295/395B: Circuit Simulation and Optimization Methods: A Power Systems Perspective

Project 1: Developing a Time-Domain Circuit Simulator

Issued: 20th January 2023

Checkpoint 0 (if not completed, negative 5 points): February 3rd, 11:59 PM EST

Checkpoint 1 (5 pts): February 14th, 11:59 PM EST

Checkpoint 2 (5 pts): February 21st, 11:59 PM EST

Final Submission (90 pts): February 28th, 11:59 PM EST

*Each deadline has a 5-day grace period, with a penalty of **1 point per day. No points after that.***

Overview

In this project, you will develop a Python implementation of a time-domain simulator using the methods and techniques taught in Jan-Feb lectures and detailed in the tutorial on GitHub (https://arxiv.org/pdf/2212.12368.pdf) and also in chapters 1 – 4 and 7 of *Electronic Circuits and Simulation Methods*.

Your solver should include all elements of a non-linear time-domain solver identified in the lecture and should use **trapezoidal integration**. You do not need to implement dynamic time steps, but you may if time permits. The solver algorithm should follow the structure shown in Figure 1. Your program should be general, meaning that it must be capable of solving any linear or non-linear circuit that includes resistors, capacitors, inductors, AC voltage sources, ideal switches, or three-phase induction motors (some exceptions for the undergraduate students apply).

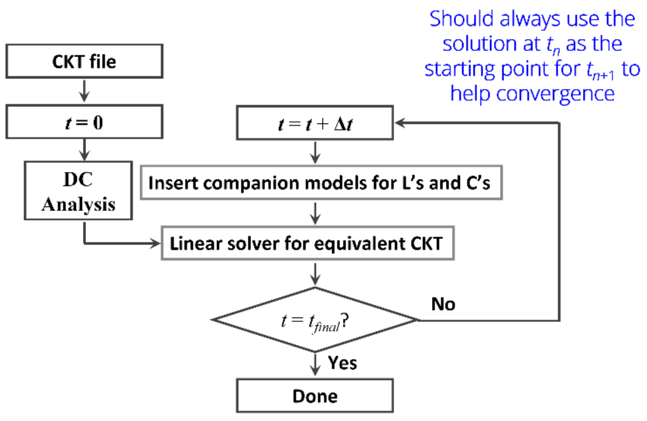
Should always use the solution at t_n as the starting point for t_{n+1} to help convergence
CKT file
$t = 0$
DC Analysis
$t = t + \Delta t$
Insert companion models for L's and C's
Linear solver for equivalent CKT
$t = t_{final}$?
No
Yes
Done

Figure 1. Basic structure of a time-domain simulator.

You can use standard open-source Python libraries to develop your solution, such as NumPy and SciPy. For plotting waveforms, you may use Matplotlib, Seaborn, or similar open-source data visualization libraries.

You will use your solver to perform a time-domain simulation on two benchmark circuits, a simple RL circuit (*testcases/RL_circuit.json*) and a more complex induction motor circuit (*testcases/IM_circuit.json)*. A netlist for each circuit, or circuit description, is provided in the course GitHub repository. The circuit elements of the netlist are in a *.json* file. To simplify your work, we have given you a reader (*lib/Reader.py*) that processes a *.json* file and then returns a dictionary of simple Python objects representing circuit elements. A description of the columns of each circuit element Python class' variables is given in the appendix.

For each benchmark circuit, you will have a set of tasks that you must complete. An explanation of the tasks is in the sections below. Your work's results must be in your final project report, and report expectations are in Deliverables.

We have provided you with some template code to start with in *run_solver.py* and *scripts/solve.py*, along with a few empty functions; you should feel free to change these as you see fit, as they are simply a suggestion.

Note: Undergraduate students will be evaluated on their output of RL_circuit alone. However, I highly recommend that they try to work on the IM_circuit as well.

