POWER ELECTRONICS

Project 3

Three-Phase Generator and Diode Rectifier Analysis

Vu Lo

Objectives

This project is to analyze and simulate the performance of a three-phase generator to a 6-diode rectifier with large DC inductance at output. This project utilizes MATLAB as the main coding program to simulate the voltages and currents that should be observed from the generator and the rectifier. With the implementation of Riemann sum, and Forward Euler algorithm, this project calculates the approximation of the behavior of the components in this circuit, shown in Figure 1.

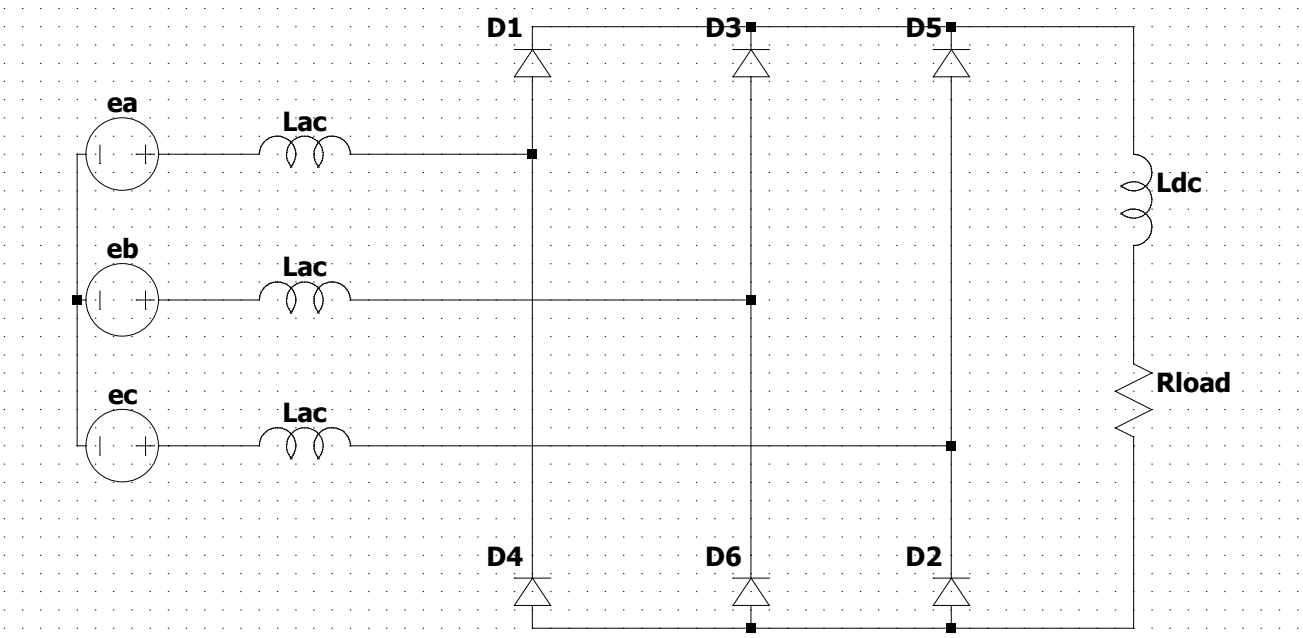

Fig. 1. Three-Phase Generator to Six-Diode Rectifier.

Analytical Analysis

The system to be analyzed is as shown in Figure 1 above.

Specifications

The three-phase generator has the source voltages in the form:

$$\epsilon_{as} = 392\cos(377t),$$
$$\epsilon_{bs} = 392\cos\left(377t - \frac{2\pi}{3}\right),$$
$$\epsilon_{cs} = 392\cos\left(377t + \frac{2\pi}{3}\right),$$

Where, 377 is the fundamental angular frequency ω, and $\frac{2\pi}{3}$ is the phase shift angle of each source voltage. The AC source inductance L_{ac} is set to be 1 mH, while the output to the rectifier has a 5 mH DC inductance and resistive load.

To analyze this inverter mathematically and logically, we need the below functions, which calculate θ, ω, and the fundamental frequency f, respectively:

$$\theta = \omega t,$$
$$\omega = 2\pi f,$$

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

From these functions, we can form the relationship between θ, ω, and f as:

$$\theta = \omega t = 2\pi f t$$

The three source voltages are described as waveforms, plotted against θ, in Figure 2 below, using the plot function in MATLAB, with the time step as small as $10^{-7}s$.

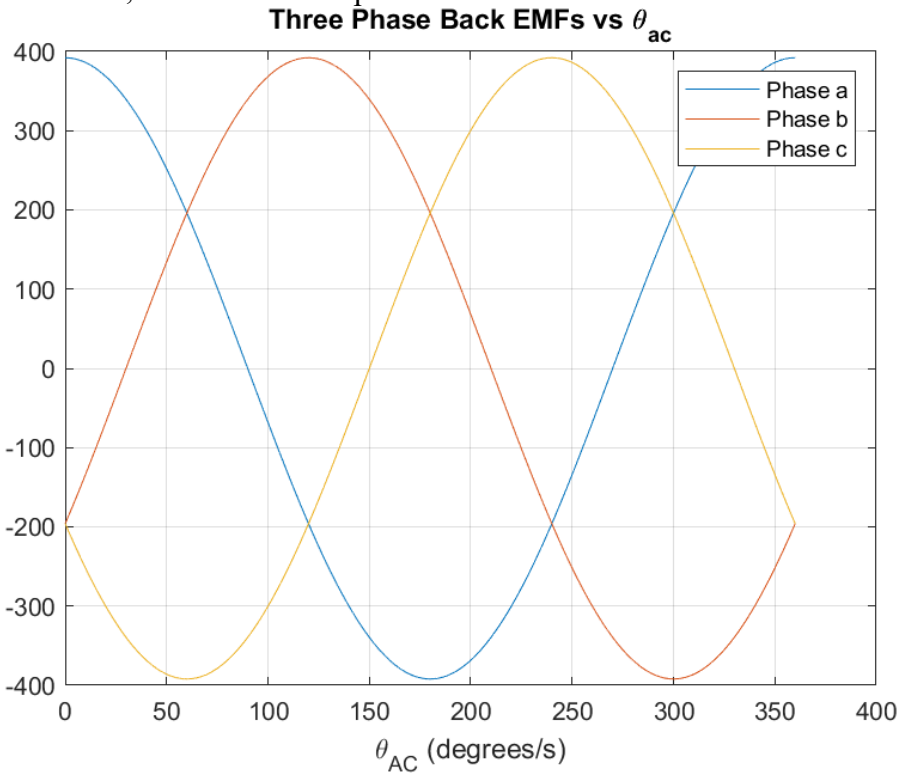

Fig. 2. Three Back EMFs vs θ_{ac}.

Procedure

Average Voltages and Currents

Using the equations for average voltage and current calculation for 3 Modes in a 3-phase generator, as shown in Table 1 below, we can arrive at the boundary values of voltages and currents for each Mode, as shown in Table 2.

Table 1. Average Voltage and Current Formulas for 3-Phase Generator Modes

Mode	$\langle V'_{DC} \rangle$	I_{DC}	Range
1	$\frac{3\sqrt{3}E}{2\pi}(1+\cos\gamma)$	$\frac{\sqrt{3}E}{2\omega L}(1-\cos\gamma)$	$0 \le \gamma \le 60$
2	$\frac{9E}{2\pi}\cos\left(\alpha+\frac{\pi}{6}\right)$	$\frac{\sqrt{3}E}{2\omega L}\sin\left(\alpha+\frac{\pi}{6}\right)$	$0 \le \alpha \le 30$
3	$\frac{9E}{2\pi}(1-\sin\left(\delta+\frac{\pi}{6}\right))$	$\frac{E}{2\omega L}(1+\sin\left(\delta+\frac{\pi}{6}\right))$	$0 \le \delta \le 60$

Table 2. Voltage and Current Boundary Values for Each Mode.

