DLTI Demo

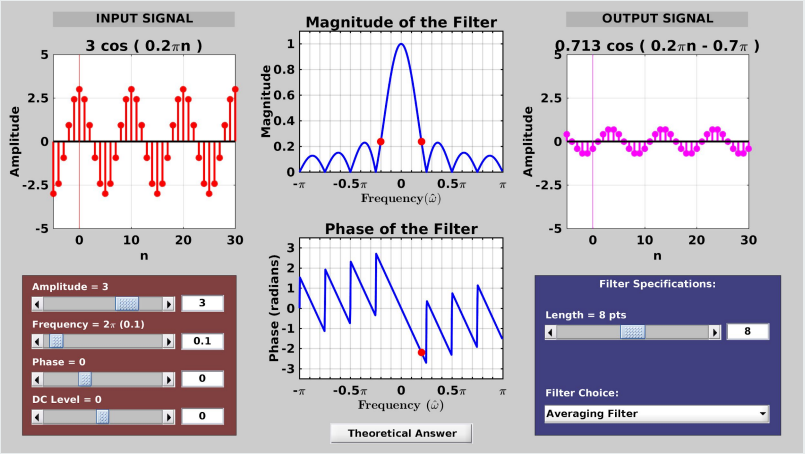
Jason Chan, John Matanin, Greg Williams, Lucas Phillips

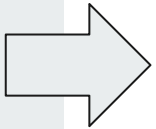
Team Members

- Jason Chan
- John Matanin
- Greg Williams
- Lucas Phillips

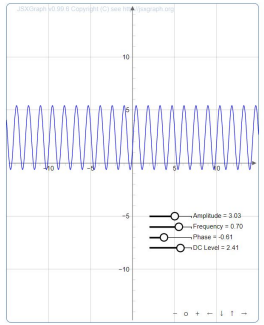
Project Motivation

- Students of Digital Signal Processing must open Matlab in order to interact with the GUIs and complete their labs.
- Intelligent Tutoring Systems intends to facilitate this process by centralizing the resources necessary to complete the labs
- We intend to do this by adding GUIs in our web application that mimic the GUIs in Matlab.
- The specific GUI we worked on this semester was the discrete-time linear time-invariant (DLTI) filter GUI.

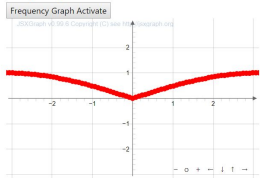




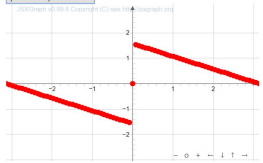
2.4 + 3.0cos(1.4πn + -0.6076π)



