Summer 2021

Bond graph modeling of critical infrastructures for cyber-physical security implementation

Michele Jane White

Department:

BOND GRAPH MODELING OF CRITICAL INFRASTRUCTURES FOR CYBER-PHYSICAL SECURITY IMPLEMENTATION

by

MICHELE JANE WHITE

A THESIS

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

in

ELECTRICAL ENGINEERING

July 2021

Approved by:

Dr. Jonathan W. Kimball, Advisor
Dr. Mehdi Ferdowsi
Dr. Bruce McMillin

ABSTRACT

In developed societies, there exists infrastructure vital to everyday life. This includes water and power systems. Technology is quickly evolving and being implemented on these utilities. This technology can range from smart metering in neighborhoods to volume sensors in local waste water treatment facilities. When networking, sensing, monitoring, or control devices are integrated with infrastructure it is considered a cyber physical system, or CPS. When information about an important physical system is connected to the virtual world, it is opened up to security risks. Cyber security can be provided to the CPS by monitoring the physical state of the system and detecting virtual attacks when unexpected changes occur. However, these systems mentioned cross multiple domains: electrical, mechanical, and hydraulic amongst others. This creates a challenge, as each domain has its own unique language, terminology, and topology. To combat this challenge, a universal representation of these systems is implemented through the use of bond graphs. Bond graphs take advantage of the commonalities found in all physical scientific domains. These similarities are found in the energy interactions throughout a given system, and bond graphs allow these relationships to be mapped graphically and mathematically. This unifying notation creates a clear picture of the energy movement throughout a physical system.

Information about the unifying bond graph method is discussed, and previous work and examples are relayed. To demonstrate the use of bond graphs on a power system, a realistic microgrid model was converted into a bond graph, simulated, and validated.

ACKNOWLEDGMENTS

I want to extend my deepest appreciation to my research advisor, Dr. Jonathan Kimball, for providing me with the ability to learn and work under his guidance. Your kindness, generosity, patience, and profound belief in my abilities as a student and a researcher allowed me to achieve what I never thought was possible. The tremendous amount of growth I have experienced from your invaluable knowledge on both physics and the world will allow me to thrive in my future endeavors.

I cannot begin to express my sincerest gratitude to Natasha Jarus for her unwavering support and mentorship throughout my research process. Without your diverse knowledge, unique perspectives, and positive outlook the completion of my thesis would not have been possible.

I would like to thank my committee members Dr. Mehdi Ferdowsi and Dr. Bruce McMillin for overseeing my work and always creating an environment where students are able to ask questions and have thoughtful discussions.

Particularly helpful to me during this time were Kartikeya Veeramraju, Angshuman Sharma, and the rest of my lab mates. Thank you all for helping me understand difficult topics and for the friendships that formed in the process.

Special thanks goes out to Blaine Allen, Sophie Holle, Mac Bailey, and Albert Perlman for believing in me when I did not believe in myself. Each of you relentlessly encouraged, supported, and uplifted me during this time. For that, I am forever grateful.

I want to thank my parents Sarah and Craig Adams for their financial and moral support throughout my education, and my father Mike White for cultivating my love for science.

Many thanks also go to the NSF, as this work was supported in part by the National Science Foundation under award 1837472.

TABLE OF CONTENTS

Page

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

In developed societies, there exists infrastructure vital to everyday life. This includes water and power systems. Technology is quickly evolving and being implemented on these utilities. This technology can range from smart metering in neighborhoods to volume sensors in local waste water treatment facilities. When networking, sensing, monitoring, or control devices are integrated with infrastructure it is considered a cyber-physical system, or CPS. When information about an important physical system is connected to the virtual world, it is opened up to security risks. Cyber security can be provided to the CPS by monitoring the physical state of the system and detecting virtual attacks when unexpected changes occur.

