EE450 - Electronic Control of Motors

Extra Credit - PWM Simulation (using Example 8.8 for values)

MATLAB code and report developed by Paco Ellaga

Student ID: 009602331

Course Instructed by Dr. Wajdi Aghnatios

May 15, 2019

Abstract

In this lab, the simulation of pulse width modulation (PWM) in power electronics was explored. Here, PWM proved correlation to power electronics through its uses in analyzing power absorption and total harmonic distortion (THD). The THD is based on the load current of the system. So, the triangulation will be compared upon the sinusoid, based on it's harmonics and the power fluctuation supported through spectral density and spectrogram results. All results were completed in code, written through MATLAB 2018a and were based on example 8.8, *A PWM Inverter*, from the textbook, *Power Electronics*, by Daniel Hart .

Method

The example, 8.8: *A PWM Inverter*, is a problem asking for the calculations for a full bridge inverter given certain values. It does not ask for any MATLAB code to be written and provides a lot to explore in terms of PWM and its characteristic elements. It asks for:

- the amplitude of a 60-Hz component (fSin -- sinusoidal frequency) of the output voltage and load current,
- the power absorbed by the load resistor,
- and the total harmonic distortion (THD) of the load current.

The preset given values were :

- For the circuit to be using bipolar pwm,
- DC voltage input to be 100v,
- the amplitude modulation ratio, ***mAmp***, to be 0.8,
- the frequency modulation ratio, ***mFreq***, to be 21,
- where the triangular frequency will be 1260Hz due to fSin*mfreq [60*21],
- and the resistive load to be RLoad = 10Ω and the inductive load to be LLoad = 20mH.

To start of the problem, all the preset values were placed independently in terms of continual development (shown in Base Variables). A possibility for phase angles were applied and were set to zero since the values were not specified. Next was to coordinate the modulated ratios of the frequency and the amplitudes of both the sinusoid (message signal) and the triangulation signal. From here, by coordinating the values, the user will be able to properly tell when the switching occurs between Vsine and Vtri, to eventually plot the PWM. For this scenario, the ***mFreq*** was already given, but the formula follows that fTri / fSin and the ***mAmp*** follows that aSin / aTri. For 21 to be the target harmonic of 1260Hz, and the 60Hz to be the 1st, the width needs relative amplitude and frequencies to properly align itself and coordinate across a varying signal. The ***mAmp***, follows a strict table as shown below:

Table 8-3 Normalized Fourier Coefficients V_n/V_{dc} for Bipolar PWM

	m_a=1	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
$n=1$	1.00	0.90	0.80	0.70	0.60	0.50	0.40	0.30	0.20	0.10
$n=m_f$	0.60	0.71	0.82	0.92	1.01	1.08	1.15	1.20	1.24	1.27
$n=mf\pm2$	0.32	0.27	0.22	0.17	0.13	0.09	0.06	0.03	0.02	0.00

Figure 1: *Detailed graph of the* ***mAmp*** *where it equals Vn/Vdc. This table was not coded, but column 0.8 was used from 27-30, where* ***mFreq*** *= 21, n == harmonics.*

From here, the amplitude of the from the fundamental frequency of 0.8*100 is produced to find the current amplitude for the 60Hz component. By following equation 8-41 this can be achieved as shown:

$$I_n = \frac{V_n}{Z_n} = \frac{V_n}{\sqrt{R^2 + (n\omega_0 L)^2}} \qquad (8\text{-}41)$$

Figure 2: *The given harmonic equation for the current amplitude, where n == harmonic.*

And as result of following this, the following result comes out and shows to coincide with the original calculations and values through here:

```
40
41 -    I_1 = Vout_1/sqrt((RLoad^2)+(1*2*pi*fSin*LLoad)^2);                  % Current amplitude from 1st harmonic
42 -    I_mFreq = Vout_mFreq/sqrt((RLoad^2)+(mFreq*2*pi*fSin*LLoad)^2);      % From mFreq value
```

Command Window

```
I_1 =

    6.3878
```

Figure 3: *Snippet of the code where line 41 solves and produces the value for the current amplitude value of 6.3878.*