Part 1: RL circuit

Consider the following RL circuit:

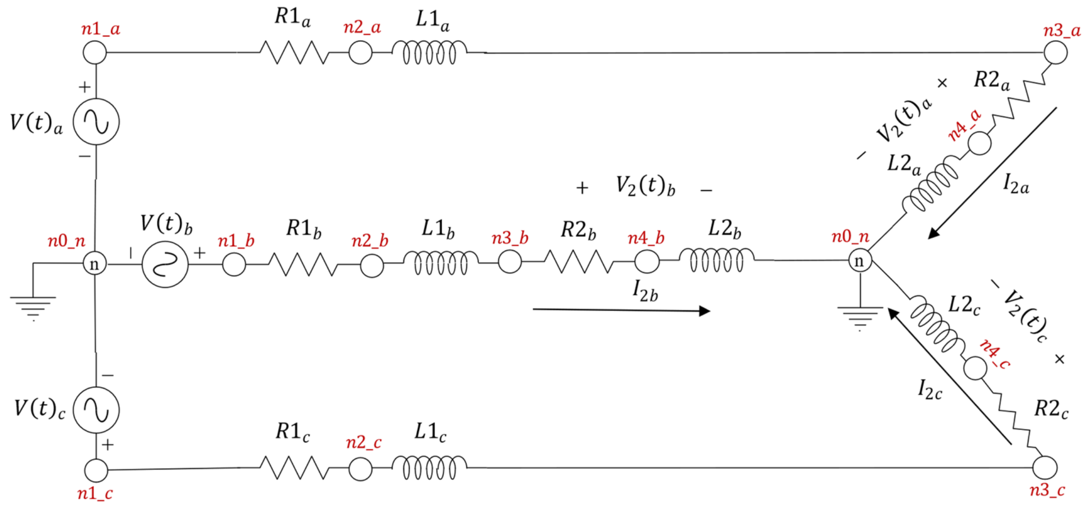

Figure 2. Three-phase series RL circuit.

Note that the phase voltages ($V_{2a}(t)$, $V_{2b}(t)$, $V_{2c}(t)$) and the phase currents ($I_{2a}(t)$, $I_{2b}(t)$, $I_{2c}(t)$) are the voltages across the output (R_{2a} and L_{2a}, R_{2b} and L_{2b}, R_{2c} and L_{2c}) and are the currents flowing into the output, respectively. Parameters for all circuit elements are in the *testcases/RL_circuit.json* file in the course GitHub repository.

For the above circuit, please complete the following tasks:

a) Perform a time-domain simulation for 0.2s.
b) Compute and plot the output voltages (V_{2a}, V_{2b}, V_{2c}) and output currents (I_{2a}, I_{2b}, I_{2c}). All voltages should be on one figure, and all currents should be on one figure.
c) Analyze the response of the output voltages and currents. No need for a lengthy analysis; a few sentences are sufficient so long as the thoughts are clear and coherent.
d) Explore the computational efficiency of your simulation when using dense vs. sparse matrices. Comment on the results.
e) Compare your solution against the signals produced by the Simulink model (*validate/project1_ rl_circuit.slx*) provided in the project distribution folder.

To solve the circuit, you will need to replace the inductors with their companion circuits, following the steps outlined in the tutorial on Github. Additionally, you will need to determine an appropriate time step for simulation.

Part 1 Bonus: RLC circuit with switches

For the bonus problem, our RL circuit will become an RLC circuit with ideal switches on each phase:

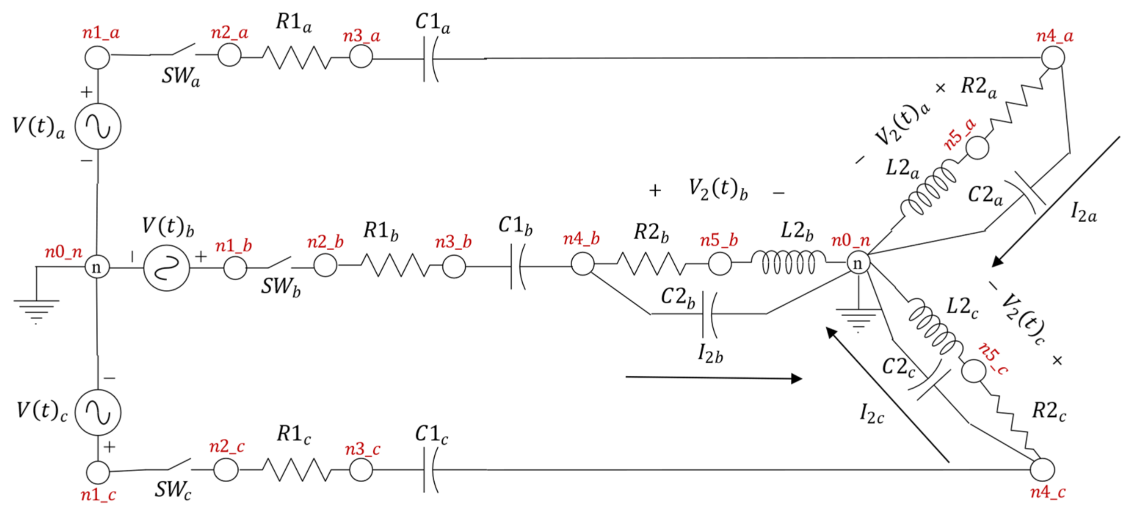

Figure 3. Three-phase series RL circuit with ideal switches on each line.

All switches are initially closed. They open at $t = 50ms\ (0.050s)$ and close at $t = 137.5ms\ (0.135s)$. Parameters for all other circuit elements are in the *testcases/RLC_circuit.json* file in the course GitHub repository.

Please complete the following tasks for the RLC circuit:

a) Perform a time-domain simulation for $0.2s$.

b) Compute and plot the output voltages ($V_{2a}(t)$, $V_{2b}(t)$, $V_{2c}(t)$) and output currents ($I_{2a}(t)$, $I_{2b}(t)$, $I_{2c}(t)$). All voltages should be on one figure, and all currents should be on one figure.

c) Analyze the response of the output voltages and currents. No need for a lengthy analysis; a few sentences are sufficient so long as the thoughts are clear and coherent.

d) Explore the computational efficiency of your simulation when using dense vs. sparse matrices. Comment on the results.

e) Compare your solution against the signals produced by the Simulink model (*validate/project1_bonus_rlc_circuit.slx*) provided in the project distribution folder.

Hint: To model the switch, consider what the switch looks like when closed and open. Draw out both states.

Part 2: Three-Phase Induction Motor

In part 2, you will test your simulator on a three-phase induction motor connected to three single-phase sinusoidal voltage sources (separated via 120 degrees) via three series RL lines with snubber resistances. The circuit for this portion of the

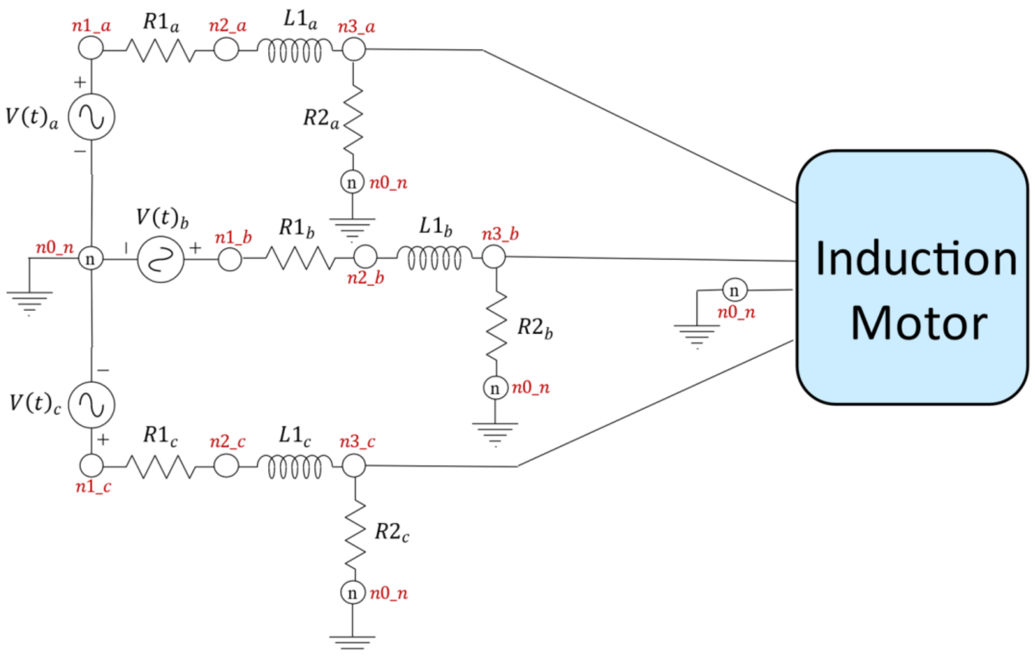

Figure 4. Three-phase induction motor circuit connected to a sinusoidal voltage source via a series RLR line.

assignment is below.