Mode	$\langle V'_{DC} \rangle$ (V)	I_{DC} (A)
1, at 0 degree	648.363	0
1, at 60 degrees	486.272	450.241
2, at 0 degree	486.272	450.241
2, at 30 degrees	280.749	779.841
3, at 0 degree	280.749	779.841
3, at 60 degrees	0	1039.788

The values from Table 2 will be used to be compared with the MATLAB Simulation section.

Commutation and Delay Scenarios

Commutation Degree of 45

For the case of a load resistance that leads to a commutation of 45 degrees, the system is in Mode 1. The general rule with commutation angle in Mode 1 is shown in the figure below, where it examines at a certain angle interval, which diodes are working, and at the same time, shows what the behavior of phase a current is like using KCL at that interval of angle.

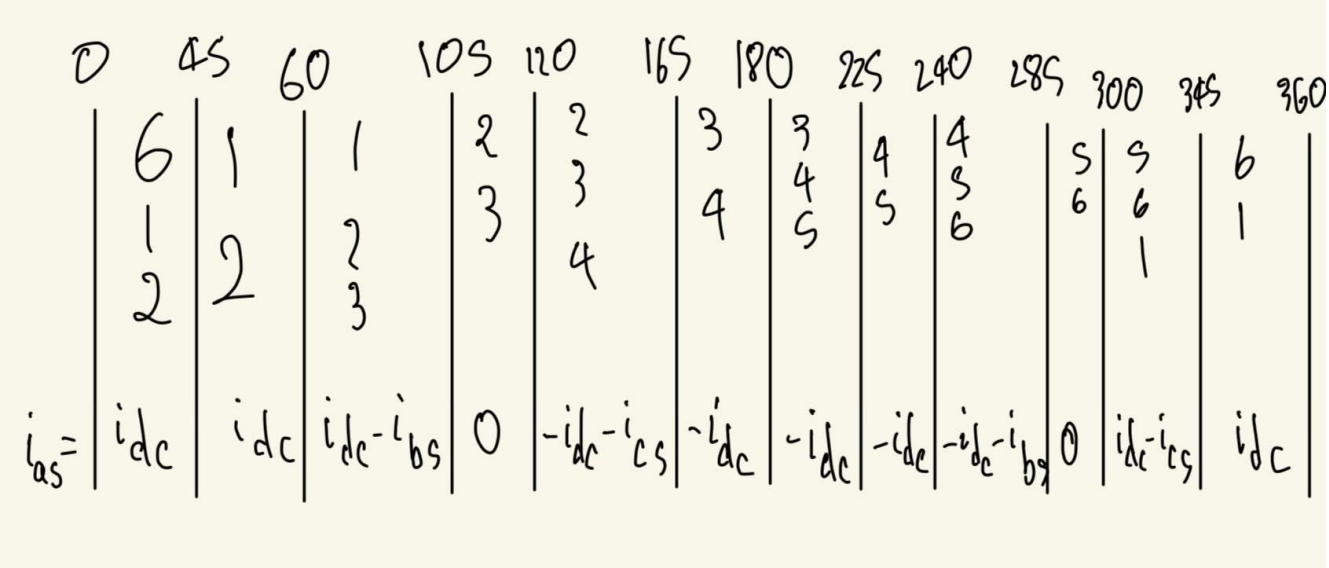

Fig. 3. Commutation Degree, Diodes Logic, and KCL of Phase-a Current.

Using derivative properties with i_{DC} is a constant value (which turns it value to 0), and KVL, we achieve the following equation to calculate the phase a current when it's not equal to 0 or i_{DC}:

$$i_{as}(\theta) = \begin{cases} \int_{\theta_0}^{\theta'} \frac{e_a - e_b}{2\omega L} d\theta + i_{as}(\theta_0), for\ when\ i_{bs}\ is\ present\ in\ the\ relationship \\ \int_{\theta_0}^{\theta'} \frac{e_a - e_c}{2\omega L} d\theta + i_{as}(\theta_0),, for\ when\ i_{cs}\ is\ present\ in\ the\ relationship \end{cases}$$

Which equals, with the function in accordance with the integrals above:

$$i_{as}(\theta) = \begin{cases} \frac{196}{\omega L}(\sin(\theta') - \sin(\theta' - 120) - \sin(\theta_0) + \sin(\theta_0 - 120)) + i_{as}(\theta_0) \\ \frac{196}{\omega L}(\sin(\theta') - \sin(\theta' + 120) - \sin(\theta_0) + \sin(\theta_0 + 120)) + i_{as}(\theta_0) \end{cases}$$

Also, using Ohm's Law in Mode 1, the load resistance for this commutation degree is:

$$R = \frac{V}{I} = \frac{V_{DC}}{I_{DC}} = \frac{\frac{3\sqrt{3}E}{2\pi}(1+\cos 45)}{\frac{\sqrt{3}E}{2\omega L}(1-\cos 45)} = 2.098\Omega$$

And the value of I_{DC} using Mode 1 equation with the 45-degree angle commutation is 263.751A.

Commutative Degree of 60 and Delay of 15

For the case of a load resistance that leads to a commutation of 60 degrees, and a delay of 15 degrees, the system is in Mode 2. The general rule with commutation angle in Mode 2 is shown in the figure below, where it examines at a certain angle interval, which diodes are working, and at the same time, shows what the behavior of phase a current is like using KCL at that interval of angle.

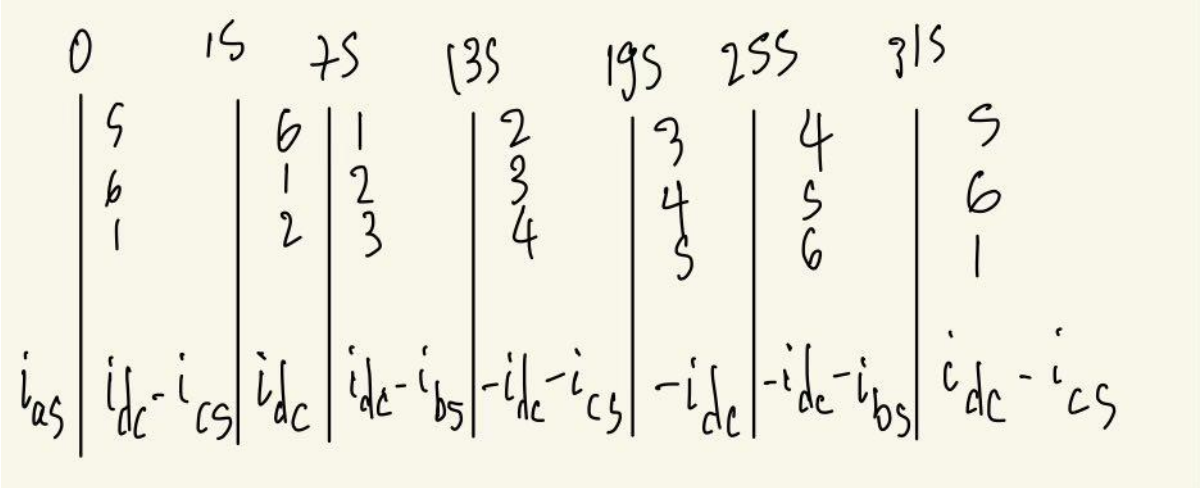

Fig. 4. Commutation Degree and Delay, Diodes Logic, and KCL of Phase-a Current.

