School of Electronics and Communication Engineering
Academic Year 2020 - 2021

Mini Project Report - Trimester VI

Title of the Project: DSBFC WITH SPECTRUM
Course Name: Analog Communication
Name of Students:

Sr. No.	PRN No.	Name of Student	Contact No.	Email ID
1	1032191969	JAY SRIVASTAVA	9096634954	jaysriva18@gmail.com
2	1032191968	CHETAN VYAS	9352383684	chetanvyas0220@gmail.com

Introduction:

The transmission of a signal, which contains a carrier along with two sidebands can be termed as Double Sideband Full Carrier system or simply DSBFC. It is plotted as shown in the following figure. However, such a transmission is inefficient. Because, two-thirds of the power is being wasted in the carrier, which carries no information.

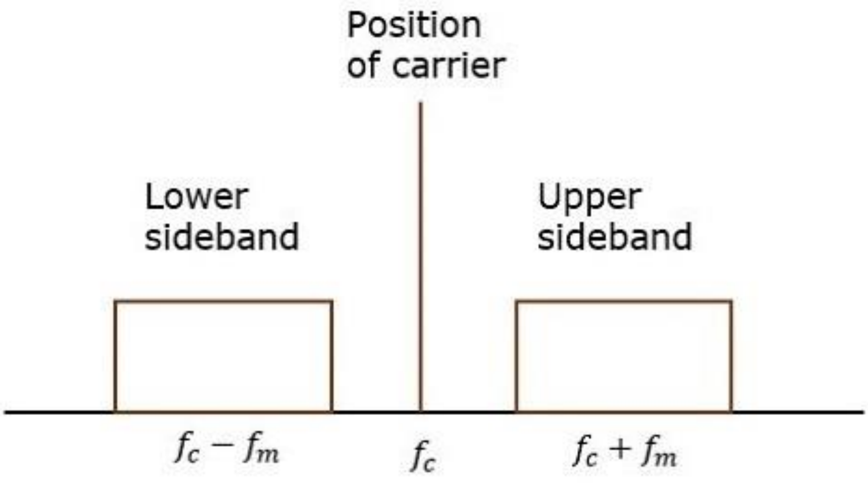


Figure 1

For a real $x(t)$, $|X(f)|$ is symmetric about $f = 0$ and $|X(f - fc)|$ is symmetric about $f = fc$. For the present discussion, it is only necessary to consider that part of

$Y(f)$ that lies in the positive half of the frequency axis. The signal content that lies in the frequency domain below fc is the lower sideband. The signal content that lies in the frequency domain above fc is the upper sideband. This is the origin of the term double sideband.

When studying and testing Analog modulation schemes, it is convenient to use a sinusoid as the message signal. This is a good choice for several reasons. First, when testing a system in the laboratory, it is desirable to use a periodic signal since a stable oscilloscope display with continuous signal capture is then possible. Second, the mathematics are usually simpler with a sinusoidal message signal. Third, a sinusoid is easy to generate in the laboratory.If $x(t)$ is a sinusoid of frequency fm, the modulated carrier can be written as

$$y(t) = \cos(2\pi fmt) \cdot \cos(2\pi fc\ t)\ 2$$

$$=1/\ 2 \cos[2\pi(fc - fm)t] + 1\ 2 \cos[2\pi(fc + fm)t]$$

This last expression indicates that when a carrier is DSB modulated by a message sinusoid, the modulated carrier is equivalent to the sum of two sinusoids: one having the difference frequency $fc - fm$ and the other with the sum frequency $fc + fm$. In the frequency domain there is signal content lying on both sides of the carrier frequency. When $x(t)$ is a sinusoid, each sideband is one discrete spectral line (on the positive half of the frequency axis). In the general case where $x(t)$ is real but not a sinusoid, each sideband is more complicated than a single discrete spectral line. However, it is still true that there are two sidebands: a lower and upper sidebands.