The induction motor block represents an induction motor. The model you implement in your simulator must use the **stationary reference frame**. Besides reviewing the GitHub tutorial, you may also consult "D, Q reference frames for the simulation of induction motors" [1] for more information about the induction motor's operation and the derivation of its equations. Parameters for the voltage sources, the induction motor, and the RLR lines are in *testcases/IM_circuit.json* file in the course GitHub repository.

Please complete the following tasks for the three-phase induction motor circuit:

a) Perform a time-domain simulation on the three-phase induction motor until it reaches the steady-state. You will need to determine when the induction motor reaches steady-state, the time step, and the simulation time.

[1] R. J. Lee, P. Pillay, and R. G. Harley, "D,Q reference frames for the simulation of induction motors," *Electr. Power Syst. Res.*, vol. 8, no. 1, pp. 15–26, 1984.

b) Report the steady-state values of the induction motor currents (I_{ds}, I_{qs}, I_{dr}, I_{qr}), electrical torque (T_e), and rotor speed (ω_r).
c) Compute and plot the induction motor currents ($I_{ds}(t)$, $I_{qs}(t)$, $I_{dr}(t)$, $I_{qr}(t)$), electrical torque ($T_e(t)$), and rotor speed ($\omega_r(t)$. All stator currents should be on one figure, and all rotor currents should be on a different figure.
d) Analyze the response of the induction motor. No need for a lengthy analysis; a few sentences are sufficient so long as the thoughts are clear and coherent.
e) Explore the computational efficiency of your simulation when using dense vs. sparse matrices. Comment on the results.
f) Compare your solution against the signals produced by the Simulink model (*validate/project1_IM_circuit.slx*) provided in the project distribution folder.

Part 2 Bonus: Three-Phase Induction Motor Disturbance

For the bonus problem, we will study the induction motor response when we temporarily remove a line. To analyze the induction motor's response to the disruption, we will add a switch to the line on phase B. Now, our circuit from part 2 becomes:

Initially, the switch is closed. The switch opens at $t = 0.7s$ and closes at $t = 0.8s$.

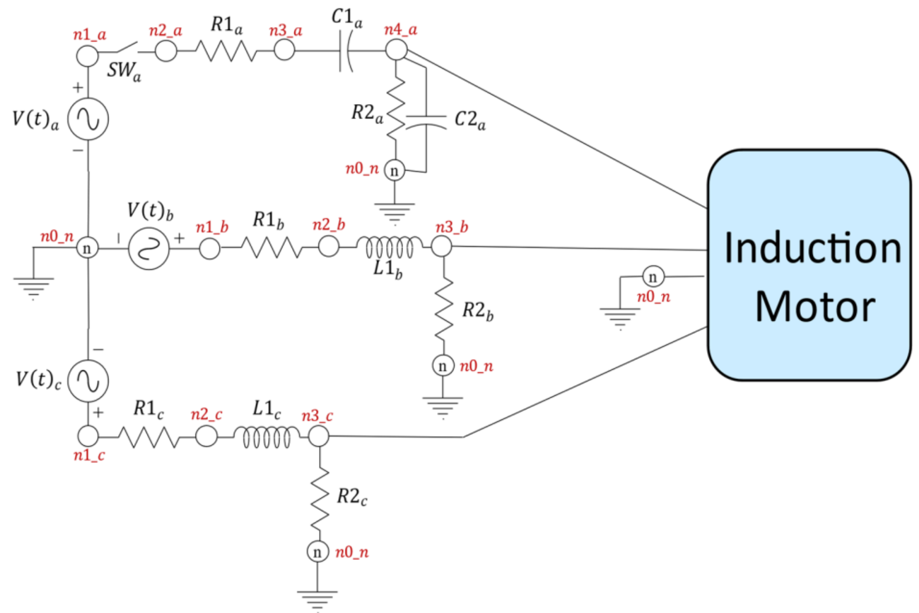

Figure 5. Three-phase induction motor with an ideal switch on phase A of the circuit.

Parameters for all circuit elements, including the ideal switch, are in the *testcases/IM_switch_circuit.json* file in the course GitHub repository.