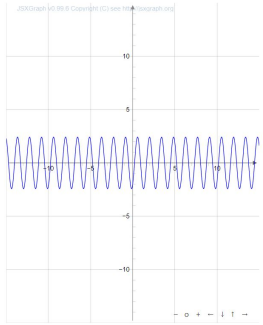
Frequency



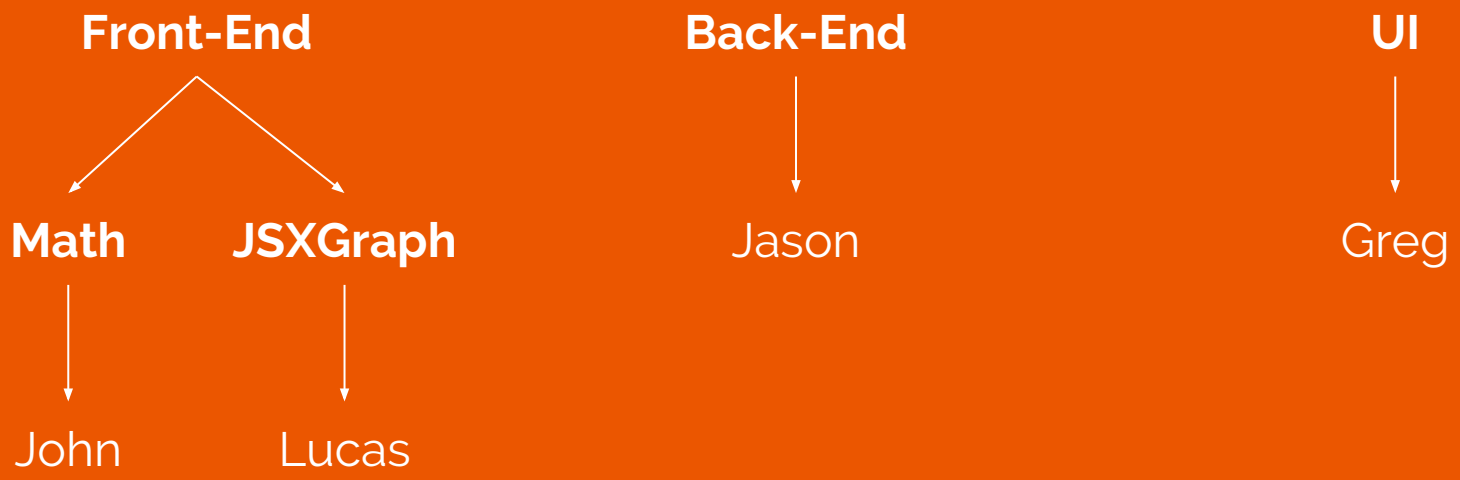
Phase



0.0 + 2.4cos(1.4πn + -0.8101π)

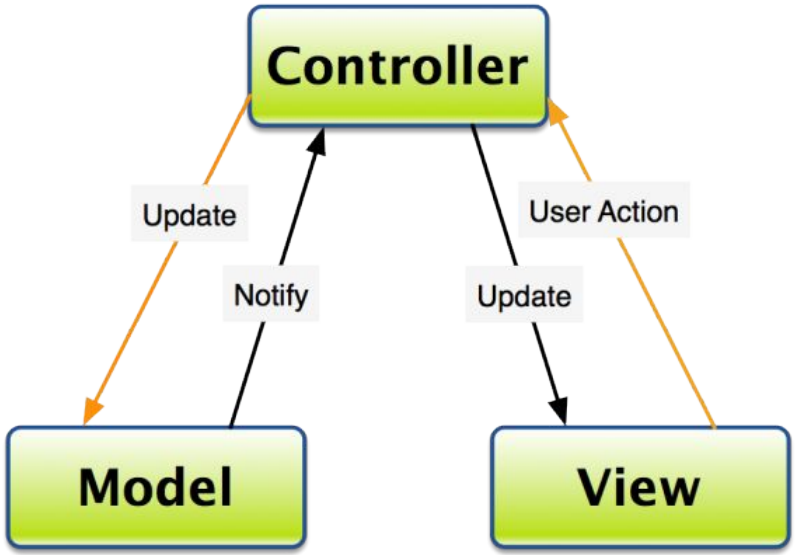


Project Structure



Model-View-Controller Structure

- We struggled with good software architecture principles in the past
- We followed the Model-View-Controller (MVC) architectural structure to create a cleaner, more modular product
- View: HTML, CSS
- Controller: Javascript / PHP
- Model: Mysql