Following this, the current amplitudes for harmonics 21, 19, and 23 must be found to calculate power absorption. By just changing the n-values and recreating the same equations, the currents were fully calculated. They are found here:

```
41 -    I_1 = Vout_1/sqrt((RLoad^2)+(1*2*pi*fSin*LLoad)^2);                  % Current amplitude from 1st harmonic
42 -    I_mFreq = Vout_mFreq/sqrt((RLoad^2)+(mFreq*2*pi*fSin*LLoad)^2);      % From mFreq value
43 -    I_mFreqP2 = Vout_mFreq2/sqrt((RLoad^2)+((mFreqP2)*2*pi*fSin*LLoad)^2);  % From mFreq value + 2
44 -    I_mFreqM2 = Vout_mFreq2/sqrt((RLoad^2)+((mFreqM2)*2*pi*fSin*LLoad)^2);  % From mFreq value - 2
45
```

Command Window

```
>> I_mFreq, I_mFreqP2, I_mFreqM2

I_mFreq =

    0.5169


I_mFreqP2 =

    0.1267


I_mFreqM2 =

    0.1532
```

Figure 4: *The next section where the rest of the current amplitudes are found.*

By having this, the individual power is calculated by the relationship of P = (Irms^2)*R for every harmonic calculated within the modulated frequency range. This total calculation of the power verifies enough information to make a solid reference for the power absorption. And by following this notion, there will always be imperfection and statistical distortion across harmonics (total harmonic distortion, THD). By following the equation (equation 8-17), the square root of the sum of the harmonics squared all divided by the initial harmonic will give a proper percent ratio of how far off the equilibrium the harmonics will continue to develop. This information found in another snippet of the code below:

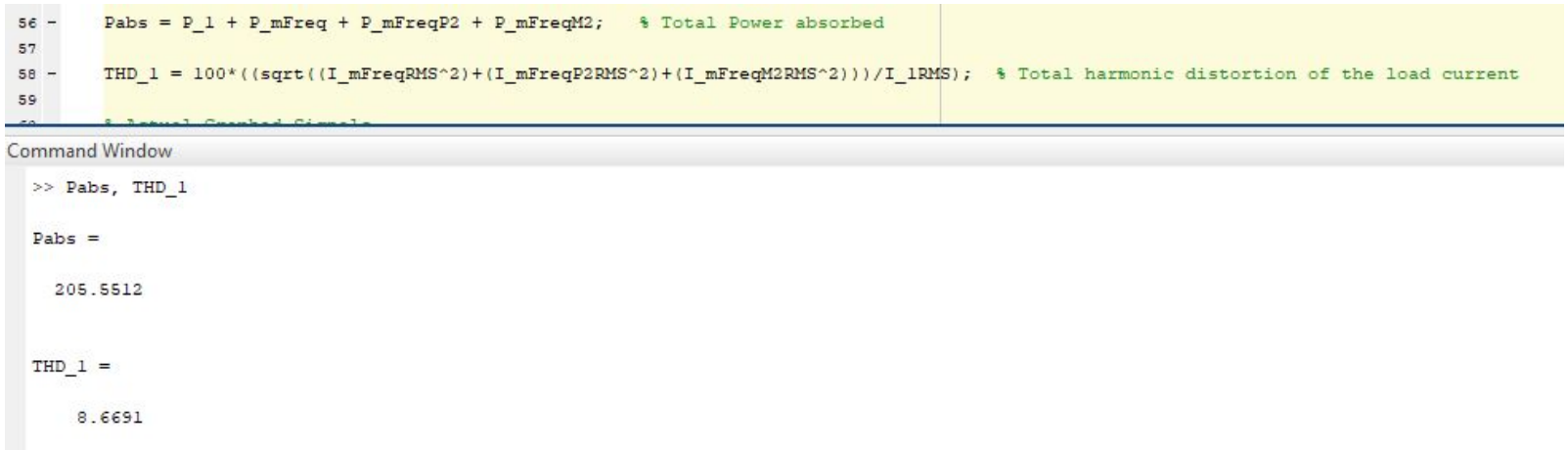

Figure 5: *Total absorption (**Pabs**) and the total harmonic distortion (**THD_1**) are displayed here.*

From this point, this fully completes the problem due to calculations, but to show a complete visual of what is happening in this problem by adding a few more lines of code, the actual pulse width modulation could be seen in relation to everything and more.

