POWER ELECTRONICS

Project 2
Inverters Analysis

Vu Lo

Objectives

This project is to analyze and simulate the performance of a single-phase inverter and a 3-phase inverter. This project utilizes MATLAB as the main coding program to simulate the voltages and currents that should be observed from those inverters. With the implementation of Riemann sum, and Fourier series, this project calculates the approximation of the behavior of the components in the circuit of the inverters.

Analytical Process Setup

Fourier Series

To do the analysis as aforementioned, we set up a script in MATLAB that transforms the input function into a Fourier series function. This script is to take a function, with its fundamental period, a time array of where we want to observe the function's behavior, and the number of terms that we desire for the Fourier series, then transforms it into a Fourier series function, whilst outputting its coefficients as arrays, the average value of said function and the error between the real inputted function and the reconstructed function.

To proceed, the function header would be:

```
function[avg,ak,bk,rcon,err] = fourseries(t,x,T,N)
```

where, the input parameters are:

- t: the time array,
- x: the function, or the input waveform that needs to be analyzed,
- T: the fundamental period of said function,
- N: the number of terms of the Fourier series that is desired,

and the output parameters are:

- avg: the average value of the waveform,
- ak and bk: the arrays of coefficients of the waveform,
- rcon: the reconstructed waveform in Fourier series,
- err: the error percentage.

Fourier series transformation is calculated and transform using the concept:

$$x(t) = \frac{1}{2}a_0 + \sum_{k=1}^{\infty} a_k \cos(kt) + \sum_{k=1}^{\infty} b_k \sin(kt)$$

where,

$$a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} x(t)dt$$

$$a_k = \frac{1}{\pi}\int_{-\pi}^{\pi} x(t)\cos(kt)\,dt$$

$$b_k = \frac{1}{\pi}\int_{-\pi}^{\pi} x(t)\sin(kt)\,dt$$

Using Riemann sum with the appropriate number of subintervals, we can approximate the integrals above to calculate each of these components. The average value of the waveform would be the a_0 value of the waveform, rcon is calculated as shown in the function above, and err is calculated by taking the RMS value of the difference between the reconstructed waveform, rcon, and the input waveform, x. The script to process this is shown in the *fourseries.m* script in the MATLAB Scripts section below.

Fourier Series Setup Method Proven

To prove the functionality and the calculations of *fourseries.m* script is correct, *provefourseries.m* script was created. The *provefourseries.m* uses the waveform:

$$x(t) = 15 - 30\cos(2\pi 40t) + 20\sin(2\pi 40t) + 8\cos(2\pi 800t) + 2\sin(2\pi 1600) - 5\cos(2\pi 2000t)$$

As observation from the waveform above, the fundamental frequency is 40Hz, which makes its fundamental period to be $T = \frac{1}{40} sec$. Furthermore, we can quickly determine the Fourier series coefficients as $a_1 = -30, a_{20} = 8, a_{50} = -5, b_1 = 20, b_{40} = 2$, while the average value, avg or a_0, is 15. N = 50 was chosen so it is possible to observe a_{50}. After running the script, the coefficient values are obtained as follows:

Coefficient	Original	Approximated
a_1	-30	-30
b_1	20	20
a_{20}	8	8
b_{40}	2	2
a_{50}	-5	-5

Table 1. Coefficient Values of Waveform x(t) Original and Approximated by provefourseries.m script.

The average value of x(t) was also obtained as 15, which agrees to the mentioned theory. The error percentage, err, was calculated to be as little as 4.5961×10^{-13}, almost to 0. The graph that shows the comparison between the original x(t) and the reconstructed waveform over five times the period is shown below:

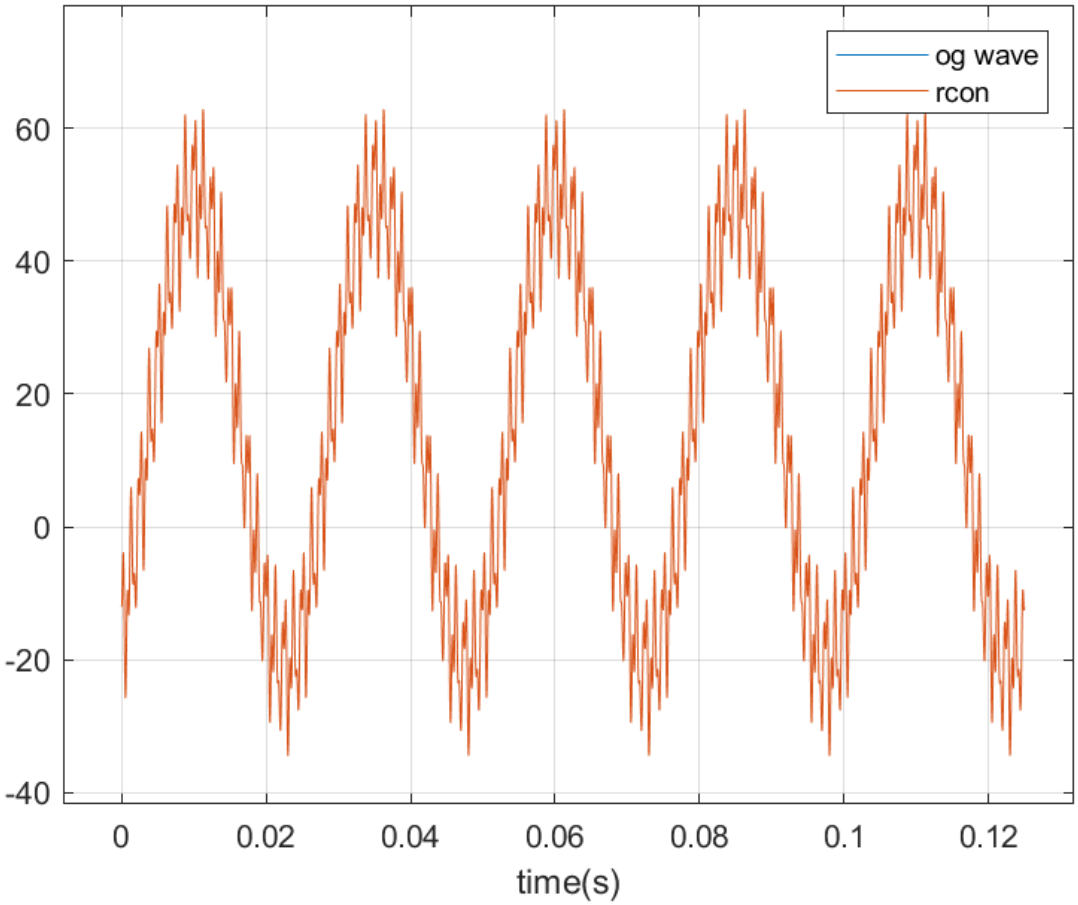

Figure 1. Original vs Reconstructed Waveform.

Therefore, from the results obtained above, we can see that *fourseries.m* script is proven to be working correctly, with very minute error. The script *provefourseries.m* is shown below in the MATLAB Scripts section.

Single-Phase Inverter Analysis

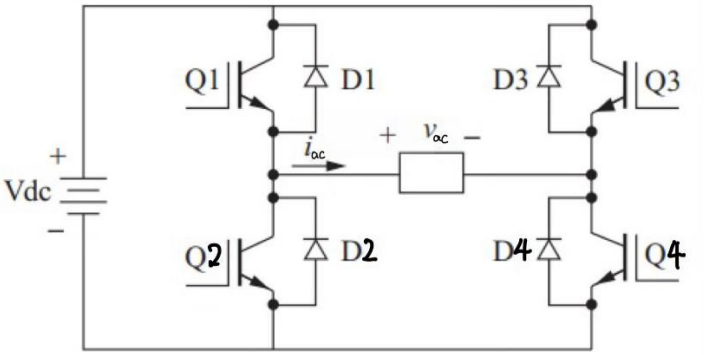

Figure 2. Single-Phase Inverter.

180 Switching Analysis

Assumptions and Specifications

The single-phase inverter to be analyzed is operating under 180 switching. The symbols Q1, Q2, Q3, Q4 represent the transistors T1, T2, T3, T4, respectively.

The load of this inverter contains a resistive load of $r = 0.5\Omega$ and an inductive load of $L = 1mH$. The desired fundamental component of AC voltage of said inverter is $V_{as} = -\frac{200}{\pi}\cos(\theta_{ac})$, where $\theta_{ac} = 120\pi t$.

Procedure

To analyze this inverter mathematically and logically, we need the below functions, which calculate θ_{ac}, ω_{ac}, and the fundamental frequency f, respectively:

$$\theta_{ac} = \omega_{ac}t,$$

$$\omega_{ac} = 2\pi f,$$

$$f = \frac{1}{T}$$

From these functions, we can form the relationship between θ_{ac}, ω_{ac}, and f as:

$$\theta_{ac} = \omega_{ac}t = 2\pi f t$$

We also made the assumption that the desired fundamental component of AC voltage is in the form of:

$$V_{as} = -\frac{200}{\pi}\cos(\theta_{ac}),$$

where $\theta_{ac} = 120\pi t$. Therefore, we can quickly calculate the fundamental frequency $f = 60Hz$, which gives the fundamental period $T = \frac{1}{60}sec$. Furthermore, for the sake of our calculations, we use trigonometric identity of:

$$\cos(\pi + a) = -\cos(a),$$

to eliminate the negative sign and turn the V_{as} equation into:

$$V_{as} = \frac{200}{\pi}\cos(\pi + \theta_{ac}),$$

Which we can use to calculate V_{dc}, using the equation of the relationship between V_{as} and V_{dc} as:

$$V_{as} = \frac{4}{\pi}V_{dc}$$

Which gives $V_{dc} = 50V$.

Analyzing the load, the AC Voltage function can be derived relating to the AC Current as:

$$V_{ac} = ri_{ac} + \frac{Ldi_{ac}}{dt},$$

In steady state, for single-phase inverter under 180 switching, the AC Current function can be described as:

$$i_o(t) = \begin{cases} \frac{V_{\text{dc}}}{R} + \left(I_{\text{min}} - \frac{V_{\text{dc}}}{R}\right)e^{-t/\tau} & \text{for } 0 < t < \frac{T}{2} \\ \frac{-V_{\text{dc}}}{R} + \left(I_{\text{max}} + \frac{V_{\text{dc}}}{R}\right) - e^{(t-T/2)/\tau} & \text{for } \frac{T}{2} < t < T \end{cases}$$

Where, when evaluating at $t = \frac{T}{2}$, and by symmetry, we can obtain that:

$$I_{max} = -I_{min} = \frac{V_{dc}}{R}\left(\frac{1 - e^{-\frac{T}{2\tau}}}{1 + e^{-\frac{T}{2\tau}}}\right),$$

Where $\tau = \frac{L}{r} = \frac{10^{-3}}{0.5} = 2 \times 10^{-3}s$, which returns the values of $I_{max} = 96.94A$, and $I_{min} = -I_{max} = -96.94A$.

Using the rules of 180 switching, we can graph the waveforms of the steady state AC current waveform, DC current waveform, the transistor and diode currents that we expect to observe over a single period versus θ_{ac} as below:

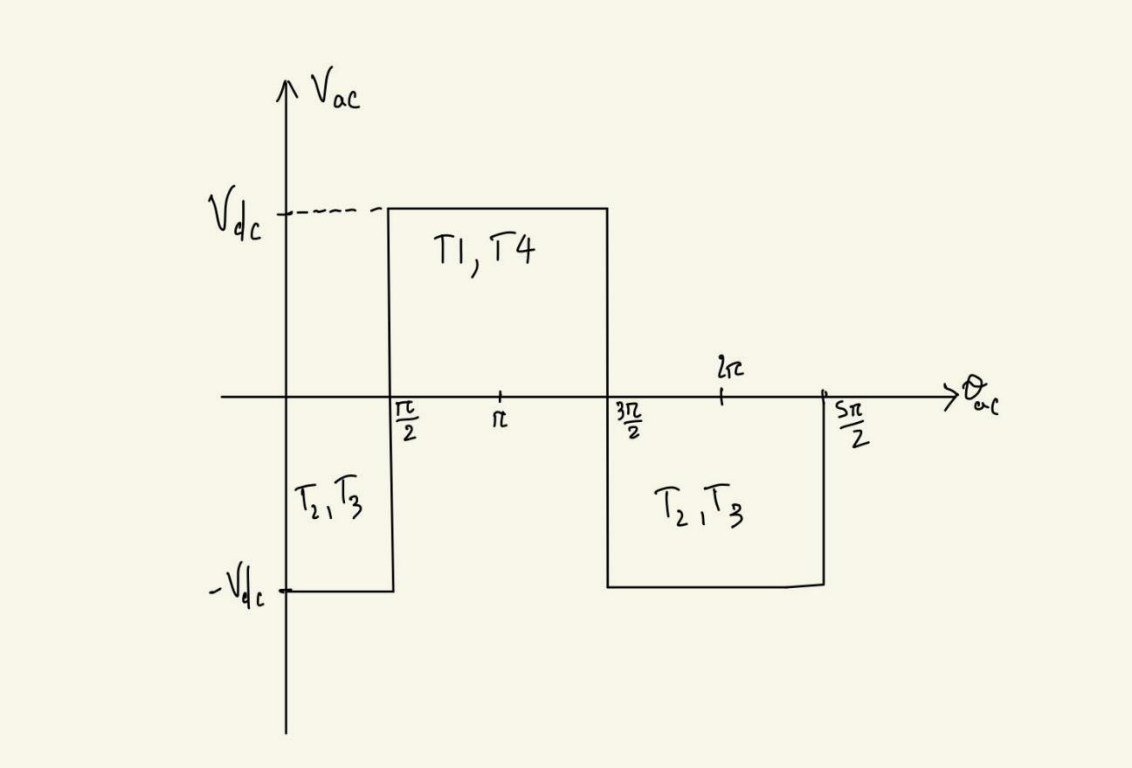

Figure 3. AC Voltage vs θ_{ac}.

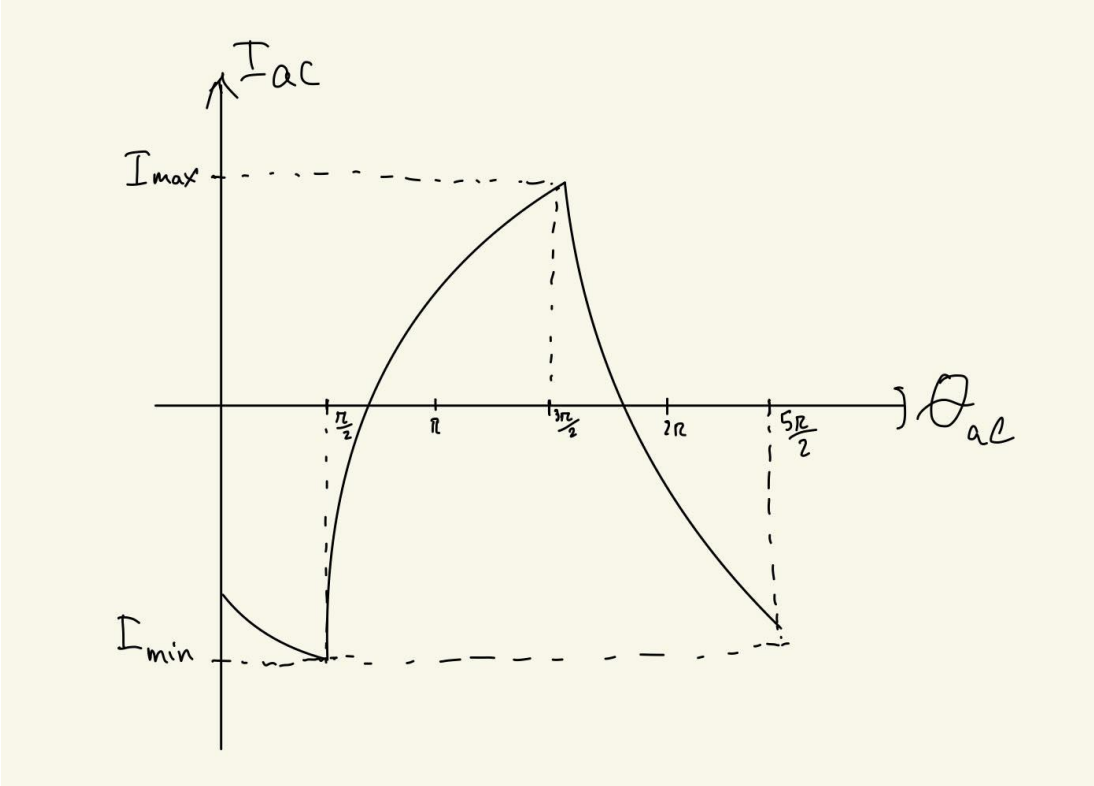

Figure 4. AC Curent vs θ_{ac}.