1.1. CHALLENGES OF MULTI-DOMAIN SYSTEM MODELING

However, these systems mentioned cross multiple domains: electrical, mechanical, and hydraulic amongst others. This creates a challenge, as each domain has its own unique language, terminology, and topology. In the electrical domain, physics is described in terms of voltage, current, resistance, capacitance, and inductance. While explaining the physics of a mechanical system, the discussion takes a different route using force, velocity, friction, spring constants, and mass. Thus, when analyzing a system involving both electrical and mechanical physics, the overall understanding of the system can become quite complicated. If we delve even deeper into these two domains, then we will notice that there are additional ways to articulate what is happening in a given system. The physics described above was merely the electrical and linear mechanics cases respectively. The electrical domain also can be described in terms of electromagnetics with the ideas of flux and reluctance. The mechanical domain has rotational expressions like angular velocity and torque.

Furthermore, if we were to aim our focus on a specific emphasis area within each domain, there would be even more ways to categorize the physical phenomenon at hand. Not only do the linguistics change between domains, but the way in which the systems are modeled is different as well. If a circuit diagram is present, then the electrical domain principles can be used to understand how it works. If a free body diagram is present, then the application of mechanical ideologies will clarify how it works. This clarification in both scenarios often comes from the mathematical equations derived from these models.

However, the ability to gain knowledge from a system given the terminology, modeling, and equations after a certain level of complexity is lost when looked upon by a scientist outside of their realm. The overall interpretation of an electromechanical system cannot be truly realized when modeled in a segregated fashion. Now, add in other scientific domains to the system. The holistic view becomes further muddled. The complexity of a system is proportional to the number of physical domains involved.

1.2. PROPOSED APPROACH

To combat the challenge that multi-domain systems produce, a universal representation of these systems is implemented through the use of bond graphs. Bond graphs take advantage of the commonalities found in physical scientific domains and consolidates them into a singular notation. These similarities are found in the energy interactions throughout a given physical system. Energy interactions occur through different physical elements that store, dissipate, source, transform and convert energy while simultaneously conserving it. The process of going from an ideal physical model of a system to a bond graph model is beneficial not only because it closes the conceptual gap between domains, but because bond graph models are straightforward to build. The building method is algorithmic in nature, and the way in which equations are derived from the model is algorithmic as well. The

most unique and valuable characteristic of bond graph modeling is its ability to tell the end user if sufficient detail of the model was provided upon inspecting causality. This unifying notation creates a clear picture of the energy movement throughout a physical system.

The usefulness of bond graph modeling for the purpose of cyber security protection will be displayed in the remainder of this paper. The following information and work is based around simulating a microgrid consisting of a synchronous generator powering an inductance motor with line inductances, loads, and a capacitor bank between the two electrical machines. There were three goals in mind when creating this system: gathering time domain imaginary and real power data, modeling the microgrid in a universal way, and finding the physics equations from the universal model to use as invariants for system protection. To achieve the first goal, a power conserving Alpha-Beta-Zero reference frame, or Clarke, transformation was made. Modelling the microgrid in this way was simpler, as the system moved from three phase power to two phase power as a result of this Clarke transformation.

To achieve the second goal, bond graph technology was applied to the ideal physical system (IPS) representation of the proposed microgrid. The microgrid had been drawn as an IPS one-line diagram where all three phases are represented as a single line. From the one-line, a bond graph model was derived and constructed. The first attempt at going from the IPS to a bond graph highlighted a key benefit of the bond graph technique. Aside from being universal, bond graphs allow the modeler to see if important physical phenomenon was left out of the configuration. When analyzing this first attempt, there was clear indication that the current bond graph model lacked crucial details of the synchronous generator and induction motor. Initially, the generator and motor were each represented as black boxes that converted energy between electrical energy and mechanical energy. To supplement the original bond graph, the black box was replaced with an energy storage device. The addition of this energy storage device incorporated multiple factors on both

the electrical and mechanical sides of the machines that the prior bond graph was lacking. Upon completion of a satisfactory bond graph, the last goal of finding equations to use for cyber security purposes could begin.