Using derivative properties with i_{DC} is a constant value (which turns it value to 0), and KVL, we achieve the following equation to calculate the phase a current when it's not equal to i_{DC}:

$$i_{as}(\theta) = \begin{cases} \int_{\theta_0}^{\theta'} \frac{e_a - e_b}{2\omega L} d\theta + i_{as}(\theta_0), for\ when\ i_{bs}\ is\ present\ in\ the\ relationship \\ \int_{\theta_0}^{\theta'} \frac{e_a - e_c}{2\omega L} d\theta + i_{as}(\theta_0), for\ when\ i_{bs}\ is\ present\ in\ the\ relationship \end{cases}$$

Which equals, with the function in accordance with the integrals above:

$$i_{as}(\theta) = \begin{cases} \frac{196}{\omega L}(\sin(\theta') - \sin(\theta' - 120) - \sin(\theta_0) + \sin(\theta_0 - 120)) + i_{as}(\theta_0) \\ \frac{196}{\omega L}(\sin(\theta') - \sin(\theta' + 120) - \sin(\theta_0) + \sin(\theta_0 + 120)) + i_{as}(\theta_0) \end{cases}$$

Also, using Ohm's Law in Mode 2, the load resistance for this commutation degree is:

$$R = \frac{V}{I} = \frac{V_{DC}}{I_{DC}} = \frac{\frac{9E}{2\pi}\cos\left(15° + \frac{\pi}{6}\right)}{\frac{\sqrt{3}E}{2\omega L}\sin\left(15° + \frac{\pi}{6}\right)} = 0.62355\Omega$$

And the value of I_{DC} using Mode 2 equation with the 15-degree angle delay is 636.737A

Maximum Power for DC Load

For the load resistance calculation for maximum power to DC Load, we can look at Mode 2 to optimize. This is due to the fact that the relationship between DC voltage and DC current for Mode 2 is not as

linear as it is for Mode 1 and 3, which means that in Mode 2, the relationship between voltage and current has a peak.
Using the Power calculation equation for Mode 2, we have:

$$P(\alpha) = V(\alpha)I(\alpha) = \frac{9E^2\sqrt{3}}{8\pi\omega L}\sin\left(2\alpha + \frac{\pi}{3}\right),$$

Using derivative property, we have:

$$\frac{dP}{dt} = \cos\left(2\alpha + \frac{\pi}{3}\right) = 0,$$

Using trigonometry rules, we know that, for a cosine function to be 0, the angle is in the form of $\theta = k\pi + \frac{\pi}{2}$, paired with the fact that $0 \leq \alpha \leq 30$, we conclude that $\alpha = 15 = \frac{\pi}{12}$. Therefore, the max power and the load resistance needed are:

$$P\left(\frac{\pi}{12}\right) = \frac{9E^2\sqrt{3}}{8\pi\omega L}\sin\left(\frac{\pi}{6} + \frac{\pi}{3}\right) = 252815.8467W,$$

$$R = \frac{V}{I} = \frac{V_{DC}}{I_{DC}} = \frac{\frac{9E}{2\pi}\cos\left(15° + \frac{\pi}{6}\right)}{\frac{\sqrt{3}E}{2\omega L}\sin\left(15° + \frac{\pi}{6}\right)} = 0.62355\Omega$$

MATLAB Simulation

MATLAB Analytical Setup Simulation

Average Voltages and Currents

Using the functions provided in Table 1, we derive a function in MATLAB to simulate the behavior of the voltages of each Mode in accordance with the currents as shown below. The points highlighted in the plot are to show the boundary values at each mode, which are equal to the values calculated analytically. The MATLAB code to calculate the process is as shown in the *dcv_i.m* as shown in the MATLAB Script section.

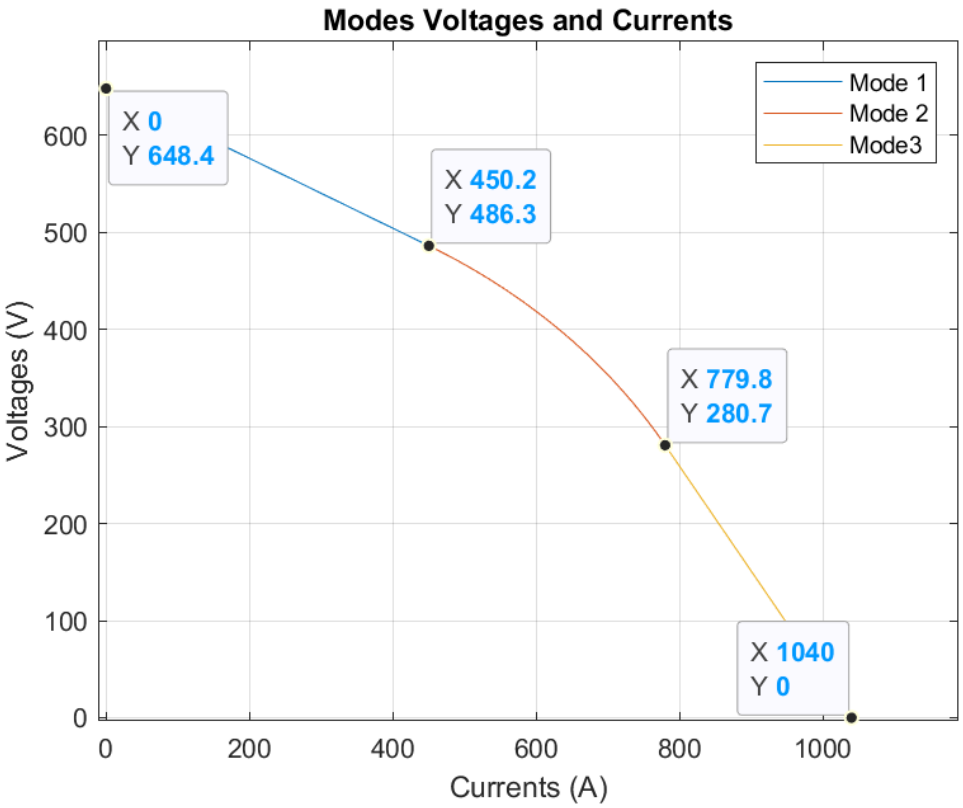

Fig. 5. MATLAB Analytical Simulation of Modes' Voltages and Currents.

Commutation and Delay Scenarios

Commutation Degree of 45

Using the code provided in *commutation.m* as shown in MATLAB Scripts section, with the equations and diode logic derived, we achieve the plot for phase a current. Using phase a current, shift the plot by 120 degree, we also achieve phase b and c currents plot against θ. The plot for all three phase currents is shown below. The data points are to be compared with the logic in the Analysis section.

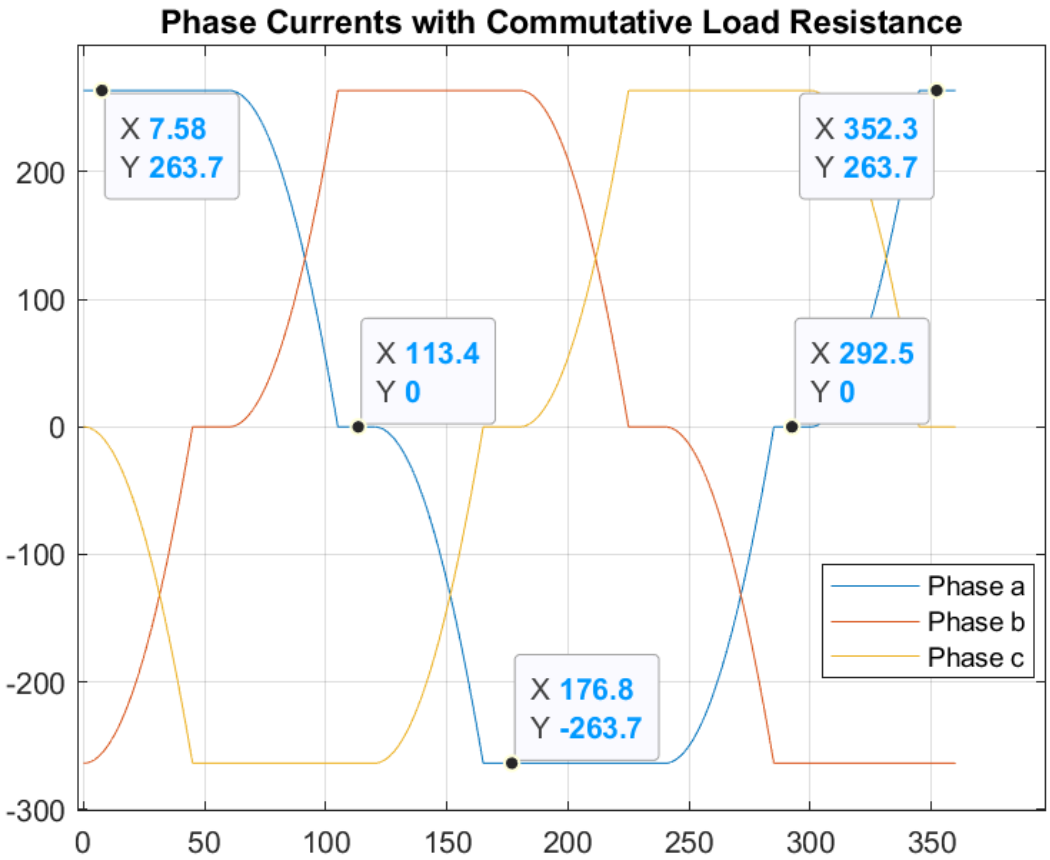

Fig. 6. Phase Currents with Commutative Load Resistance.

