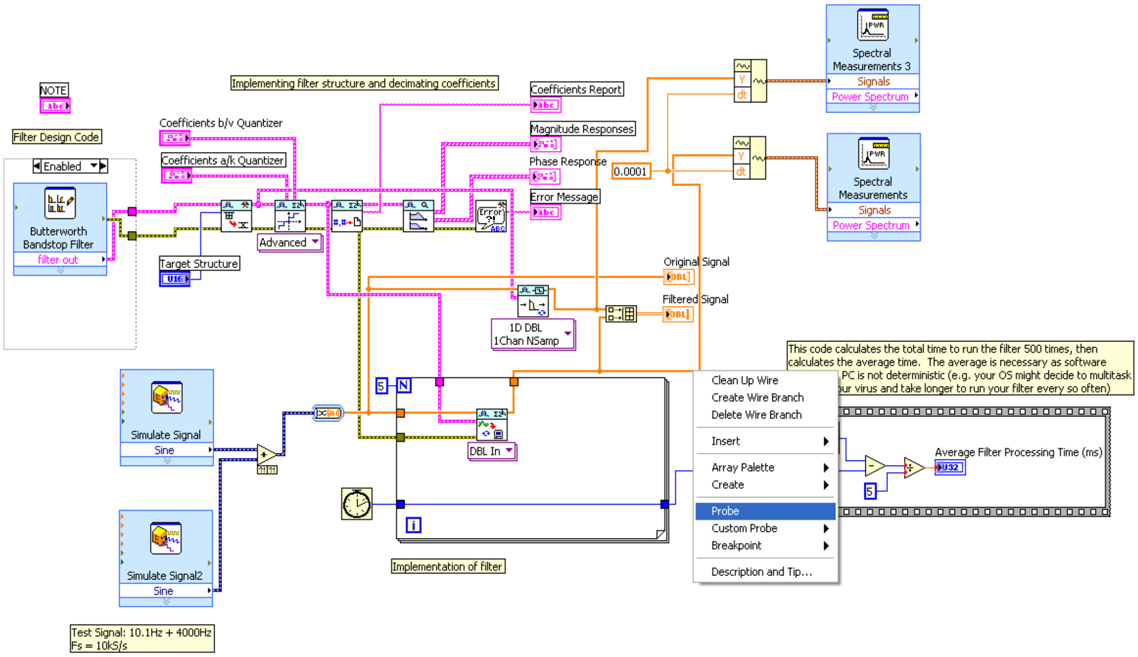
   Why do you think changing the word length has less effect on the filter performance than removing the sign bit of the coefficients?

IIR Filter Implementation

5. Go back to the Filter Design VI and change it to an IIR filter (the other filter requirements remain the same as Step 2). Take a screen shot of the design panel (of the Express VI) of the filter.

6. Implement a Butterworth filter with the structures in the Table 2. Observe if the fixed point filter meets the requirements and what the average processing time is. In cases where the requirements are not met, try looking at the Coefficients Report and see if you can do anything about the quantization to obtain a filter that meets the requirements.

Explain your solution.



This code calculates the total time to run the filter 500 times, then calculates the average time. The average is necessary as software PC is not deterministic (e.g. your OS might decide to multitask our virus and take longer to run your filter every so often)

Test Signal: 10.1Hz + 4000Hz
Fs = 10kS/s

Figure 2: Block Diagram of Lab 3 Structure VI

Filter Structure	**Meet Requirements?**	**Average Processing Time (ms)**
IIR Direct Form I		
IIR Direct Form I Transposed		
IIR Direct Form II		
IIR Direct Form II Transposed		
IIR Cascaded 2nd Order Sections Form I		
IIR Cascaded 2nd Order Form I Transposed		

Table 2: IIR Filter Structure Performance

Which structure would you choose to implement your filter? Explain.

Repeat Table 2 for a Chebyshev filter. Take a screen shot of the design panel (of the Express VI) of the filter.

Which IIR filter type and structure gave you the best performance regardless of processing time?

1. Depending on the filter design, it is possible you will not see a Fixed-Point waveform in the Filtered Waveform graph. This happens when the values of the FXP waveform are NaN (not a number). You can see this by probing the wire of the FXP waveform (right click on wire and choose probe – see Figure 2). This happens when we divide zero by zero (among other reasons). Why do you think we're getting a divide-by-zero situation here?

If you recall, IIR filters are more sensitive to quantization than FIR filters. We are seeing this effect again. Why is that?

In this exercise, you should be able to observe that fixed point filters are sensitive to filter structure. You should also note that DSP filter design is an iterative design process requiring compromise with respect to various parameters. This lab did not require the quantization of the input and output values. This would make the filter design process yet more challenging.