This final goal was accomplished by mathematically evaluating the bond graph of the microgrid. From the bond graph we extracted state variables that set the flow of energy in the system, as well as a set of algebraic constraints on those variables. Consequently, differential algebraic equations (DAEs) are formed which we used to evaluate the physics of the microgrid. We can then use these DAEs to check if our assumptions based on the physics of the microgrid match what is currently happening in the system. Miss matches in the system can indicate that there is either a cyber-security attack or a system malfunction occurring, making these equations a valuable tool to cyber security programmers.

The rest of the paper is as follows: first, a review of different modeling methods will be presented, along with the roles of said modeling in correspondence with CPS security. Then, previous works that have utilized bond graph modeling for CPSs and microgrids will be discussed. Afterwards, the necessary background for microgrids and bond graphs will be given. Next, the experimental method and results will be shown.

2. LITERATURE REVIEW AND BACKGROUND

In this section, various modeling methods and software options will be discussed to illustrate why bond graphs were the optimal choice to model and analyze our microgrid. Additionally, the usefulness of system modeling in CPSs for cyber security purposes will be explained. This will be followed by specific examples of CPSs that use the bond graph method for modeling, along with other previous works using bond graph modeling for microgrids and power applications. An in-depth review of foundational material on microgrids and bond graphs will be provided to better understand the importance and usefulness of this project.

2.1. SYSTEM MODELING

There are many different paths that can be taken on the journey from transforming a multidomain CPS concept to a working simulation. The paths that will be discussed include analogous system models, block diagrams, software modeling, and bond graphs.

2.1.1. Analogous Systems. Analogous system modeling is the transformation of a system model in one scientific domain to another while remaining mathematically sound[1]. Modeling in this manner is beneficial as it allows for scientists to comprehend systems outside of their field of study by converting the unfamiliar material to information they know. Moreover, creating and testing a physical working model of a system is easier if it can be mirrored as an electrical circuit[2]. For example, this strategy is used in the 1933 paper [1] to go from mechanical diagrams to electrical circuits by means of the newly created force-current analogy rather than the previously used force-voltage analogy. The force-voltage, or impedance, analogy relates force to voltage, velocity to current, masses to inductances, and springs to capacitors. The main issue with the impedance analogy is the lack of intuitiveness involved due to through elements, like force, being mapped to across elements, like voltage. Thus, applying inverse relationships between the mechanical and

electrical domains, so mechanical elements in series are mapped to electrical elements in parallel and vice versa. The mobility analogy alleviates these issues by relating force to current, velocity to voltage, masses to capacitors, and springs to inductances.

Analogous systems are not limited to just the mechanical and electrical domains. The force-voltage analogy was applied in [3] to analyze how vibrations at different frequencies affect biological systems ranging from tree branches to human muscles. The same force-voltage, or impedance, analogy was used in [4] to relate electrical resistances to different tribology — the study of wear on interacting surfaces — aspects like lubricants to calculate the life span of different objects.

Although the result of analogous system modeling yields a transformed and digestible replica of the original system, this method has its limitations. An analogous system model may only be understood by two groups of scientists: those who have studied the domain used in the original system model, and those who have studied the domain used in the transformed system model. Therefore, an analogous system model would have to be created for every scientific domain in order to be universally understood.

2.1.2. Block Diagrams. Block diagrams are another way in which a system can be modeled. Block diagrams are built using blocks containing transfer functions, summing junctions, and arrows dictating the linear direction of inputs, outputs, and feedback information[2]. The main advantage of block diagrams over analogous systems can be found in their transfer functions. Transfer functions are not limited to one scientific domain or another, and they can incorporate the mathematical purposes of multiple domains in one equation. Transfer functions are able to do so because they utilise the Laplace Transform, which takes an equation in the time domain, and transforms it into the frequency domain. To show the multi-domain application of a block diagram, a moving coil loud speaker system is modeled in [2]. Although the speaker system involves electrical, mechanical, and acoustic elements, they are able to be used in unison through the block diagram approach.

Block diagrams are especially useful when analyzing a CPS because they can directly show the physical system itself and how this system interacts with controllers. Modeling a CPS in this way can aid in the discovery of controller coefficients and assuring the system remains stable. Looking at examples of CPSs, specifically microgrids, being modeled by block diagrams are significant as microgrid modeling is the experimental subject matter of this paper.