Commutative Degree of 60 and Delay of 15

Using the code provided in *delay.m* as shown in MATLAB Scripts section, with the equations and diode logic derived, we achieve the plot for phase a current. Using phase a current, shift the plot by 120 degree, we also achieve phase b and c currents plot against θ. The plot for all three phase currents is shown below. The data points are to be compared with the logic in the Analysis section.

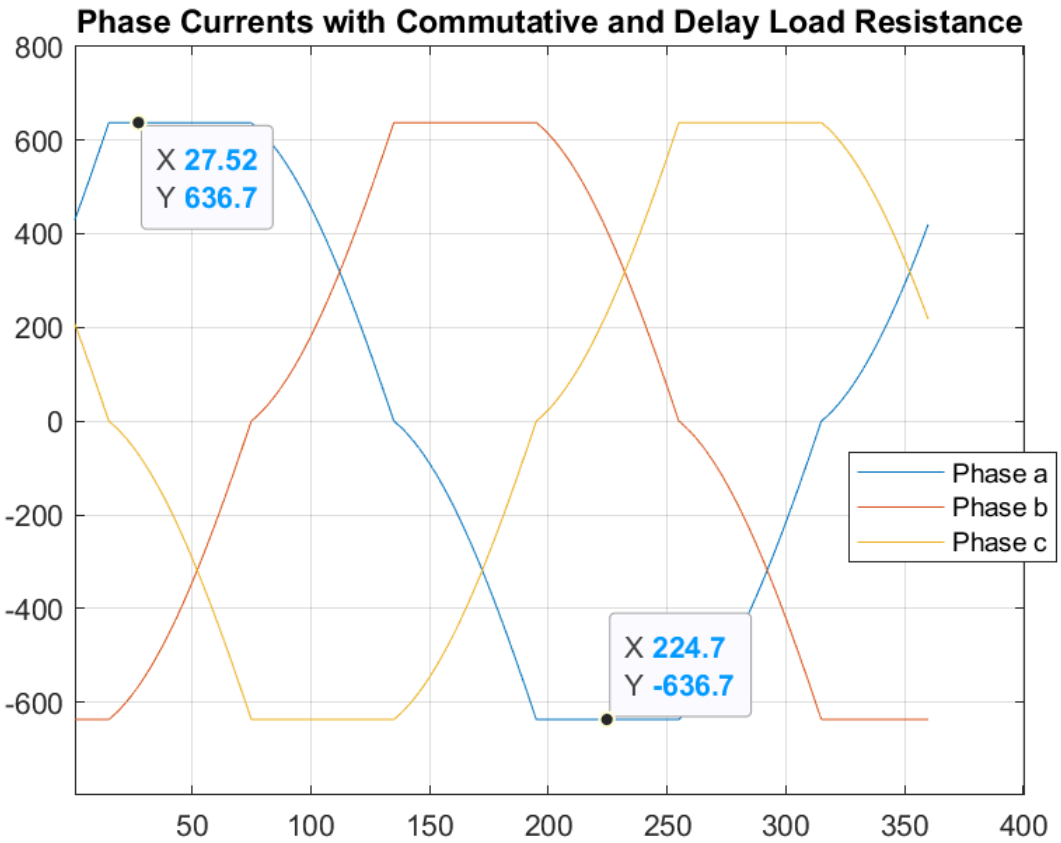

Fig. 7. Phase Currents with Commutative and Delay Load Resistance.

Maximum Power for DC Load

Using the code provided in *genproc.m, Maximum Power Load* subsection as shown in MATLAB Scripts section, we calculated the maximum power and the load resistance needed for that case. To perform the calculations, we form 2 arrays for DC voltages and currents for all modes. From then, we calculate a power array by multiplying the 2 arrays and find the maximum value and its index within that power array. Using that index, we find the voltage at that point of maximum power, we use the equation below to calculate the load resistance:

$$R = \frac{V^2}{P}$$

The results are:

$$P_{max} = 2.5281 * 10^5 W$$
$$R_{min} = 0.6236\Omega$$

MATLAB Forward Euler Simulation

Forward Euler Setup

To set up the Forward Euler, we provide *forwardEuler.m* as shown in MATLAB Scripts section. This function simulates the system using a Forward Euler algorithm. It takes inputs of an array of resistive loads, phase shift angle, fundamental period, back EMF amplitudes, AC inductance, and DC inductance. It outputs the phase currents arrays, angle θ, and DC voltage and current arrays.

```
function[Ias, Ibs, Ics, theta, Idc, Vdc] = forwardEuler(Rload, E_amp, T)
```

This function calculates the voltages and currents utilizing the Forward Euler algorithm, which can be described as the equation below:

$$y_{n+1} = y_n + hf(t_n, y_n),$$

Where h is the step size, f is defined by $f(t, y) = y(t)$, n denotes the stage that we are on, and y is the function in question.

However, we might run into noise issue, which would be reduced greatly with some additional parameters, ϵ and a transfer function, which is used for calculating I_{DC}. We also had an added parameter of DC inductance, 5 mH, which allows the system to have a low ripple DC current. This is all to help with the noise we might encounter when plotting.

Average function

To set up the Forward Euler, we provide *average.m*as shown in MATLAB Scripts section. This function calculates average value using Riemann sum where it takes in dt - time step, x - the waveform that is being analyzed, T - its fundamental period of the waveform; and outputs the avg – average value.

```
function[avg] = average(dt,x,T)
```

Resistance Values of Mode 1 and Mode 2

Using the Mode 1 and Mode 2 Voltage and Current equations:

$$R_{Mode1} = \frac{V_{Mode1}}{I_{Mode1}} = \frac{\frac{3\sqrt{3}E}{2\pi}(1+\cos\gamma)}{\frac{\sqrt{3}E}{2\omega L}(1-\cos\gamma)},$$

$$R_{Mode2} = \frac{V_{Mode2}}{I_{Mode2}} = \frac{\frac{9E}{2\pi}\cos\left(\alpha+\frac{\pi}{6}\right)}{\frac{\sqrt{3}E}{2\omega L}\sin\left(\alpha+\frac{\pi}{6}\right)},$$

Also utilize an increment of 5° for both Mode 1 and Mode 2 calculations, we achieve resistances for Forward Euler calculations as shown in the table below:

Table 3. Resistance Values for Mode 1 and Mode 2.