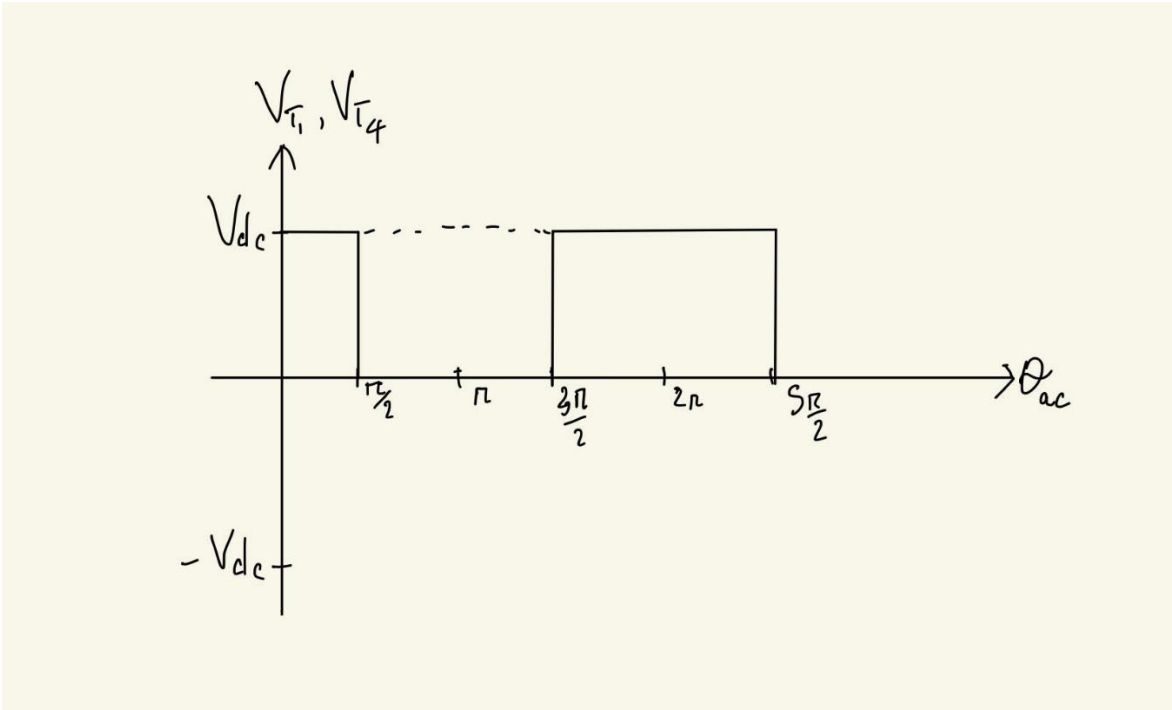

Figure 5. Transistor T1 and T4 Voltages vs θ_{ac}.

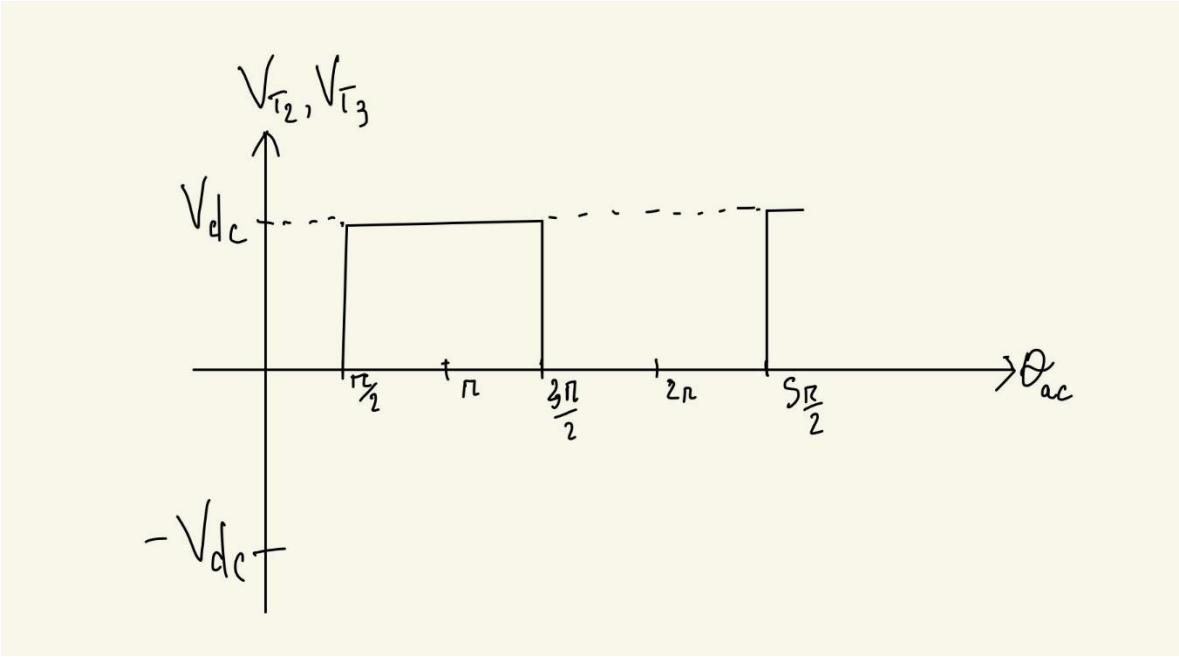

Figure 6. Transistor T2 and T3 Voltages vs θ_{ac}.

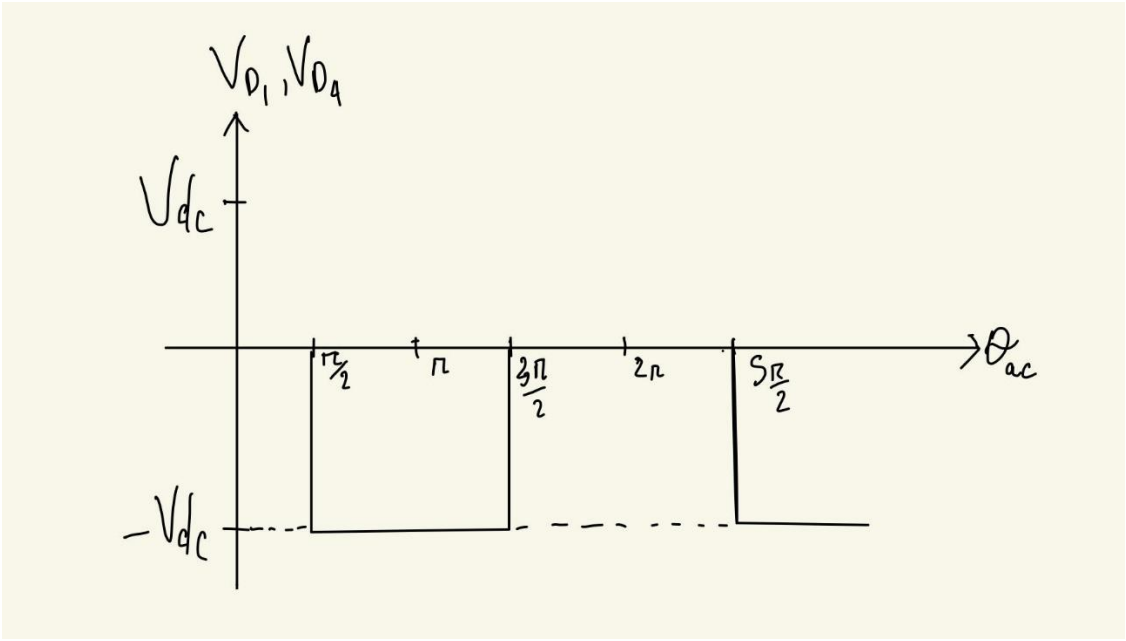

Figure 7. Diode D1 and D4 Voltages vs θ_{ac}.

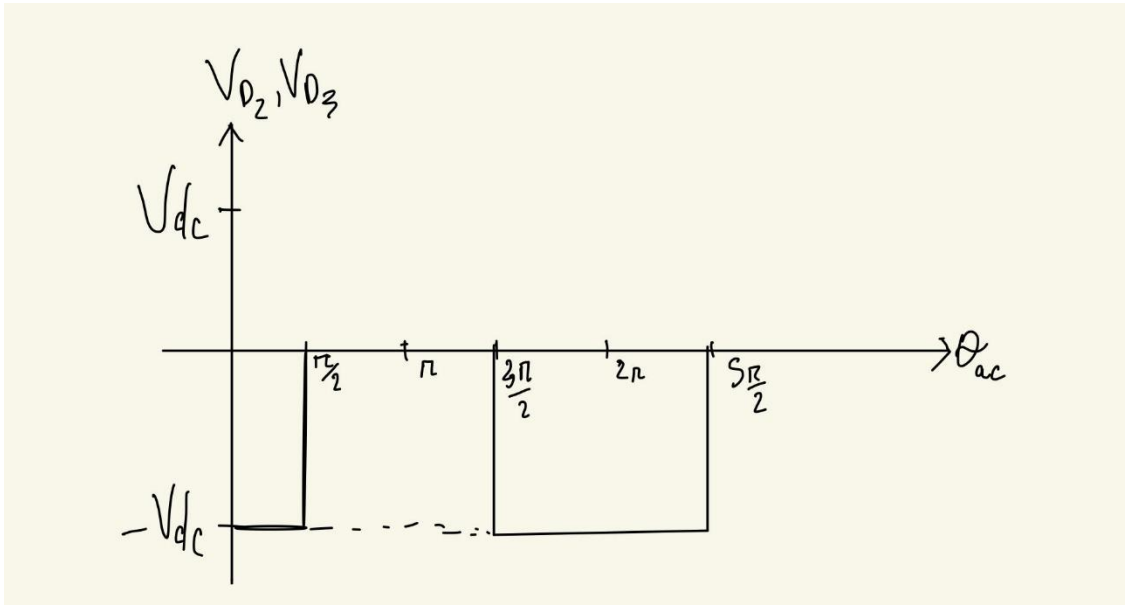

Figure 8. Diode D2 and D3 Voltages vs θ_{ac}.

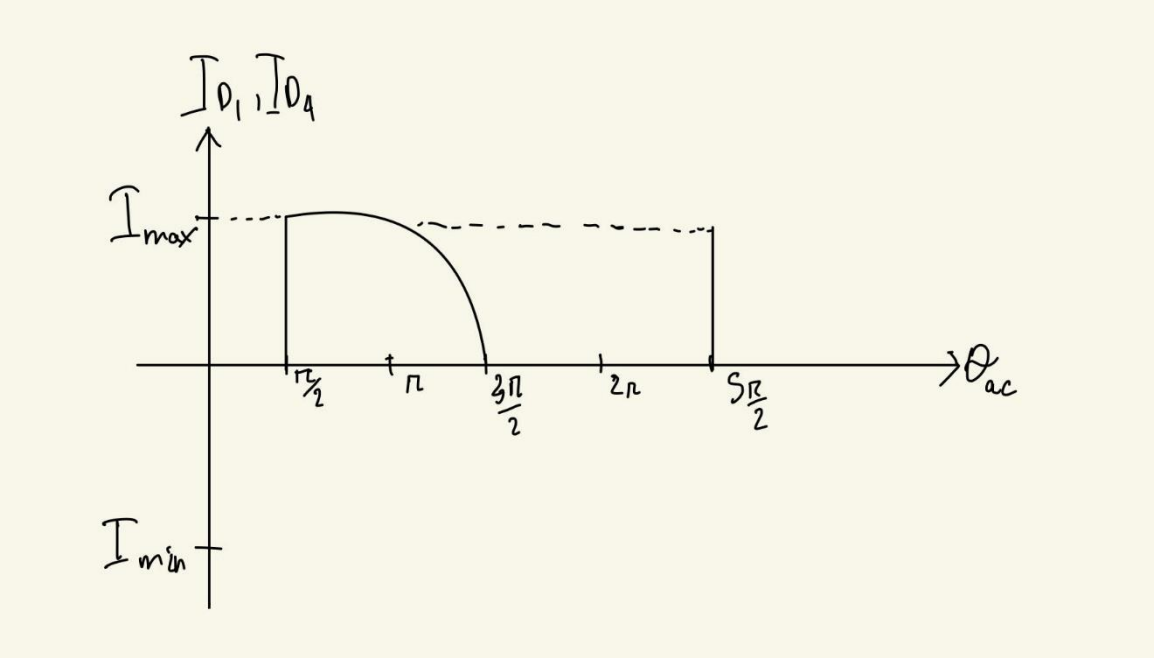

Figure 9. Diode D1 and D4 Currents vs θ_{ac}.

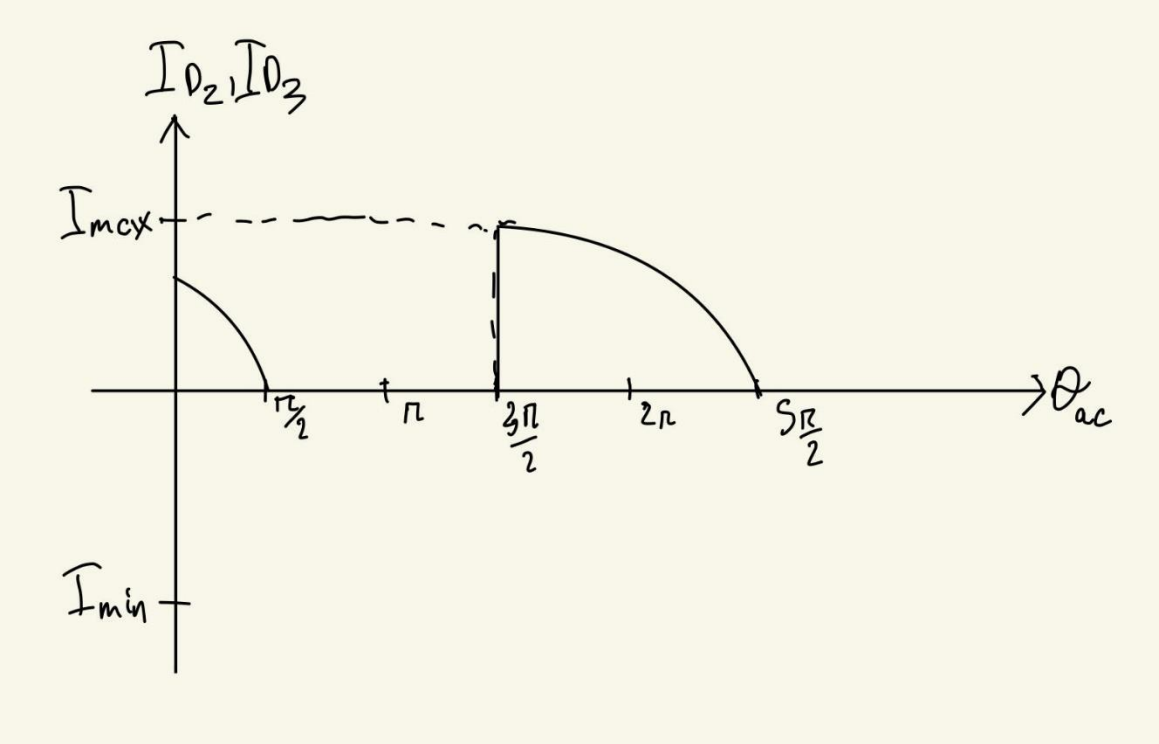

Figure 10. Diode D2 and D3 Currents vs θ_{ac}.

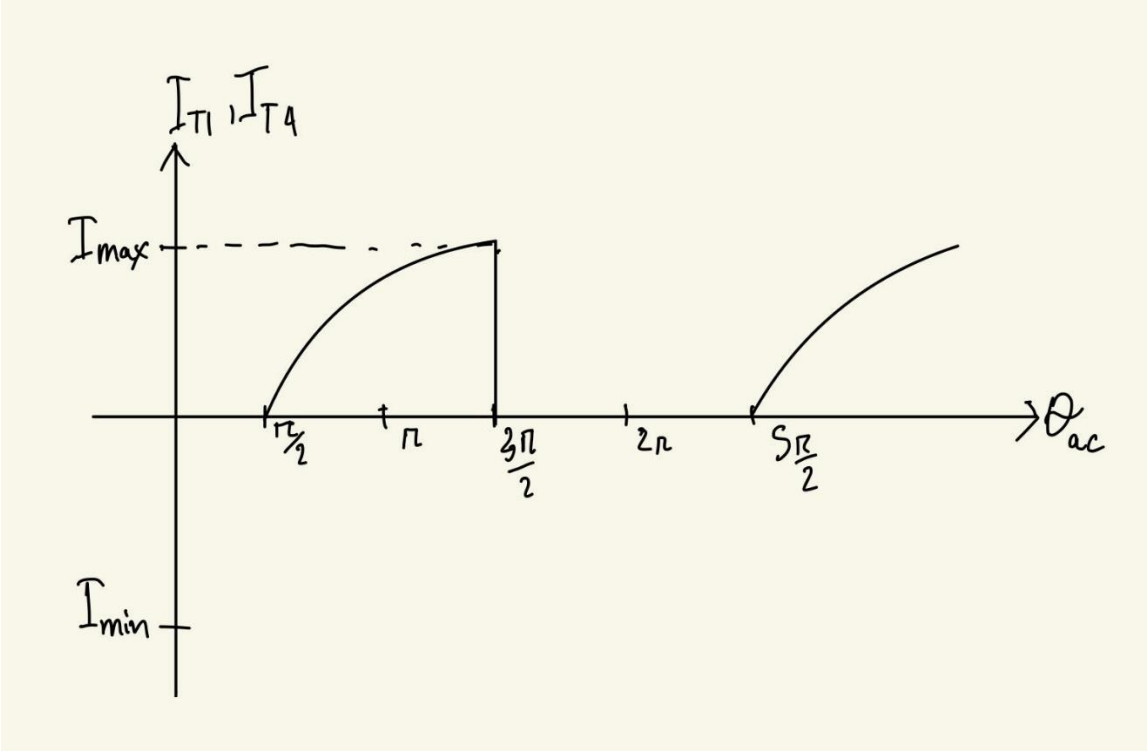

Figure 11. Transistor T1 and T4 Currents vs θ_{ac}.

