



Dr. Vishwanath Karad

**MIT WORLD PEACE  
UNIVERSITY** | PUNE

TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

**School of Electronics and Communication Engineering**

**Second Year B. Tech. (ECE)**

**Analog Communication**

**Course Code: ECE212A**

**PULSE WIDTH MODULATION**

**A Project Report**

**Submitted by**

**Smit Arekar(PB40)**

**Sarthak Nisal (PB48)**

**Under guidance of**

**Prof. Raghunath S. Bhadade**

## AKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, Professor Raghunath S. Bhadade, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We also take this opportunity to thank all the faculty of the School of Electronics and Communication Engineering for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

It helped us in doing a lot of Research and we as a team came to know about a lot of things related to this topic.

Finally, we would say that this project was a team effort and we helped each other a lot in finalizing this project within the limited time frame and also implementing it well. It was a really good time to experience new thrills working with the GUI.

We all did face some difficulties, but we all came over it and learnt a lot even during the COVID-19 pandemic.

Sr. No.	PRN No.	Name of Student	Contact No.	Email ID
1	1032191122	Smit Arekar	85509 83213	smit.arekar123@gmail.com
2	1032191190	Sarthak Nisal	83810 23891	Sarthak.nisal007@gmail.com

## INDEX

Topics	Page No.
<b>1. Introduction</b>	4
1.1 Aim	4
1.2 Basic Scope and Background	4
<b>2. Literature Survey</b>	5
<b>3. Simulation Design</b>	6
3.1 Block Schematic	6
3.2 System Specifications	6
3.3 Concept Explanation	9
<b>4. Software Design</b>	10
4.1 Software Stack	10
4.2 Actual Implementation	10
4.3 Software Program	11
4.4 GUI	13
4.5 Sample output	14
<b>5. Conclusion</b>	15
<b>6. References</b>	15

## **1. INTRODUCTION:**

Pulse-width modulation (PWM) is a method of reducing the average power delivered by an electrical signal, by effectively chopping it up into discrete parts. The average value of voltage (and current) fed to the load is controlled by turning the switch between supply and load on and off at a fast rate. The longer the switch is on compared to the off periods, the higher the total power supplied to the load. PWM is particularly suited for running inertial loads such as motors, which are not as easily affected by this discrete switching, because their inertia causes them to react slowly. The PWM switching frequency must be high enough not to affect the load, which is to say that the resultant waveform perceived by the load must be as smooth as possible.

### **1.1 AIM:**

To create an application that generates PWM output as per the requirement and inputs given by the user.

### **1.2 BASIC SCOPE AND BACKGROUND:**

Pulse-width modulation (PWM), or pulse-duration modulation (PDM), is a method of reducing the average power delivered by an electrical signal, by effectively chopping it up into discrete parts. The average value of voltage fed to the load is controlled by turning the switch between supply and load on and off at a fast rate. The longer the switch is on compared to the off periods, the higher the total power supplied to the load. Along with maximum power point tracking (MPPT), it is one of the primary methods of reducing the output of solar panels to that which can be utilized by a battery. PWM is particularly suited for running inertial loads such as motors, which are not as easily affected by this discrete switching, because their inertia causes them to react slowly. The PWM switching frequency has to be high enough not to affect the load, which is to say that the resultant waveform perceived by the load must be as smooth as possible.

The frequency at which the power supply must switch can vary greatly depending on load and application. For example, switching has to be done several times a minute in an electric stove; 120 Hz in a lamp dimmer; between a few kilohertz (kHz) and tens of kHz for a motor drive;

and well into the tens or hundreds of kHz in audio amplifiers and computer power supplies. The main advantage of PWM is that power loss in the switching devices is very low. When a switch is off there is practically no current, and when it is on and power is being transferred to the load, there is almost no voltage drop across the switch. Power loss, being the product of voltage and current, is thus in both cases close to zero. PWM also works well with digital controls, which, because of their on/off nature, can easily set the needed duty cycle. PWM has also been used in certain communication systems where its duty cycle has been used to convey information over a communications channel.

