JavaScript GUIs

Continuous Convolution
Filter Design

Team Members

Reema Patel - 4th year Computer Science

Adam Chau - 2nd year Computer Science

Jessica Bishop - 2nd year Computer Science

Nick Bennett - 3rd year Computer Science

Chawalit Saetiew - 3rd year Computer Science

David Saiontz - 2nd year Computer Science

Project Motivation

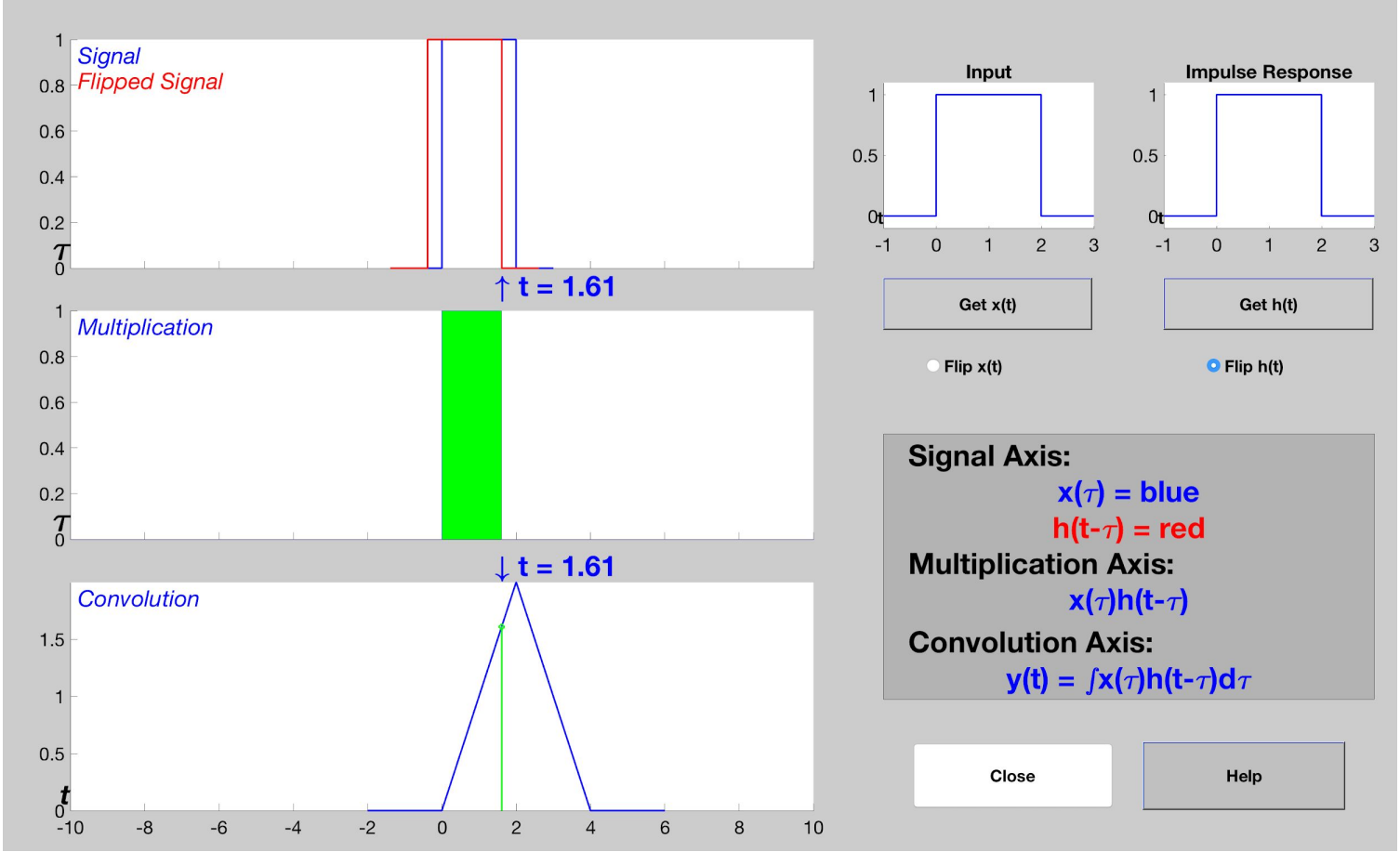
MATLAB required to run current GUIs

Bothersome to have both lab questions and GUI open

To learn and apply JavaScript to convert a MATLAB GUI

Useful for future applications (gathering information about student performance, linking to database, etc.)