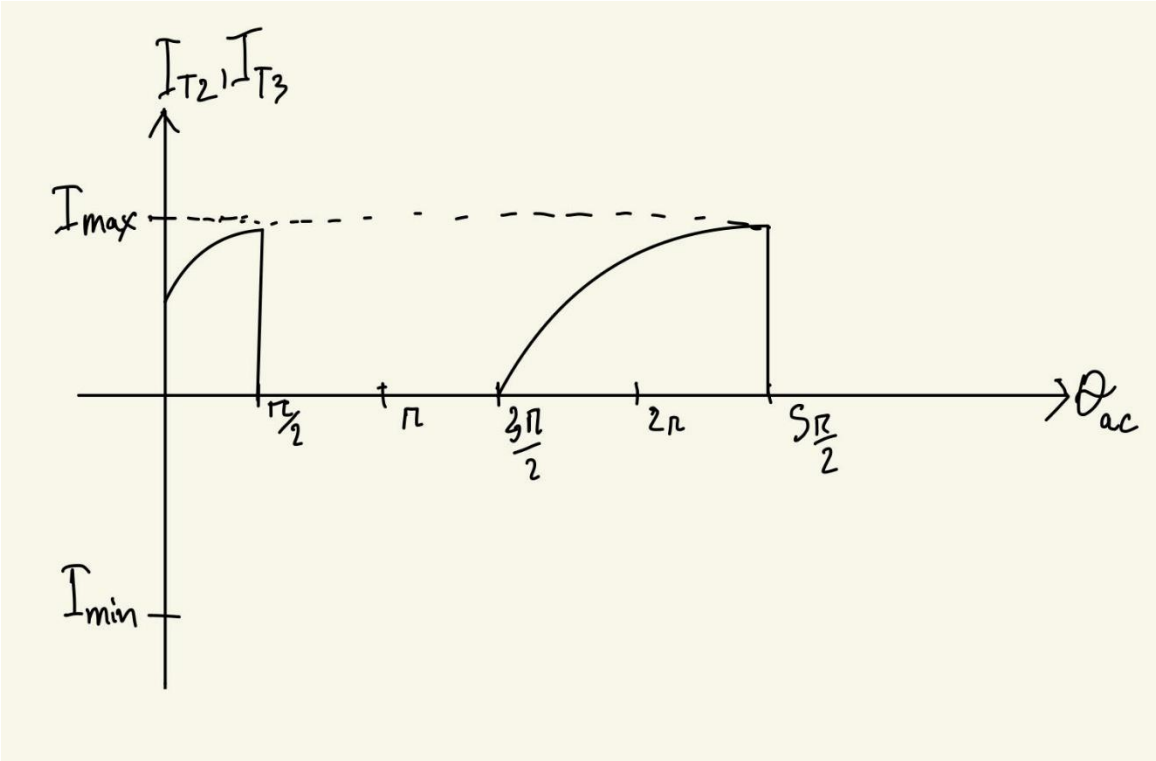

Figure 12. Transistor T2 and T3 Currents vs θ_{ac}.

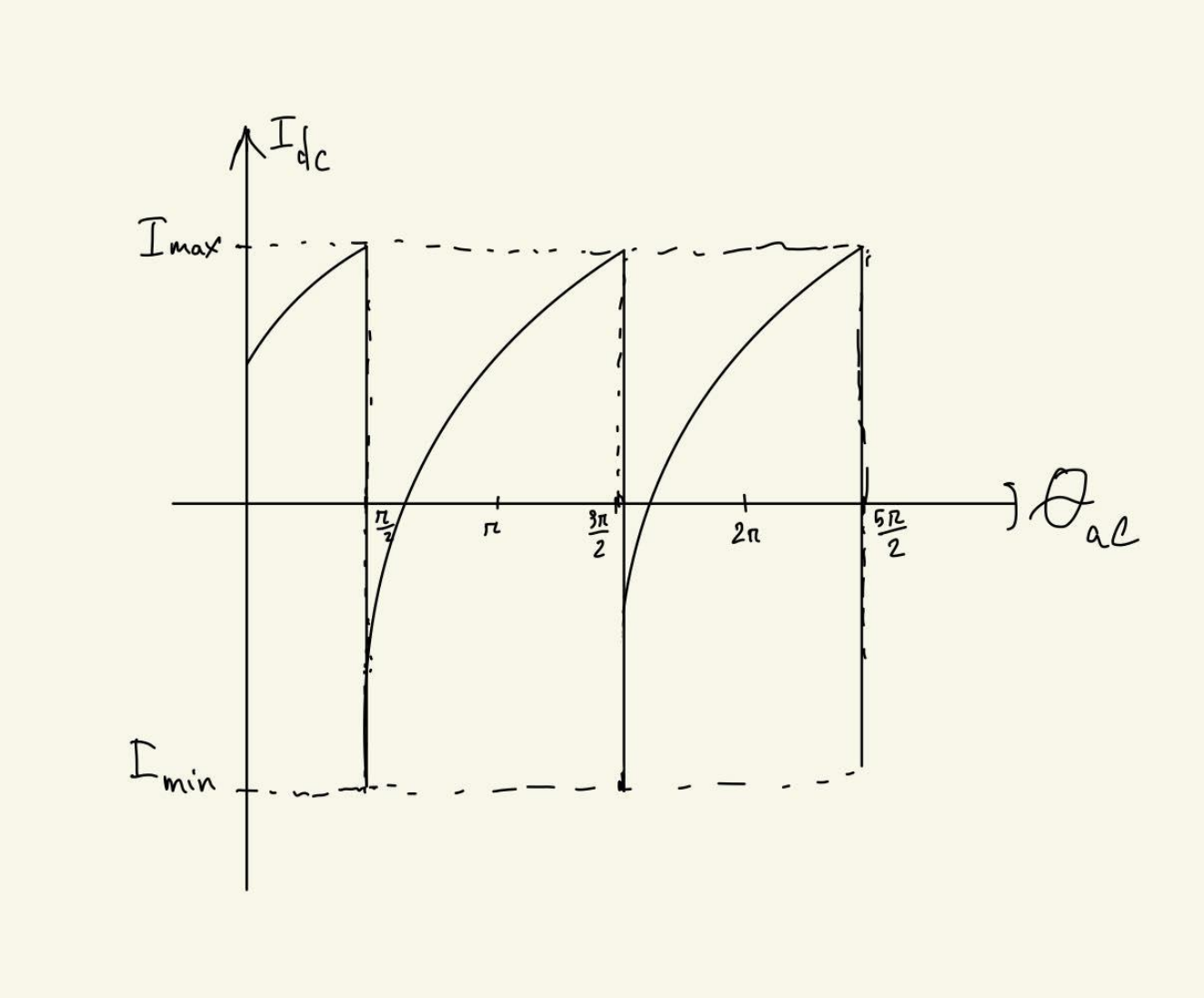

Figure 13. DC Current vs θ_{ac}.

Based on the logic of 180 switching, we graphed the voltages, as well as currents, of diodes D1 and D4, and transistors T1 and T4 on the same graphs just as shown in Figure 5, 7, 9, 11 as they are similar to each other; and with the same logic and assumptions in mind, we graphed voltages and currents for diodes D2 and D3, and transistors T2 and T3.

MATLAB Analysis

Applying the logic of 180 switching, we formulated *switch180.m* script, in which we plot and simulate the behavior of this single-phase inverter. The specifications and assumptions made above are used as initial values and parameters for this script.

All voltages and currents, except for DC voltage, are initialized as arrays, which serves the purpose of plotting them versus θ_{ac}. This script utilizes a while loop to update θ_{ac}, which is used to regulate the switching logic of this inverter. For diodes and transistors, all voltages and currents graphs are separated, not joined like in the Procedure section. This is only to show they are similar to each other. All components are plotted against θ_{ac} as the x-axis. This script makes use a mod function in the calculation of θ_{ac}, which helps keep it between 0 and 360 degrees; however, this function has a flaw that if we

intend to graph over more than one period, whenever the value hits 360, it returns to 0, making it looks like there are many periods overlapping. To fix this issue, we had to use indexing when plotting our data. The graphs are as below:

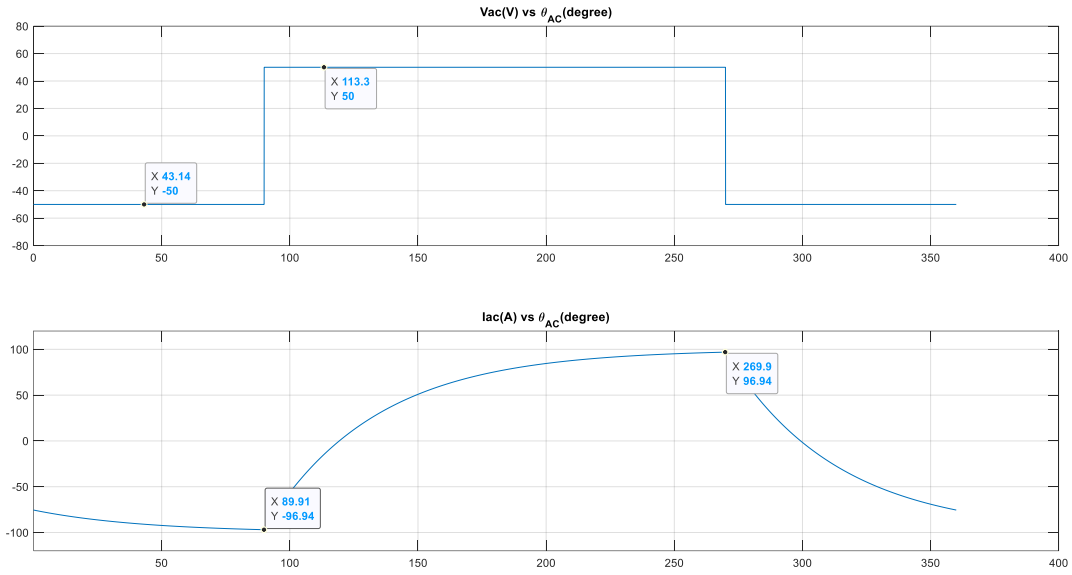

Figure 14. AC Voltage and Current vs θ_{ac}.

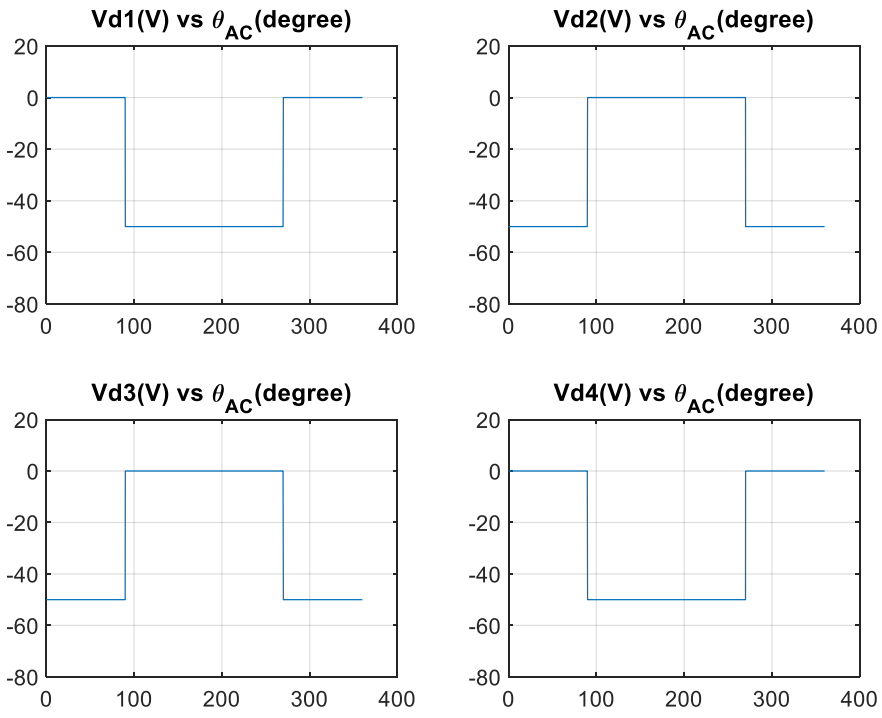

Figure 15. Diodes Voltages vs θ_{ac}.

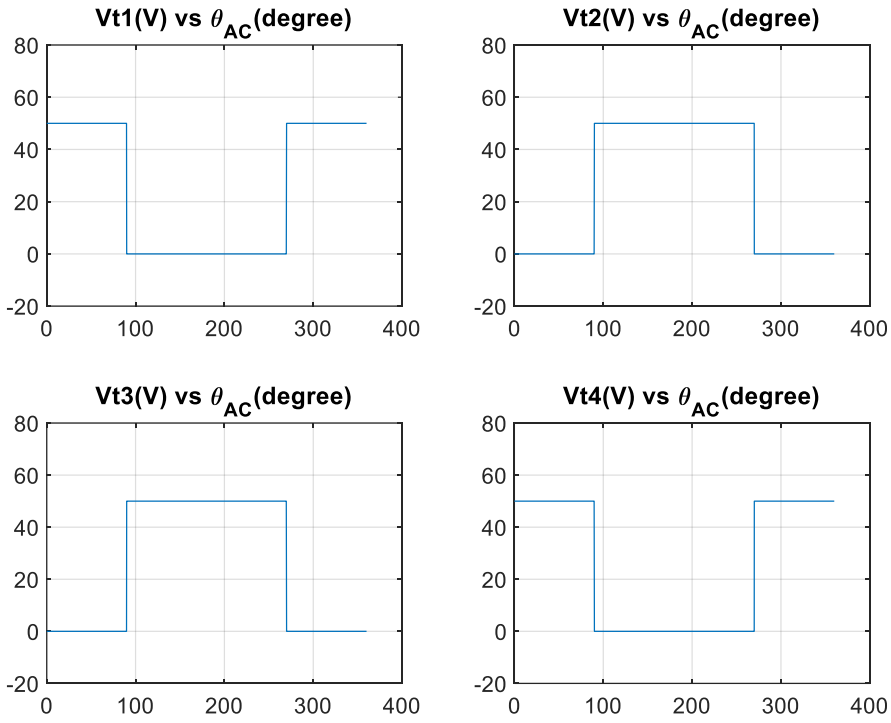

Figure 16. Transistors Voltages vs θ_{ac}.

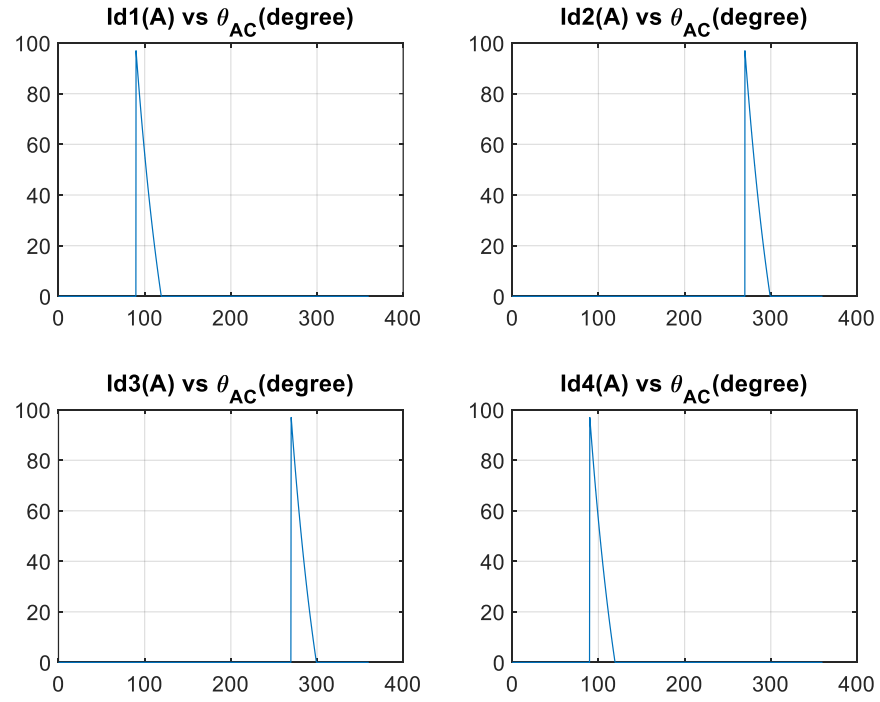

Figure 17. Diodes Currents vs θ_{ac}.

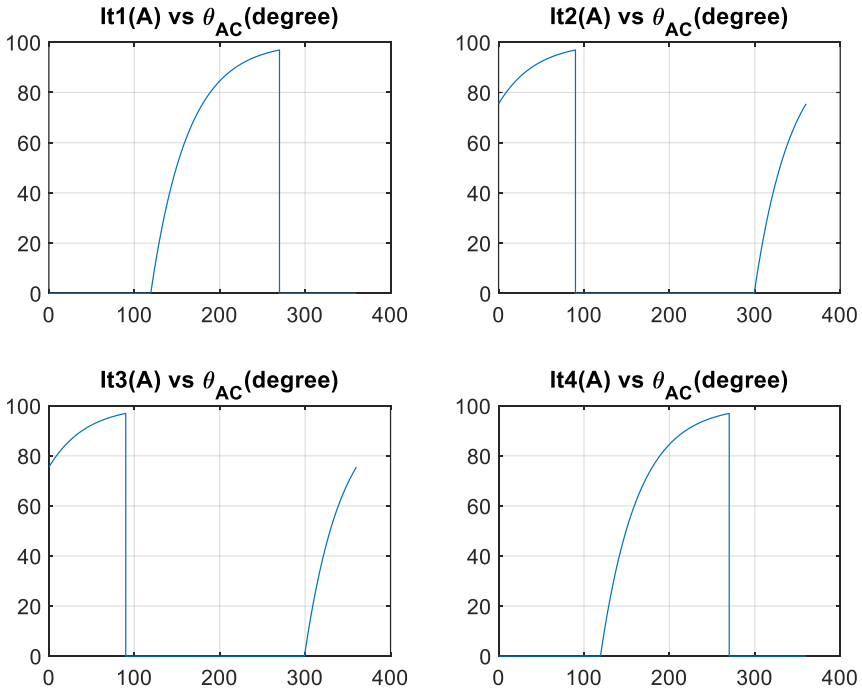

Figure 18. Transistors Currents vs θ_{ac}.