One such example comes from [5], where the motivation to model a microgrid using a block diagram comes from the need improve the method of how data is gathered and utilized of said system. Specifically, this paper is analyzing ways to gather information about a microgrid's system dynamics during times of transience. The author then uses this enhanced understanding of the microgrid to improve controller strategies for the system; previously, creating controllers for microgrids manifested from a guess and check method. Another example of a microgrid being represented by a block diagram can be seen in [6], which details what classifies as a microgrid, applications of a microgrid, microgrid topologies, how they are modeled, and more. Further discussion of microgrids will occur later in this section.

Though block diagrams can be used to model microgrids, they are not the optimal choice. While block diagrams allow for multiple domains to be used and mathematically united, there is still an issue not accounted for. This issue being that microgrids do not just produce energy for the owner, but they can also produce energy for the nearby utility. Meaning, energy can flow in two different directions in this system. Block diagrams only allow for linear energy flow depiction. Even though the linear case can be used successfully to model a migrogrid, it is not accurately depicting the bilateral energy exchange occurring in the system.

2.1.3. Bond Graphs. When choosing between methods for modeling a system, key elements of the system being modeled should be taken into consideration. For this paper, a microgrid is being modeled. The two key elements of a microgrid to be mindful

of include: working in both the electrical and mechanical domains and distributing energy in a bilateral fashion. Analogous systems only allow for one domain to be considered at a time, and block diagrams only permit linear energy movement. Therefore, the bond graph method would be the best choice in this case. The bond graph method provides a way to use multiple scientific domains concurrently, and it allows for bilateral energy flow throughout the system being modeled. How a bond graph is able to achieve these tasks will be covered in the following section.

2.2. BOND GRAPH OVERVIEW

This Section is a summary of information about bond graphs gathered from [7] and [8]. Unlike analogous system modeling and block diagrams, bond graphs are uniquely able to encompass many different scientific fields at once, and they are able to represent bilateral energy movement. Another advantage to bond graphs, is that they are able to be constructed easily from an ideal physical model, or IPM, through a process of algorithmic steps.

2.2.1. Bond Graph Basics. A bond graph is a power conserving model that represents energy interactions throughout a physical system using a uniform notation across scientific domains. Specifically, these models focus on the power transfer among various elements in the system at hand. Bond graphs are able to be represented with a single nomenclature utilizing effort and flow variables, along with port variables. The power transfer $P(t)$ between system components is the product of these effort $e(t)$ and flow $f(t)$ variables,

$$P(t) = e(t)f(t). \tag{2.1}$$

In the electrical domain, the effort variable is equivalent to voltage and the flow variable to current. In the mechanical domain, the effort and flow variables are equivalent to force and velocity, respectively. These effort and flow domain mappings along with others can be seen

Table 2.1. Scientific domains effort and flow equivalents.

Domain	**Effort**	**e(t)**	**Flow**	**f(t)**
Mechanical Translation	Force	F(t)	Velocity	V(t)
Mechanical Rotation	Torque	$\tau(t)$	Angular Velocity	$\omega(t)$
Electrical	Voltage	v(t)	Current	i(t)
Hydraulic	Pressure	$\rho(t)$	Volume Flow Rate	Q(t)

in Table 2.1. While the effort and flow variables take care of unifying power relationships in a system, port variables take care of unifying the energy interactions within the system and surrounding environment. In general, all physical systems include elements that serve the same energy interaction purposes. Bond graphs take advantage of these commonalities to create the port variables. There are seven basic port elements, which will be elaborated on further in Section 2.2.2, to represent the seven common physics principles occurring in physical systems. These principles include: two types of energy storage, energy dissipation, two types of energy sourcing, energy transformation, and energy conversion. A bond graph illustrates these unifying power and energy relationships through a system model diagram comprised of different components.

2.2.2. Components of a Bond Graph. There are five main components that compose a bond graph: bonds, ports, port elements, junctions, and causality bars.