Results

For the final results of this procedure, the following will show how the sinusoid, triangulation, and PWM were plotted and found.

```
% Actual Graphed Signals
Tri = aTri.*sawtooth(2*pi*fTri*t+thetaTri);     % Triangular signal
Sin = aSin.*sin(2*pi*fSin*t+thetaSin);          % Sinusoidal signal
L = length(Tri);

% Simulation loop
for i = 1:L
    if (Sin(i) > Tri(i))
        PWM(i) = -1;     % if M value is greater than C return pulse
    else
        PWM(i) = 1;      % else return positive PWM signal
    end
end
```

Figure 6: Tri *is the triangular signal, the* ***Sin*** *is the sinusoidal (message signal), and the PWM is formulated by the switch construct scheme based on intersection of the* ***Tri*** *and the* ***Sin*** *signals and as defined in* ***mAmp*** *and* ***mFreq***.

In this following section of results, the first graph (***Figure 7***) is the direct visual of how the sinusoid and the triangular signals correlates together (top) create the PWM (bottom). The PWM is also matched with the triangular signal again to display proper time markers where the intersections line up with the sinusoid. As shown from a distance, the signal fully repeats after 0.1 seconds and is actually sloped in a continuous format versus having discrete impulses. By having a continuous measurement versus a discrete signal, the PWM will be more a continuous flowing value and essentially follow the signal more smoothly than fully jumping back from positive and negative (bipolar) values, possibly losing essential values of distortion.