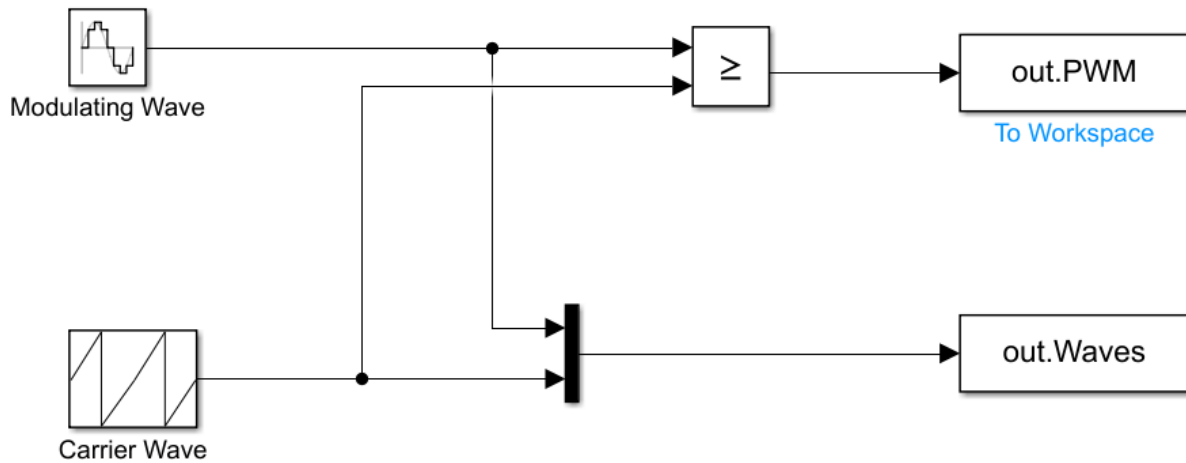
## **2. LITERATURE SURVEY:**

Various studies and research have been carried out over PWM design, and PWM techniques. Some of the research works which inspired this project are mentioned below. "A review on different PWM techniques for five leg voltage source inverters," by A. Dixit, N. Mishra, S. K. Sinha and P. Singh. Presently, many industrial processes require high performance switching control of a number of inverter-fed induction machines. Multi-motor drives employ 3 various methods to reduce complexity and minimize the cost. Independent control of motors is required as these motors are subjected to frequently changing operating conditions. Recent research have ascertained the independent control of parallel connected dual induction motor drive system fed through a five leg voltage source inverter (FLVSI). Implementation of suitable Pulse Width Modulation (PWM) technique to the inverter fed system is of paramount importance. This paper reviews the execution method of recently proposed PWM schemes for independent control of dual induction motor drive system employing a Five Leg Inverter (FLI). "A novel multiple modes PWM controller for LEDs," by J. Lu and X. Wu, A monolithic controller for pulse width modulation (PWM) DC-DC converter was presented in this paper. The controller was designed especially for LED (light emitting diode) driver circuit with four operation modes, current feedback mode, constant current mode, no sense resistor mode and PWM dimming mode. The controller can be adapted to almost all current DC-DC topologies such like Boost, Buck-Boost and etc. It also features the different load current sense methods for different topologies. For LED lighting, both the digital and analog dimming modules were integrated onto the chip, which were used to meet the demands of two kinds of dimming

applications, respectively. The controller integrated circuit (IC) was designed, simulated and. And both the simulation and test results were consistent with expectations well.