2.2.2.1. Bonds. Bonds, also referred to as power bonds, play a few different roles in a bond graph. Bonds indicate the direction power is moving in, connect ports and junctions together, and demonstrate how physical system components relate to each other. Bonds are portrayed as half arrows as seen in Figure 2.2, and they specify the direction of power through the orientation of the half arrow. The half arrow shows which ports are producing power and which ports are consuming power. Figure 2.1 shows an ideal physical model of a simple circuit containing a voltage source, v, and a load, R. This circuit is producing power at the voltage source and consuming power at the load resistor. Figure 2.2 demonstrates

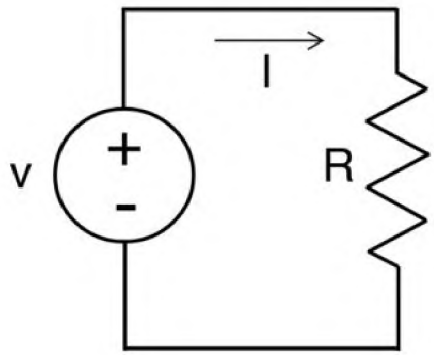

Figure 2.1. A simple ideal physical model of a source and load circuit.

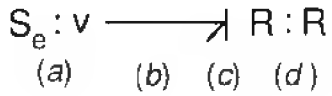

Figure 2.2. A bond graph of the simple circuit in Figure 2.1.
This bond graph illustrates three of the four parts of a bond graph: (*a*) and (*d*) port elements and ports, (*b*) bonds, and (*d*) causal strokes.

the same power production and consumption across the bond (*b*) with the tail of the half arrow on the source (*a*) and the tip pointing towards the load (*d*). Each power bond contains an effort and a flow variable, as shown in (2.1). This allows for unifying and singular nomenclature of power relationships to be depicted.

2.2.2.2. Ports and port elements. Located at the tail and tip of a power bond are ports. Ports are where 1-port, 2-port, and multi-port elements attach to the bond graph as seen in Figure 2.2 at points (*a*) and (*d*). Port elements indicate how energy exchanges across bonds, and the energy exchange is based off of the underlying physics principles involved in physical systems.

There are seven basic port elements to represent these physics principles, five of which are 1-port elements. The 1-port elements have exactly that: 1 port. This means that 1-port elements connect to the rest of the bond graph using exactly one bond. The 1-port elements are C-elements, I-elements, R-elements, S_e elements, and S_f elements. C-elements and I-elements portray the physics principle of energy storage. C-elements store energy through effort and displacement. Examples of C-elements as ideal physical model, or IPM, components are capacitors, springs, and water tanks. Capacitors store energy by accumulating charge, or displacement, between two parallel plates with the charge being

$$q = \int i \, dt \tag{2.2}$$

where q is the charge on the capacitor and i is the current flowing into the capacitor. This accumulation of charge creates a potential difference that results in voltage, or effort, and is expressed as

$$v = \frac{q}{C} \tag{2.3}$$

where v is the produced voltage and C is the capacitance. Springs store energy by changing the distance, or displacement, between the ends of the spring. The displacement can be defined as

$$x = \int V \, dt \tag{2.4}$$

where x is the displacement of the spring and V is the velocity the spring is moving at. This distance change will put the spring under tension or compression, resulting in force, or effort as well. This force is

$$F = kx \tag{2.5}$$

where F is the resulting force and k is the spring constant where $k = 1/C$. These electrical and mechanical specific equations can be represented by the universal bond graph nomenclature as

$$q = \int f \, dt \tag{2.6}$$

where the generalized displacement, q, is a state variable that *sets* the flow, f on the the physical system, and

$$e = \frac{q}{C} \tag{2.7}$$

is the effort at that specific C-element with capacitance C.