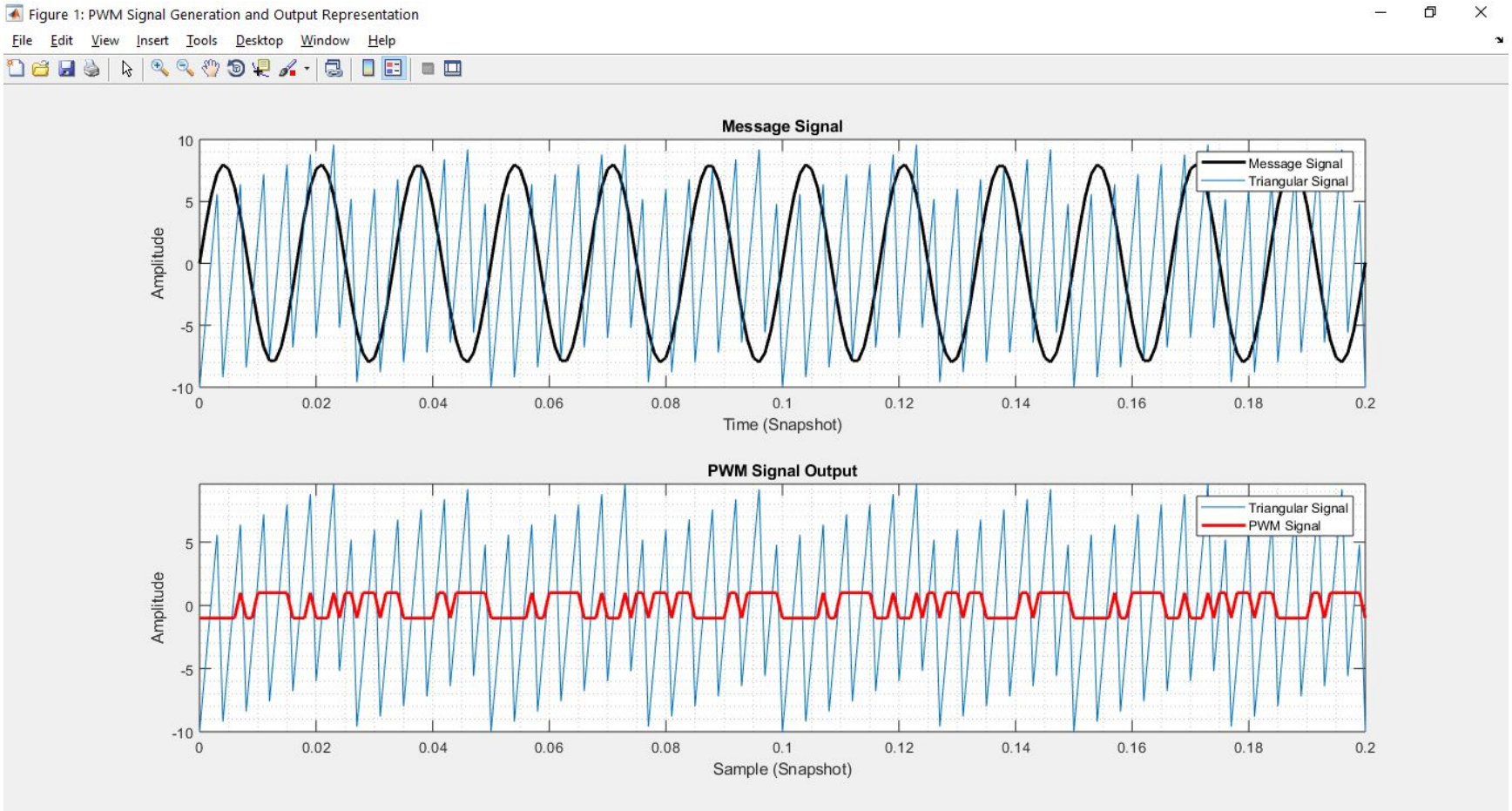

Figure 7: *The top graph is the sinusoid overlapped with the triangular and the bottom is overlapping the triangular with the PWM sloped impulses to create a bipolar PWM inverter of a RL load design.*

In addition just the PWM, it was also worthwhile to explore other representations and characteristics of the signal and its structure. This was taken advantage by taking the spectrogram. The spectrogram was evaluated on both the signal and it's PWM output. Since spectrograms measure a signal's intensity over across their projected time, it was interesting to what occurred... As shown below, the sinusoid actually will dissipate to practically a very minimal amount of energy, minus an unusually large spike in dB(rad/samples) around ~0.12 pi rad/sample across the normalized frequency. This is also seen in the PWM chart and seems to repeat itself only midway through on the triangular signal. As it repeats in the triangular, the PWM responds as it seems to echo it way till the end of the signal. The intensity is measured in terms of specifically where the triangular will fully reach the max and minimum limits of the measurable scale. It will appear completely dark for a split moment when it fully dips and will appear quite bright when the signal reaches the peak. As shown below, it is interesting that the pulse width modulation is patterned to repeat on the way of how the signal interacts on itself with applied modulation.