Continuous Convolution


Signal
Flipped Signal
τ
t = 1.61
Multiplication
τ
t = 1.61
Convolution
t
Input
Impulse Response
Get x(t)
Get h(t)
Flip x(t)
Flip h(t)
Signal Axis:
x(τ) = blue
h(t-τ) = red
Multiplication Axis:
x(τ)h(t-τ)
Convolution Axis:
y(t) = ∫x(τ)h(t-τ)dτ
Close
Help

MATLAB GUI

Code Structure: React

Separated into 2 components:

- Graph component
- Logic component that took in graph and executed commands on it

Problem: Could not figure out a way to make one board a child of another

Code Structure: HTML/JS

Copied structure from previous semesters' GUIs

Separated into HTML page and JS graph

- JS page holds all logic for the graphs
- HTML page displays graphs

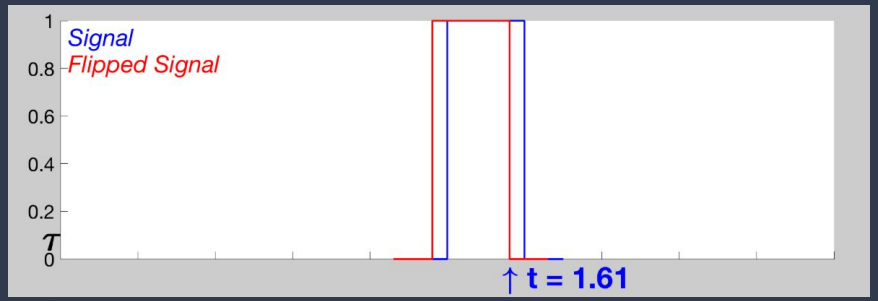