Angle (degrees)		$R_{load}(\Omega)$
γ	0	infinite
	5	188.85
	10	47.03
	15	20.77
	20	11.58
	25	7.33
	30	5.01
	35	3.62
	40	2.72
	45	2.1
	50	1.66
	55	1.33
	60	1.08
α	0	1.08
	5	0.89
	10	0.74
	15	0.62

	20	0.52
	25	0.44
	30	0.36

Average Voltages and Currents

Using the Forward Euler as mentioned above in the *forwardEuler.m* script, and the two modes' load resistances values calculated in Table 3, we achieved the graph plotting the DC Voltages and Currents comparison between the analytical and Forward Euler simulation is as shown in Figure 8 below:

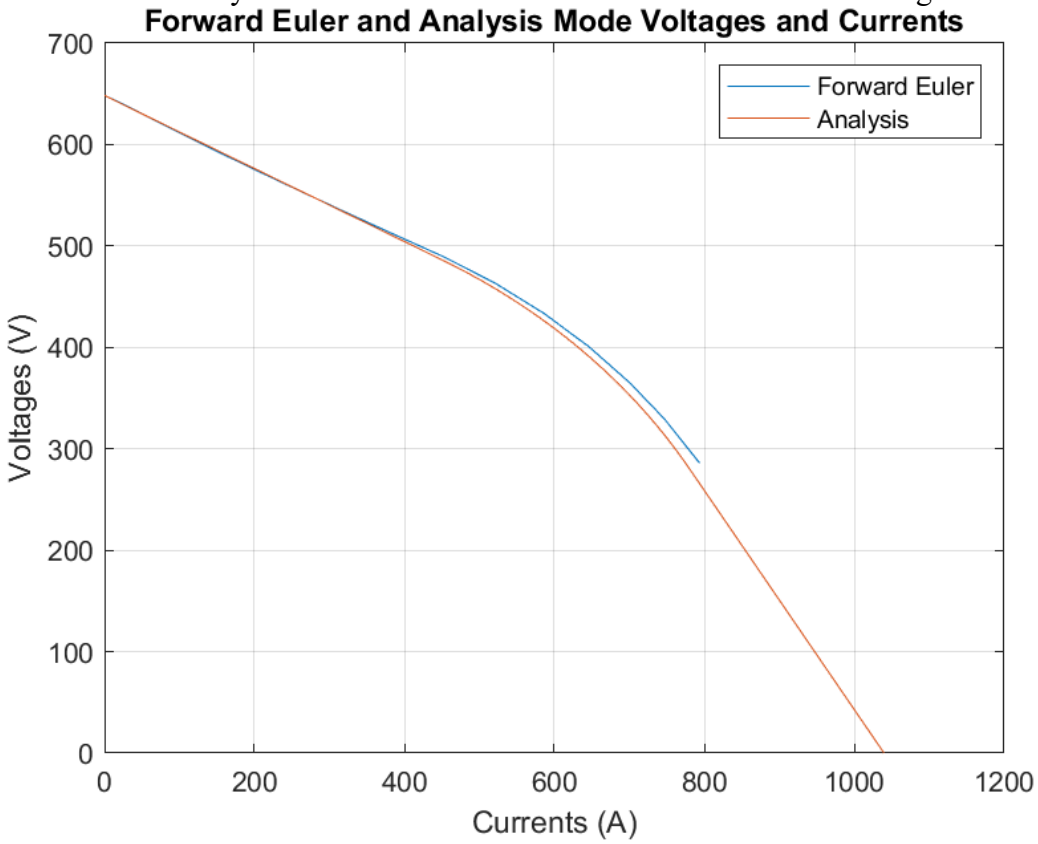

Fig. 8. Analytically and Forward Euler Calculated Mode Voltages and Currents.

Commutation Degree Scenario

For the scenario when load resistance of 2.098Ω has the system under 45° commutation, we call *forwardEuler.m* to calculate the waveform of the DC current and the phase currents of said scenario as the figures below, with the average $I_{DC} = 263.6828A$ calculated by the Forward Euler method:

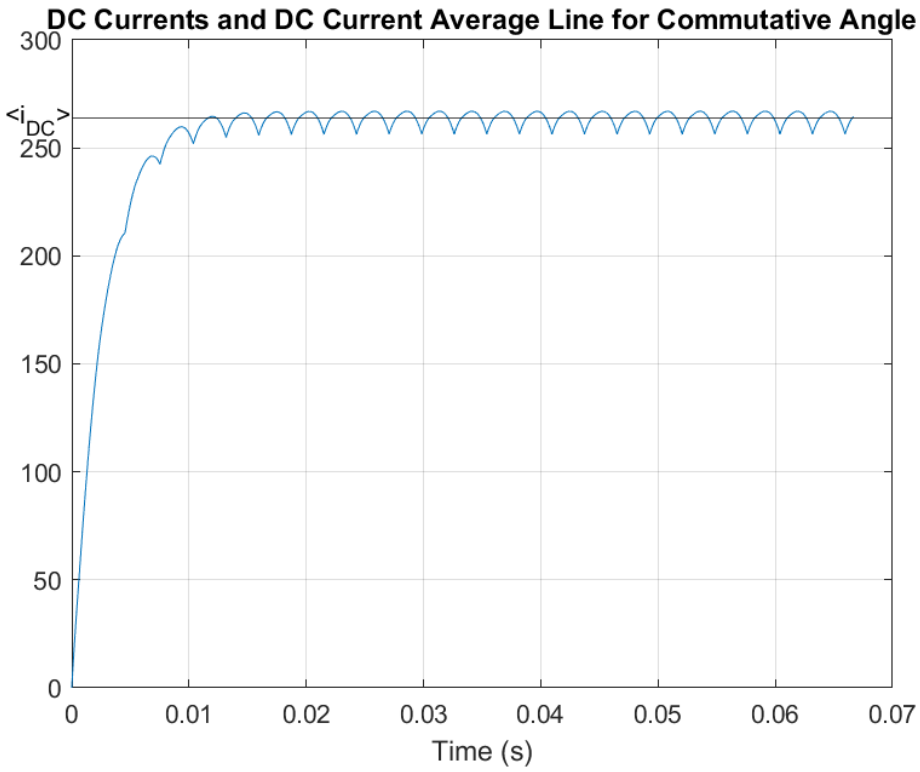

Fig. 9. DC Current with Its Average Value for 45° Commutation.

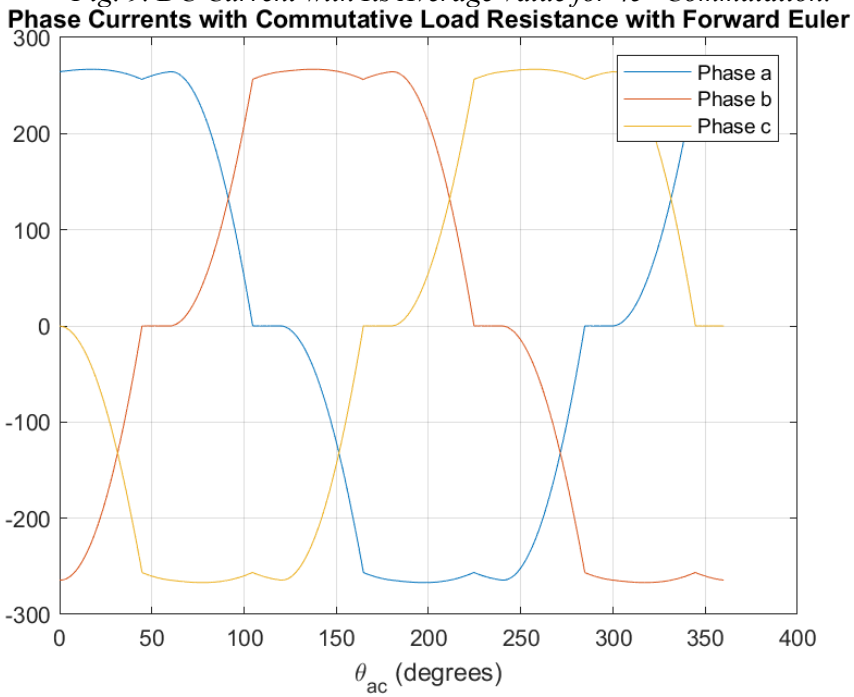

Fig. 10. Phase Currents with 45° Commutation by Forward Euler.