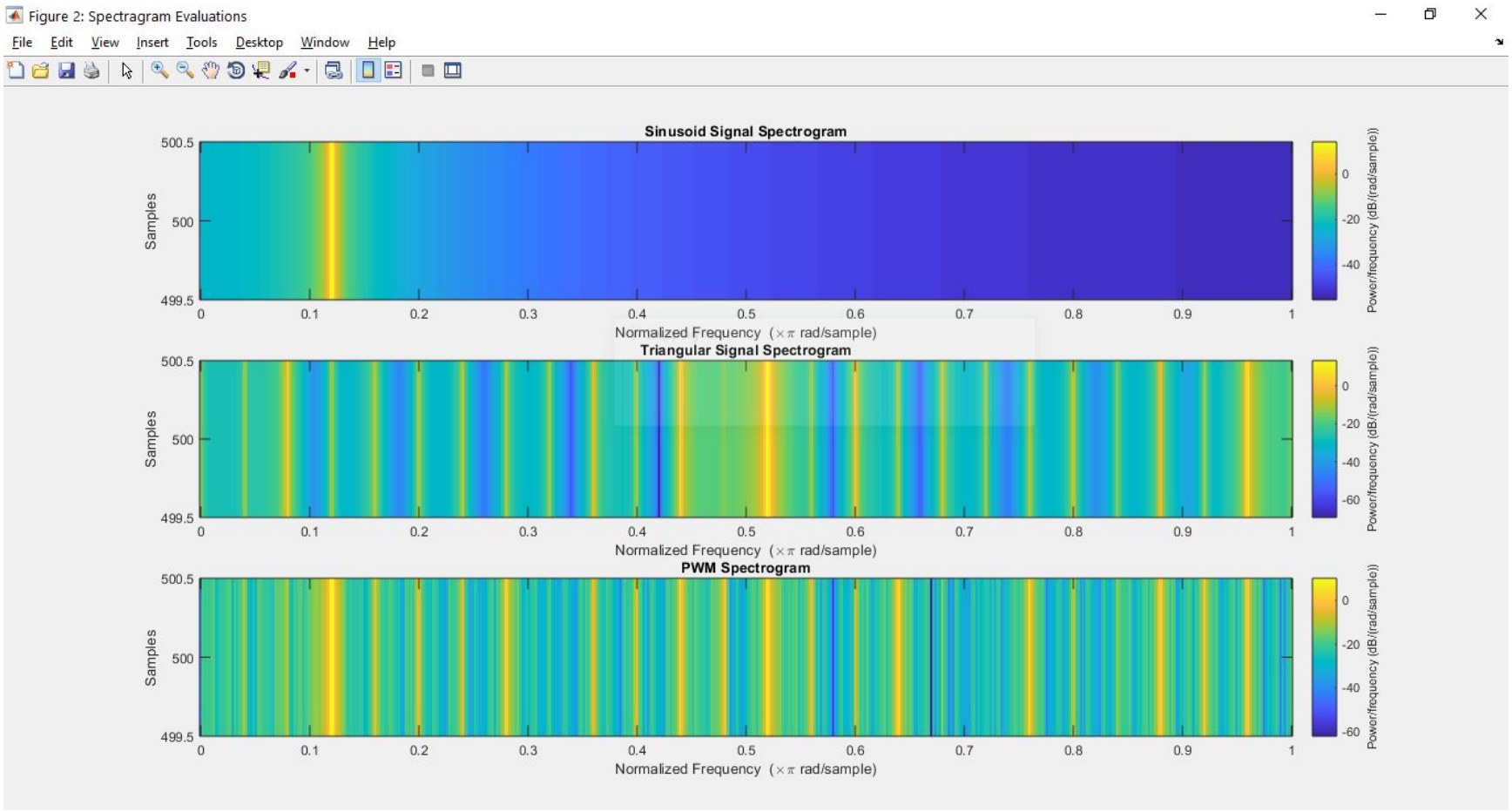

Figure 8: *Top graph represents the sinusoid, the middle graph is the triangular, and the bottom is the PWM output.*

Additionally, to really see the value in their amplitude level by using the command ***pwelch()***, the amplitude in the PWM output can accurately be displayed and analyzed. This command is the Welch's power spectral density estimate, where the distribution of power can be analyzed. This spectrum really an interesting insight of how the power was actually dispersed and eventually nullified as the spikes from ***Figure 7 & 8*** can be estimated in terms of amplitude. From the ~0.12pi rad/sample portion of the spectrogram, the PSD now shows to be actually a very significant large portion of the power distribution of the signal. At the point of repetition, it is shown to be about ~70 frequency samples in, where the power has dropped by almost 75%. As per exact values, these were the following for the highest and lowest point where the PWM achieved an amplitude to be processed:

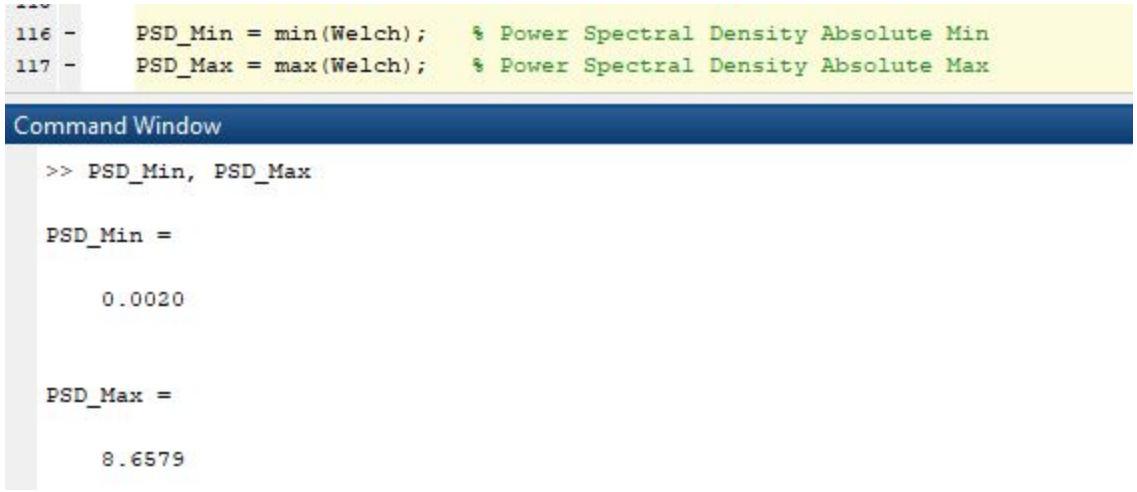

Figure 9: *The most minimum amplitude for an intersection of the* ***Sin*** *and the* ***Tri*** *to occur was at an amplitude 0.002 and the highest was at 8.6579.*

As shown in the final figure below, it is clearly illustrated how it fully developed. Where the signal is repeated halfway through such significant power intensity loss, it is also reflective of the spectrogram represented the power dissipation of the sinusoid.