### 3. SIMULATION DESIGN

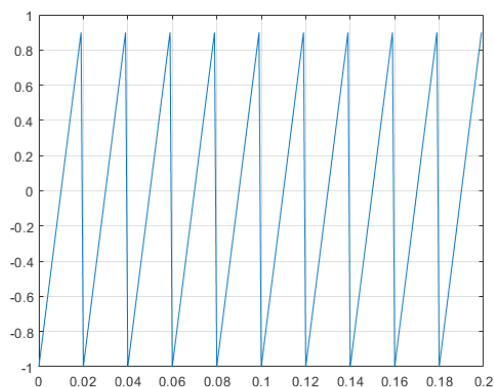
#### 3.1 BLOCK SCHEMATIC:



#### 3.2 SYSTEM SPECIFICATIONS

##### SAWTOOTH GENERATOR

The Sawtooth Generator block generates a sawtooth waveform, with peak amplitude of  $\pm 1$ , at regular intervals. The figure shows how the **Frequency** and **Phase** block parameters affect the output waveform.



### ***Parameters***

#### *Frequency (Hz)*

Specify the frequency of the sawtooth waveform, in hertz. Default is 1e3.

#### *Phase (degrees)*

Specify the delay of the sawtooth waveform, in degrees. When the phase is set to 0, the waveform starts with an amplitude of  $-1$  and a positive slope. Default is 180.

#### *Sample time*

Specify the sample time of the block, in seconds. Set to 0 to implement a continuous block. Default is 0.

### **RELATIONAL PERATOR:**

The Relational Operator block performs the specified relational operation on the input. The value you choose for the Relational operator parameter determines whether the block accepts one or two input signals.

#### **Two-Input Mode**

By default, the Relational Operator block compares two inputs using the Relational operator parameter that you specify. The first input corresponds to the top input port and the second input to the bottom input port.

Operation	Description
==	True if the first input is equal to the second input
~=	True if the first input is not equal to the second input
<	True if the first input is less than the second input
<=	True if the first input is less than or equal to the second input
>=	True if the first input is greater than or equal to the second input
>	True if the first input is greater than the second input

### **MUX:**

The Mux block combines inputs with the same data type and complexity into a vector output. The output mux signal is flat, even if you create the mux signal from other mux signals.

However, you can use multiple Mux blocks to create a mux signal in stages. A mux signal simplifies the visual appearance of a model by combining two or more signal lines into one line. Mux signals do not affect simulation or code generation.

### **SINE WAVE GENERATOR:**

The Sine Wave block generates a multichannel real or complex sinusoidal signal, with independent amplitude, frequency, and phase in each output channel. The block supports floating point and signed fixed-point data types.

The block generates a real sinusoidal signal when you set the Output complexity parameter to Real. The real sinusoidal output is defined by an expression of the type

$$y = A \sin(2\pi f t + \phi)$$

where you specify A in the Amplitude parameter, f in hertz in the Frequency parameter, and  $\phi$  in radians in the Phase offset parameter.

#### ***Parameters***

##### *Frequency (Hz)*

Specify the frequency of the sawtooth waveform, in hertz. Default is 100.

##### *Amplitude*

Specify the Amplitude of the sine wave. Default is 1.

##### *Phase offset(degrees)*

Specify the delay of the sine waveform, in degrees. When the phase is set to 0, the waveform starts with an amplitude of 0 and a positive slope. Default is 0.

##### *Sample time*

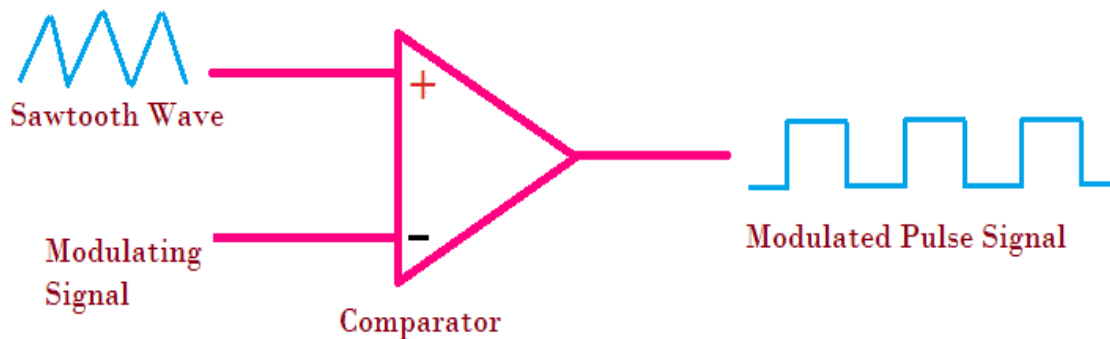
Specify the sample time of the block, in seconds. Set to 0 to implement a continuous block. Default is 0.