Please complete the following tasks for the three-phase induction motor circuit:

a) Perform a time-domain simulation on the induction motor until it returns to steady-state after the disturbance. You will need to determine when the induction motor returns to steady-state, the time step, and the simulation time.

b) Compute and plot the induction motor currents ($I_{ds}(t)$, $I_{qs}(t)$, $I_{dr}(t)$, $I_{qr}(t)$), electrical torque ($T_e(t)$), and rotor speed ($\omega_r(t)$). You limit your plots to include only the results of the disturbance and its aftermath. All stator currents should be on one figure, and all rotor currents should be on a different figure.

c) Analyze the response of the induction motor. No need for a lengthy analysis; a few sentences are sufficient so long as the thoughts are clear and coherent.

d) Explore the computational efficiency of your simulation when using dense vs. sparse matrices. Comment on the results.
e) Compare your solution against the signals produced by the Simulink model (*validate/project1_IM__switch_circuit.slx*) provided in the project distribution folder.

Hint: To model the switch, consider what the switch looks like when closed and open. Draw out both states.

Deliverables

Checkpoint 0 (Due: February 3rd, 11:59 PM EST**)**

Given a matrix $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 3 \\ 2 & 0 & 1 \end{bmatrix}$ and column vector $B = \begin{bmatrix} 6 \\ 8 \\ 5 \end{bmatrix}$, solve for elements in vector x:

$$A \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = B$$

The students must use numpy or scipy functions in Python to solve for x. Submit your code and pdf with a snapshot of your solution on Blackboard Checkpoint 0 submission.

Checkpoint 1 (Graduate Students- Due: February 14th, 11:59 PM EST**)**

Submit code that can parse and simulate the network defined in *testcases/RL_circuit.json* and produce plots of the requested signals. For full credit, your submission need not be 100% correct, but it should be able to run a time-domain simulation for the requested period without crashing and produce plots. We expect the stamping functions for resistors, capacitors, inductors, and voltage sources to be completed.

Checkpoint 1 (Undergraduate Students: Due: February 14th, 11:59 PM EST**)**

Submit code that can parse and simulate the network defined in *testcases/single_phase_RL_circuit.json* and testcases/*single_phase_R_circuit.json* and produce plots of the requested signals. For full credit, your submission need not be 100% correct, but it should be able to run a time-domain simulation for the requested period without crashing and producing plots. We expect the stamping functions for resistors, inductors, and voltage sources to be completed.

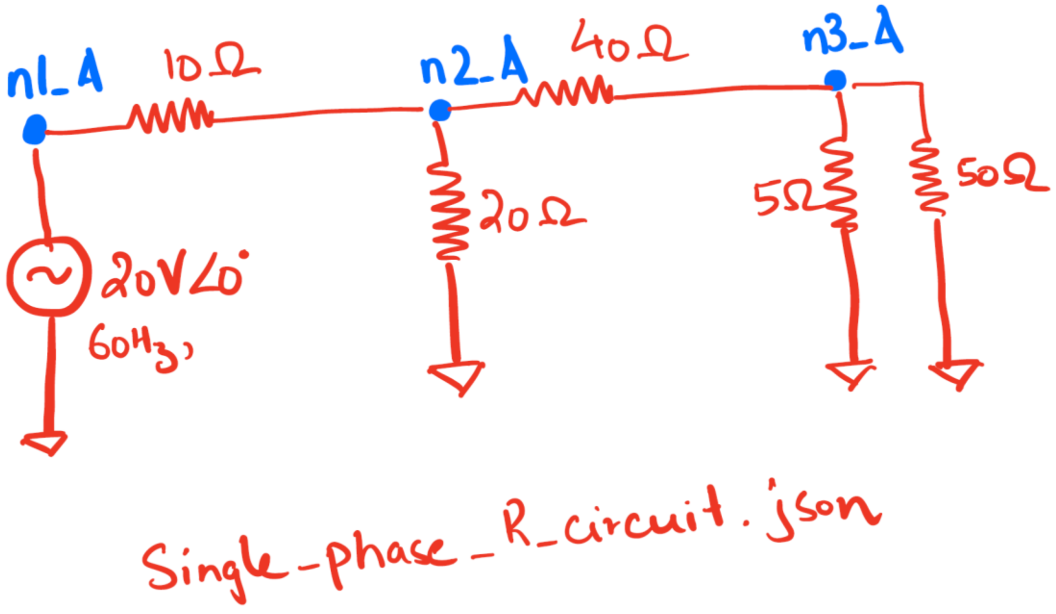

Figure 1:single_phase_R_circuit that undergraduate students need to simulate for checkpoint1.