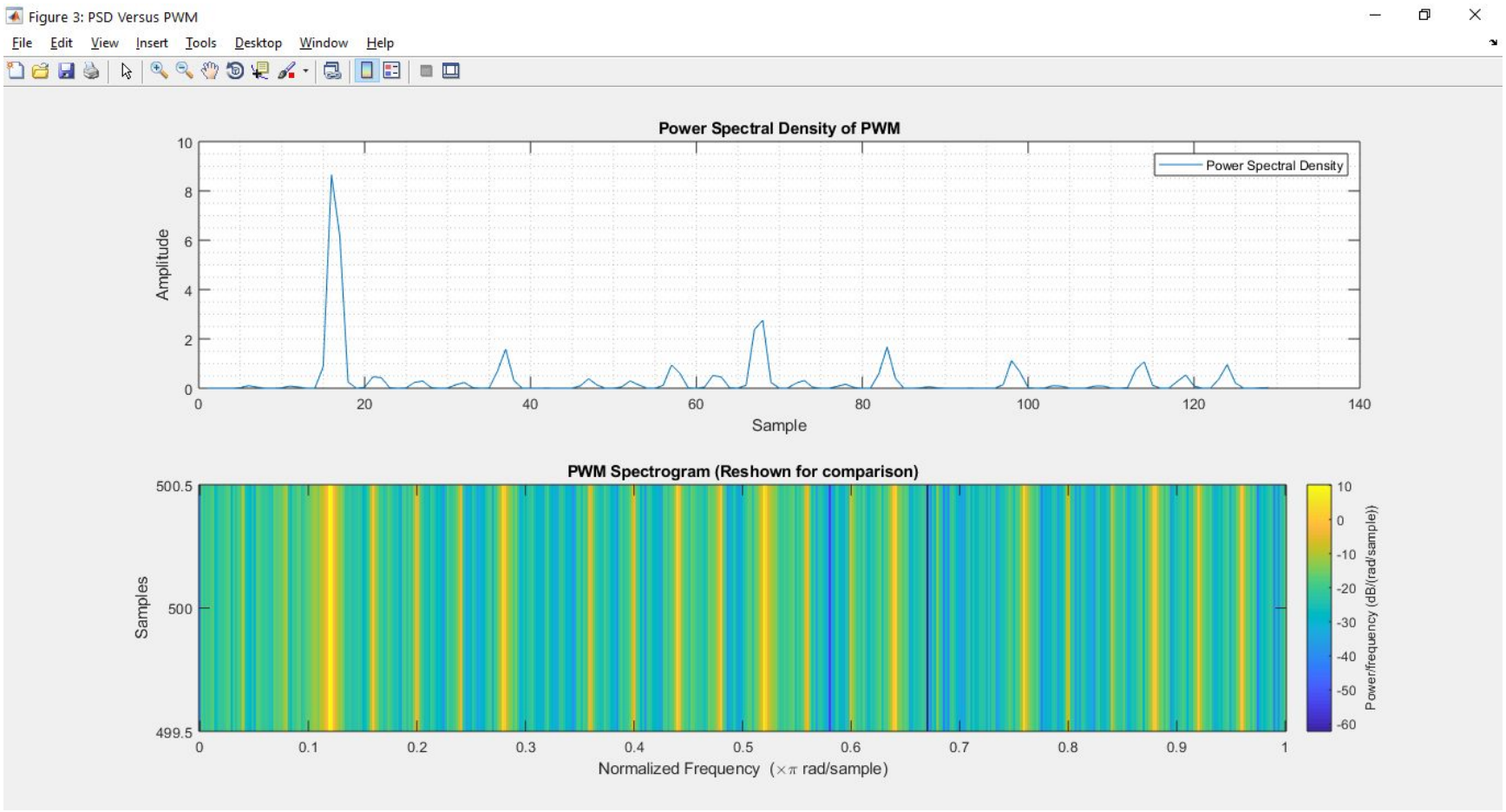

Figure 10: *The top is the power spectral density and the bottom is the direct comparison to the PWM's spectrogram*

Conclusion

In conclusion, the code could essentially be optimized, there are many variables and even a table missing of the normalized fourier coefficients from ***Figure 1***. This doesn't mean it was all for a loss, as the code showed more than the what the problem initially offered, by viewing the signal physically in signal message, spectrogram, and power spectral density format. The problem does however show a significant amount of information that proves to be essential in covering the topic of RL loaded inverters to have PWM measurement applied. The code is mostly designed to follow the problem, but if one were to re-assign values, it could essentially be used again plot various signals. By simulating the PWM in this example, it provided an unique way of approaching values for and revolving around power electronics.