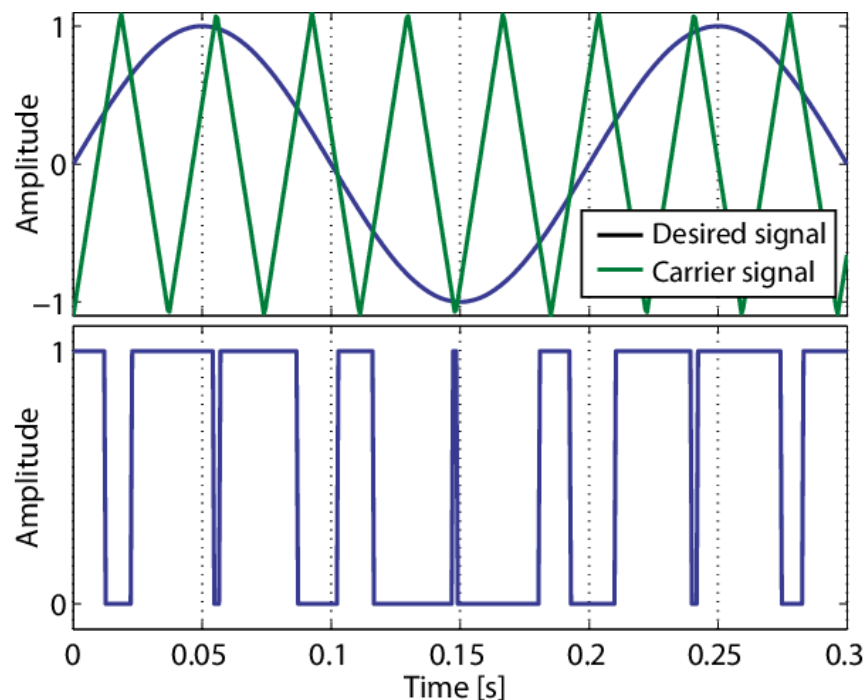
### 3.3 CONCEPT EXPLANATION

#### PWM GENERATION:

Pulse Width Modulating signal can be generated using a Comparator (relational operator block in our case) as shown. Modulating signal forms one of the input to the Comparator and the other input is fed with a non-sinusoidal wave or sawtooth wave. It operates at carrier frequency. The Comparator compares the two signals and generates a PWM signal as its output waveform.



If the value of the Sawtooth triangle signal is more than the modulation signal, then the PWM output signal is at “High” else it’s in “Low” state. Thus, the value of the input signal magnitude determines the comparator output which defines the width of the pulse generated at the output.



SPWM modulation is based on constant amplitude pulses with different duty cycles for each period. The width of pulses is obtained by modulation of a carrier to obtain the desired output voltage and to reduce its harmonic content. The carrier signal of SPWM is usually a triangular wave with a high frequency, generally in several KHz. The modulation signal of SPWM is a sinusoidal waveform with a frequency equal to the desired output voltage frequency.

In this PWM technique, the sinusoidal AC voltage reference  $V_{ref}$  is compared with the high-frequency triangular carrier wave  $V_c$  in real time to determine switching states for each pole in the inverter. After comparing, the switching states for each pole can be determined based on the following rule:

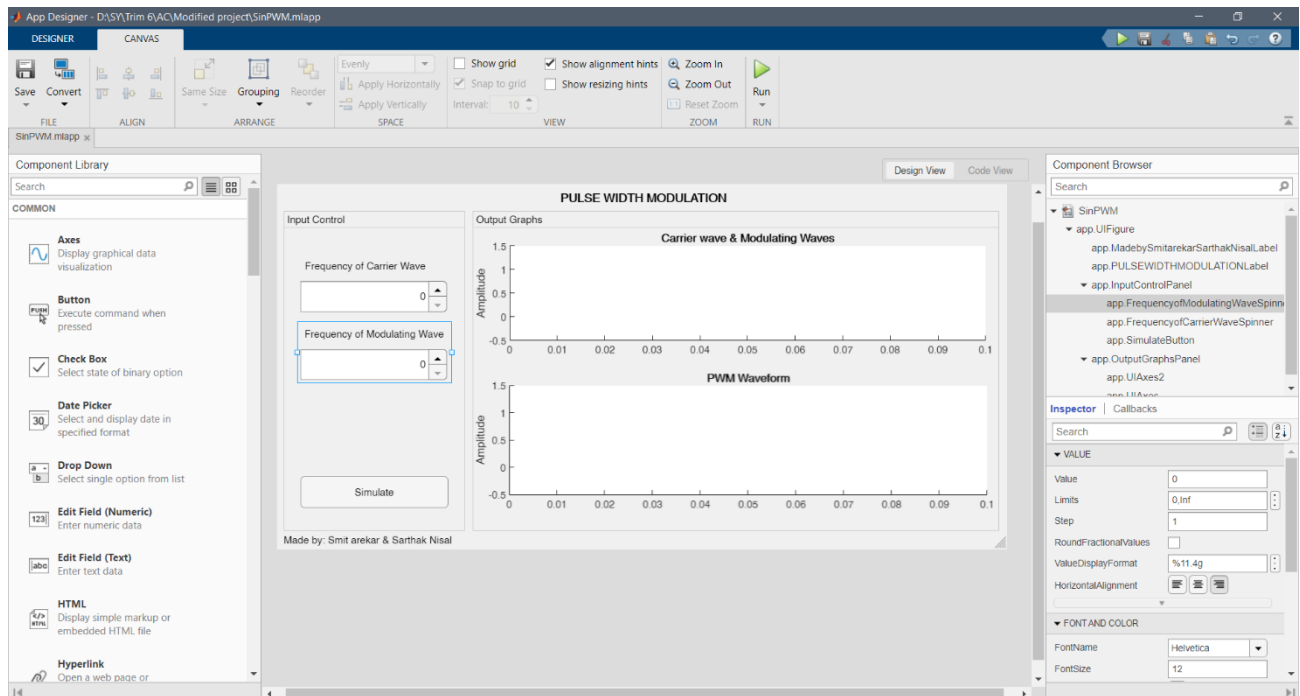
- Voltage reference  $v_{ref} > \text{Triangular carrier } v_c$ : upper switch is turned on
- Voltage reference  $v_{ref} < \text{Triangular carrier } v_c$ : lower switch is turned on