In the time domain, a carrier that is DSB modulated by a message sinusoid looks like this:

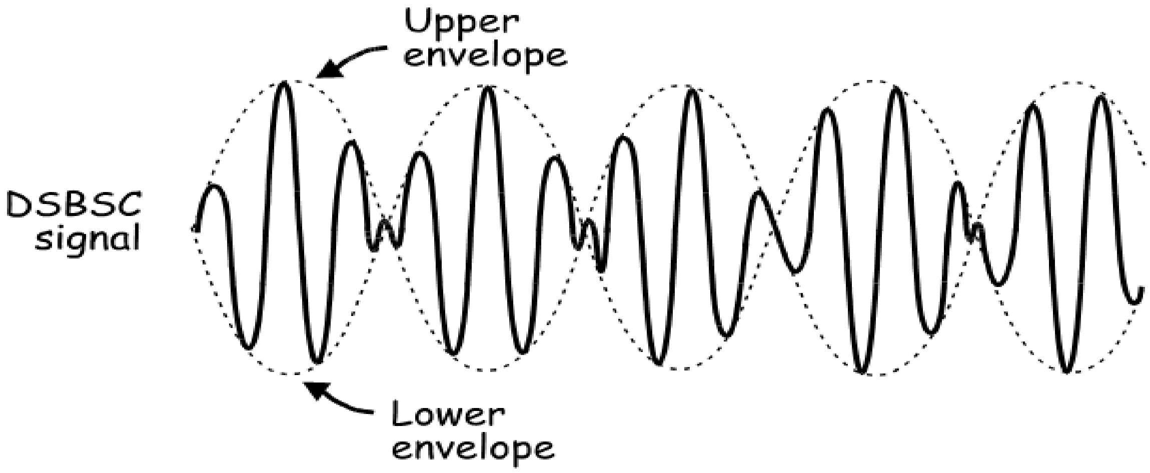


Figure 2

Block Schematic and Explanation:

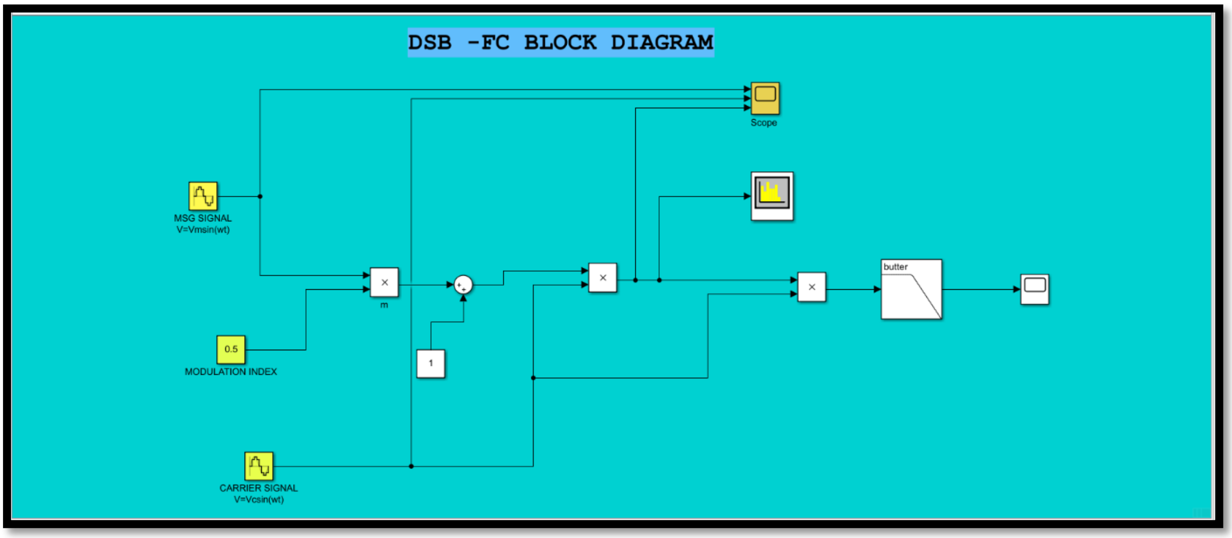


Figure 3

Here the modulating signals might be an audio or video signal. These are also called as baseband signals as these are modulated with the carrier signals. Carriers are extremely high-frequency radio signals, In general, carrier signals are received from the RF oscillators. These two signals are combined in a modulator. The modulator considers the instant amplitude of the modulating signal and modifies it as per the amplitude of the carrier signal. So, the resultant signal amplitude is the amplitude of the modulated signal. The modulated signal is passed through the amplifier for the amplitude modulation and then transmitted through an antenna or a co-axial cable.

After the process of modulation, the modulated level of the carrier is calculated, and this exertion is termed as Modulation index. It determines the level of the modulation that a carrier wave undergoes. Below is the modulated signal.

$$\boxed{M(t) = [A_m + A_c \sin (2\pi (f_m + f_c)t)]}$$

So, When the amplitudes of both the carrier and modulating signal are known, the modulation index can be known. The resultant modulated wave shows maximum amplitude in the condition $\sin (2\pi f_m * t)$ is 1.

$$\boxed{\mathbf{A_{max} = A_i + A_c}}$$

Similarly, the resultant modulated wave shows minimum amplitude in the condition $\sin(2\pi f_m * t)$ is -1.

$$Amin = Ai – Ac$$

Combining and solving the maximum and minimum amplitude equations, the amplitude of the carrier signal can be known, where

$$Amax + Amin = Ai + Ac + Ai - Ac = 2Ac$$

$$Ac = (Amax + Amin)/2$$

$$\text{Whereas } Amax – Amin = Ai + Ac – (Ai - Ac) = 2Ai$$

$$Ai = (Amax – Amin)/2$$

With the above Ai and Ac equations, the ratio of these can be calculated as

$$Ai/Ac = [(Amax – Amin)/2]/[(Amax + Amin)/2]$$

$$\boxed{\mu = (Amax – Amin)/(Amax + Amin)}$$

Specifications:

- The frequency band used for AM radio is about 550 to 1720 kHz.
- The information transmitted is music and talk which falls in the audio spectrum. The full audio spectrum ranges up to 20 kHz, but AM radio limits the upper modulating frequency to 5 kHz
- This results in a maximum bandwidth of 10 kHz. Therefore, the FCC can assign stations frequencies that are 10 kHz apart without fear of overlap in reality, there still can be some overlap because the spectrum doesn't just end at the side-band, it actual kind of tapers off slowly.
- These "tails" can overlap if the signal is strong enough. You can make you receiver more selective by changing from the "distant" to the "local" setting to eliminate this at the expense of sensitivity). So if we fill up the AM band, assigning stations every 10 kHz, there are 107 available transmitter frequencies.
- The practice of limiting the upper frequency to 5 kHz removes some of the original information (that which falls in the 5-20 kHz) range. Since the ability to exactly reproduce the signal is called fidelity, there is a loss of fidelity in AM broadcasts.

- This is one of the reasons that AM radio doesn't sound that good (compared to FM radio). Talk radio is relatively unaffected because conversation has very little of its signal above 5 kHz anyway. This might explain why talk radio is much more common on AM than FM.

Screen shots:

1. Actual Implementation (Front View)

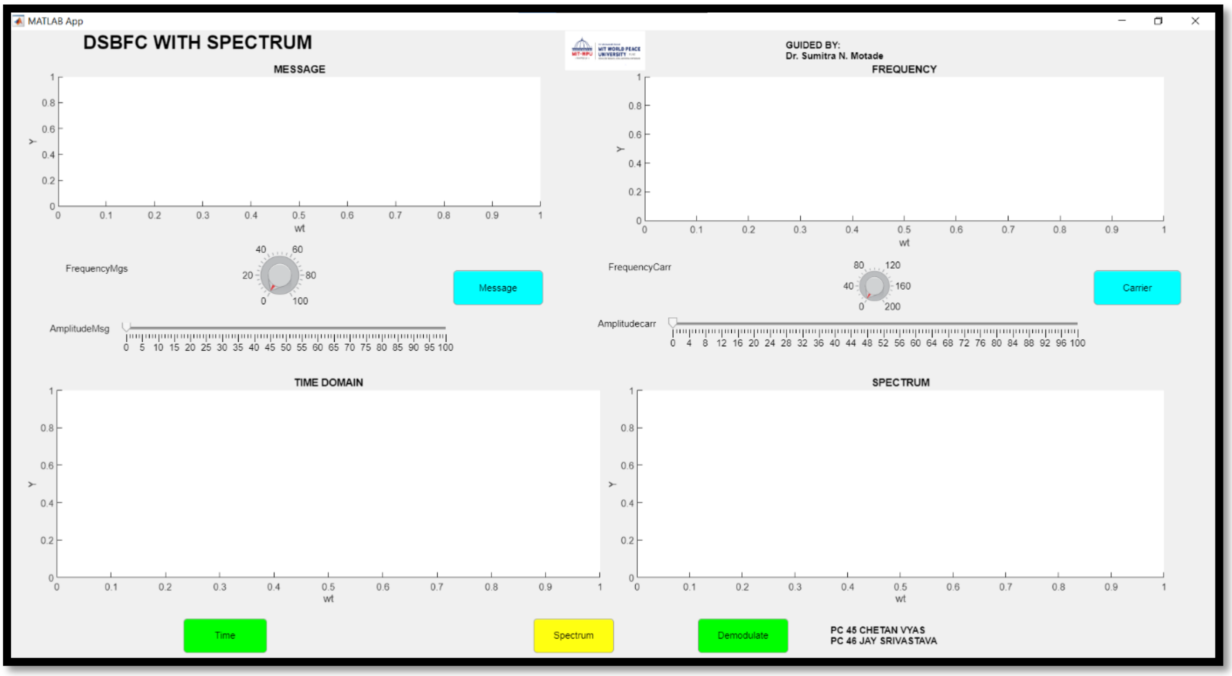


Figure 4

2. Outputs

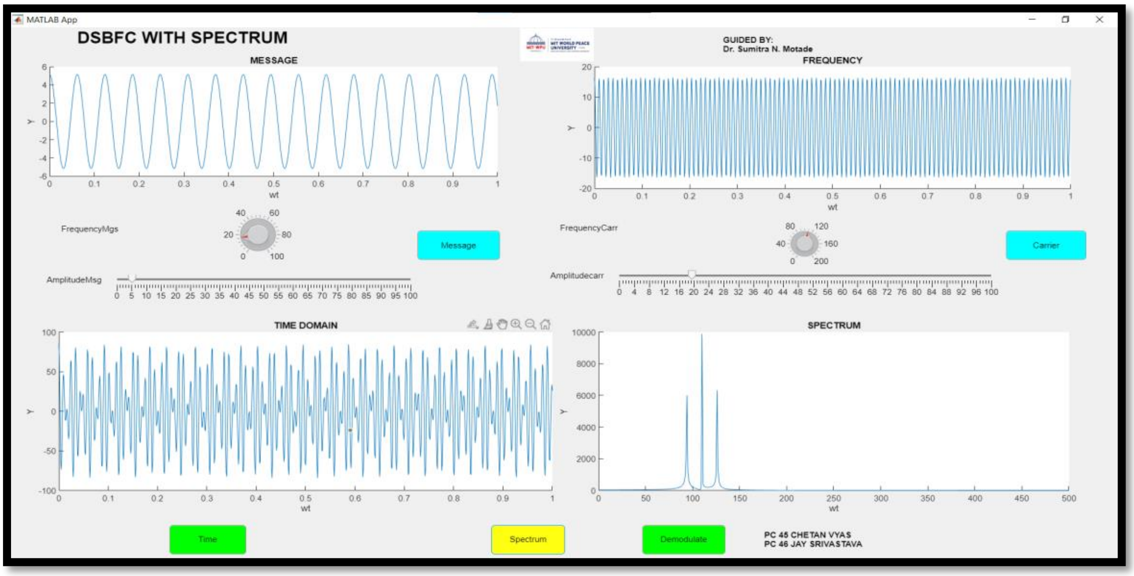


Figure 5

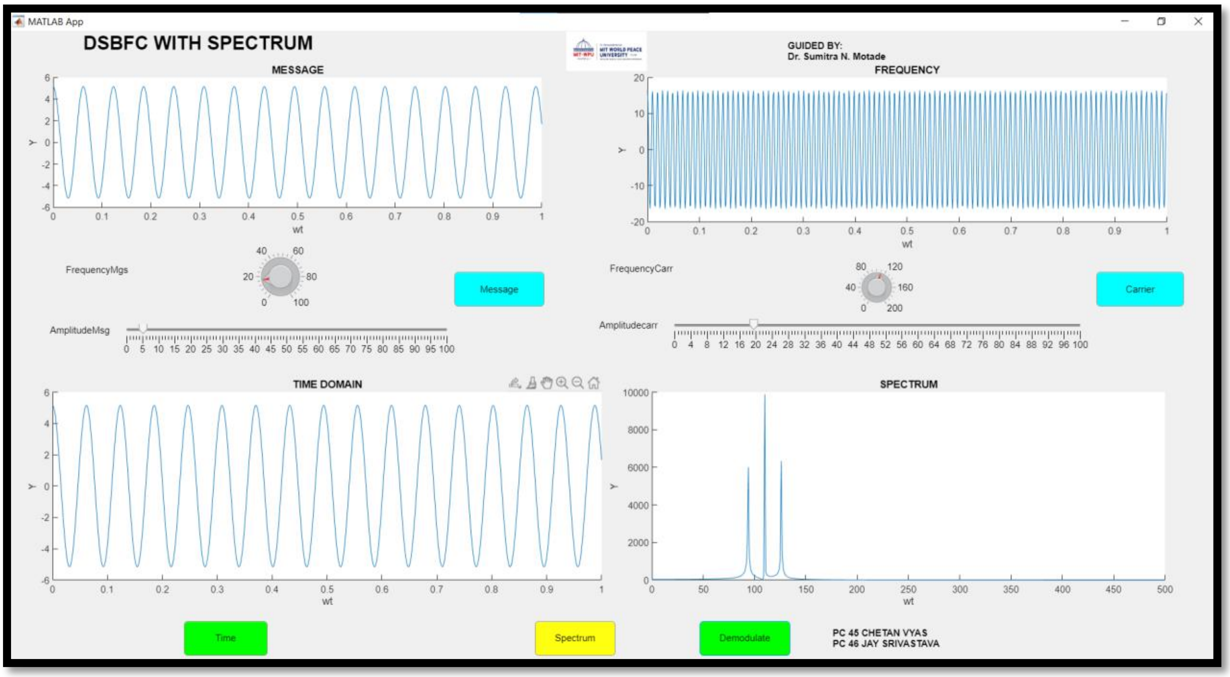


Figure 6

3. Program

```
classdef AMDSBFC < matlab.apps.AppBase


% Properties that correspond to app components
properties (Access = public)
UIFigure matlab.ui.Figure
AmplitudeMsgSliderLabel matlab.ui.control.Label
AmplitudeMsgSlider matlab.ui.control.Slider
AmplitudecarrSliderLabel matlab.ui.control.Label
AmplitudecarrSlider matlab.ui.control.Slider
MessageButton matlab.ui.control.Button
CarrierButton matlab.ui.control.Button
SpectrumButton matlab.ui.control.Button
TimeButton matlab.ui.control.Button
FrequencyCarrKnobLabel matlab.ui.control.Label
FrequencyCarrKnob matlab.ui.control.Knob
FrequencyMgsKnobLabel matlab.ui.control.Label
FrequencyMgsKnob matlab.ui.control.Knob
Image matlab.ui.control.Image
DemodulateButton matlab.ui.control.Button
DSBFCWITHSPECTRUMLabel matlab.ui.control.Label
PC45CHETANVYASPC46JAYSRIVASTAVALabel matlab.ui.control.Label
GUIDEDBYDrSumitraNMotadeLabel matlab.ui.control.Label
UIAxes matlab.ui.control.UIAxes
UIAxes2 matlab.ui.control.UIAxes
UIAxes3 matlab.ui.control.UIAxes
UIAxes4 matlab.ui.control.UIAxes
end
```