MATLAB Code

```
%% EE450 - Pulse Width Modulation (PWM) Simulation
% MATLAB Code developed by Paco Ellaga
% SID: 009602331

clc, clear all, close all;

% Base Variables (temp values, change if needed)
fs = 1000;          % sampling frequency
dt = 1/fs;          % differential of time
t = 0:dt:1-dt;      % time interval and spacing

Vdc = 100;          % Source Voltage
RLoad = 10;         % Resistor Load
LLoad = 20e-3;      % Inductor Load

fSin = 60;          % Sinusoidal frequency (1st amplitude harmonic)
aSin = 8;           % amplitude of Sinusoidal (less than or equal to value of aTri)
aTri = 10;          % amplitude of Triangular
thetaTri = 0;       % Triangular phase delay in terms of pi
thetaSin = 0;       % Sinusoid phase delay in terms of pi

% Signal modulation ratios (temp values, change if needed)
mFreq = 21;             % fTri/fSin;  % frequency modulation ratio and target harmonic
mFreqP2 = mFreq + 2;    % harmonic + 2
```

```
mFreqM2 = mFreq - 2;    % harmonic - 2

mAmp = aSin/aTri;       % amplitude modulation ratio
% Following 2 variables can be plugged by matrix
mAmp_mFreq = 0.82;      % Note: Just plugged these from table 8-3
mAmp_mFreq2 = 0.22;     % I can't do this anymore, will come back later

fTri = mFreq*fSin;      % Triangular modulation frequency (amplitude harmonic)
fTriP2 = mFreqP2*fSin;  % harmonic + 2
fTriM2 = mFreqM2*fSin;  % harmonic - 2

% Signal Outputs
Vout_1 = mAmp*Vdc;              % Output voltage of the fundamental frequency from 1st harmonic
Vout_mFreq = mAmp_mFreq*Vdc;    % From target harmonic
Vout_mFreq2 = mAmp_mFreq2*Vdc;  % From target harmonic +- 2

I_1 = Vout_1/sqrt((RLoad^2)+(1*2*pi*fSin*LLoad)^2);                  % Current amplitude from 1st harmonic
I_mFreq = Vout_mFreq/sqrt((RLoad^2)+(mFreq*2*pi*fSin*LLoad)^2);      % From mFreq value
I_mFreqP2 = Vout_mFreq2/sqrt((RLoad^2)+((mFreqP2)*2*pi*fSin*LLoad)^2);  % From mFreq value + 2
I_mFreqM2 = Vout_mFreq2/sqrt((RLoad^2)+((mFreqM2)*2*pi*fSin*LLoad)^2);  % From mFreq value - 2

I_1RMS = I_1/sqrt(2);                % Current RMS from 1st harmonic
I_mFreqRMS = I_mFreq/sqrt(2);        % From mFreq harmonic
I_mFreqP2RMS = I_mFreqP2/sqrt(2);    % From mFreq harmonic + 2
I_mFreqM2RMS = I_mFreqM2/sqrt(2);    % From mFreq harmonic - 2
```

```
P_1 = ((I_1RMS)^2)*RLoad;              % Power absorbed from 1st harmonic
P_mFreq = ((I_mFreqRMS)^2)*RLoad;      % From mFreq harmonic
P_mFreqP2 = ((I_mFreqP2RMS)^2)*RLoad;  % From mFreq harmonic + 2
P_mFreqM2 = ((I_mFreqM2RMS)^2)*RLoad;  % From mFreq harmonic - 2

Pabs = P_1 + P_mFreq + P_mFreqP2 + P_mFreqM2;  % Total Power absorbed

THD_1 = 100*((sqrt((I_mFreqRMS^2)+(I_mFreqP2RMS^2)+(I_mFreqM2RMS^2)))/I_1RMS);  % Total harmonic distortion of the load current

% Actual Graphed Signals
Tri = aTri.*sawtooth(2*pi*fTri*t+thetaTri);     % Triangular signal
Sin = aSin.*sin(2*pi*fSin*t+thetaSin);          % Sinusoidal signal
L = length(Tri);

% Simulation loop
for i = 1:L
   if (Sin(i) > Tri(i))
      PWM(i) = -1;    % if M value is greater than C return pulse
   else
      PWM(i) = 1;     % else return positive PWM signal
   end
end

figure('Name','PWM Signal Generation and Output Representation'),figure(1)
% Representation of the Sinusoidal Signal
subplot(2,1,1)
```

```
hold on, plot(t,Sin,'black','LineWidth',2)
plot(t,Tri), hold off
xlabel('Time (Snapshot)')
xlim([0 0.2])   % may be adjusted based on various input scenarios
ylabel('Amplitude'), % ylim([min(Sin) max(Sin)])
title('Message Signal'), legend('Message Signal','Triangular Signal')
grid minor, box on

% Representation of Triangular Signal
subplot(2,1,2)
hold on, plot(t,Tri)
plot(t,PWM,'red','LineWidth',2), hold off
xlabel('Sample (Snapshot)')
xlim([0 0.2])   % may be adjusted based on various input scenarios
ylabel('Amplitude'), ylim([min(Tri) max(Tri)])
title('PWM Signal Output'), legend('Triangular Signal','PWM Signal')
grid minor, box on

figure('Name','Spectragram Evaluations'),figure(2)
% Spectrogram representations
subplot(3,1,1)
spectrogram(t,Sin)                  % time versus Sinusoid
title('Sinusoid Signal Spectrogram')
subplot(3,1,2)
spectrogram(t,Tri)                  % time versus Triangular
title('Triangular Signal Spectrogram')
```

```
subplot(3,1,3)
spectrogram(t,PWM)                  % time versus PWM
title('PWM Spectrogram')

figure('Name','PSD Versus PWM'),figure(3)
% Power Spectral Density (PWelsh)
Welch = pwelch(PWM);    % allocation of Welch's PSD estimate
subplot(2,1,1)
plot(Welch)           % Spectrum of PWelch returned values
xlabel('Sample'), ylabel('Amplitude')
title('Power Spectral Density of PWM'), legend('Power Spectral Density')
grid minor, box on

PSD_Min = min(Welch);   % Power Spectral Density Absolute Min
PSD_Max = max(Welch);   % Power Spectral Density Absolute Max

subplot(2,1,2)
spectrogram(t,PWM)                  % time versus PWM
title('PWM Spectrogram (Re-shown for comparison)')
```