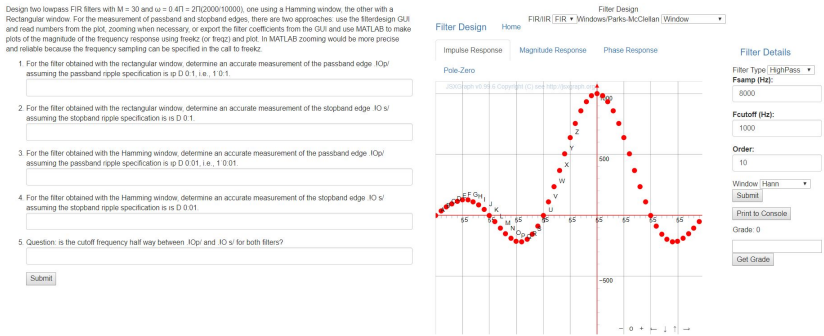
Back-End

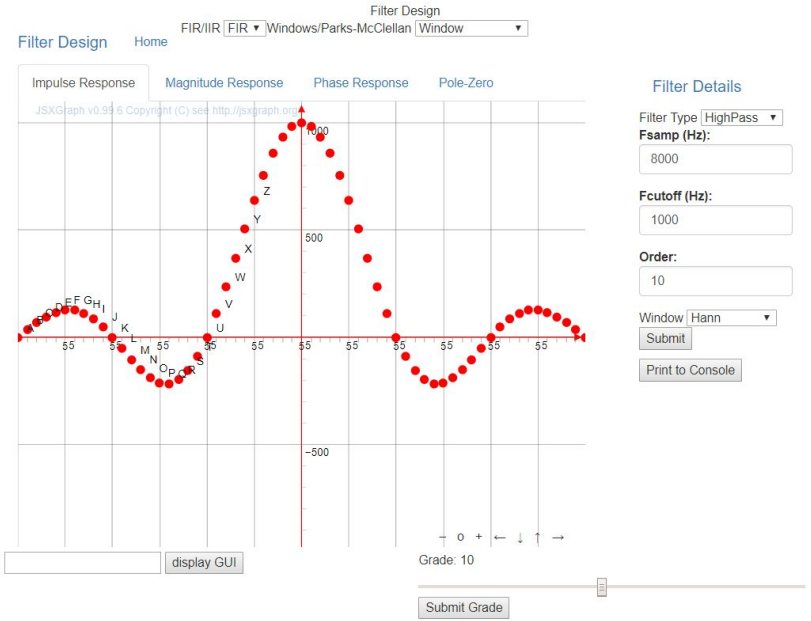
Goal: Be able to save a DLTI state in a database.

- Stretch: Be able to retrieve the state to populate an instructor page
- Use FIR as a model for implementation

Motivation: Instructors need to be able to see student responses as well as provide feedback

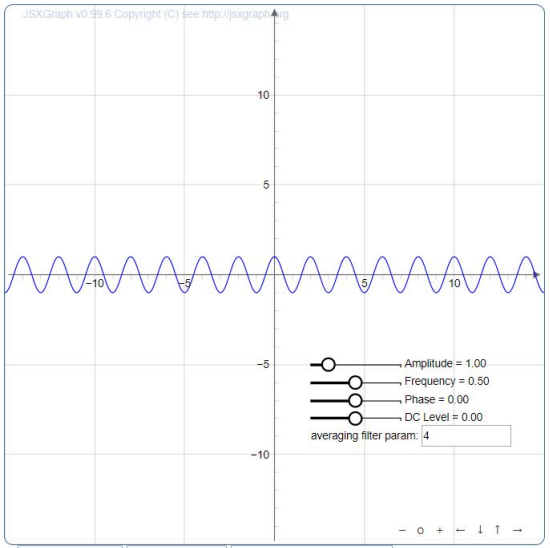
Filter Design Model



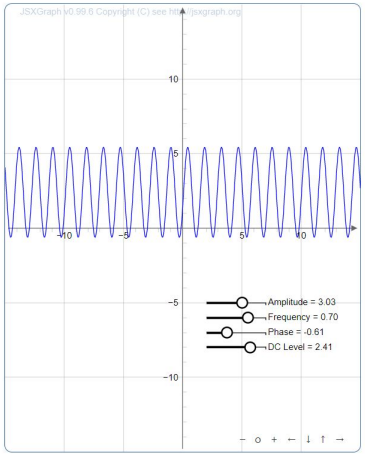


DLTI

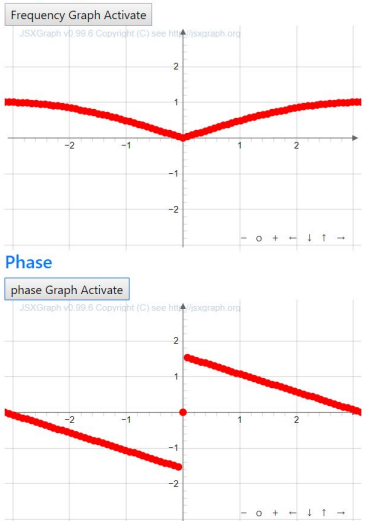
0.00 + 1.00cos(0.00πn + 0.00π)