Three Main Components

Signal and Flipped Signal Graph

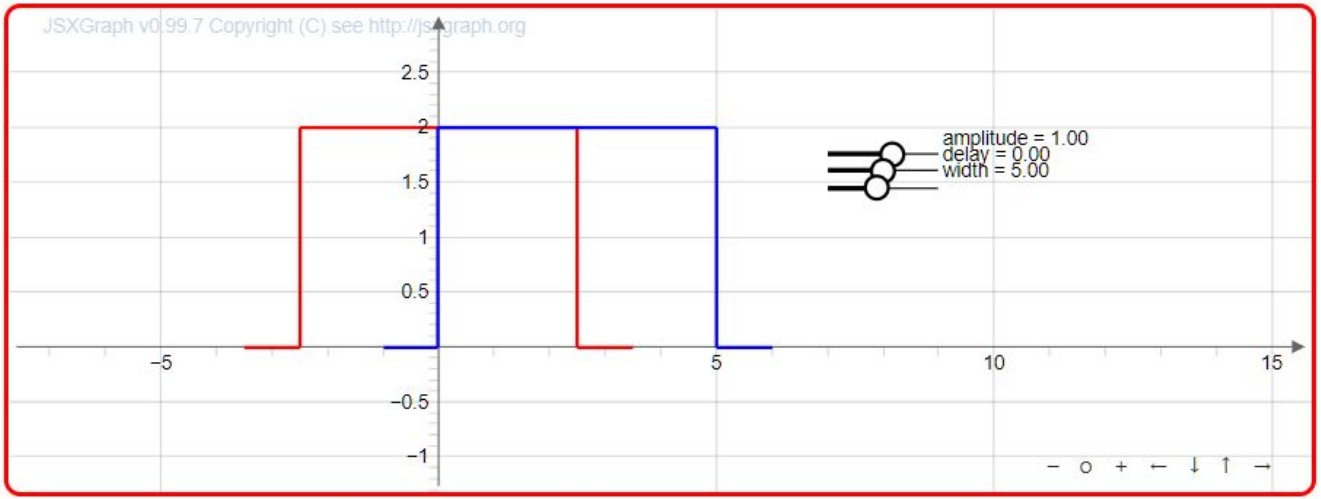
Multiplication Graph

Convolution Graph

Signal Graph



Creating a time pulse graph

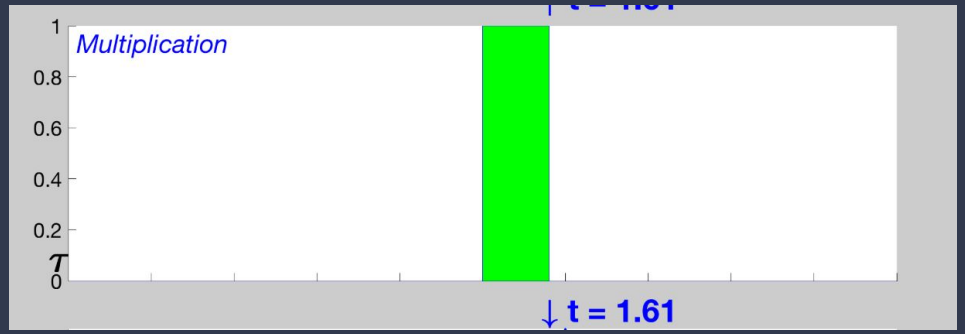


No time pulse function in JSXGraph

5 components

Needed to make each component move with each other when user uses slider

Multiplication Graph

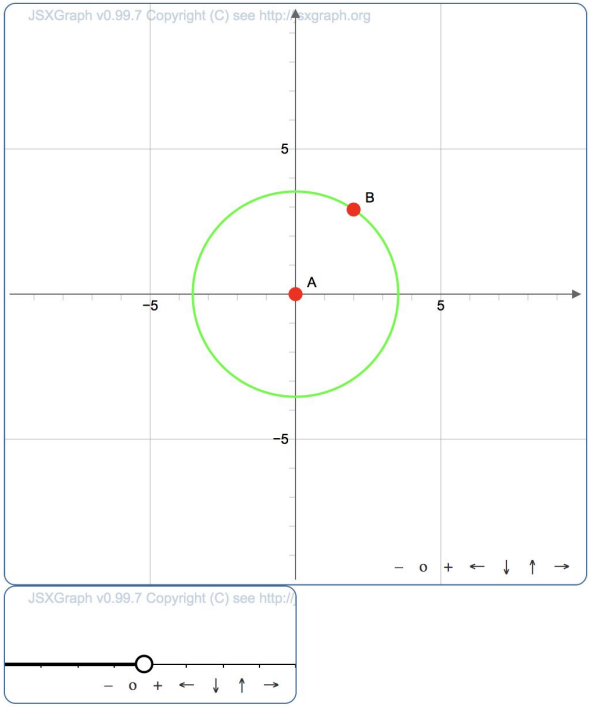


Getting Started...

Something will be controlling the area shown in multiplication graph

Abstract this into a more general device

Use a slider to understand more




```
7  sliderBoard.addChild(board);
```

Showing Area with a Slider

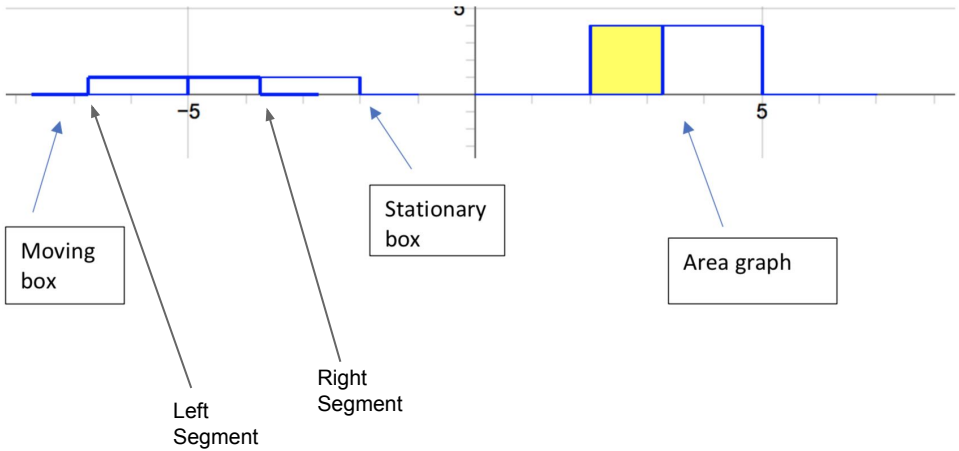
How will we display the shaded area?