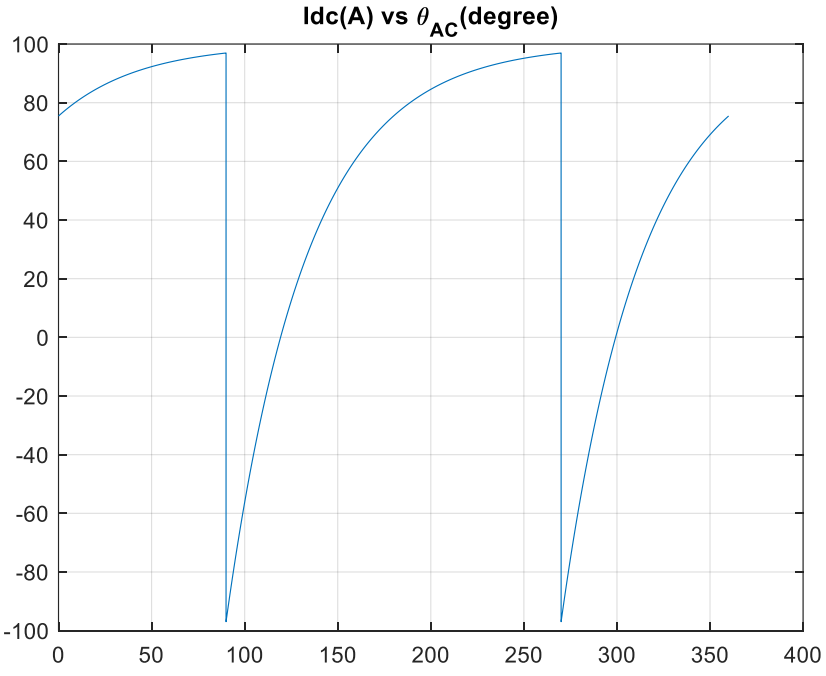

Figure 19. DC Current vs θ_{ac}.

This script also calls *fourseries.m* to analyze the frequency spectrum of the output voltage from 0 to 1200Hz as shown below:

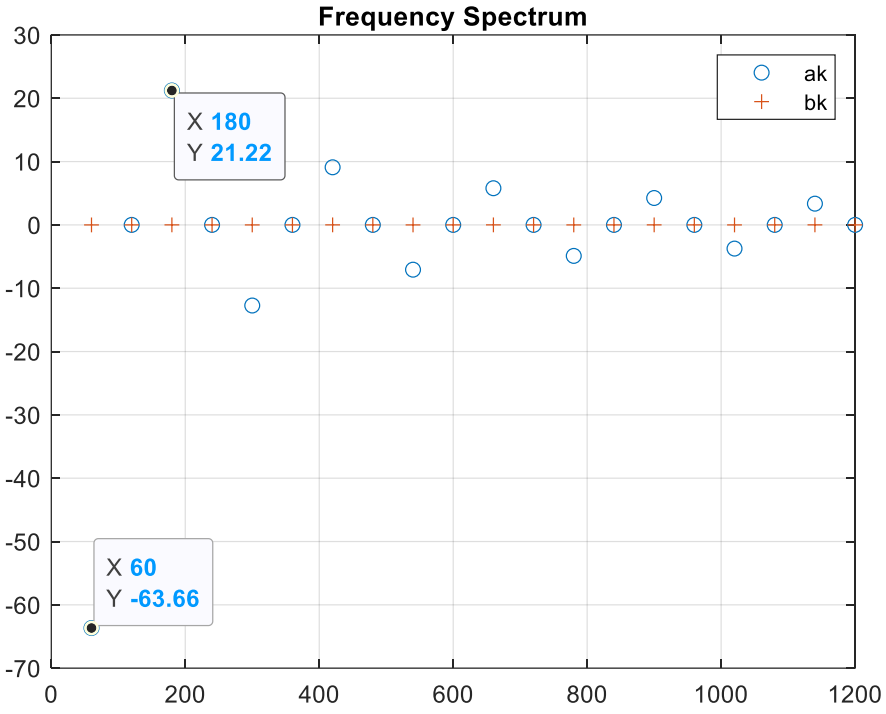

Figure 20. AC Voltage Amplitudes vs Frequency.

The *switch180.m* script is shown in MATLAB Scripts section.

Results and Conclusion

Calling *fourseries.m* and getting error value of the spectrum analyzer to be 7.1157, which is low enough to be acceptable.

Furthermore, revisiting the desired fundamental component of AC voltage, we have:

$$V_{as} = -\frac{200}{\pi}\cos(\theta_{ac}),$$

Where,

$$V_{as} = \frac{4}{\pi}V_{dc}$$

With $V_{dc} = 50V$ as previously calculated, we have $V_{as} \approx -63.662 = a_0$, since this is the fundamental component of AC voltage. Looking at the frequency spectrum in Figure 20, we have the fundamental component $a_0 = -63.66$, which yields us the error percentage of only 0.00314%.

Examining Figure 14, we can see our maximum and minimum AC current values to be $I_{max} = 96.94$, and $I_{min} = -96.94$. Compared with our calculations for maximum and minimum AC current values, we can see they match together, yielding 0 percent error.

This concludes that our analysis and simulation of the inverter under 180 switching is correct, all specifications are met and proven.

Sine-Triangle PWM Analysis

Assumptions and Specifications

The same single-phase inverter is to be analyzed but operating under sine-triangle PWM.

The load of this inverter contains a resistive load of $r = 1\Omega$ and an inductive load of $L = 1mH$. The fundamental frequency is 400Hz, DC voltage is 100V, and a phase angle of $\phi_v = -\frac{\pi}{2}$. Assuming switching frequency is 7600Hz, and m is 1.

Procedure

To analyze this inverter mathematically and logically, we use the similar functions to calculate frequencies and periods to get θ_{ac}, ω_{ac}, and switching θ_{sw}, ω_{sw}. These give us the values:

$$\omega_{ac} = 2\pi 400,$$

$$\omega_{sw} = 2\pi 7600,$$

$$\theta_{ac} = \omega_{ac} t = 2\pi 400t,$$

$$\theta_{sw} = \omega_{sw} t = 2\pi 7600t,$$

With m = 1 and $\phi_v = -\frac{\pi}{2}$ as our assumption, we have the duty cycle waveform as:

$$d = 1\cos(\theta_{ac} + \phi_v) = \cos\left(\theta_{ac} - \frac{\pi}{2}\right)$$

In order to complete the switching logic of sine-triangle PWM, we implemented a triangle waveform function that goes from -1 to 1 every half period. The triangle wave is as followed:

$$tri(t) = \frac{1}{2} + \sum_{k=1}^{\infty} \frac{(2\cos(k\pi) - \cos(2\pi k) - 1)}{(k\pi)^2} \cos(k\omega_{sw} t),$$

However, this triangle function only go from 0 to 1 every half period. To make it go from -1 to 1 every half period, an extra step needs to be made:

$$tri = 2tri - 1,$$

Therefore, we can make our triangle wave in a for loop for however long we want k to be and after the loop, we would do that extra step.

This triangle wave function and the duty cycle function are the main logic of how sine-triangle PWM switching is. At any time t, if the duty cycle waveform is larger than the triangle waveform, transistors T1 and T4 are on, while T2 and T3 are off, and vice versa.

MATLAB Analysis

Applying the logic of sine-triangle PWM switching as aforementioned, we formulated *sinetri.m* script as shown in MATLAB Scripts section below, in which we plot and simulate the behavior of this single-phase inverter. The specifications and assumptions made above are used as initial values and parameters for this script.

All voltages and currents, except for DC voltage, are initialized as arrays, which serves the purpose of plotting them versus θ_{ac}. This script utilizes a while loop to update θ_{ac}, and $d(t)$, which are used to

regulate the switching logic of this inverter. All components are plotted against θ_{ac} as the x-axis. This script still makes use of the mod function in the calculation of θ_{ac}, which helps keep it between 0 and 360 degrees. The graphs are as below:

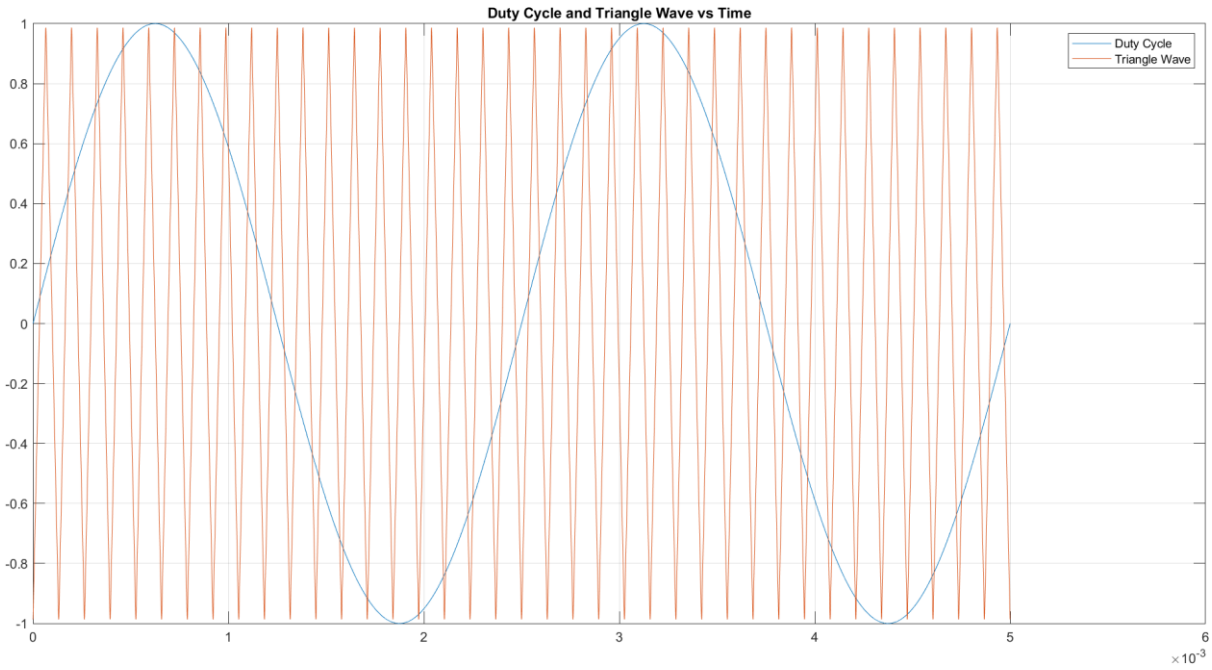

Figure 21. Duty Cycle and Triangle Modulation Waveform vs Time.

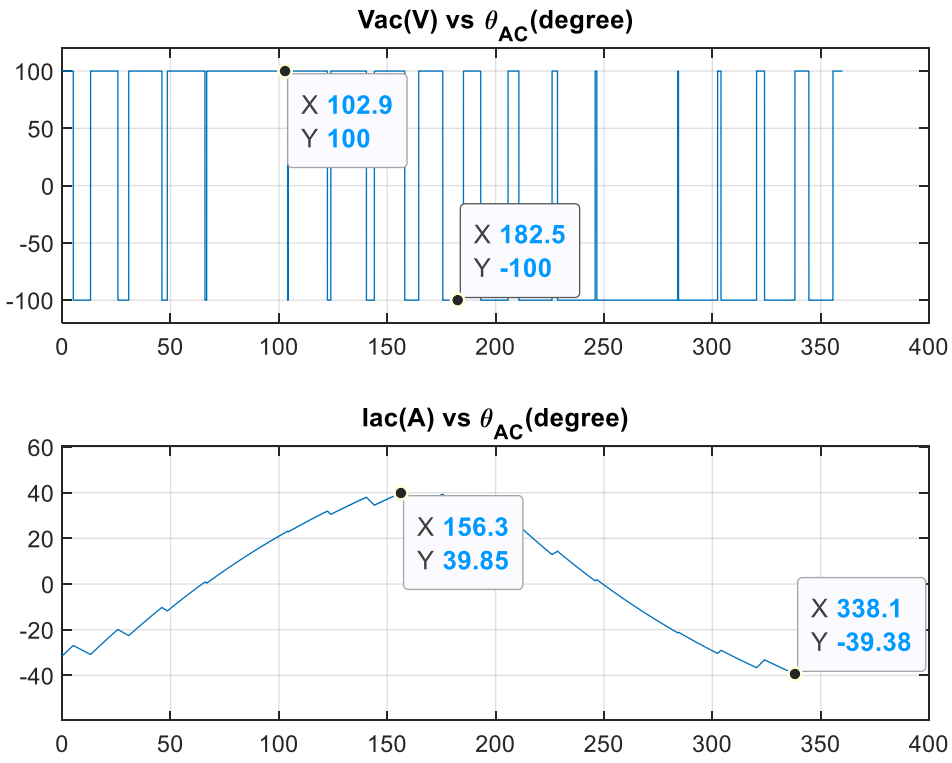

Figure 22. AC Voltage and Current vs θ_{ac}.

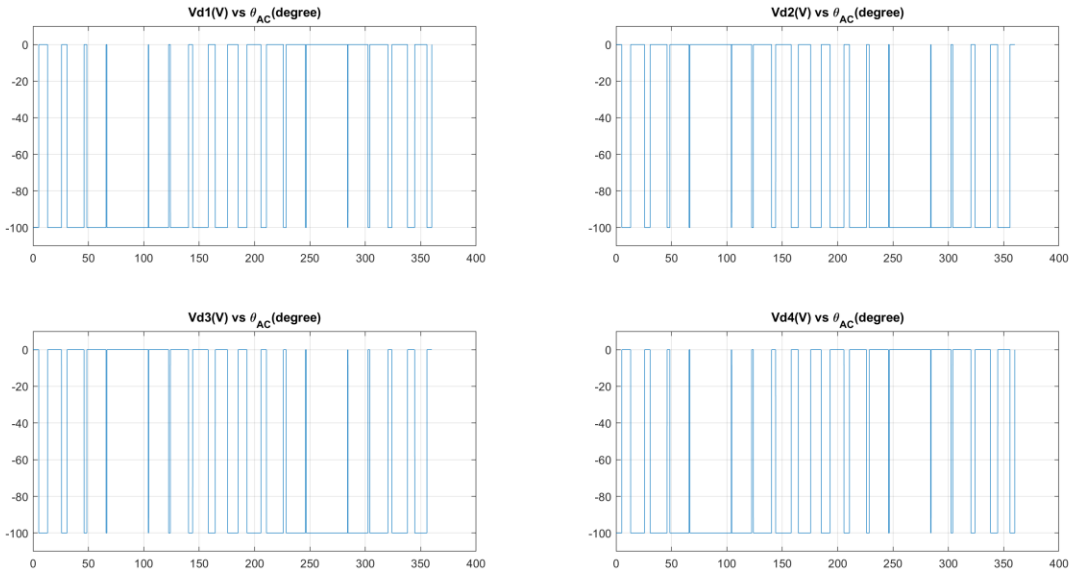

Figure 23. Diodes Voltages vs θ_{ac}.

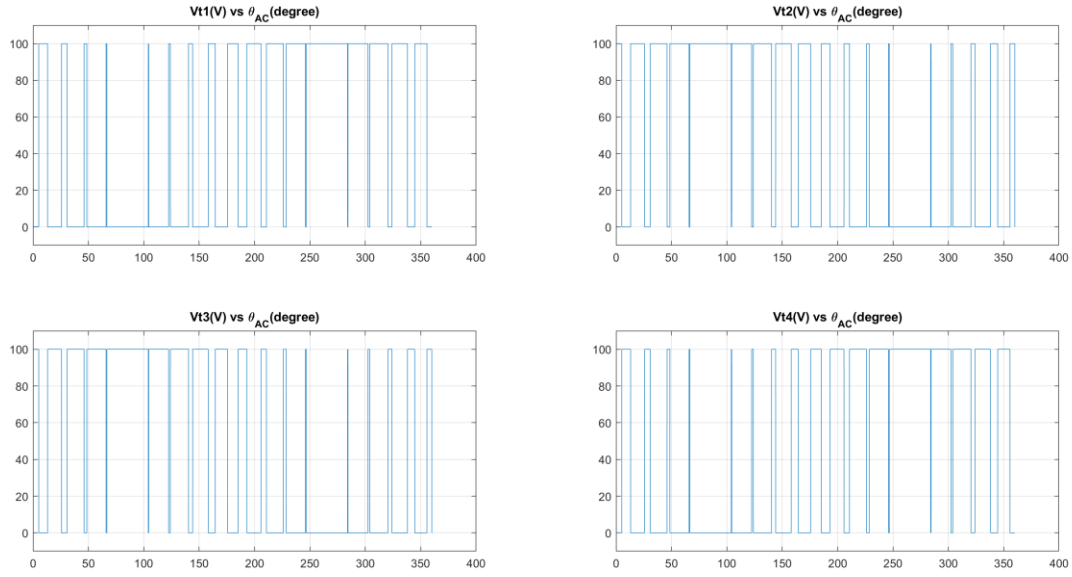

Figure 24. Transistors Voltages vs θ_{ac}.