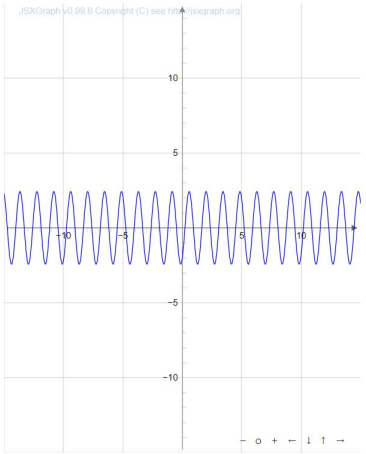
2.4 + 3.0cos(1.4πn + -0.6076π)



Frequency

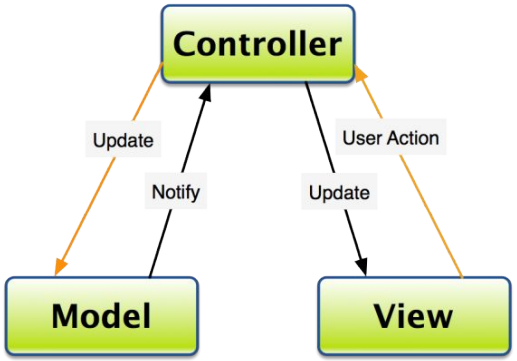


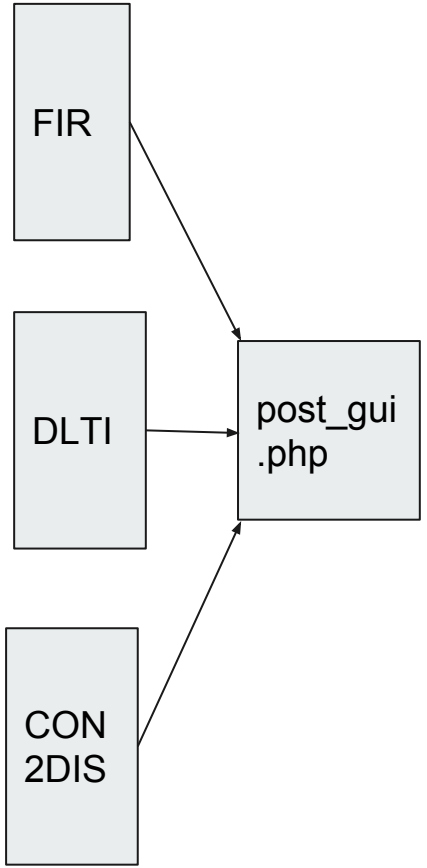
0.0 + 2.4cos(1.4πn + -0.8101π)



Modular Design

- Modular Design is where we would write our code such that it would not be specific to any single GUI, so that we could use the controller of our MVC for any view.
- When the controller updates the model, instead of simply passing in the value of the parameters, it passes in the name of the parameter and the value (ex. Amplitude: 5).





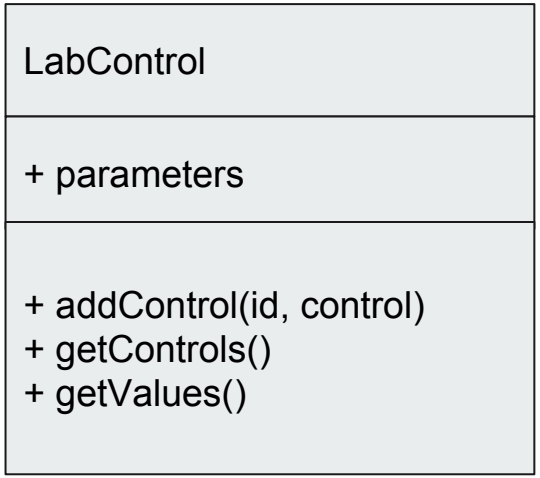
Modular Design Cont.