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


% Button pushed function: MessageButton
function MessageButtonPushed(app, event)
fm=app.FrequencyMgsKnob.Value;
vm=app.AmplitudeMsgSlider.Value;
time=0:1/1000:1;
v_m=vm*cos(2*pi*fm*time);
plot(app.UIAxes,time,v_m);
end


% Button pushed function: CarrierButton
function CarrierButtonPushed(app, event)
fc=app.FrequencyCarrKnob.Value;
vc=app.AmplitudecarrSlider.Value;
time=0:1/1000:1;
v_c=vc*cos(2*pi*fc*time);
plot(app.UIAxes2,time,v_c);
end


% Button pushed function: TimeButton
function TimeButtonPushed(app, event)
time=0:1/1000:1;
fm=app.FrequencyMgsKnob.Value;
vm=app.AmplitudeMsgSlider.Value;
v_m=vm*cos(2*pi*fm*time);
fc=app.FrequencyCarrKnob.Value;
vc=app.AmplitudecarrSlider.Value;
v_c=vc*cos(2*pi*fc*time);
v_am=v_m.*v_c;
plot(app.UIAxes3,time,v_am);
end


% Button pushed function: SpectrumButton
function SpectrumButtonPushed(app, event)
time=0:1/1000:1;
fm=app.FrequencyMgsKnob.Value;
vm=app.AmplitudeMsgSlider.Value;
v_m=vm*cos(2*pi*fm*time);
fc=app.FrequencyCarrKnob.Value;
vc=app.AmplitudecarrSlider.Value;
ma=vm/vc;
%v_am=v_m.*v_c;
v_am= vc*(1+ma*v_m).*sin(2*pi*fc*time);
N=length(time);
y=fft(v_am,N);
z=y(1:(floor(N/2)+1));
k=0:(floor(N/2));
plot(app.UIAxes4,k,abs(z));
```

```
end

% Button pushed function: DemodulateButton
function DemodulateButtonPushed(app, event)
fm=app.FrequencyMgsKnob.Value;
vm=app.AmplitudeMsgSlider.Value;
time=0:1/1000:1;
v_m=vm*cos(2*pi*fm*time);
plot(app.UIAxes3,time,v_m);
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.Color = [0.9412 0.9412 0.9412];
app.UIFigure.Position = [100 100 913 666];
app.UIFigure.Name = 'MATLAB App';

% Create AmplitudeMsgSliderLabel
app.AmplitudeMsgSliderLabel = uilabel(app.UIFigure);
app.AmplitudeMsgSliderLabel.HorizontalAlignment = 'right';
app.AmplitudeMsgSliderLabel.Position = [45 347 82 22];
app.AmplitudeMsgSliderLabel.Text = 'AmplitudeMsg';

% Create AmplitudeMsgSlider
app.AmplitudeMsgSlider = uislider(app.UIFigure);
app.AmplitudeMsgSlider.Position = [148 356 150 3];

% Create AmplitudecarrSliderLabel
app.AmplitudecarrSliderLabel = uilabel(app.UIFigure);
app.AmplitudecarrSliderLabel.HorizontalAlignment = 'right';
app.AmplitudecarrSliderLabel.Position = [488 353 79 22];
app.AmplitudecarrSliderLabel.Text = 'Amplitudecarr';

% Create AmplitudecarrSlider
app.AmplitudecarrSlider = uislider(app.UIFigure);
app.AmplitudecarrSlider.Position = [588 362 150 3];