## 4. SOFTWARE DESIGN

### 4.1 Software Stack

Matlab 2021b Simulink and Matlab App designer was used to design the GUI

### 4.2 ACTUAL IMPLEMENTATION



### 4.3 SOFTWARE PROGRAM

```
classdef SinPWM < matlab.apps.AppBase

% Properties that correspond to app components
properties (Access = public)
    UIFigure                                matlab.ui.Figure
    MadebySmitarekarSarthakNisallLabel      matlab.ui.control.Label
    PULSEWIDTHMODULATIONLabel              matlab.ui.control.Label
    InputControlPanel                       matlab.ui.container.Panel
    FrequencyofModulatingWaveSpinner         matlab.ui.control.Spinner
    FrequencyofModulatingWaveSpinnerLabel    matlab.ui.control.Label
    FrequencyofCarrierWaveSpinner           matlab.ui.control.Spinner
    FrequencyofCarrierWaveSpinnerLabel      matlab.ui.control.Label
    SimulateButton                          matlab.ui.control.Button
    OutputGraphsPanel                       matlab.ui.container.Panel
    UIAxes2                                 matlab.ui.control.UIAxes
    UIAxes                                  matlab.ui.control.UIAxes
end

% Callbacks that handle component events
methods (Access = private)

% Button pushed function: SimulateButton
function SimulateButtonPushed(app, event)
    assignin("base", "Fc", app.FrequencyofCarrierWaveSpinner.Value);
    assignin("base", "Fm", app.FrequencyofModulatingWaveSpinner.Value);
    output = sim("D:\SY\Trim 6\AC\Modified project\SPWM.slx");
    plot(app.UIAxes, output.Waves.time, output.Waves.data);
    plot(app.UIAxes2, output.PWM.time, output.PWM.data);
end

end

% Component initialization
methods (Access = private)

% Create UIFigure and components
function createComponents(app)

% Create UIFigure and hide until all components are created
app.UIFigure = uifigure('Visible', 'off');
app.UIFigure.Position = [100 100 869 438];
app.UIFigure.Name = 'MATLAB App';

% Create OutputGraphsPanel
app.OutputGraphsPanel = uipanel(app.UIFigure);
app.OutputGraphsPanel.Title = 'Output Graphs';
app.OutputGraphsPanel.Position = [234 23 626 380];

% Create UIAxes
app.UIAxes = uiaxes(app.OutputGraphsPanel);
title(app.UIAxes, 'Carrier wave & Modulating Waves')
ylabel(app.UIAxes, 'Amplitude')
zlabel(app.UIAxes, 'Z')
app.UIAxes.XLim = [0 0.1];
```

```

app.UIAxes.YLim = [-0.5 1.5];
app.UIAxes.Position = [0 191 622 167];

% Create UIAxes2
app.UIAxes2 = uiaxes(app.OutputGraphsPanel);
title(app.UIAxes2, 'PWM Waveform')
ylabel(app.UIAxes2, 'Amplitude')
zlabel(app.UIAxes2, 'Z')
app.UIAxes2.XLim = [0 0.1];
app.UIAxes2.YLim = [-0.5 1.5];
app.UIAxes2.Position = [0 7 623 185];

% Create InputControlPanel
app.InputControlPanel = uipanel(app.UIFigure);
app.InputControlPanel.Title = 'Input Control';
app.InputControlPanel.Position = [9 23 216 380];

% Create SimulateButton
app.SimulateButton = uibutton(app.InputControlPanel, 'push');
app.SimulateButton.ButtonPushedFcn = createCallbackFcn(app,
@SimulateButtonPushed, true);
app.SimulateButton.Position = [20 27 176 38];
app.SimulateButton.Text = 'Simulate';

% Create FrequencyofCarrierWaveSpinnerLabel
app.FrequencyofCarrierWaveSpinnerLabel = uilabel(app.InputControlPanel);
app.FrequencyofCarrierWaveSpinnerLabel.HorizontalAlignment = 'right';
app.FrequencyofCarrierWaveSpinnerLabel.Position = [20 304 149 22];
app.FrequencyofCarrierWaveSpinnerLabel.Text = 'Frequency of Carrier Wave';

% Create FrequencyofCarrierWaveSpinner
app.FrequencyofCarrierWaveSpinner = uispinner(app.InputControlPanel);
app.FrequencyofCarrierWaveSpinner.Limits = [0 Inf];
app.FrequencyofCarrierWaveSpinner.Position = [20 260 176 37];

% Create FrequencyofModulatingWaveSpinnerLabel
app.FrequencyofModulatingWaveSpinnerLabel=uilabel(app.InputControlPanel);
app.FrequencyofModulatingWaveSpinnerLabel.HorizontalAlignment = 'right';
app.FrequencyofModulatingWaveSpinnerLabel.Position = [20 223 171 22];
app.FrequencyofModulatingWaveSpinnerLabel.Text='Frequency of Modulating
Wave';

% Create FrequencyofModulatingWaveSpinner
app.FrequencyofModulatingWaveSpinner = uispinner(app.InputControlPanel);
app.FrequencyofModulatingWaveSpinner.Limits = [0 Inf];
app.FrequencyofModulatingWaveSpinner.Position = [20 179 176 37];

% Create PULSEWIDTHMODULATIONLabel
app.PULSEWIDTHMODULATIONLabel = uilabel(app.UIFigure);
app.PULSEWIDTHMODULATIONLabel.HorizontalAlignment = 'center';
app.PULSEWIDTHMODULATIONLabel.FontSize = 14;
app.PULSEWIDTHMODULATIONLabel.FontWeight = 'bold';
app.PULSEWIDTHMODULATIONLabel.Position = [264 406 346 26];
app.PULSEWIDTHMODULATIONLabel.Text = 'PULSE WIDTH MODULATION';