ID	Amplitude	Frequency	Phase	DC Level

ID	GUI name	Param 1	Value 1	Param 2	Value 2	...

Lab Control

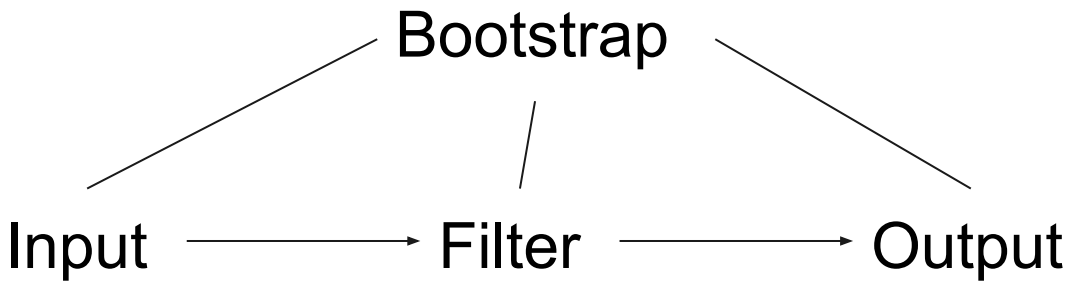
- Wrapper class for a map of GUI parameters that stores the id and html / JSX graph element that affects the GUI
- Example of Lab Control Instance
  - {“ampSlider”: <JSX Graph Slider>,
  - “averagingFilterLength”: <html input>, ...}
- This class allows us to easily access all of our parameters and manipulate them for our back end, so adding another parameter is simple.



Front-End

Goal: Use web-based development tools to replicate the functionality of the MATLAB DLTI GUI

- JSXGraph provides a graphing library able to display these kinds of graphs.
- Bootstrap can help structure the components of the GUI