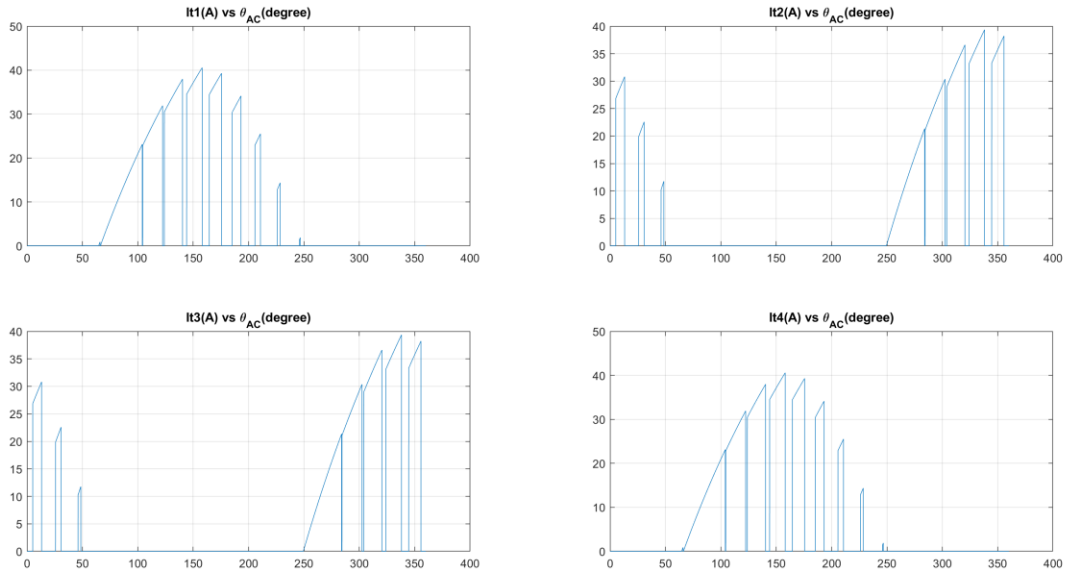

Figure 25. Transistors Currents vs θ_{ac}.

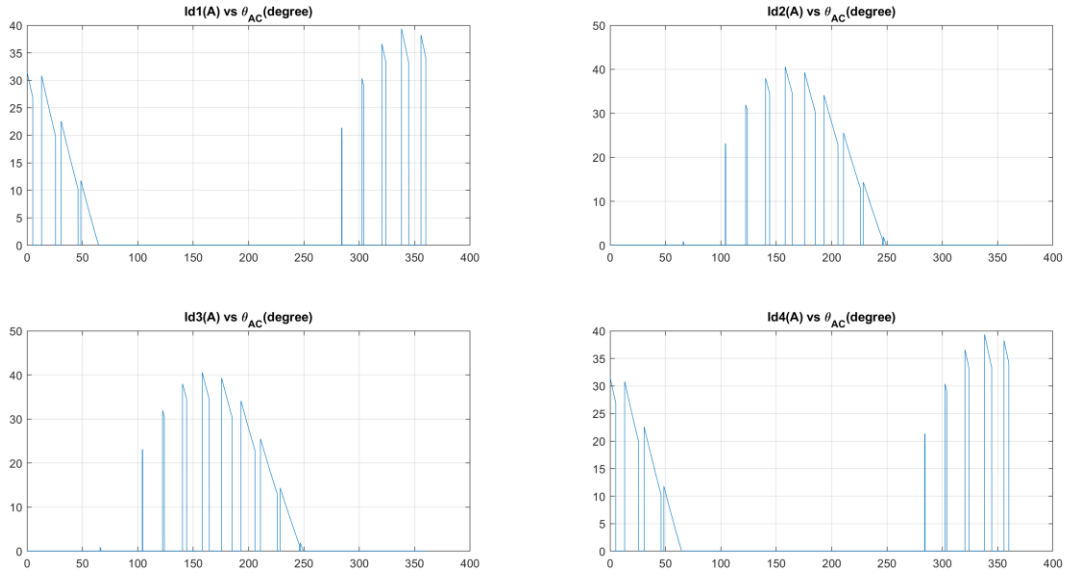

Figure 26. Diodes Currents vs θ_{ac}.

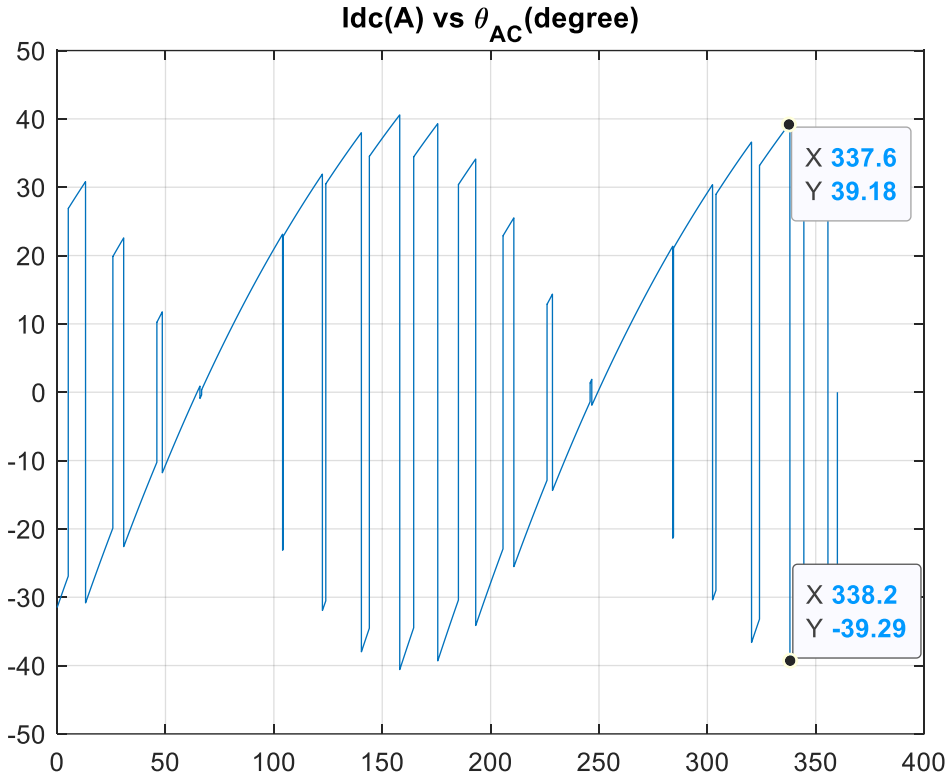

Figure 27. DC Current vs θ_{ac}.

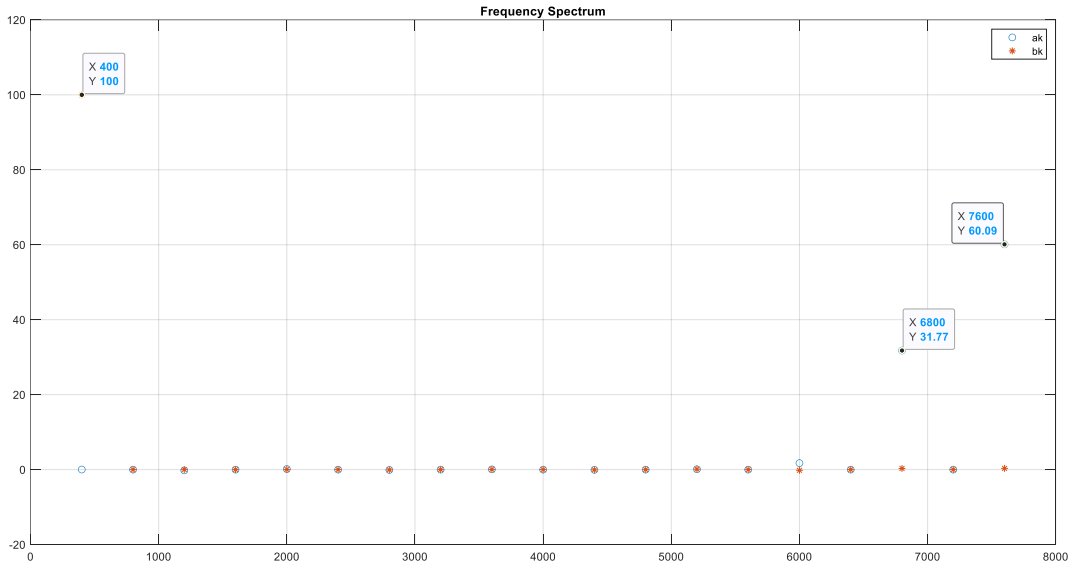

Figure 28. AC Voltage Amplitude vs Frequency.

Results and Conclusion

This concludes that our analysis and simulation of the inverter under 180 switching is correct, all specifications are met and proven.

From the MATLAB script and plot of AC Voltage and Current in Figure 22, we obtain the values as followed:

$$V_{max} = 100V = -V_{min} = V_{dc},$$

$$I_{max} = 39.85A,$$

$$I_{min} = -39.38A,$$

Using our assumption of m = 1, we can calculate the value for V_{ac} as:

$$V_{ac} = mV_{dc} = 100V,$$

This value agrees with the min and max values as shown in the MATLAB graph.

Using m = 1 and phasor analysis on this load, we can obtain the values for maximum and minimum I_{ac}:

$$I_{max} = \frac{V_{dc}}{Z} = 36.9698 \approx 37A, and\ I_{min} = -I_{max} = -37A,$$

With these values, we acquire error percentages of 7.70% and 6.43% for the max and min current respectively.

Furthermore, as shown in Figure 28, we can see that at frequency 400Hz, the value of $b_k = 100$. This agrees and satisfies the desired value of V_{ac} as:

$$V_{ac} = mV_{dc}\cos(\theta_{ac} + \phi_v) = 100\sin(\theta_{ac}),$$

Which explains we only observed a b_k value, while other coefficients values stay zero. This also means that we have successfully eliminated low frequency harmonics. However, as shown in Figure 28, at frequency 7600Hz and 6800Hz, there are some b_k values. These are high frequency harmonics, which would require us to implement active filters to get rid of.

In conclusion, this step has been successfully performed.

Amplitudes and Frequency Spectrum Analysis

Procedure

Using the single-phase inverter operating under sine-triangle PWM as above with the same frequencies, periods, and components, we are to test how different values of m would affect the AC voltage amplitudes.

To be more precise, the amplitude that we are to observe are the amplitude of the fundamental frequency component of the output voltage, which means that we want to get the amplitude at $k = 1$ for all a_k and b_k values. To calculate amplitudes, we use the function:

$$V_{amp} = \sqrt{a_1^2 + b_1^2},$$

The value of m can alter the modulation regime, which we want to get it to linear, to acquire the minimum DC voltage amplitude.

MATLAB Analysis

We altered the *sinetri.m* script to make *amp.m* to calculate and plot the fundamental amplitudes as a function of duty cycle. The *amp.m* script is shown in MATLAB Scripts section below.

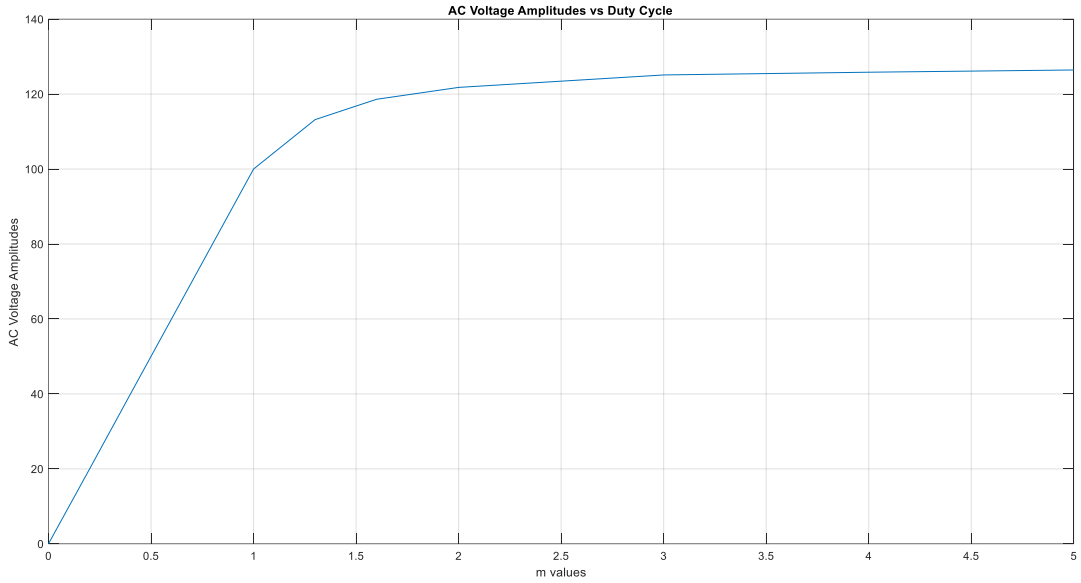

Figure 29. Fundamental Amplitudes of Output Voltage as a Function of Duty Cycle.

Results and Conclusions

As we can see in the graph, the value of the amplitude is in linear for $0 \leq m \leq 1$, this agrees with the behavior of the linear regime of the PWM output in single-phase inverter that

$$mV_{dc} = V_{amp}, as\ 0 \leq m \leq 1$$

And for $m > 1$, the function becomes:

$$f(m) = \frac{1}{2}\sqrt{\left(1 - \left(\frac{1}{m}\right)^2\right)} + \frac{m}{4}(\pi - 2\cos^{-1}\left(\frac{1}{m}\right))$$

This is when we are in overmodulation regime.

Three-Phase Inverter

Assumptions and Specifications

The three-phase inverter to be analyzed is operating under sine-triangle PWM. The symbols Q1, Q2, Q3, Q4 represent the transistors T1, T2, T3, T4, respectively.

The load of this inverter contains a resistive load of $r = 1\Omega$ and an inductive load of $L = 1mH$. The desired AC voltage of said inverter has a fundamental component at 50Hz. Assuming the phase-a current to be $I_{as} = 20\cos(\theta_{ac})$, and a switching frequency of 3100Hz. Assuming m = 1.

Procedure

To analyze this inverter mathematically and logically, we use the similar functions to calculate frequencies and periods to get θ_{ac}, ω_{ac}, and switching θ_{sw}, ω_{sw}. These give us the values:

$$\omega_{ac} = 2\pi 50,$$

$$\omega_{sw} = 2\pi 3100,$$

$$\theta_{ac} = \omega_{ac} t = 2\pi 50t,$$

$$\theta_{sw} = \omega_{sw} t = 2\pi 3100t,$$

We are given:

$$I_{as} = 20\cos(\theta_{ac})$$

Using the relationship between I and V, we can quickly form:

$$V_{as} = rI_{as} + \frac{LdI_{as}}{dt} = C \times \cos(\theta_{ac} + \phi_v)$$

Deriving this function with the specifications given, we have:

$$V_{as} = 20\cos(\theta_{ac}) - 2\pi\sin(\theta_{ac})$$

From here, using the Vamp equation to calculate magnitude, we can derive C and using arctan, we can get ϕ_v. Hence, we get:

$$V_{as} = C \times \cos(\theta_{ac} + \phi_v) = \sqrt{20^2 + (-2\pi)^2}\cos\left(\theta_{ac} + \arctan\left(-\frac{2\pi}{20}\right)\right)$$

Therefore,

$$V_{as} = 20.96\cos(\theta_{ac} - 17.441)$$

We know,

$$\frac{m}{2}V_{dc} = V_{as},$$

Gives

$$V_{dc} = 41.928V$$

The switching logic for this inverter resembles that for the single-phase inverter under sine-triangle PWM, with a duty cycle function for each leg, with the phase shift of 120 degrees.

MATLAB Analysis

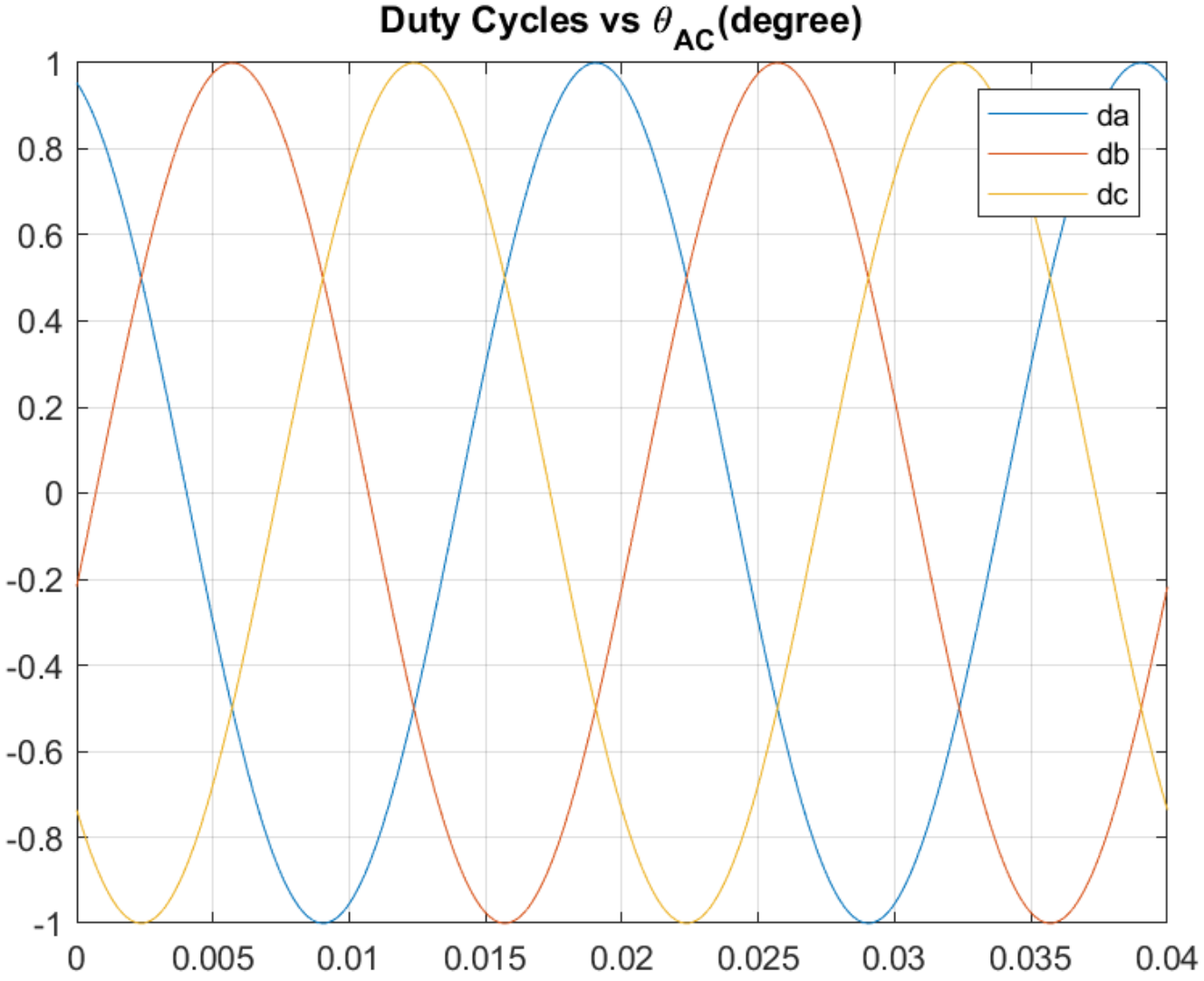

Figure 30. Duty Cycle vs Time.