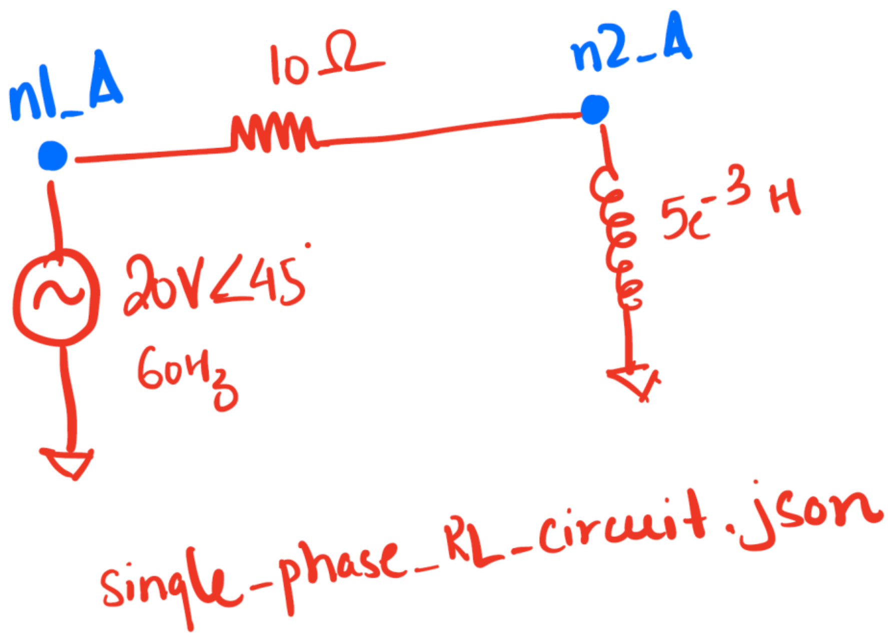

Figure 2:single_phase_RL_circuit.json that undergraduate students need to solve for checkpoint 1.

Checkpoint 2 (Might be slightly different for undergraduate students)

In order to help you with correctly implementing the model for the three-phase induction motor, we have created a worksheet to organize your derivations of the trapezoidal approximation and linearization of the model's underlying nonlinear differential equations. We will release this worksheet by 2/9/23. Complete it to receive full credit for Checkpoint 2.

Final Submission (Might be slightly different for undergraduate students)

For the final submission, you must provide two deliverables:

1. Python code for your simulator as a forked repository of the main course GitHub repository. You must push the final version of your code and submit a merge request by the deadline. More details about the submission process are in the GitHub repository. We will not grade any changes made after the deadline.
2. Upload a project report in **.pdf* format to Blackboard that includes the following items:
    a. A brief explanation of your code's structure, classes, methods, functions, and anything else that is unique about your codebase.
    b. Responses to the tasks outlined in Part 1 and Part 2. All output waveforms or figures should be appropriately sized and have labels, legends, and captions. Captions should be below the figure.
    c. A discussion and comparison of your results when using sparse and non-sparse matrices.

Grading

We will grade your project submission based on the rubric below. Both bonus problems are worth 5 points each.

Component	Sophisticated	Competent	Needs Work
Checkpoint 1 (5 points)	5 pts	1-4 pts	0 pts
	Code turned in that can run RL_circuit.json network and generate correct plots for the requested signal.	Some, but not all, stamping functions completed for RLC and sources. Incomplete plotting.	No code submitted.
Checkpoint 2 (5 points)	5 pts	1-4 pts	0 pts
	Derivation worksheet fully completed.	Derivation worksheet partially completed.	Derivation worksheet not turned in.
	36 – 40 pts	30 – 35 pts	0 – 30 pts