Create four points for corners

Use four corners to create a shaded polygon

Can also be used for curves

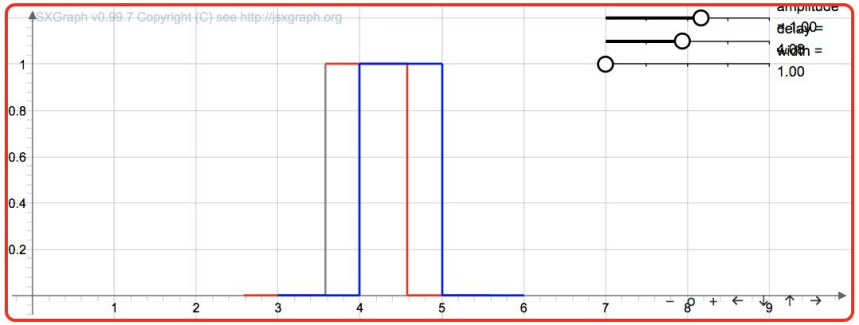
Moving Away from Slider



Keep track of position of right and left segment of moving box

What happens when one of the segments cross the bounds of the stationary box

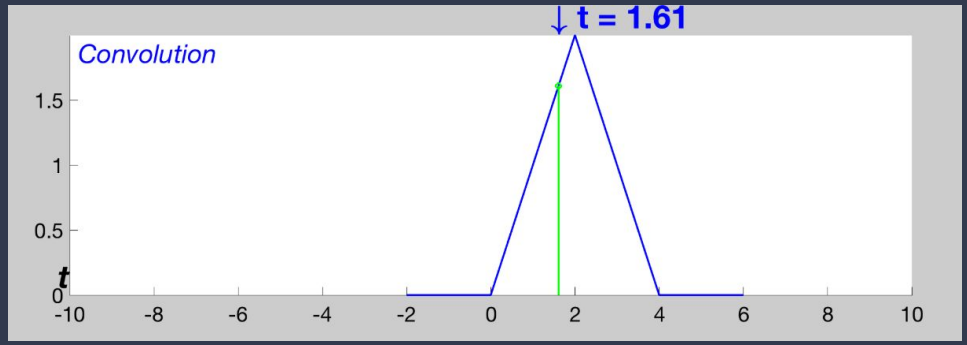
Putting it all together...