% Create MessageButton
```

```
app.MessageButton = uibutton(app.UIFigure, 'push');
app.MessageButton.ButtonPushedFcn = createCallbackFcn(app,
@MessageButtonPushed, true);
app.MessageButton.BackgroundColor = [0 1 1];
app.MessageButton.Position = [308 374 114 44];
app.MessageButton.Text = 'Message';


% Create CarrierButton
app.CarrierButton = uibutton(app.UIFigure, 'push');
app.CarrierButton.ButtonPushedFcn = createCallbackFcn(app,
@CarrierButtonPushed, true);
app.CarrierButton.BackgroundColor = [0 1 1];
app.CarrierButton.Position = [759 374 111 44];
app.CarrierButton.Text = 'Carrier';


% Create SpectrumButton
app.SpectrumButton = uibutton(app.UIFigure, 'push');
app.SpectrumButton.ButtonPushedFcn = createCallbackFcn(app,
@SpectrumButtonPushed, true);
app.SpectrumButton.BackgroundColor = [1 1 0.0667];
app.SpectrumButton.Position = [366 9 103 44];
app.SpectrumButton.Text = 'Spectrum';


% Create TimeButton
app.TimeButton = uibutton(app.UIFigure, 'push');
app.TimeButton.ButtonPushedFcn = createCallbackFcn(app, @TimeButtonPushed,
true);
app.TimeButton.BackgroundColor = [0 1 0];
app.TimeButton.Position = [221 9 106 44];
app.TimeButton.Text = 'Time';


% Create FrequencyCarrKnobLabel
app.FrequencyCarrKnobLabel = uilabel(app.UIFigure);
app.FrequencyCarrKnobLabel.HorizontalAlignment = 'center';
app.FrequencyCarrKnobLabel.Position = [503 408 86 22];
app.FrequencyCarrKnobLabel.Text = {'FrequencyCarr'; ''};


% Create FrequencyCarrKnob
app.FrequencyCarrKnob = uiknob(app.UIFigure, 'continuous');
app.FrequencyCarrKnob.Limits = [0 200];
app.FrequencyCarrKnob.Position = [636 388 30 30];


% Create FrequencyMgsKnobLabel
app.FrequencyMgsKnobLabel = uilabel(app.UIFigure);
app.FrequencyMgsKnobLabel.HorizontalAlignment = 'center';
app.FrequencyMgsKnobLabel.Position = [68 407 85 22];
app.FrequencyMgsKnobLabel.Text = {'FrequencyMgs'; ''};
```

```
% Create FrequencyMgsKnob
app.FrequencyMgsKnob = uiknob(app.UIFigure, 'continuous');
app.FrequencyMgsKnob.Position = [184 396 38 38];


% Create Image
app.Image = uiimage(app.UIFigure);
app.Image.Position = [366 617 183 50];
app.Image.ImageSource = 'MITLOGO.png';


% Create DemodulateButton
app.DemodulateButton = uibutton(app.UIFigure, 'push');
app.DemodulateButton.ButtonPushedFcn = createCallbackFcn(app,
@DemodulateButtonPushed, true);
app.DemodulateButton.BackgroundColor = [0 1 0];
app.DemodulateButton.Position = [576 9 115 44];
app.DemodulateButton.Text = 'Demodulate';


% Create DSBFCWITHSPECTRUMLabel
app.DSBFCWITHSPECTRUMLabel = uilabel(app.UIFigure);
app.DSBFCWITHSPECTRUMLabel.FontSize = 24;
app.DSBFCWITHSPECTRUMLabel.FontWeight = 'bold';
app.DSBFCWITHSPECTRUMLabel.Position = [93 638 308 29];
app.DSBFCWITHSPECTRUMLabel.Text = 'DSBFC WITH SPECTRUM ';