Commutation Degree and Delay Degree Scenario

For the scenario when load resistance of 0.62355Ω has the system under 60° commutation and 15° delay, we call *forwardEuler.m* to calculate the waveform of the DC current and the phase currents of said

scenario as the figures below, with the average $I_{DC} = 644.3169A$ calculated by the Forward Euler method:

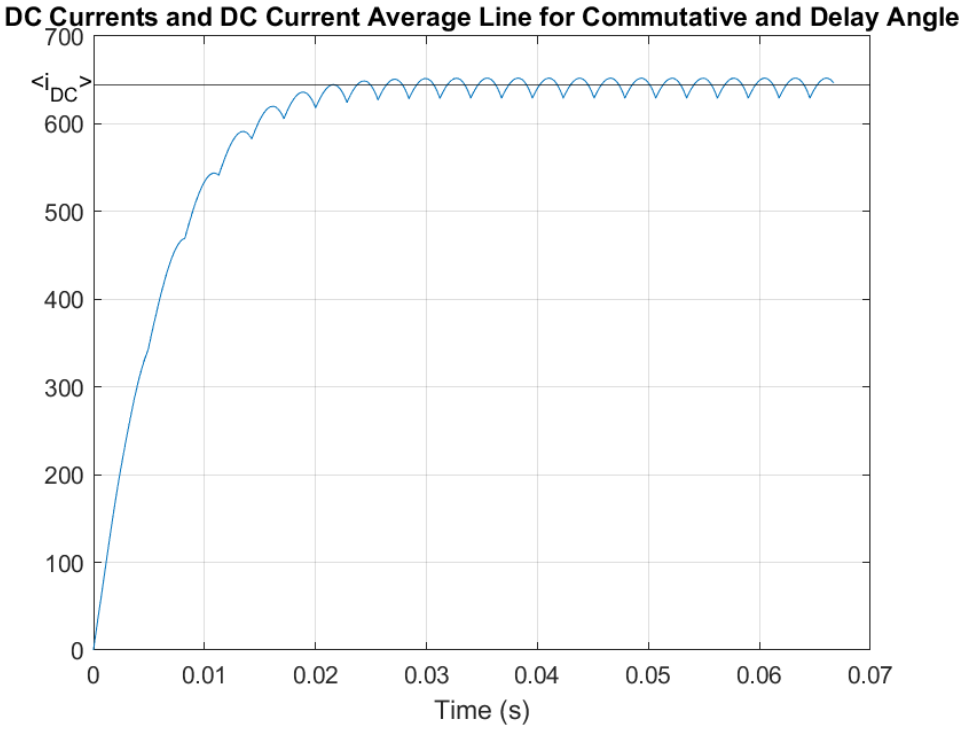

Fig. 11. DC Current with Its Average Value for Commutation and Delay Angle.

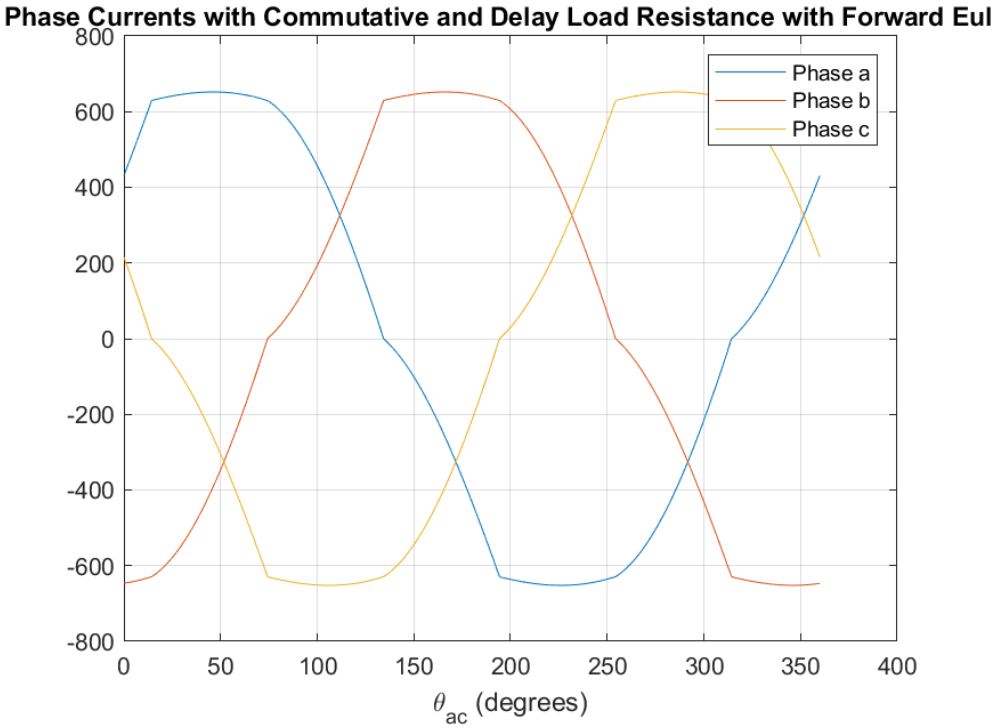

Fig. 12. Phase Currents with Commutation and Delay Angle by Forward Euler.

Results and Conclusion

Results

For the results of this project, we reevaluate each part of the analysis compared to the according part in the MATLAB Simulation part by comparing the MATLAB Simulation according to the analysis and the Forward Euler algorithm.

Average Voltages and Currents

For the average output voltage and current behavior for mode 1, 2, and 3, we can observe from Figure 5 and Figure 8 that the behavior of the voltage and current agrees for mode 1, but started to separate away as it goes deeper into mode 2 and towards mode 3. I believe the Forward Euler method isn't as accurate when our system moves into mode 3, so it was not plotted for that mode.

Commutation and Delay Scenarios

Commutation Degree of 45

For the 45° commutative angle scenario, we can observe from Figure 6 and Figure 10 that the behavior of the phase currents agrees with the logic demonstrated in the analysis. At the same time, the DC current as analysis is 263.751A, while that as Forward Euler algorithm is 263.6828A. This gives the error percentage of as low as 0.0259%.

Commutative Degree of 60 and Delay of 15

For the 60° commutative and 15° delay angles scenario, we can observe from Figure 7 and Figure 12 that the behavior of the phase currents agrees with the logic demonstrated in the analysis. At the same time, the DC current as analysis is 636.737A, while that as Forward Euler algorithm is 644.3169A. This gives the error percentage of as low as 1.1764%, but as we can see it's a bit higher than the scenario before, since this scenario, the system is in mode 2.

Conclusion

As seen from the results section, the error percentages were acceptable for the system. Hence, Forward Euler algorithm is effective enough to be put into consideration when analyzing the system of 3-phase generator to 6-diode rectifier. However, it should be noted that Forward Euler would only be keeping its integrity for mode 1 and 2.

Appendix

As mentioned below in MATLAB Scripts section, *genproc.m* is the main script that calls and plots the acquired graphs and calculations in this paper.