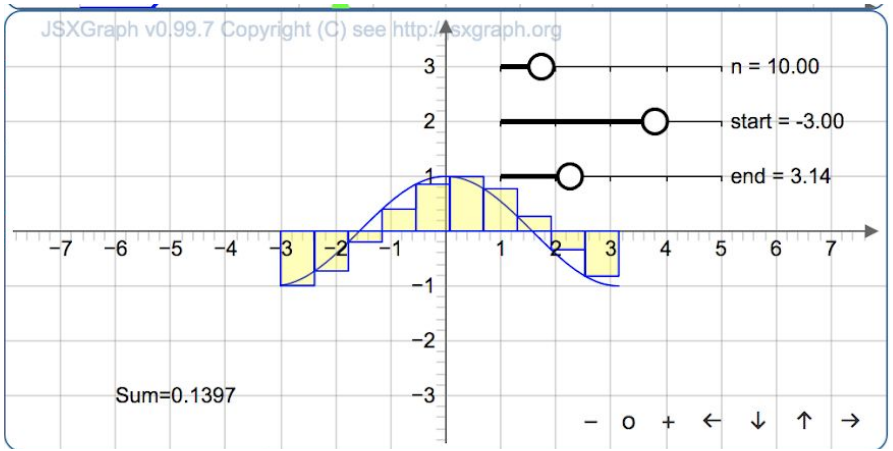


Pulling location data from graph above

Can composite for different lengths

Convolution Graph





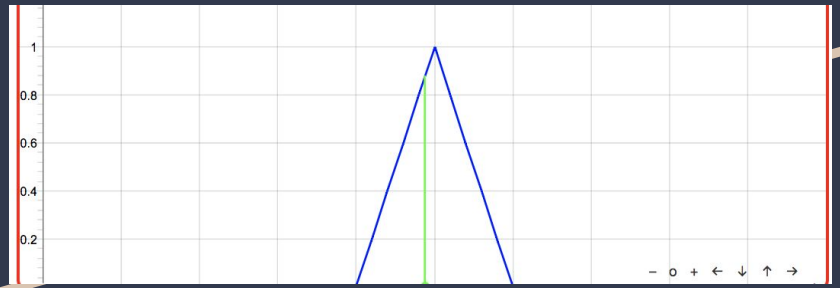
Use of the Riemann sum to try and figure out a solution to the convolution graph

Didn't work because of the lack of time

Aids other VIP-ITS members for future projects involving continuous convolution or Riemann sums

```
brd2.create('text',[-6,-3,function(){ return 'Sum='+(JXG.Math.Numerics.riemannsum(v,
  s.Value(),5,t.Value(),u.Value())).toFixed(4); }]);
```

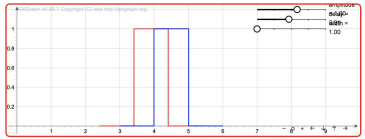
Our Convolution Graph Now

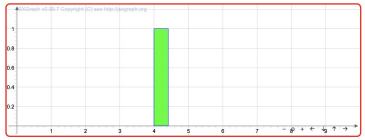


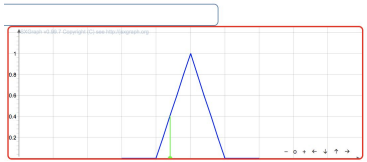
Line has full functionality

Convolution graph is static

Convolution Graph:







Hard coded points

Represent the area of a pulse that is 1 unit wide and 1 unit tall

Trouble making it dynamic

Convolution Line:

X position of line comes from the right segment of moving time pulse graph

Height comes from current area of the shaded polygon in the multiplication graph

```
var top = res.create('point', [function() {
  return d.X();
}, function() {
  return pol1.Area();
}], {trace: false, visible: false});
```

Next Steps

Incorporate GUI with different types of function

- Sine, Cosine, Impulse, Exponential

Connect the x(t) and h(t) input boxes to the graph

Implement flip x(t) and h(t)

Add a dynamically changing convolution graph

Challenges

Learning JavaScript, React, and Matlab

Understanding and graphing continuous convolution

Changing code structure

```
let logicJS = (brd) => {
    var amplitude = brd.create('slider',
```

vs.

```
var sig1 = JXG.JSXGraph.initBoard('box',
```

```
sig1.addChild(sig2);
sig1.addChild(res);
```