The I-element also stores energy, but instead of doing so by means of effort and displacement, the I-element stores energy with flow and generalized momentum. A few components that store energy in this manner are inductors, masses, and fluid-filled pipes. An inductor stores energy when its coil is energized from voltage and a magnetic field is formed. A piece of this magnetic field will pass through the inductor. This piece of magnetic field will be referred to as flux linkage, which can be defined as

$$\lambda = \int v \, dt \tag{2.8}$$

where λ is flux linkage, and v is the energizing voltage. The flux linkage, or in bond graph terms momentum, will then induce a current, or flow, on the coil of the inductor. This current can be shown as

$$i = \frac{\lambda}{L} \tag{2.9}$$

where i is the induced current, and L is the inductance. In a similar way, a mass stores energy when it is accelerated by a force. The momentum accumulates the force, or effort, via

$$p = \int F \, dt \tag{2.10}$$

where p is momentum and F is force. which then reflects in its velocity, or flow, as

$$V = \frac{p}{m} \tag{2.11}$$

where V is velocity and m is mass. The bond graph notation for these domain specific equations can be shown as

$$p = \int e \, dt \tag{2.12}$$

where the momentum, p, is a state variable that *sets* the effort, e, on the physical system, and

$$f = \frac{p}{I} \tag{2.13}$$

is the flow at that specific I-element with an inertia I.

Again, this nomenclature is not only for mechanical and electrical domains. Any scientific domain with energy storage formed by either effort and displacement, or by flow and momentum, will translate into the bond graph terminology.

The R-element dissipates energy. In a physical system, the R-element can appear as a resistor, a damper, or a fluid moving through a pipe. The R-element restricts flow given a certain effort. This is the same relationship seen in Ohm's Law, which is

$$v = iR. \tag{2.14}$$

Table 2.2. Effort and flow source domain equivalents.

Domain	**Effort Source, $S_e(t)$**	**Flow Source, $S_f(t)$**
Mechanical Translation	Force Source	Velocity Source
Mechanical Rotation	Torque Source	Angular Velocity Source
Electrical	Voltage Source	Current Source
Hydraulic	Pressure Source	Volume Flow Rate Source

A resistor will resist an amount of current proportional to the amount of voltage provided to the circuit. The subsequent mechanical relationship involving force, velocity, and damping is

$$F = VB. \tag{2.15}$$

The equivalent bond graph representation of the previous domain specific equations is

$$e = fR \tag{2.16}$$

where e is effort, f is flow, and R is the resistance. In any physical system where there is power loss, the dissipative bond graph R-element can be used.

The last two 1-port elements are the S_e, or effort source, and S_f, or flow source, elements. These elements source energy from their surrounding environment. Examples of an effort source are voltage sources, the force of gravity, or a fluid pressure source. A flow source may appear as a current source, a velocity source, or a hydraulic pump. A list of effort and flow sources is presented in 2.2; however, this table does not encompass all possible scientific domains this universal bond graph naming convention can be applied too.

Finally, 2-port elements will be discussed: the transformer and the gyrator. The 2-port elements have two ports, and they can connect to the rest of the bond graph using exactly two bonds. A transformer has the ability to either increase or decrease effort and flow values using a transformer ratio, or modulus. The transformer ratio with respect to an electrical transformer correlates to the number of turns on the coil of the transformer itself. An electrical transformer can be used in circuits to step up or step down power, and is defined as

$$v_1 = Nv_2 \tag{2.17}$$

for voltage transformation, or as

$$Ni_1 = i_2 \tag{2.18}$$

for current transformation, where N is the transformer ratio. In mechanical systems, a lever may be the transformer used to increase or decrease the force needed to move said lever. The transformer ratio here comes from the distance the fulcrum lies from each end of the lever, and is defined as

$$\frac{b}{a}V_1 = V_2 \tag{2.19}$$

for velocity transformation, or as

$$F_1 = \frac{b}{a}F_2 \tag{2.20}$$

for force transformation, where a and b are the distances on each side of the fulcrum to the end of the lever. In bond graph notation, the transformer equations are

$$e_1 = me_2 \tag{2.21}$$

and

$$mf_1 = f_2 \tag{2.22}$$

where m is the general transformer modulus.