MATLAB Scripts

```
%% genproc.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%{
This function is to simulate the system and plot the required graphs.
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear;
```

```
%% Initial Parameters
E_amp = 392;
w = 377;
T = 2 * pi / w;
phi = 120; % Phase angle
l = 1e-3;
x_ac = w * l;
gam1 = 45;
gam2 = 60;
d = 15;
l_dc = 5e-3;
%% Time and Angle Arrays
dt = 1e-7;
t = 0:dt:T;
t2 = 0:dt:4*T; % Same in forwardEuler.m
theta = 0:0.01:360;
theta_len = length(theta);
%% Arrays of Mode Voltages and Currents
V_m1 = zeros(1,60);
V_m2 = zeros(1,30);
V_m3 = zeros(1,60);
i_m1 = zeros(1,60);
i_m2 = zeros(1,30);
i_m3 = zeros(1,60);
%% Arrays of Phase Currents in Commutation and Delay Cases
i_as45 = zeros(1,theta_len);
i_bs45 = zeros(1,theta_len);
i_cs45 = zeros(1,theta_len);
i_asd = zeros(1,theta_len);
i_bsd = zeros(1,theta_len);
i_csd = zeros(1,theta_len);
%% Arrays of Forward Euler Mode Voltages and Currents
V_m1fe = zeros(1,60);
V_m2fe = zeros(1,30);
V_m3fe = zeros(1,60);
i_m1fe = zeros(1,60);
i_m2fe = zeros(1,30);
i_m3fe = zeros(1,60);
%% Resistance Values for Forward Euler calculations
R_1 = [inf 188.85 47.03 20.77 11.58 7.33 5.01 3.62 2.72 2.1 1.66 1.33
1.08];
R_2 = [1.08 0.89 0.74 0.62 0.52 0.44 0.36];
R = [R_1, R_2];
Rlen = length(R);
Vfe_fe = zeros(1,Rlen); % Voltages calculated using Forward Euler
Ife_fe = zeros(1,Rlen); % Currents calculated using Forward Euler
```

```
%% PLOTS & CALCULATIONS
%% Back EMFs vs Theta
e_as = E_amp * cosd(theta);
e_bs = E_amp * cosd(theta - phi);
e_cs = E_amp * cosd(theta + phi);
figure(1)
plot(theta, e_as, theta, e_bs, theta, e_cs)
title('Three Phase Back EMFs vs \theta_{ac}');
legend('Phase a','Phase b','Phase c')
grid on
%% Average Voltages and DC Currents Plot
[V_m1, i_m1, V_m2, i_m2, V_m3, i_m3] = dcv_i(E_amp, x_ac);
figure(2)
plot(i_m1, V_m1, i_m2, V_m2, i_m3, V_m3)
title('Modes Voltages and Currents');
xlabel('Currents (A)')
ylabel('Voltages (V)')
grid on
legend('Mode 1', 'Mode 2', 'Mode3')
%% Phase Currents vs Theta_ac with commutation of 45
[i_as45, i_bs45, i_cs45] = commutation(gam1, x_ac, i_m1(46), theta);
figure(3)
plot(theta, i_as45, theta, i_bs45, theta, i_cs45)
title('Phase Currents with Commutative Load Resistance');
grid on
legend('Phase a', 'Phase b', 'Phase c')
%% Phase Currents vs Theta_ac with commutation of 60 & delay of 15
[i_asd, i_bsd, i_csd] = delay(gam2, x_ac, i_m2(16), theta, d);
figure(4)
plot(theta, i_asd, theta, i_bsd, theta, i_csd)
title('Phase Currents with Commutative and Delay Load Resistance');
grid on
legend('Phase a', 'Phase b', 'Phase c')
%% Maximum Power Load
P = zeros(1,150); % Power array
V = [V_m1, V_m2, V_m3];
I = [i_m1, i_m2, i_m3];
P = V .* I;
P_max = max(P);
P_pos = find(P == P_max);
R_min = V(P_pos) ^ 2 / P_max;
disp(P_max);
disp(R_min);
%% Forward Euler Mode Voltages and Currents
% Note: the indexing seen here in every arrays is to compensate for the mod
% function used earlier to calculate and update theta. 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.
[V_m1fe, i_m1fe, V_m2fe, i_m2fe, V_m3fe, i_m3fe] = dcv_i(E_amp, x_ac);
Vfe_a = [V_m1fe, V_m2fe, V_m3fe];
Ife_a = [i_m1fe, i_m2fe, i_m3fe];
for i = 1:Rlen
    [~, ~, ~, ~, Idc_fe, Vdc_fe] = forwardEuler(R(i), E_amp, T);
    Vdc_fe_fe(i) = average(dt, Vdc_fe, T);
    Idc_fe_fe(i) = average(dt, Idc_fe, T);
end
figure(5)
plot(Idc_fe_fe, Vdc_fe_fe, Ife_a, Vfe_a)
title('Forward Euler and Analysis Mode Voltages and Currents');
xlabel('Currents (A)')
ylabel('Voltages (V)')
grid on
legend('Forward Euler', 'Analysis')
%% Commutation Degree of 45 Scenario with Forward Euler Algorithm
Rcomm = 2.098;
[Ias45, Ibs45, Ics45, theta45, Idc45, ~] = forwardEuler(Rcomm, E_amp, T);
Idc_comm = average(dt, Idc45, T);
disp(Idc_comm);
figure(6)
plot(t2, Idc45(1:end-1))
yline(Idc_comm,'k')
text(0, Idc_comm, "<i_{DC}>", "HorizontalAlignment", "right");
title('DC Currents and DC Current Average Line for Commutative Angle');
xlabel('Time (s)')
grid on
figure(7)
plot(theta45(500002:end), Ias45(500002:end), ...
     theta45(500002:end), Ibs45(500002:end), ...
     theta45(500002:end), Ics45(500002:end))
xlabel('\theta_{ac} (degrees)')
title('Phase Currents with Commutative Load Resistance with Forward
Euler');
grid on
legend('Phase a', 'Phase b', 'Phase c')
%% Commutation Degree and Delay Scenario with Forward Euler Algorithm
Rdelay = 0.62355;
[Ias60, Ibs60, Ics60, theta60, Idc60, ~] = forwardEuler(Rdelay, E_amp, T);
Idc_delay = average(dt, Idc60, T);
disp(Idc_delay);
figure(8)
plot(t2, Idc60(1:end-1))
yline(Idc_delay,'k')
```

```
text(0, Idc_delay, "<i_{DC}>", "HorizontalAlignment", "right");
title('DC Currents and DC Current Average Line for Commutative and Delay
Angle');
xlabel('Time (s)')
grid on
figure(9)
plot(theta60(500002:end), Ias60(500002:end), ...
     theta60(500002:end), Ibs60(500002:end), ...
     theta60(500002:end), Ics60(500002:end))
xlabel('\theta_{ac} (degrees)')
title('Phase Currents with Commutative and Delay Load Resistance with
Forward Euler');
grid on
legend('Phase a', 'Phase b', 'Phase c')

%% dcv_i.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%{
This function is to simulate the average output voltage versus
current for modes 1, 2, and 3. It takes in E_amp - back EMF
amplitude, x_ac - product of fundamental angular frequency and AC source
inductance. It returns the modes' voltages and currents to plot.dc
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function[V_m1, i_m1, V_m2, i_m2, V_m3, i_m3] = dcv_i(E_amp, x_ac)
    %% Mode 1 Average Voltage and DC Current
    for gamma = 0:60
        V_m1(gamma + 1) = 3 * sqrt(3) * E_amp * (1 + cosd(gamma)) / (2 *
pi);
        i_m1(gamma + 1) = sqrt(3) * E_amp * (1 - cosd(gamma)) / (2 * x_ac);
    end

    %% Mode 2 Average Voltage and DC Current
    for alpha = 0:30
        V_m2(alpha + 1) = 9 * E_amp * cosd(alpha + 30) / (2 * pi);
        i_m2(alpha + 1) = sqrt(3) * E_amp * sind(alpha + 30) / (2 * x_ac);
    end

    %% Mode 3 Average Voltage and DC Current
    for delta = 0:60
        V_m3(delta + 1) = 9 * E_amp * (1 - sind(delta + 30)) / (2 * pi);
        i_m3(delta + 1) = E_amp * (1 + sind(delta + 30)) / (2 * x_ac);
    end

%% commutation.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%{
This function is to simulate the system with a commutation angle.
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function[i_as, i_bs, i_cs] = commutation(g, x, i_dc, theta_ac)
    for k = 1:length(theta_ac)
        %% Phase a current
        if(theta_ac(k) >= 0 && theta_ac(k) <= 60)
            i_as(k) = i_dc;