Demo

Filter Design

End Goal

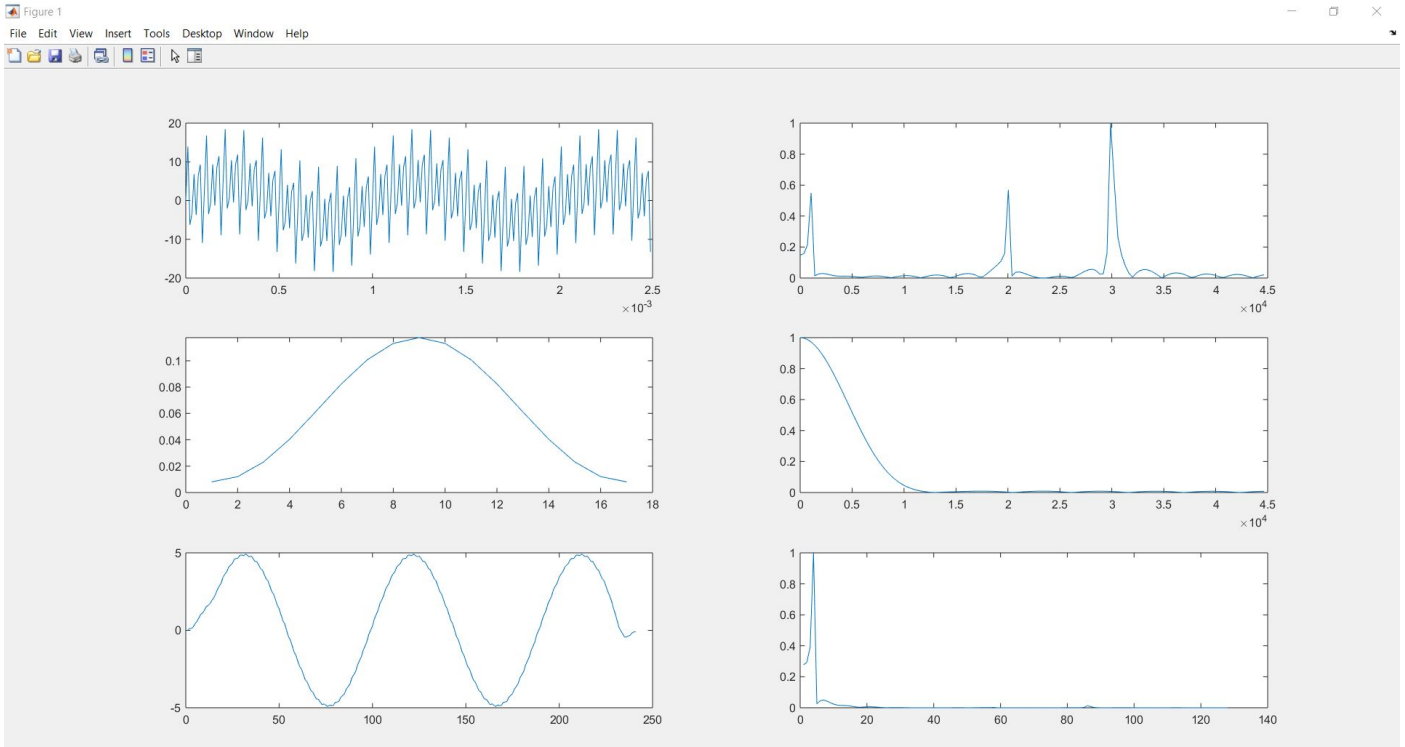
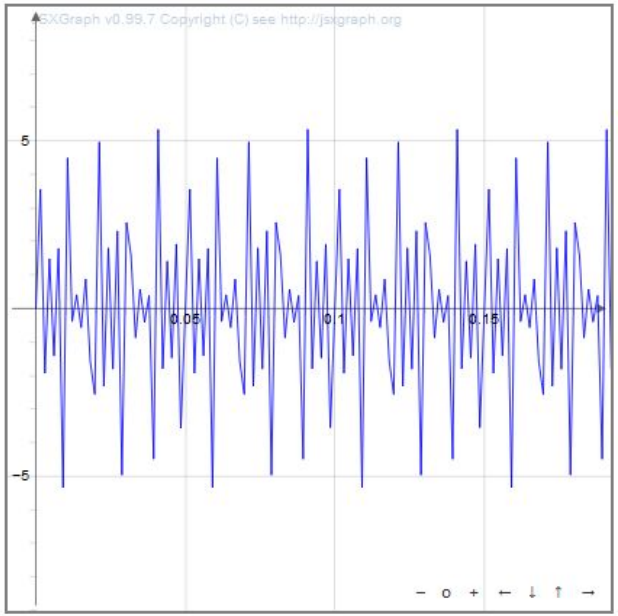
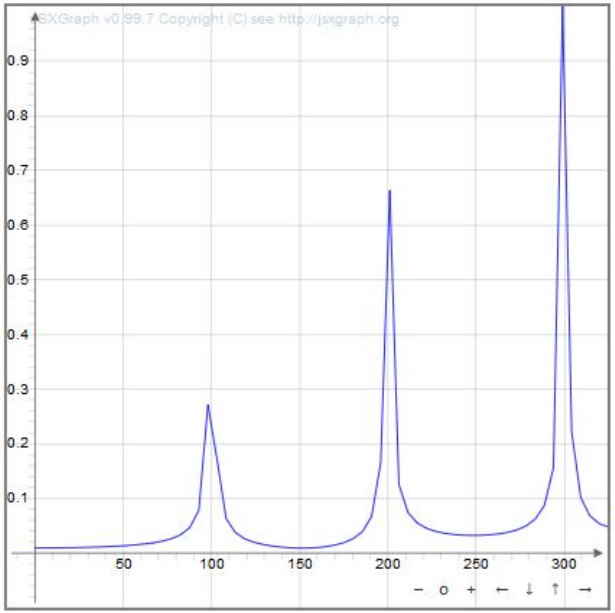