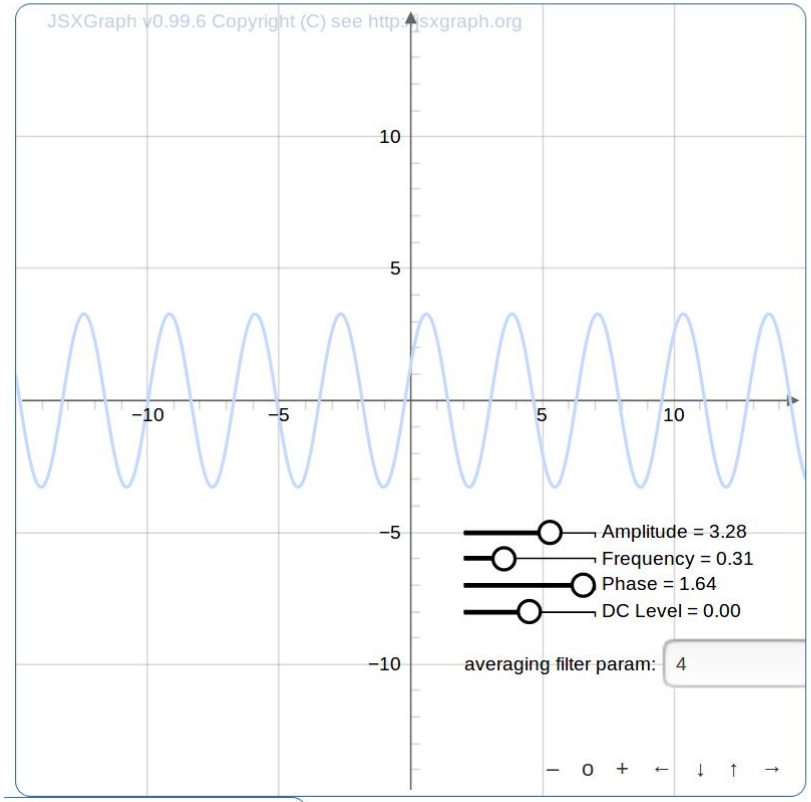
Connection Between the Boards

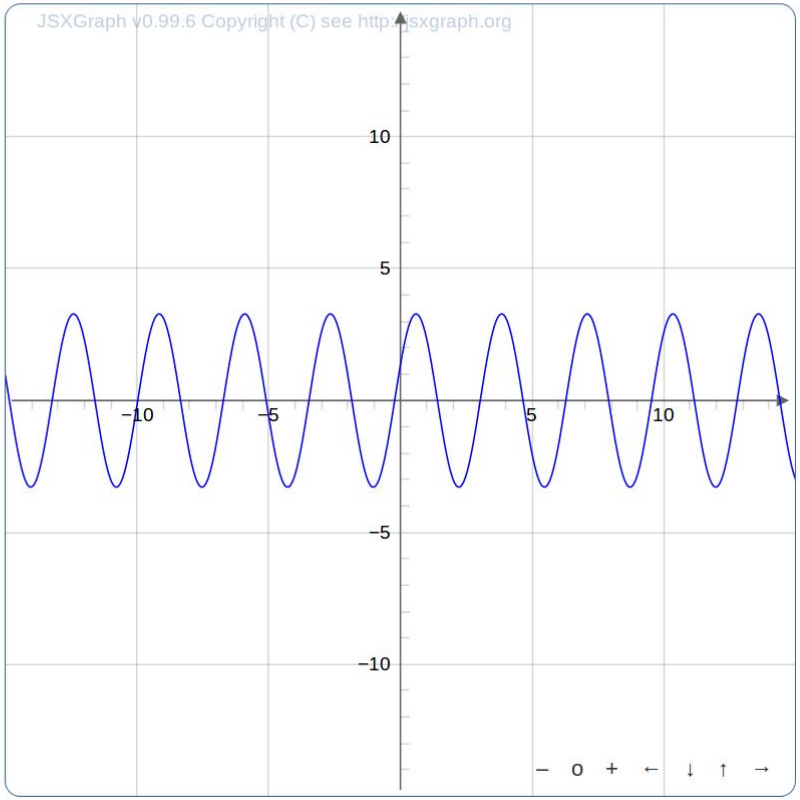
How to connect the inputs from the user to the filter and then the output graph?

Simplest Approach →

```
var glo_amp, glo_freq, glo_phase, glo_dc;
```

$0.00 + 3.28\cos(0.62\pi n + 0.52\pi)$





Input/Output graphs matching

Input Function

$$DC\ Level + Amplitude * \cos(2\pi * Frequency + Phase * \pi)$$

- The input function may be created using 4 slider parameters
- These being amplitude, frequency, phase, and DC level
- The format of the input function is shown to the left

Magnitude and Phase

$$H(e^{j\hat{\omega}}) = \sum_{k=0}^{M} b_k e^{-j\hat{\omega}k}$$

- The frequency response of a filter is specific to its coefficients
- To the left, the formula for frequency response is shown
- The frequency response equation can be broken down into two parts, magnitude and phase
- Thus given only an array [x,y,z...], the magnitude and phase may be calculated

Output

$$x[n] = A\cos(\hat{\omega}_1 n + \phi)$$
$$\Rightarrow y[n] = A\left|H(e^{j\hat{\omega}_1})\right|\cos(\hat{\omega}_1 n + \phi + \angle H(e^{j\hat{\omega}_1}))$$

```
var dcLevelTwo = 0;
for (var i = 0; i <= userInput.length - 1; i++) {
    dcLevelTwo = dcLevelTwo + userInput[i] * dc;
}
```

- The output of a filter may be determined based off of the input parameters, and the filter coefficients of the specified filter
- Magnitude and phase which are seen to the left as |H(e^jw)| and angH(e^jw)
- x[n] represents the input and y[n] represents the output
- In the case that the input function has a non-zero DC level, the output DC level may be determined as shown to the left where userInput holds the filter coefficients and dc holds the input DC level