        elseif(theta_ac(k) > 60 && theta_ac(k) < (60 + g))
            i_as(k) = 196 * (sind(theta_ac(k)) - ...
                      sind(theta_ac(k) - 120) ...
                      - sqrt(3)) / x + i_dc;
        elseif(theta_ac(k) >= (60 + g) && theta_ac(k) < (120))
            i_as(k) = 0;
        elseif(theta_ac(k) >= 120 && theta_ac(k) < (120 + g))
            i_as(k) = 196 * (sind(theta_ac(k)) - ...
                      sind(theta_ac(k) + 120) ...
                      - sqrt(3)) / x;
        elseif(theta_ac(k) >= 165 && theta_ac(k) < 240)
            i_as(k) = -i_dc;
        elseif(theta_ac(k) >= 240 && theta_ac(k) < (240 + g))
            i_as(k) = 196 * (sind(theta_ac(k)) - ...
                      sind(theta_ac(k) - 120) ...
                      + sqrt(3)) / x - i_dc;
        elseif(theta_ac(k) >= (240 + g) && theta_ac(k) < 300)
            i_as(k) = 0;
        elseif(theta_ac(k) >= 300 && theta_ac(k) < (300 + g))
            i_as(k) = 196 * (sind(theta_ac(k)) - ...
                      sind(theta_ac(k) + 120) ...
                      + sqrt(3)) / x;
        elseif(theta_ac(k) >= (300 + g) && theta_ac(k) <= 360)
            i_as(k) = i_dc;
        end
    end

    shift_num = round(length(theta_ac) * 120 / 360);
    i_bs = circshift(i_as, [0, shift_num]);
    i_cs = circshift(i_as, [0, -shift_num]);
end
%% delay.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%{
This function is to simulate the system with a delay and a commutation
angle.
%}
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function[i_as, i_bs, i_cs] = delay(g, x, i_dc, theta_ac, d)
    for k = 1:length(theta_ac)
        %% Phase a current
        if(theta_ac(k) >= 0 && theta_ac(k) <= d)
            i_as(k) = 196 * (sind(theta_ac(k)) - ...
                      sind(theta_ac(k) + 120) - sind(315) ...
                      + sind(435)) / x;
        elseif(theta_ac(k) > d && theta_ac(k) < (g + d))
            i_as(k) = i_dc;
        elseif(theta_ac(k) >= (g + d) && theta_ac(k) < (60 + g + d))
            i_as(k) = 196 * (sind(theta_ac(k)) - ...
                      sind(theta_ac(k) - 120) - sind(g + d) ...
                      + sind(g + d - 120)) / x + i_dc;
        elseif(theta_ac(k) >= (60 + g + d) && theta_ac(k) < (120 + g + d))
            i_as(k) = 196 * (sind(theta_ac(k)) - ...
                      sind(theta_ac(k) + 120) - sind(60 + g + d) ...
                      + sind(g + d + 180)) / x;
        elseif(theta_ac(k) >= (120 + g + d) && theta_ac(k) < (180 + g + d))
            i_as(k) = -i_dc;
        elseif(theta_ac(k) >= (180 + g + d) && theta_ac(k) < (240 + g + d))
            i_as(k) = 196 * (sind(theta_ac(k)) - ...
                      sind(theta_ac(k) - 120) - sind(180 + g +d) ...
                      + sind(g + d + 60)) / x - i_dc;
         elseif(theta_ac(k) >= (240 + g + d) && theta_ac(k) <= 360)
            i_as(k) = 196 * (sind(theta_ac(k)) - ...
                      sind(theta_ac(k) + 120) - sind(240 + g + d) ...
                      + sind(g + d + 360)) / x;
        end
    end
    disp(196/x);
    shift_num = round(length(theta_ac) * 120 / 360);
    i_bs = circshift(i_as, [0, shift_num]);
    i_cs = circshift(i_as, [0, -shift_num]);
end

%% forwardEuler.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%{
This function is to simulate the system and plot the required graphs for
the analytical process.
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function[Ias, Ibs, Ics, theta, Idc, Vdc] = forwardEuler(R, E_amp, T)
    %% Initial Parameters and Arrays
    w = 377;
```

```
phi = 120;
tau = 1e-5;                 % Time constant
Ls = 1e-3;                  % AC Source Inductance
L_dc = 5e-3;                % DC Inductance
%% Timing Array
tend = 4 * T;
dt = 1e-7;
t = 0:dt:tend;
tlen = length(t)+1;
%% Initial Arrays
theta = zeros(1,tlen);
e_as = zeros(1,tlen);
e_bs = zeros(1,tlen);
e_cs = zeros(1,tlen);
Ias = zeros(1,tlen);
Ibs = zeros(1,tlen);
Ics = zeros(1,tlen);
VLa = zeros(1,tlen);
VLb = zeros(1,tlen);
VLc = zeros(1,tlen);
Va = zeros(1,tlen);
Vb = zeros(1,tlen);
Vc = zeros(1,tlen);
id1 = zeros(1,tlen);
id3 = zeros(1,tlen);
id5 = zeros(1,tlen);
Itransfer = zeros(1,tlen); % Transfer function for Idc
Vdc = zeros(1,tlen);
Idc = zeros(1,tlen);
%% Forward Euler Algorithm
epsilon = 0.01;
k = 1;
while t(k) < tend
    %% Phase and Angle Update
    theta(k + 1) = w * t(k) * 180 / pi;
    theta(k + 1) = mod(theta(k + 1), 360);
    %% Source Voltages
    e_as(k) = E_amp * cosd(theta(k));
    e_bs(k) = E_amp * cosd(theta(k) - phi);
    e_cs(k) = E_amp * cosd(theta(k) + phi);
    %% Forward Euler Calculation
    if Ias(k) > epsilon
        VLa(k) = Vdc(k);
    elseif Ias(k) < -epsilon
        VLa(k) = 0;
    elseif ((Ias(k) > -epsilon) && (Ias(k) < epsilon))
```

```
    VLa(k) = ((2 * epsilon) ^ (-1) * Ias(k) + 0.5) * Vdc(k);
end
if Ibs(k) > epsilon
    VLb(k) = Vdc(k);
elseif Ibs(k) < -epsilon
    VLb(k) = 0;
elseif ((Ibs(k) > -epsilon) && (Ibs(k) < epsilon))
    VLb(k) = ((2 * epsilon) ^ (-1) * Ibs(k) + 0.5) * Vdc(k);
end
if Ics(k) > epsilon
    VLc(k) = Vdc(k);
elseif Ics(k) < -epsilon
    VLc(k) = 0;
elseif ((Ics(k) > -epsilon) && (Ics(k) < epsilon))
    VLc(k) = ((2 * epsilon) ^ (-1) * Ics(k) + 0.5) * Vdc(k);
end
%% Update Phase Voltage Arrays
Va(k) = (2 * VLa(k) - VLb(k) - VLc(k)) / 3;
Vb(k) = (2 * VLb(k) - VLc(k) - VLa(k)) / 3;
Vc(k) = (2 * VLc(k) - VLa(k) - VLb(k)) / 3;
%% Update Phase Current Arrays
Ias(k+1) = dt * (e_as(k) - Va(k)) / Ls + Ias(k);
Ibs(k+1) = dt * (e_bs(k) - Vb(k)) / Ls + Ibs(k);
Ics(k+1) = dt * (e_cs(k) - Vc(k)) / Ls + Ics(k);
%% Diode Current Logic
if Ias(k+1) > 0
    id1(k+1) = Ias(k+1);
else
    id1(k+1) = 0;
end
if Ibs(k+1) > 0
    id3(k+1) = Ibs(k+1);
else
    id3(k+1) = 0;
end
if Ics(k+1) > 0
    id5(k+1) = Ics(k+1);
else
    id5(k+1) = 0;
end
%% Calculate Vdc and Idc
Idc(k+1) = id1(k+1) + id3(k+1) + id5(k+1);
Itransfer(k+1) = dt * (R * Idc(k) - Vdc(k)) / tau + Itransfer(k);
Vdc(k+1) = L_dc * Idc(k+1) / tau + Itransfer(k+1);
%% Time Update
t(k+1) = t(k) + dt;
```

```
        k = k + 1;
    end
end

%% average.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%{
dt is time step, x is the waveform that is being analyzed, T is its
fundamental
period of the waveform.
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function[avg] = average(dt,x,T)
    %% Parameters
    avg = 0;
    N = T / dt; % Number of subintervals
    k = length(x);
    %% Calculate average (avg) of a waveform using Riemann sum
    for i = k:-1:k-N
        avg = avg + dt * x(i);
    end
    avg = avg / T;
```