Figure 1
File Edit View Insert Tools Desktop Window Help
×10^-3
×10^4
×10^4

Multiple Frequency Sine Function

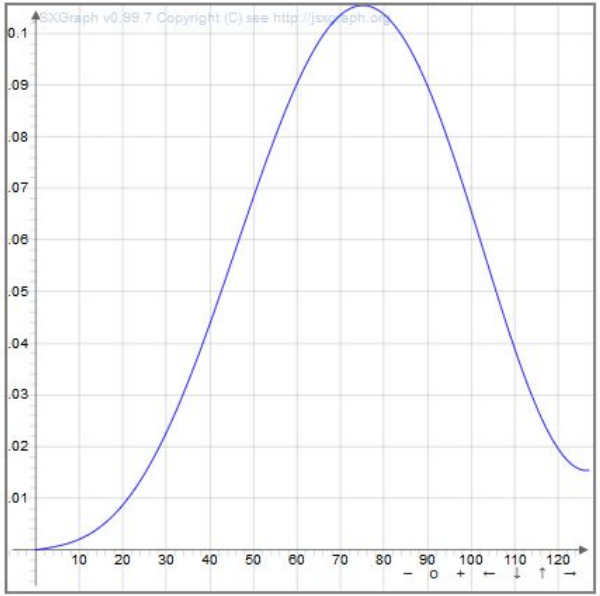


Time Domain

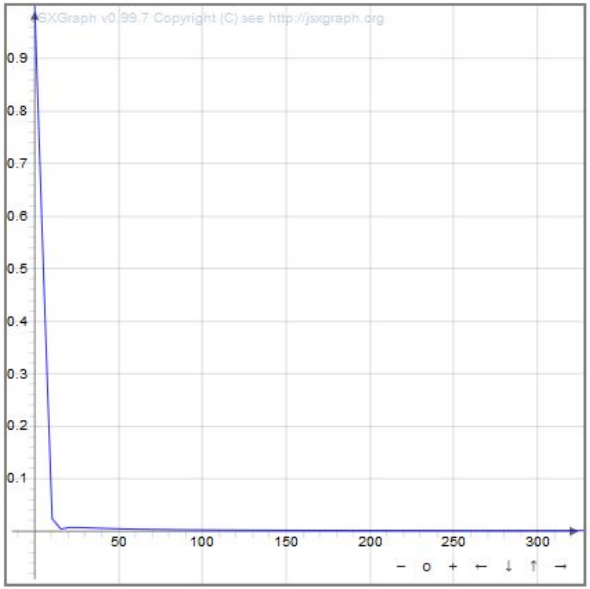


Frequency Domain

Hamming Window

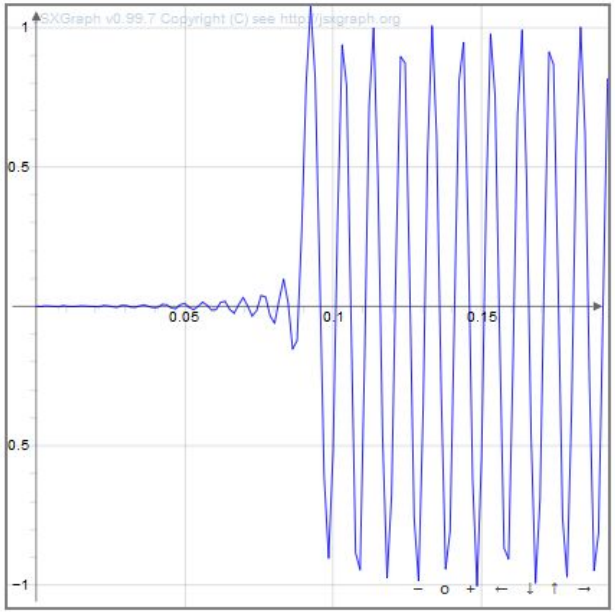


Time Domain

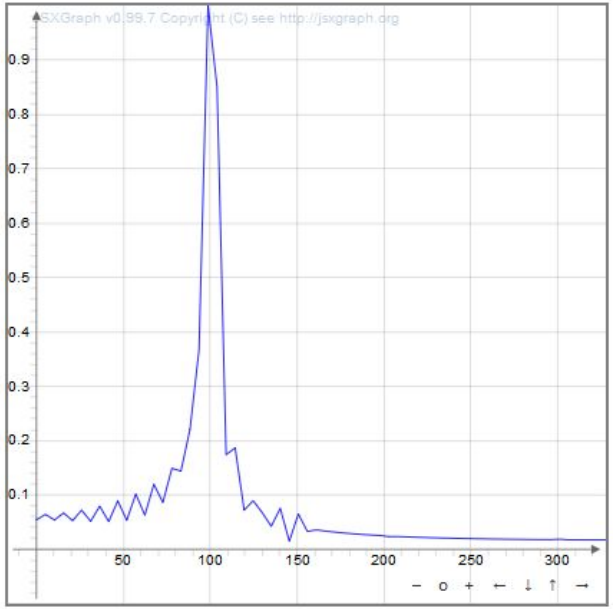


Frequency Domain

Filtered Sine Function



Time Domain



Frequency Domain

dsp and fili library

```
var fft = new dspStuff.FFT(bufferSize, fs);
fft.forward(filY);
var spectrum = fft.spectrum;
var normal = normalizeyfft(spectrum);
var xfft = createXfft(fs, bufferSize - 1);
var yfft = Array.from(spectrum);
```

```
var firCalculator = new Fili.FirCoeffs();
var firFilterCoeffs = firCalculator.lowpass({
    order: order, // filter order
    Fs: fs, // sampling frequency
    Fc: cutoff // cutoff frequency
});
var firFilter = new Fili.FirFilter(firFilterCoeffs);
var filY = firFilter.multiStep(y);
```

NPM create-react-app

Creates foundation of React-based project

Automatically installs many useful libraries, allowing for inline-CSS, creation of HTML objects in JS, etc

React Components

Modular, allows for many HTML and JSXGraph objects to be created and altered quickly in one file

Include states which can be used to store and share information

HTML Listeners

Problem: JSXGraph designed as a closed system within React

Solution: Event listeners for HTML inputs inserted into the body of JSXGraph code

Bypasses closed structure of JSXGraph-react package

Slow compared to native JSXGraph controls

GUI Code Structure

```
class Amplitude3 extends React.Component {
    render() {
        return (
            <input
                type="number"
                id="amp3"
                name="inputbox"