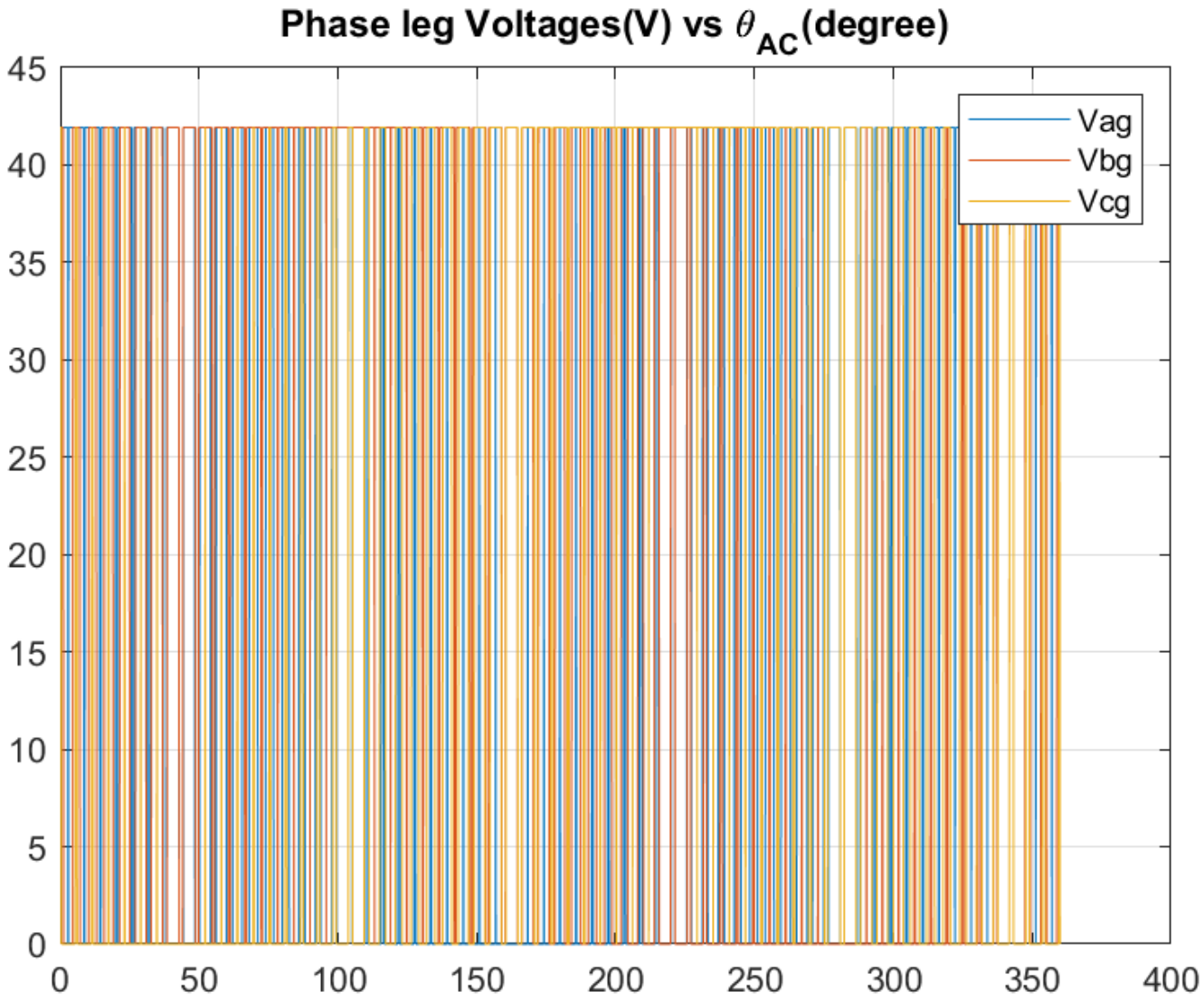

Figure 31. Phase Leg Voltages vs θ_{ac}.

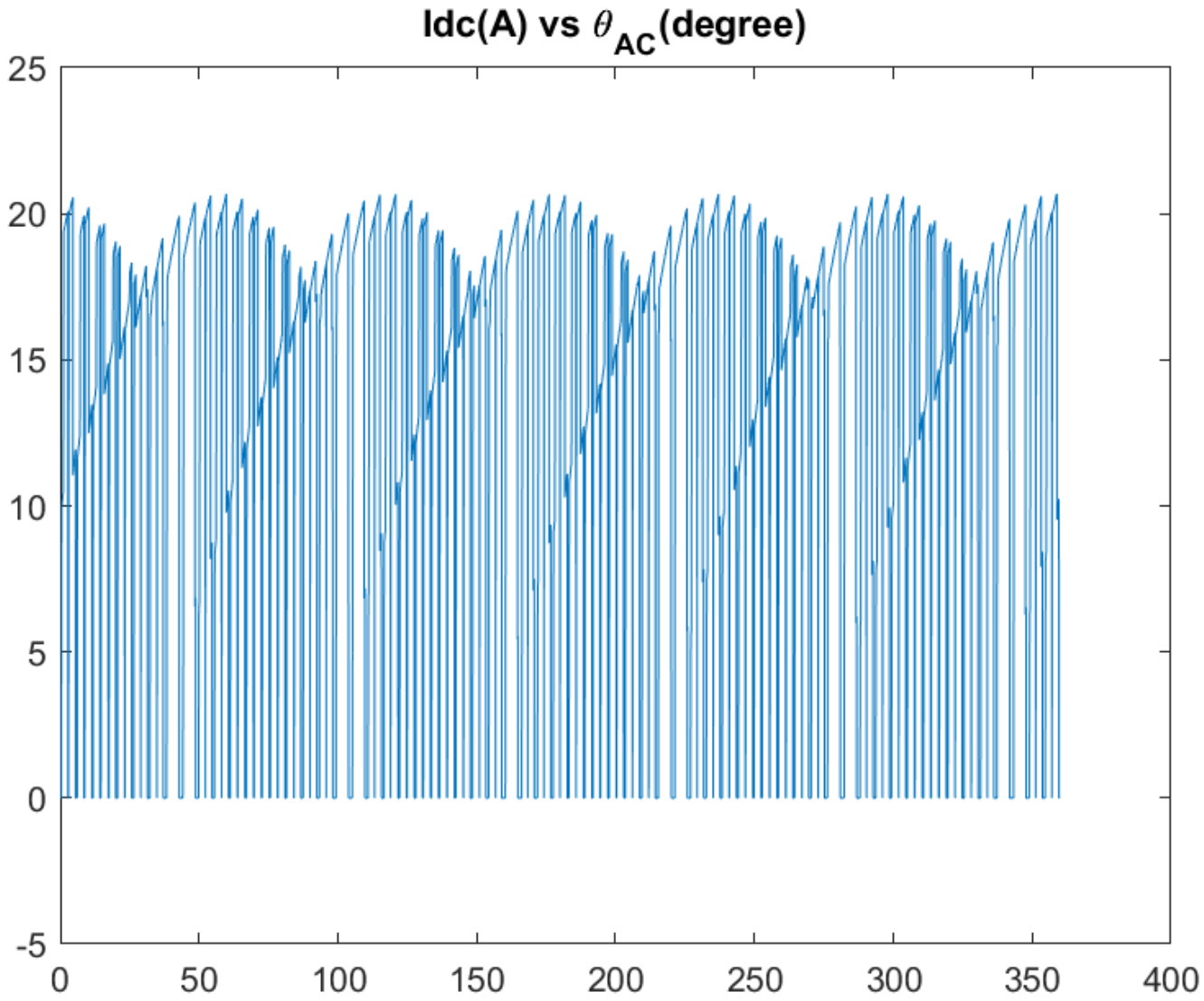

Figure 32. DC Current vs θ_{ac}.

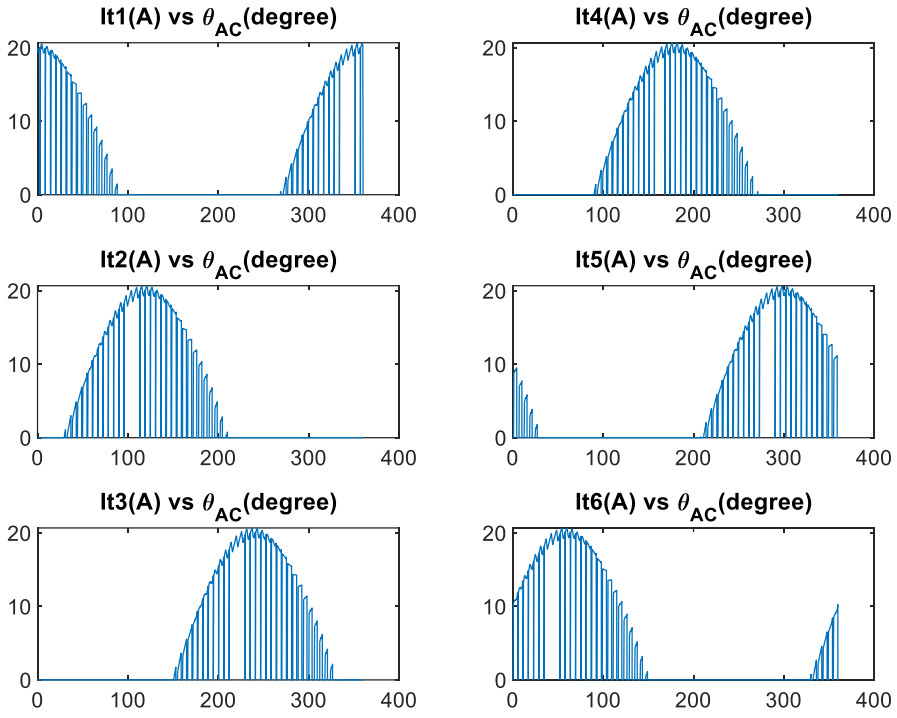

Figure 33. Transistor Currents vs θ_{ac}.

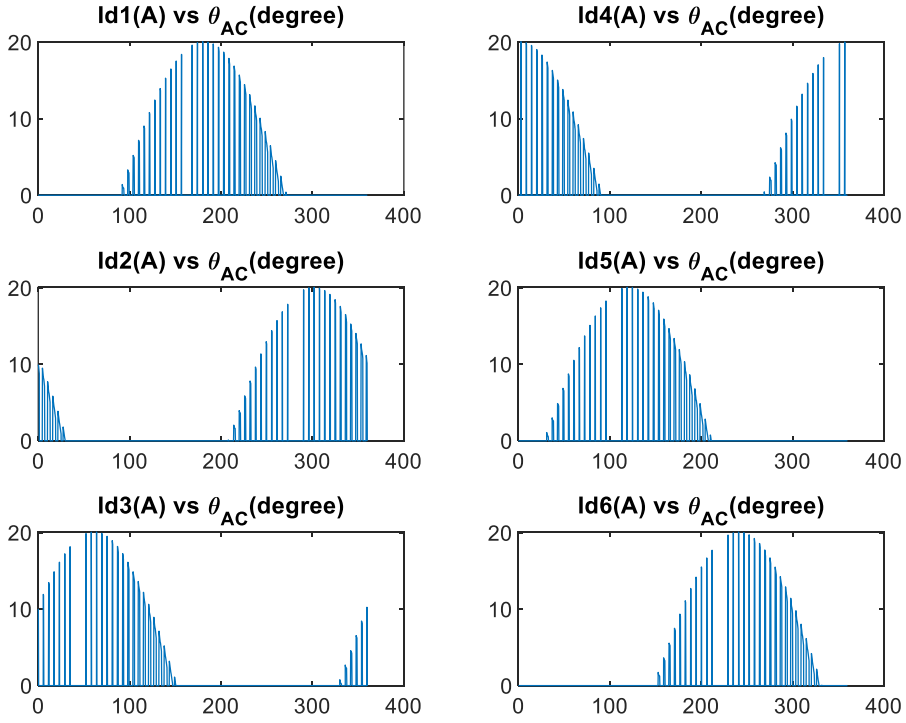

Figure 34. Diode Currents vs θ_{ac}.

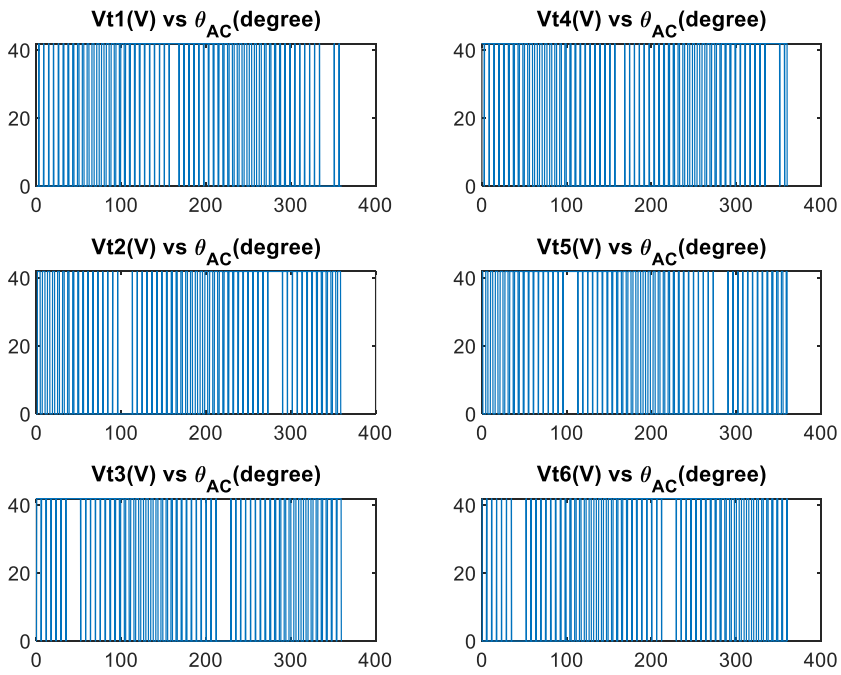

Figure 35. Transistor Currents vs θ_{ac}.

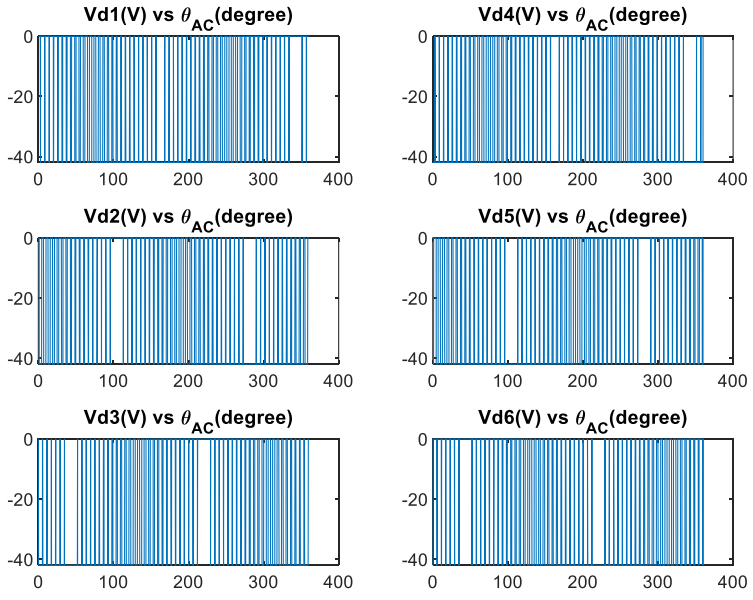

Figure 36. Diode Voltages vs θ_{ac}.

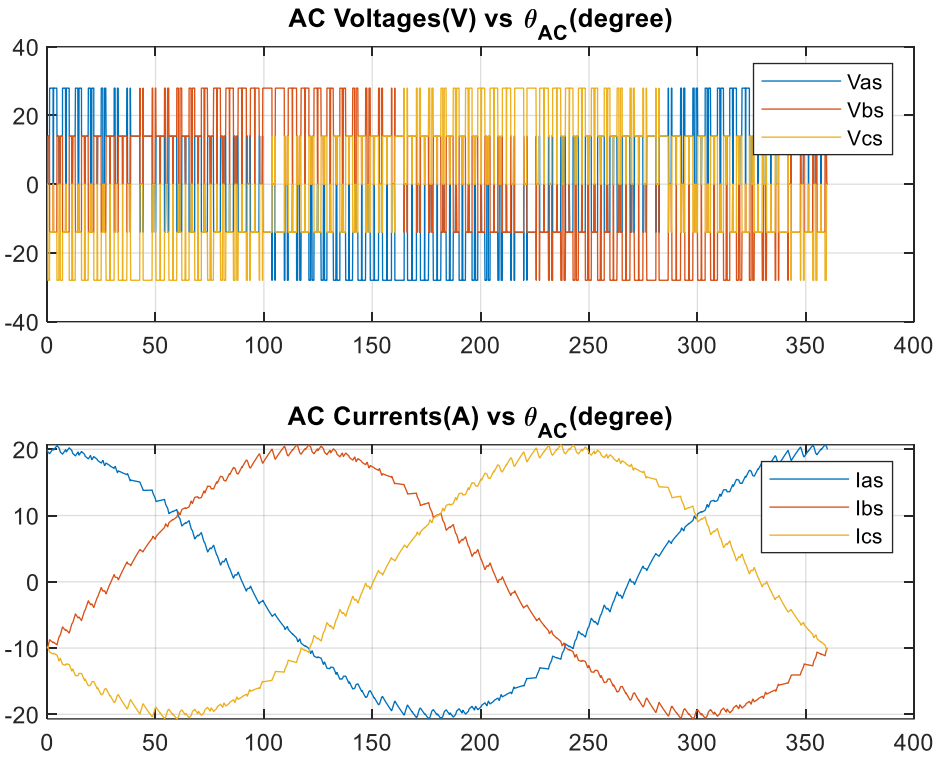

Figure 37. AC Voltages and Currents vs θ_{ac}.