Solver Implementation (30 points)	The solution implements all time-domain simulator components correctly. The solver always works correctly. Code has clear and sufficient comments.	The solution implements most time-domain simulator components correctly. The solver works the majority of the time correctly. The code could use a few additional comments but is generally logical and easy to follow.	Significant elements of the time-domain simulator are missing. The program does not often work correctly or work at all. Few to no comments make the code hard to understand.
RL Circuit Accuracy (25 points)	23 (46) – 25 (50) pts	16 (32) – 22 (44) pts	0 – 15 (30) pts
	Output voltage and current waveforms match the solution exactly. Computational efficiency metrics reported are reasonable and justified.	The output voltage or current waveforms are close, but one or both are not exact. Computational efficiency metrics reported are mostly reasonable and but not justified.	Output voltage and current waveforms are not accurate. No computation efficiency metrics reported.
Three-Phase Induction Motor Accuracy (25 points)	23 – 25 pts	16 – 22 pts	0 – 15 pts
	All steady-state induction motor outputs match the solution exactly. Waveform plots match the answer exactly. Computational efficiency metrics reported are reasonable and justified.	Steady-state induction motor outputs are off by a few digits. Waveform plots are not exact but close to the correct solution. Computational efficiency metrics reported are mostly reasonable and but not justified.	Steady-state induction motor outputs are not at all within the range of the right answer. Waveform plots are not accurate at all. No computation efficiency metrics reported.
Written Communication (10 points)	8 – 10 pts	5 – 7 pts	0 – 4 pts
	The report is well-organized and coherent. All figures and tables are readable, appropriately sized, and properly labeled, and captioned correctly. There are no spelling or grammar errors.	The report is, for the most part, well-organized and coherent. Some figures and tables are hard to follow or labeled and captioned inappropriately. There are a few spelling and grammatical errors, but they do not detract from the work.	The report has little to no structure or organization. No labels and captions on figures and tables. Many spelling and grammatical errors make the report difficult to read.

Note: Undergraduate students will be evaluated on their output of RL_circuit alone, therefore you can see the parentheses score, for how their projects will be evaluated. However, I highly recommend that they try to work on the IM_circuit as well.

Bonus points will be given for completion of checkpoints.

Appendix: Circuit Element Array Descriptions

Nodes.py		
Variable Name	`name`	`phase`
Type	String	String
Description	Name of the node	Phase of the node

VoltageSources.py (Sinusoidal Voltage Sources)						
Variable Name	`name`	`vp_node`	`vn_node`	`amp_ph_ph_rms`	`phase_deg`	`frequency_hz`
Type	String	String	String	Float	Float	Float
Description	Name of the voltage source	The node connection at the positive terminal of the voltage source.	The node connection at the negative terminal of the voltage source.	The phase-phase RMS amplitude in volts.	The phase angle in degrees.	The frequency in Hz.

Resistors.py				
Variable Name	`name`	`from_node`	`to_node`	`r`
Type	String	String	String	Float
Description	The name of the resistor.	The node connection at the sending end of the circuit element.	The node connection at the receiving end of the circuit element.	The resistance in Ω.

InductionMotors.py						
Variable Name	`name`	`phase_a_node`	`phase_b_node`	`phase_c_node`	`power_nom`	
Type	String	String	String	String	Float	
Description	Name of the induction motor	The node that is connected to phase A of the induction motor.	The node that is connected to phase B of the induction motor.	The node that is linked to phase C of the induction motor.	The nominal power of the motor.	
Variable Name	`v_nom`	`lm`	`rs`	`rs`	`rr`	`lls`
Type	Float	Float	Float	Float	Float	Float
Description	Line-line voltage	Motor frequency	Mutual inductance	Stator resistance	Rotor resistance	Stator inductance
Variable Name	`llr`	`j`	`tm`	`d_fric`	`n_pole_pairs`	
Type	Float	Float	Float	Float	Float	
Description	Rotor inductance	Motor Inertia	Mechanical torque	Friction factor	Number of pole pairs	

Capacitors.py				
Variable Name	`Name`	`from_node`	`to_node`	`c`
Type	String	String	String	Float
Description	The capacitor name	The node connection at the sending end of the circuit element.	The node connection at the receiving end of the circuit element.	The capacitance in F.

Inductors.py				
Variable Name	`name`	`from_node`	`to_node`	`L`
Type	String	String	String	Float
Description	The inductor name	The node connection at the sending end of the circuit element.	The node connection at the receiving end of the circuit element.	The inductance in H.

Switches.py					
Variable Name	`name`	`from_node`	`to_node`	`t_open`	`t_close`
Type	String	String	String	Float	Float
Description	The ideal switch name	The node connection at the sending end of the circuit element.	The node connection at the receiving end of the circuit element.	The time in seconds that the switch opens.	The time in seconds when the switch closes.