                step="any"
                placeholder="Amplitude 3"
                style={{
                    position: "fixed",
                    left: '100px',
                    top: '450px'
                }}
            />
        )
    }
}
```

Each input is its own React component

Each graph is also a component

All components on the page are encapsulated in a single wrapper which allows for a shared scope

MathJax

JavaScript display engine for LaTeX

$$w_r[n]x[n+n_0] = \left\{ \begin{array}{cc} 0 & n < 0 \\ x[n+n_0] & 0 \leq n < L \\ 0 & n \geq l \end{array} \right\}$$

Used CodeCogs LaTeX editor to create equations

```
let drawEquations = () => {
    return (
        <div
            style={{
                position:'fixed',
                right:'50px',
                top:'125px'
            }}>
            Rectangular Window Equation
            <MathJax.Context input='tex'>
                <div>
                    <MathJax.Node>{rectangularWindowEquation}</MathJax.Node>
                </div>
            </MathJax.Context>
```

Next Steps

Generalize GUI structure (more modular)

Add backend infrastructure

Save user information for persistence

Improve graph controls and communication among components

Add true noise generator

Debug functions for the filtration graphs

Update equations to substitute variables for user inputs

Demo

Conclusion

Learning Outcomes

Became more proficient in JavaScript and MATLAB

Real-world software development skills (GitHub, working in a team, research)

Test out different approaches before picking one

Future Applications

Web-enabled GUI for ECE 2026

Learn information from user interaction with GUIs

Have a basic structure for what a GUI should look like

Questions?