Results and conclusions

After running the script *sinetri3phase.m*, found below in MATLAB Scripts section, we acquire the value for fundamental amplitude of:

$$V_{as} = 20.9652V$$

$$I_{as} = 18.1917V$$

These values compared to the voltage and current values of Vas and Ias we desired and calculated above provided errors of 0.005 and 9 for voltage and current respectively.

This concludes that we successfully achieved the desired three-phase inverter.

MATLAB Scripts

%% fourseries.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%{
t is time, x is the waveform that is being analyzed, T is its fundamental
period of the waveform, N is the number of terms of the fourier series
that is desired.
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function[avg,ak,bk,rcon,err] = fourseries(t,x,T,N)
    %% Parameters
    avg = 0;
    err = 0;
    dt = t(2) - t(1); % Size of subinterval
    sample = T / dt; % Number of subintervals
    ak = zeros(1, N);
    bk = zeros(1, N);
    w = 2 * pi / T;
    rcon = zeros(size(t));

    %% Calculate average (avg) of a waveform using Riemann sum
    for i = 1:sample
        avg = avg + dt * x(i);
    end
    avg = avg / T;

    %% Calculate ak and bk using Riemann sum
    for k = 1:N
        for n = 1:sample
            ak(k) = ak(k) + x(n) * dt * cos(k * w * t(n));
            bk(k) = bk(k) + x(n) * dt * sin(k * w * t(n));
        end
    end
    ak = 2 * ak / T;
    bk = 2 * bk / T;

    %% Calculate rcon, the reconstructed approximation of x
    % First we get a0 term, which is also the average
    rcon = avg;
    % Reconstruct x
    for k = 1:N
        rcon = rcon + ak(k) * cos(k * w .* t) + bk(k) * sin(k * w .* t);
    end

    %% Calculate error
    for i = 1:sample
```

```
        err = err + (rcon(i) -  x(i)) ^ 2;
    end
    err = sqrt(err / sample);
end
```

%% provefourseries.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%{
This is the script to call the function fourseries to prove its correctness.
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear

T = 1 / 40; % Fundamental Period
t = 0:1e-7:5*T; % Time array
 x = 15 - 30*cos(2 * pi * 40 * t) + 20*sin(2 * pi * 40 * t)...
     + 8*cos(2 * pi * 800 * t) + 2*sin(2 * pi * 1600 * t)...
     - 5*cos(2 * pi * 2000 * t);
N = 50; % Number of Fourier series terms

[avg, ak, bk, rcon, err] = fourseries(t, x, T, N);

disp(avg);
disp(err);

figure(1)
plot(t, x, t, rcon)
xlabel('time(s)')
legend('og wave', 'rcon')
grid on
```

%% switch180.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%{
This function is to simulate the system of a single-phase inverter
operating under 180 switching with the load is an rl load with
r = 0.5 ohm and l = 1 mH and the desired fundamental component
of AC voltage is Vas = -200/pi*cos(theta_ac) where theta_ac = 120*pi*t
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear
%% Initial Parameters
r = 0.5; % resistive load
l = 1e-3; % inductive load
f = 60; % fundamental frequency
T = 1 / f; % fundamental period
```

```
Vdc = 50; % Vdc
w = 2 * pi * f;
dt = 1e-7; % time step
tend = 2*T; % end of loop
t = 0:dt:tend; % time array
t_len = length(t);

%% Components Arrays of Values
Vac = zeros(1, t_len);
Idc = zeros(1, t_len);
Iac = zeros(1, t_len);

Id1 = zeros(1, t_len);
Id2 = zeros(1, t_len);
Id3 = zeros(1, t_len);
Id4 = zeros(1, t_len);
It1 = zeros(1, t_len);
It2 = zeros(1, t_len);
It3 = zeros(1, t_len);
It4 = zeros(1, t_len);

theta_ac = zeros(1, t_len);

Vd1 = zeros(1, t_len);
Vd2 = zeros(1, t_len);
Vd3 = zeros(1, t_len);
Vd4 = zeros(1, t_len);
Vt1 = zeros(1, t_len);
Vt2 = zeros(1, t_len);
Vt3 = zeros(1, t_len);
Vt4 = zeros(1, t_len);

%% while loop for simulation
k = 1; % while loop counter
while (t(k) < tend)
    theta_ac(k + 1) = w * t(k) * 180 / pi;
    theta_ac(k + 1) = mod(theta_ac(k + 1), 360);
    %% Switching logic
    if (theta_ac(k) >= 0 && theta_ac(k) < 90) || (theta_ac(k) > 270 &&
theta_ac(k) <= 360)
        T2 = 1;
        T3 = 1;
        T1 = 0;
        T4 = 0;
    elseif (theta_ac(k) >= 90 && theta_ac(k) <= 270)
        T2 = 0;
        T3 = 0;
```

```
        T1 = 1;
        T4 = 1;
    end
    %% AC Voltage and Current
    Vac(k + 1) = (2 * T1 * T4 - 1) * Vdc;
    Iac(k + 1) = Iac(k) + dt * (Vac(k) - r * Iac(k)) / l;
    %% Diodes and Transistors Voltages
    Vt1(k) = (1 - T1) * Vdc;
    Vt4(k) = Vt1(k);
    Vt2(k) = (1 - T2) * Vdc;
    Vt3(k) = Vt2(k);
    Vd1(k) = -T1 * Vdc;
    Vd4(k) = Vd1(k);
    Vd2(k) = -T2 * Vdc;
    Vd3(k) = Vd2(k);
    %% Diodes and Transistors Currents
    if (T1 == 1 && T4 == 1)
        if (Iac(k) >= 0)
            It1(k) = Iac(k);
            It4(k) = It1(k);
            Id1(k) = 0;
            Id4(k) = 0;
        else
            It1(k) = 0;
            It4(k) = 0;
            Id1(k) = -Iac(k);
            Id4(k) = -Iac(k);
        end
    else
        if (Iac(k) >= 0)
            It2(k) = 0;
            It3(k) = 0;
            Id2(k) = Iac(k);
            Id3(k) = Iac(k);
        else
            It2(k) = -Iac(k);
            It3(k) = -Iac(k);
            Id2(k) = 0;
            Id3(k) = 0;
        end
    end
    Idc(k) = It1(k) + It2(k) - Id1(k) - Id2(k);
    % Time array and steps update
    t(k + 1) = t(k) + dt;
    k = k + 1;
end
```

```
%% Calculate frequency spectrum
f_spec = f:f:1200;
N = length(f_spec);
[avg, ak, bk, rcon, err] = fourseries(t, Vac, T, N);
disp(err)

%% Plot
% Note: the indexing seen here in every arrays is to compensate for the mod
% function used earlier to calculate and update theta_ac. This function,
% after hitting 360, returns straight back to zero, which results in an
% overlap of many periods, which affects the integrity of the analysis.
% Index 166669 is where theta_ac returns back to zero.
figure(1)
subplot(2,1,1)
plot(theta_ac(166669:end), Vac(166669:end))
ylim([-80 80])
grid on
title('Vac(V) vs \theta_{AC}(degree)')
subplot(2,1,2)
plot(theta_ac(166669:end), Iac(166669:end))
ylim([-120 120])
grid on
title('Iac(A) vs \theta_{AC}(degree)')

figure(2)
subplot(2,2,1)
plot(theta_ac(166669:end), Vd1(166668:end))
ylim([-80 20])
grid on
title('Vd1(V) vs \theta_{AC}(degree)')
subplot(2,2,2)
plot(theta_ac(166669:end), Vd2(166668:end))
ylim([-80 20])
grid on
title('Vd2(V) vs \theta_{AC}(degree)')
subplot(2,2,3)
plot(theta_ac(166669:end), Vd3(166668:end))
ylim([-80 20])
grid on
title('Vd3(V) vs \theta_{AC}(degree)')
subplot(2,2,4)
plot(theta_ac(166669:end), Vd4(166668:end))
ylim([-80 20])
grid on
title('Vd4(V) vs \theta_{AC}(degree)')

figure(3)
```

```
subplot(2,2,1)
plot(theta_ac(166669:end), Vt1(166668:end))
ylim([-20 80])
grid on
title('Vt1(V) vs \theta_{AC}(degree)')
subplot(2,2,2)
plot(theta_ac(166669:end), Vt2(166668:end))
ylim([-20 80])
grid on
title('Vt2(V) vs \theta_{AC}(degree)')
subplot(2,2,3)
plot(theta_ac(166669:end), Vt3(166668:end))
ylim([-20 80])
grid on
title('Vt3(V) vs \theta_{AC}(degree)')
subplot(2,2,4)
plot(theta_ac(166669:end), Vt4(166668:end))
ylim([-20 80])
grid on
title('Vt4(V) vs \theta_{AC}(degree)')

figure(4)
subplot(2,2,1)
plot(theta_ac(166669:end), Id1(166668:end))
grid on
title('Id1(A) vs \theta_{AC}(degree)')
subplot(2,2,2)
plot(theta_ac(166669:end), Id2(166668:end))
grid on
title('Id2(A) vs \theta_{AC}(degree)')
subplot(2,2,3)
plot(theta_ac(166669:end), Id3(166668:end))
grid on
title('Id3(A) vs \theta_{AC}(degree)')
subplot(2,2,4)
plot(theta_ac(166669:end), Id4(166668:end))
grid on
title('Id4(A) vs \theta_{AC}(degree)')

figure(5)
subplot(2,2,1)
plot(theta_ac(166669:end), It1(166668:end))
grid on
title('It1(A) vs \theta_{AC}(degree)')
subplot(2,2,2)
plot(theta_ac(166669:end), It2(166668:end))
grid on
```

```
title('It2(A) vs \theta_{AC}(degree)')
subplot(2,2,3)
plot(theta_ac(166669:end), It3(166668:end))
grid on
title('It3(A) vs \theta_{AC}(degree)')
subplot(2,2,4)
plot(theta_ac(166669:end), It4(166668:end))
grid on
title('It4(A) vs \theta_{AC}(degree)')

figure(6)
plot(theta_ac(166669:end), Idc(166668:end))
grid on
title('Idc(A) vs \theta_{AC}(degree)')

figure(7)
plot(f_spec, ak, 'o')
hold on
plot(f_spec, bk, '+')
title('Frequency Spectrum')
legend('ak','bk')
grid on
hold off
```

%% sinetri.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%{
This function is to simulate the system of a single-phase inverter
operating under sine-triangle PWM with the load is an rl load with
r = 0.5 ohm and l = 1 mH and the desired ac voltage has a fundamental
frequency of 400Hz and a phase angle phi = -pi/2. We assume DC voltage
is 100V and switching frequency is 7600Hz.
}%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear
%% Initial Parameters
r = 1; % resistive load
l = 1e-3; % inductive load
phi = (-pi / 2) * 180 / pi; % phase angle
f = 400; % fundamental frequency
fsw = 7600; % switching frequency
T = 1 / f; % fundamental period
Vdc = 100; % DC voltage
w_ac = 2 * pi * f; % omega
w_sw = 2 * pi * fsw; % omega switch
dt = 1e-7; % time step
```

```
tend = 2*T; % end of loop
t = 0:dt:tend; % time array
t_len = length(t);

tri = 1 / 2; % tri function intial value
for i = 1:30
    tri = tri + (2 * cos(pi * i) - cos(2 * pi * i) - 1) / ((pi ^ 2) ...
            * (i ^ 2)) * cos(w_sw * i * t);
end
tri = 2 * tri - 1;
m = 1;

%% Components Arrays of Values
Vac = zeros(1, t_len);
Idc = zeros(1, t_len);
Iac = zeros(1, t_len);
Id1 = zeros(1, t_len);
Id2 = zeros(1, t_len);
Id3 = zeros(1, t_len);
Id4 = zeros(1, t_len);
It1 = zeros(1, t_len);
It2 = zeros(1, t_len);
It3 = zeros(1, t_len);
It4 = zeros(1, t_len);
theta_ac = zeros(1, t_len);
Vd1 = zeros(1, t_len);
Vd2 = zeros(1, t_len);
Vd3 = zeros(1, t_len);
Vd4 = zeros(1, t_len);
Vt1 = zeros(1, t_len);
Vt2 = zeros(1, t_len);
Vt3 = zeros(1, t_len);
Vt4 = zeros(1, t_len);
d = zeros(1, t_len); % duty cycle array

%% while loop for simulation
k = 1; % while loop counter
while (t(k) < tend)
    theta_ac(k + 1) = (w_ac * t(k)) * 180 / pi;
    theta_ac(k + 1) = mod(theta_ac(k + 1), 360);
    d(k) = m * cosd(theta_ac(k) + phi);
    %% Switching logic
    if (d(k) >= tri(k))
        T2 = 0;
        T3 = 0;
        T1 = 1;
        T4 = 1;
```

```
elseif (d(k) < tri(k))
    T2 = 1;
    T3 = 1;
    T1 = 0;
    T4 = 0;
end
%% AC Voltage and Current
Vac(k + 1) = (2 * T1 * T4 - 1) * Vdc;
Iac(k + 1) = Iac(k) + dt * ((Vac(k) - r * Iac(k)) / l);
%% Diodes and Transistors Voltages
Vt1(k) = (1 - T1) * Vdc;
Vt4(k) = Vt1(k);
Vt2(k) = (1 - T2) * Vdc;
Vt3(k) = Vt2(k);
Vd1(k) = -T1 * Vdc;
Vd4(k) = Vd1(k);
Vd2(k) = -T2 * Vdc;
Vd3(k) = Vd2(k);
%% Diodes and Transistors Currents
if (T1 == 1 && T4 == 1)
    if (Iac(k) >= 0)
        It1(k) = Iac(k);
        It4(k) = It1(k);
        Id1(k) = 0;
        Id4(k) = 0;
    else
        It1(k) = 0;
        It4(k) = 0;
        Id1(k) = -Iac(k);
        Id4(k) = Id1(k);
    end
elseif (T2 == 1 && T3 == 1)
    if (Iac(k) >= 0)
        Id2(k) = Iac(k);
        Id3(k) = Iac(k);
        It2(k) = 0;
        It3(k) = 0;
    else
        Id2(k) = 0;
        Id3(k) = 0;
        It2(k) = -Iac(k);
        It3(k) = -Iac(k);
    end
end
Idc(k) = It1(k) + It2(k) - Id1(k) - Id2(k);
% Time array and steps update
t(k + 1) = t(k) + dt;
```

```
    k = k + 1;
end

%% Calculate frequency spectrum
f_spec = f:f:fsw;
N = length(f_spec);
[avg, ak, bk, rcon, err] = fourseries(t, Vac, T, N);
disp(err)
disp(avg)

%% Plot
% Note: the indexing seen here in every arrays is to compensate for the mod
% function used earlier to calculate and update theta_ac. This function,
% after hitting 360, returns straight back to zero, which results in an
% overlap of many periods, which affects the integrity of the analysis.
% Index 25003 is where theta_ac returns back to zero.
figure(1)
subplot(2,1,1)
plot(theta_ac(25003:end), Vac(25003:end))
ylim([-120 120])
grid on
title('Vac(V) vs \theta_{AC}(degree)')
subplot(2,1,2)
plot(theta_ac(25003:end), Iac(25003:end))
ylim([-60 60])
grid on
title('Iac(A) vs \theta_{AC}(degree)')

figure(2)
subplot(2,2,1)
plot(theta_ac(25003:end), Vd1(25003:end))
ylim([-110 10])
grid on
title('Vd1(V) vs \theta_{AC}(degree)')
subplot(2,2,2)
plot(theta_ac(25003:end), Vd2(25003:end))
ylim([-110 10])
grid on
title('Vd2(V) vs \theta_{AC}(degree)')
subplot(2,2,3)
plot(theta_ac(25003:end), Vd3(25003:end))
ylim([-110 10])
grid on
title('Vd3(V) vs \theta_{AC}(degree)')
subplot(2,2,4)
plot(theta_ac(25003:end), Vd4(25003:end))
ylim([-110 10])
```

```
grid on
title('Vd4(V) vs \theta_{AC}(degree)')

figure(3)
subplot(2,2,1)
plot(theta_ac(25003:end), Vt1(25003:end))
ylim([-10 110])
grid on
title('Vt1(V) vs \theta_{AC}(degree)')
subplot(2,2,2)
plot(theta_ac(25003:end), Vt2(25003:end))
ylim([-10 110])
grid on
title('Vt2(V) vs \theta_{AC}(degree)')
subplot(2,2,3)
plot(theta_ac(25003:end), Vt3(25003:end))
ylim([-10 110])
grid on
title('Vt3(V) vs \theta_{AC}(degree)')
subplot(2,2,4)
plot(theta_ac(25003:end), Vt4(25003:end))
ylim([-10 110])
grid on
title('Vt4(V) vs \theta_{AC}(degree)')

figure(4)
subplot(2,2,1)
plot(theta_ac(25003:end), Id1(25003:end))
grid on
title('Id1(A) vs \theta_{AC}(degree)')
subplot(2,2,2)
plot(theta_ac(25003:end), Id2(25003:end))
grid on
title('Id2(A) vs \theta_{AC}(degree)')
subplot(2,2,3)
plot(theta_ac(25003:end), Id3(25003:end))
grid on
title('Id3(A) vs \theta_{AC}(degree)')
subplot(2,2,4)
plot(theta_ac(25003:end), Id4(25003:end))
grid on
title('Id4(A) vs \theta_{AC}(degree)')

figure(5)
subplot(2,2,1)
plot(theta_ac(25003:end), It1(25003:end))
grid on
```

```
title('It1(A) vs \theta_{AC}(degree)')
subplot(2,2,2)
plot(theta_ac(25003:end), It2(25003:end))
grid on
title('It2(A) vs \theta_{AC}(degree)')
subplot(2,2,3)
plot(theta_ac(25003:end), It3(25003:end))
grid on
title('It3(A) vs \theta_{AC}(degree)')
subplot(2,2,4)
plot(theta_ac(25003:end), It4(25003:end))
grid on
title('It4(A) vs \theta_{AC}(degree)')

figure(6)
plot(theta_ac(25003:end), Idc(25003:end))
grid on
title('Idc(A) vs \theta_{AC}(degree)')

figure(7)
plot(f_spec, ak, 'o')
hold on
plot(f_spec, bk, '*')
title('Frequency Spectrum')
ylim([-20 120])
legend('ak','bk')
grid on
hold off

figure(8)
plot(t, d, t, tri)
legend('Duty Cycle','Triangle Wave')
title('Duty Cycle and Triangle Wave vs Time')
grid on


%% amp.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%{
This function is to simulate the system of a single-phase inverter
operating under sinetriangle PWM. this function varies m values from 0,
0.2, 0.4, 1.0, 1.3, 1.6, 2.0, 3.0, 4, 5 then plot amplitude as a function
of duty cycle
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear
%% Initial Parameters
```

```
r = 1; % resistive load
l = 1e-3; % inductive load
phi = (-pi / 2) * 180 / pi; % phase angle
f = 400; % fundamental frequency
T = 1 / f; % fundamental period
w_ac = 2 * pi * f; % omega
fsw = 7600; % switching frequency
Tsw = 1 / fsw;
w_sw = 2 * pi * fsw; % omega switch

Vdc = 100; % DC voltage
dt = 1e-7; % time step
tend = 2*T; % end of loop
t = 0:dt:tend; % time array
t_len = length(t);
m_array = [0, 0.2, 0.4, 1.0, 1.3, 1.6, 2.0, 3.0, 4, 5];
m_len = length(m_array);

tri = 1 / 2; % tri function intial value
for i = 1:30
    tri = tri + (2 * cos(pi * i) - cos(2 * pi * i) - 1) / ((pi ^ 2) ...
            * (i ^ 2)) * cos(w_sw * i * t);
end
tri = 2 * tri - 1;
m = 1;

%% Components Arrays of Values
Vac = zeros(1, t_len);
theta_ac = zeros(1, t_len);
d = zeros(1, t_len);
amplitude = zeros(1, m_len);

%% Procedure
for i = 1:m_len
    %% while loop for simulation
    k = 1; % while loop counter
    while (t(k) < tend)
        theta_ac(k + 1) = (w_ac * t(k)) * 180 / pi;
        theta_ac(k + 1) = mod(theta_ac(k + 1), 360);
        d(k) = m_array(i) * cosd(theta_ac(k) + phi);
        %% Switching logic
        if (d(k) >= tri(k))
            T1 = 1;
            T4 = 1;
        else
            T1 = 0;
            T4 = 0;
```

```
        end
        %% AC Voltage
        Vac(k + 1) = (2 * T1 * T4 - 1) * Vdc;
        % Time array and steps update
        t(k + 1) = t(k) + dt;
        k = k + 1;
    end
    [avg,ak,bk,rcon,err] = fourseries(t, Vac, T, 10);
    amplitude(i) = sqrt((ak(1)) ^ 2 + (bk(1)) ^ 2);
end

%% Plot
figure(1)
plot(m_array, amplitude);
title('AC Voltage Amplitudes vs Duty Cycle')
xlabel('m values')
ylabel('AC Voltage Amplitudes')
grid on
```