% Create PC45CHETANVYASPC46JAYSRIVASTAVALabel
app.PC45CHETANVYASPC46JAYSRIVASTAVALabel = uilabel(app.UIFigure);
app.PC45CHETANVYASPC46JAYSRIVASTAVALabel.FontWeight = 'bold';
app.PC45CHETANVYASPC46JAYSRIVASTAVALabel.Position = [745 18 142 27];
app.PC45CHETANVYASPC46JAYSRIVASTAVALabel.Text = {'PC 45 CHETAN VYAS'; 'PC 46
JAY SRIVASTAVA'};


            % Create GUIDEDBYDrSumitraNMotadeLabel
            app.GUIDEDBYDrSumitraNMotadeLabel = uilabel(app.UIFigure);
            app.GUIDEDBYDrSumitraNMotadeLabel.FontWeight = 'bold';
            app.GUIDEDBYDrSumitraNMotadeLabel.Position = [527 623 271 38];
            app.GUIDEDBYDrSumitraNMotadeLabel.Text = {'GUIDED BY:'; 'Dr.
Sumitra N. Motade'};


            % Create UIAxes
            app.UIAxes = uiaxes(app.UIFigure);
            title(app.UIAxes, 'MESSAGE')
            xlabel(app.UIAxes, 'wt')
            ylabel(app.UIAxes, 'Y')
            zlabel(app.UIAxes, 'Z')
            app.UIAxes.Position = [21 457 363 170];


            % Create UIAxes2
```

```
            app.UIAxes2 = uiaxes(app.UIFigure);
            title(app.UIAxes2, 'FREQUENCY')
            xlabel(app.UIAxes2, 'wt')
            ylabel(app.UIAxes2, 'Y')
            zlabel(app.UIAxes2, 'Z')
            app.UIAxes2.Position = [469 442 388 185];


            % Create UIAxes3
            app.UIAxes3 = uiaxes(app.UIFigure);
            title(app.UIAxes3, 'TIME DOMAIN')
            xlabel(app.UIAxes3, 'wt')
            ylabel(app.UIAxes3, 'Y')
            zlabel(app.UIAxes3, 'Z')
            app.UIAxes3.Position = [20 70 425 228];


            % Create UIAxes4
            app.UIAxes4 = uiaxes(app.UIFigure);
            title(app.UIAxes4, 'SPECTRUM')
            xlabel(app.UIAxes4, 'wt')
            ylabel(app.UIAxes4, 'Y')
            zlabel(app.UIAxes4, 'Z')
            app.UIAxes4.Position = [444 70 413 228];


            % 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 = AMDSBFC


            % Create UIFigure and components
            createComponents(app)


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


            if nargout == 0
                clear app
            end
        end
```

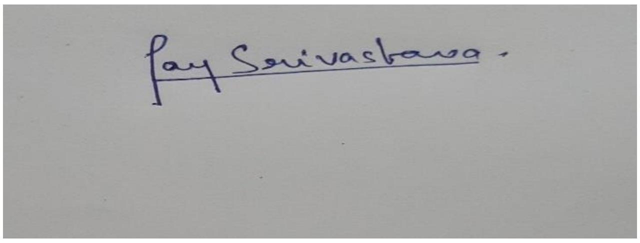
```
        % Code that executes before app deletion
        function delete(app)

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

References:

- http://zimmer.fresnostate.edu/~pkinman/pdfs/DSB%20and%20AM.pdf
- https://fas.org/man/dod-101/navy/docs/es310/AM.htm
- https://www.watelectronics.com/what-is-amplitude-modulation-derivatives-typesandbenefits/#:~:text=Amplitude%20Modulation%20Block%20Diagram&text=These%20are%20also%20called%20as,modulated%20with%20the%20carrier%20signals.&text=These%20two%20signals%20are%20combined,amplitude%20of%20the%20carrier%20signal.
- https://www.tutorialspoint.com/analog_communication/analog_communicationm_modulators.htm
- http://ecelabs.njit.edu/ece489v2/Lab1.php

Signature of Students

1. **Pc 46 JAY SRIVASTAVA**



2. **PC 45 CHETAN VYAS**