Approach

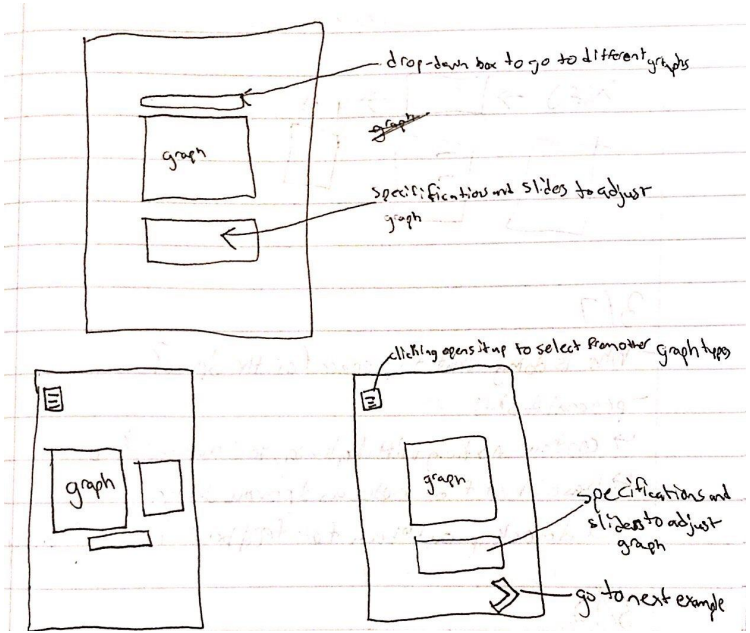
- In order to go from input function to output function, I wrote a series of functions to be used
- The first function, getFREMAGPHA, returns the array [FRE, MAG, PHA]
- FRE, MAG, and PHA are all arrays used to plot the magnitude/phase response of the filter
- The second function, getFilterPoint, returns the array [frequencyFilter, magnitudeFilter, phaseFilter]
- frequencyFilter, magnitudeFilter, and phaseFilter are all decimal values which indicate the frequency of input/output, the magnitude of the filter at that frequency, and the phase of the filter at that frequency
- The third function, getOutputParameters, returns the array [amplitudeTwo, frequencyTwo, phaseTwo, dcLevelTwo]
- amplitudeTwo, frequencyTwo, phaseTwo, and dcLevelTwo are decimal values corresponding to the output parameters so that the output may be graphed in the same format as the input

UI

Goal: Develop a user-friendly layout and make the design more intuitive to users

- Easy access and navigation to different filter types and specifications
- Think about the design in the context of mobile