%% sinetri3phase.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%{
This function is to simulate the system of a 3-phase inverter
operating under sine-triangle PWM with the load is an rl load with
r = 0.5 ohm and l = 1 mH and the desired ac current has a fundamental
component at 50Hz and phase a current is Ias = 20cos(theta_ac).
We assume switching frequency is 3100Hz.
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear
%% Initial Parameters
r = 1; % resistive load
l = 1e-3; % inductive load
phi_dis = 120; % displayed phase for each leg

f = 50; % fundamental frequency
T = 1 / f; % fundamental period
w_ac = 2 * pi * f; % omega
fsw = 3100; % switching frequency
w_sw = 2 * pi * fsw; % omega switch
T_sw = 1 / fsw; % switching period

dt = 1e-7; % time step
tend = 2*T; % end of loop
t = 0:dt:tend; % time array
t_len = length(t);
```

```
%% Calculate Vdc and phase shift angle
Vdc = 2 * sqrt(20 ^ 2 + (2 * pi) ^ 2); % Vdc derived from given Ias
phi = atand((-2 * pi) / 20); % Phase shift angle

%% Triangle function
tri = 1 / 2; % tri function intial value
for i = 1:100
    tri = tri + (2 * cos(pi * i) - cos(2 * pi * i) - 1) / ((pi ^ 2) ...
            * (i ^ 2)) * cos(w_sw * i * t);
end
tri = 2 * tri - 1;
m = 1;

%% Components Arrays of Values
Vas = zeros(1, t_len);
Vbs = zeros(1, t_len);
Vcs = zeros(1, t_len);
Vag = zeros(1, t_len);
Vbg = zeros(1, t_len);
Vcg = zeros(1, t_len);

Idc = zeros(1, t_len);

Ias = zeros(1, t_len);
Ibs = zeros(1, t_len);
Ics = zeros(1, t_len);

Id1 = zeros(1, t_len);
Id2 = zeros(1, t_len);
Id3 = zeros(1, t_len);
Id4 = zeros(1, t_len);
Id5 = zeros(1, t_len);
Id6 = zeros(1, t_len);

It1 = zeros(1, t_len);
It2 = zeros(1, t_len);
It3 = zeros(1, t_len);
It4 = zeros(1, t_len);
It5 = zeros(1, t_len);
It6 = zeros(1, t_len);

theta_ac = zeros(1, t_len);

Vd1 = zeros(1, t_len);
Vd2 = zeros(1, t_len);
Vd3 = zeros(1, t_len);
```

```
Vd4 = zeros(1, t_len);
Vd5 = zeros(1, t_len);
Vd6 = zeros(1, t_len);

Vt1 = zeros(1, t_len);
Vt2 = zeros(1, t_len);
Vt3 = zeros(1, t_len);
Vt4 = zeros(1, t_len);
Vt5 = zeros(1, t_len);
Vt6 = zeros(1, t_len);

da = zeros(1, t_len); % duty cycle array for phase leg a
db = zeros(1, t_len); % duty cycle array for phase leg b
dc = zeros(1, t_len); % duty cycle array for phase leg c

%% while loop for simulation
k = 1; % while loop counter
while (t(k) < tend)
    %% Phases and Angles
    theta_ac(k + 1) = (w_ac * t(k)) * 180 / pi;
    theta_ac(k + 1) = mod(theta_ac(k + 1), 360);
    %% Duty Cycles
    da(k) = m * cosd(theta_ac(k) - phi);
    db(k) = m * cosd(theta_ac(k) - phi - phi_dis);
    dc(k) = m * cosd(theta_ac(k) - phi + phi_dis);
    %% Switching logic
    if (da(k) >= tri(k))
        T1 = 1;
        T4 = 0;
    elseif (da(k) < tri(k))
        T1 = 0;
        T4 = 1;
    end
    if (db(k) >= tri(k))
        T2 = 1;
        T5 = 0;
    elseif (db(k) < tri(k))
        T2 = 0;
        T5 = 1;
    end
    if (dc(k) >= tri(k))
        T3 = 1;
        T6 = 0;
    elseif (dc(k) < tri(k))
        T3 = 0;
        T6 = 1;
    end
```

```
%% Phase Leg, Diodes, and Transistors Voltages
if T1 == 1
    Vag(k) = Vdc;
    Vt1(k) = 0;
    Vd1(k) = 0;
    Vt4(k) = Vdc;
    Vd4(k) = -Vdc;
elseif T4 == 1
    Vag(k) = 0;
    Vt4(k) = 0;
    Vd4(k) = 0;
    Vt1(k) = Vdc;
    Vd1(k) = -Vdc;
end
if T2 == 1
    Vbg(k) = Vdc;
    Vt2(k) = 0;
    Vd2(k) = 0;
    Vt5(k) = Vdc;
    Vd5(k) = -Vdc;
elseif T5 == 1
    Vbg(k) = 0;
    Vt5(k) = 0;
    Vd5(k) = 0;
    Vt2(k) = Vdc;
    Vd2(k) = -Vdc;
end
if T3 == 1
    Vcg(k) = Vdc;
    Vt3(k) = 0;
    Vd3(k) = 0;
    Vt6(k) = Vdc;
    Vd6(k) = -Vdc;
elseif T6 == 1
    Vcg(k) = 0;
    Vt6(k) = 0;
    Vd6(k) = 0;
    Vt3(k) = Vdc;
    Vd3(k) = -Vdc;
end
%% AC Voltages and Currents
Vas(k) = (2 / 3) * Vag(k) - (1 / 3) * Vbg(k) - (1 / 3) * Vcg(k);
Vbs(k) = (2 / 3) * Vbg(k) - (1 / 3) * Vag(k) - (1 / 3) * Vcg(k);
Vcs(k) = (2 / 3) * Vcg(k) - (1 / 3) * Vbg(k) - (1 / 3) * Vag(k);

Ias(k + 1) = Ias(k) + dt * (Vas(k) - r * Ias(k)) / l;
Ibs(k + 1) = Ibs(k) + dt * (Vbs(k) - r * Ibs(k)) / l;
```

```
Ics(k + 1) = Ics(k) + dt * (Vcs(k) - r * Ics(k)) / l;
%% Diodes and Transistors Currents
if (T1 == 1)
    if (Ias(k) >= 0)
        It1(k) = Ias(k);
        Id1(k) = 0;
    else
        Id1(k) = -Ias(k);
        It1(k) = 0;
    end
end
if (T2 == 1)
    if (Ibs(k) >= 0)
        It2(k) = Ibs(k);
        Id2(k) = 0;
    else
        Id2(k) = -Ibs(k);
        It2(k) = 0;
    end
end
if (T3 == 1)
    if (Ics(k) >= 0)
        It3(k) = Ics(k);
        Id3(k) = 0;
    else
        Id3(k) = -Ics(k);
        It3(k) = 0;
    end
end
if (T4 == 1)
    if (Ias(k) >= 0)
        Id4(k) = Ias(k);
        It4(k) = 0;
    else
        It4(k) = -Ias(k);
        Id4(k) = 0;
    end
end
if (T5 == 1)
    if (Ibs(k) >= 0)
        Id5(k) = Ibs(k);
        It5(k) = 0;
    else
        It5(k) = -Ibs(k);
        Id5(k) = 0;
    end
end
```

```
    if (T6 == 1)
        if (Ics(k) >= 0)
            Id6(k) = Ics(k);
            It6(k) = 0;
        else
            It6(k) = -Ics(k);
            Id6(k) = 0;
        end
    end

    Idc(k) = It1(k) - Id1(k) + It2(k) - Id2(k) + It3(k) - Id3(k);
    % Time array and steps update
    t(k + 1) = t(k) + dt;
    k = k + 1;
end

%% Calculate amplitudes of fundamental frequency component of Vas and Ias
[avg, ak, bk, rcon, err] = fourseries(t, Vas, T, 100);
Vamp = sqrt((ak(1)) ^ 2 + (bk(1)) ^ 2);
disp(Vamp);
[avg, ak, bk, rcon, err] = fourseries(t, Ias, T, 100);
Iamp = sqrt((ak(1)) ^ 2 + (bk(1)) ^ 2);
disp(Iamp);

%% Plots
figure(1)
subplot(2,1,1)
plot(theta_ac(200003:end), Vas(200003:end), ...
    theta_ac(200003:end), Vbs(200003:end), ...
    theta_ac(200003:end), Vcs(200003:end));
title('AC Voltages(V) vs \theta_{AC}(degree)')
grid on
legend('Vas','Vbs','Vcs')
subplot(2,1,2)
plot(theta_ac(200003:end), Ias(200003:end), ...
    theta_ac(200003:end), Ibs(200003:end), ...
    theta_ac(200003:end), Ics(200003:end));
title('AC Currents(A) vs \theta_{AC}(degree)')
grid on
legend('Ias','Ibs','Ics')

figure(2)
subplot(3,2,1)
plot(theta_ac(200003:end), Vd1(200003:end));
%xlim([0 360])
title('Vd1(V) vs \theta_{AC}(degree)')
subplot(3,2,3)
```

```
plot(theta_ac(200003:end), Vd2(200003:end));
%xlim([0 360])
title('Vd2(V) vs \theta_{AC}(degree)')
subplot(3,2,5)
plot(theta_ac(200003:end), Vd3(200003:end));
%xlim([0 360])
title('Vd3(V) vs \theta_{AC}(degree)')
subplot(3,2,2)
plot(theta_ac(200003:end), Vd4(200003:end));
%xlim([0 360])
title('Vd4(V) vs \theta_{AC}(degree)')
subplot(3,2,4)
plot(theta_ac(200003:end), Vd5(200003:end));
%xlim([0 360])
title('Vd5(V) vs \theta_{AC}(degree)')
subplot(3,2,6)
plot(theta_ac(200003:end), Vd6(200003:end));
%xlim([0 360])
title('Vd6(V) vs \theta_{AC}(degree)')

figure(3)
subplot(3,2,1)
plot(theta_ac(200003:end), Vt1(200003:end));
%xlim([0 360])
title('Vt1(V) vs \theta_{AC}(degree)')
subplot(3,2,3)
plot(theta_ac(200003:end), Vt2(200003:end));
%xlim([0 360])
title('Vt2(V) vs \theta_{AC}(degree)')
subplot(3,2,5)
plot(theta_ac(200003:end), Vt3(200003:end));
%xlim([0 360])
title('Vt3(V) vs \theta_{AC}(degree)')
subplot(3,2,2)
plot(theta_ac(200003:end), Vt4(200003:end));
%xlim([0 360])
title('Vt4(V) vs \theta_{AC}(degree)')
subplot(3,2,4)
plot(theta_ac(200003:end), Vt5(200003:end));
%xlim([0 360])
title('Vt5(V) vs \theta_{AC}(degree)')
subplot(3,2,6)
plot(theta_ac(200003:end), Vt6(200003:end));
%xlim([0 360])
title('Vt6(V) vs \theta_{AC}(degree)')

figure(4)
```

```
subplot(3,2,1)
plot(theta_ac(200003:end), Id1(200003:end));
%xlim([0 360])
title('Id1(A) vs \theta_{AC}(degree)')
subplot(3,2,3)
plot(theta_ac(200003:end), Id2(200003:end));
%xlim([0 360])
title('Id2(A) vs \theta_{AC}(degree)')
subplot(3,2,5)
plot(theta_ac(200003:end), Id3(200003:end));
%xlim([0 360])
title('Id3(A) vs \theta_{AC}(degree)')
subplot(3,2,2)
plot(theta_ac(200003:end), Id4(200003:end));
%xlim([0 360])
title('Id4(A) vs \theta_{AC}(degree)')
subplot(3,2,4)
plot(theta_ac(200003:end), Id5(200003:end));
%xlim([0 360])
title('Id5(A) vs \theta_{AC}(degree)')
subplot(3,2,6)
plot(theta_ac(200003:end), Id6(200003:end));
%xlim([0 360])
title('Id6(A) vs \theta_{AC}(degree)')

figure(5)
subplot(3,2,1)
plot(theta_ac(200003:end), It1(200003:end));
%xlim([0 360])
title('It1(A) vs \theta_{AC}(degree)')
subplot(3,2,3)
plot(theta_ac(200003:end), It2(200003:end));
%xlim([0 360])
title('It2(A) vs \theta_{AC}(degree)')
subplot(3,2,5)
plot(theta_ac(200003:end), It3(200003:end));
%xlim([0 360])
title('It3(A) vs \theta_{AC}(degree)')
subplot(3,2,2)
plot(theta_ac(200003:end), It4(200003:end));
%xlim([0 360])
title('It4(A) vs \theta_{AC}(degree)')
subplot(3,2,4)
plot(theta_ac(200003:end), It5(200003:end));
%xlim([0 360])
title('It5(A) vs \theta_{AC}(degree)')
subplot(3,2,6)
```

```
plot(theta_ac(200003:end), It6(200003:end));
%xlim([0 360])
title('It6(A) vs \theta_{AC}(degree)')

figure(6)
plot(theta_ac(200003:end), Idc(200003:end));
%xlim([0 360])
title('Idc(A) vs \theta_{AC}(degree)')

figure(7)
plot(theta_ac(200003:end), Vag(200003:end), theta_ac(200003:end), ...
    Vbg(200003:end), theta_ac(200003:end), Vcg(200003:end));
title('Phase leg Voltages(V) vs \theta_{AC}(degree)')
grid on
legend('Vag','Vbg','Vcg')

figure(8)
plot(t(1:end-1), da(1:end-1), t(1:end-1), db(1:end-1), t(1:end-1),
dc(1:end-1));
title('Duty Cycles vs \theta_{AC}(degree)')
grid on
legend('da','db','dc')
```