```

```

% Create MadebySmitarekarSarathakNisalLabel
app.MadebySmitarekarSarathakNisalLabel = uilabel(app.UIFigure);
app.MadebySmitarekarSarathakNisalLabel.Position = [9 2 253 22];
app.MadebySmitarekarSarathakNisalLabel.Text = 'Made by: Smit arekar & Sarthak Nisal';

% Show the figure after all components are created
app.UIFigure.Visible = 'on';
end
end

% App creation and deletion
methods (Access = public)

% Construct app
function app = SinPWM

% Create UIFigure and components
createComponents(app)

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

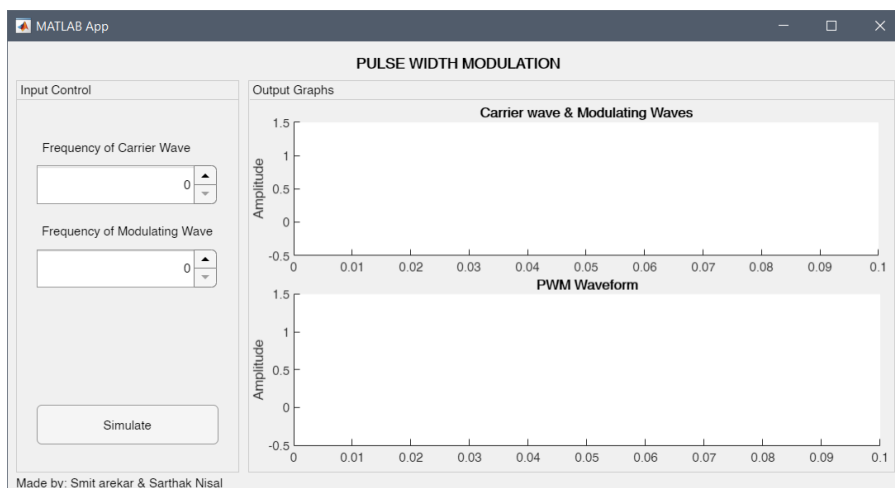
if nargin == 0
    clear app
end
end

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

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

```

## 4.4 GUI



## 4.5 SAMPLE OUTPUT

$F_c = 300\text{Hz}$  ;  $F_m = 50\text{ Hz}$



$F_c = 200\text{Hz}$  ;  $F_m = 30\text{ Hz}$




## 5. CONCLUSION:

Pulse Width modulation is an important method used in many fields, from driving servo motors to control its speeds, or in telecommunication where PWM is a form of signal modulation where the widths of the pulses correspond to specific data values encoded at one end and decoded at the other. In this project we have implemented a Sinusoidal Pulse Width Modulation using Matlab. The GUI deals with a sawtooth Carrier and Sinusoidal modulating wave. The project can be further developed to provide user options of different types of carrier and modulation waves with their adjustable parameters.

## 6. REFERENCE:

- 1] Jalnekar, Rajesh & Jog, K. (2015). "Pulse-Width-Modulation Techniques: A Review", IETE Journal of Research., 46. 175-183. 10.1080/03772063.2000.11416153.
- 2] Barr, M. (n.d.). Introduction to Pulse Width Modulation (PWM). Retrieved 6 12, 2021.
- 3] Bitar, S. J., Crowley, I. F., & Leung, H. F. (2011). PWM Techniques: A Pure Sine Wave Inverter. Retrieved 6 12, 2021.



Smit Arekar



Sarthak Nisal