Initial Sketches

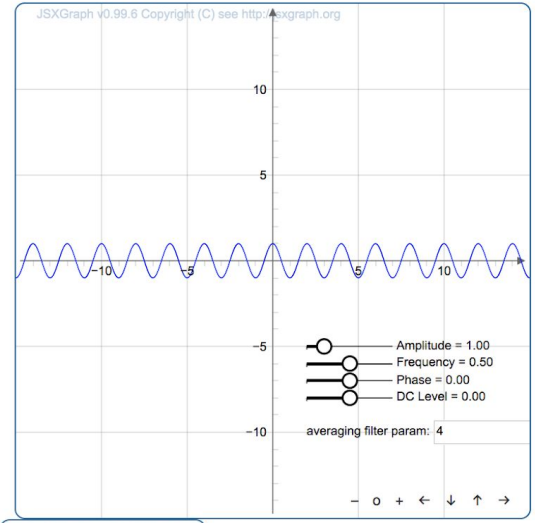


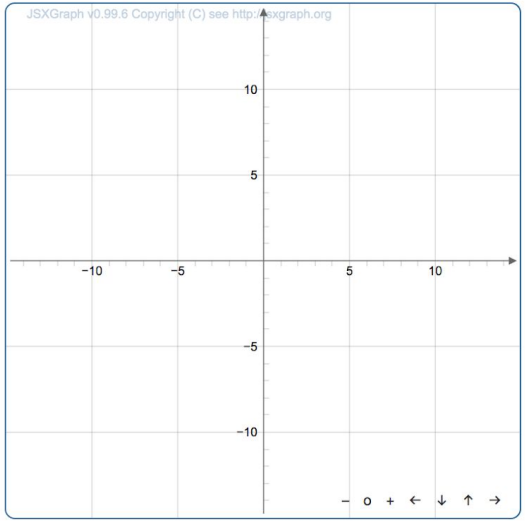
Implementation

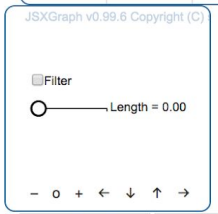
- HTML, CSS, and JavaScript
- Navbar to navigate between different filters
    - Averaging
    - First difference
- Bootstrap to automatically adjust for varying screen sizes

DLTI Demo Home Filter 1 Filter 2

0.00 + 1.00cos(0.00πn + 0.00π)



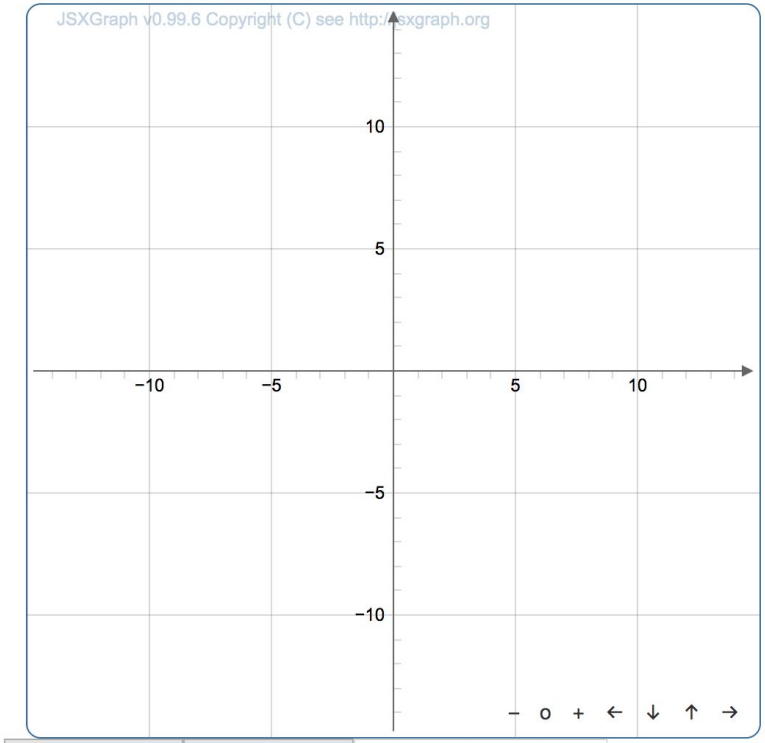


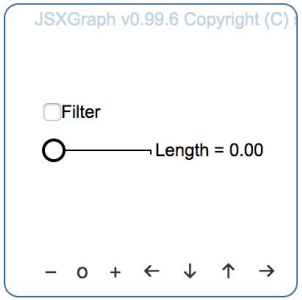


Post GUI State Get GUI State

DLTI Demo Home Filter 1 Filter 2

0.00 + 1.00cos(0.00πn + 0.00π)





Post GUI State Get GUI State

Outcome

- Back End: We have the ability to capture a GUI state by recording the 4 parameters: Amplitude, Frequency, Phase, and DC Level.
- Front-End: A GUI with an input/output that talks to each other and phase/magnitude filter implementation.
- UI: Simplified version with ability to hide certain filters

Next Steps

- Back End: Record the filter type and filter input using modular design
- Front-End: Implement the other filter types
- UI: Working accordion to hide frequency and phase types

Questions?