A gyrator has the ability to convert an effort to a flow or a flow to an effort using a gyrator ratio, or modulus, r. The conversion between effort and flow can happen within a singular scientific domain, or it can be used to convert from one scientific domain to another. Examples of a gyrator include turbines, hydroelectric dams, and electromechanical machines such as a motor or generator. Gyrators are able to shift effort and flow variables through many different ways across the scientific domains; therefore, only the bond graph equations will be shown for simplicity, which are

$$e_1 = r f_2 \tag{2.23}$$

and

$$r f_1 = e_2. \tag{2.24}$$

A more in depth look at how gyrators function will be examined in the Methodology Section of this paper.

Additionally, any of the port elements explained above can be modulated. A port element can be modulated by denoting this on the bond graph with an M in front of the port element name, and with a full arrow pointing to the modulated port element with the modulation portrayed at the tail of the arrow. Modulation occurs when there is some form of monitoring or control system involved with the physical system. The full arrow is used to indicate that there is no power transfer occurring between the virtual modulation device and the physical system component it is associated with. Modulation can appear on an effort source in order to keep said effort source constant, or it can exist as an input signal on a transformer to alter the transformer modulus without disrupting the conservation of energy.

2.2.2.3. Junctions. The next component of bond graphs to discuss are junctions. Junctions are N-port components, meaning that they can connect to the rest of the bond graph with as many bonds as necessary. Junctions are used to connect port elements together. There are two types of junctions: 0-junctions and 1-junctions. When a 0-junction

is used, all the efforts at that 0-junction will be equal to each other, and all flows at that 0-junction will sum to zero. When a 1-junction is used, all the flows at that 1-junction will be equal to each other, and all efforts at the 1-junctions will sum to zero. In the electrical domain, these can be equated to Kirchhoff's Current and Voltage laws, or KCL and KVL respectively. The 0-junction relates to KCL, as this law indicates that the sum of current, or flow, at a node will equal zero, and all branches parallel to said node will have the same voltage, or effort. The 1-junction pertains to KVL, as this law indicates that the current, or flow, within a closed loop will be equal at any point in the loop, and that the voltage, or effort, within that loop will sum to zero. Although an electrical domain analogy was made here, that does not limit the use of the 1- and 0-junctions to just the electrical domain.

2.2.2.4. Causal strokes. The last component of bond graphs to cover is the causal stroke. The causal stroke is represented by a short line perpendicular to the bond, as shown in Figure 2.2 (*d*). The causal stroke can lie at either the tip or tail of the half arrow depending on *causality*, and the causality is independent of the direction of the power bond. A simple example of causality between two elements is shown in Figures 2.3 and 2.4. Figure 2.3 shows an *effort-in* causality, while Figure 2.4 depicts an *effort-out* causality. However, effort-in and -out are not all that is indicated with a causal stroke, as they are an indication of flow as well. Power bonds allow for bilateral signal movement; so, when effort is an input to Element B, as seen in Figure 2.3, then flow will be an output from Element B. These effort and flow inputs and outputs are determined by the *preferred causality* of certain port elements and junctions. Preferred causality is the ideal location of a causal stroke for each element and junction, with the exception of the R-element. The R-element can have an effort-in or effort-out causality, and the R-element causality is determined by the causality of the rest of the system. For instance, consider an effort source. An effort source will always have effort-in causality because it will supply effort into the system. A flow source will always have effort-out causality because it will supply flow as an input to the system; therefore, effort must be an output. A list of the port elements preferred causalities is

Figure 2.3. A representation of an effort-in causal stroke with respect to Element B.

Figure 2.4. A representation of an effort-out causal stroke with respect to Element B.

provided in 2.3. The C- and I-elements of a bond graph do not have to follow their preferred causality, but this is typically an indication that the IPM was not properly transferred into a bond graph. This can mean that either the IPM was not detailed enough and resulted in an inaccurate bond graph, that an error was made when building the the bond graph, or that there was detail in the IPM left out of the bond graph. Furthermore, using non-preferred causality makes equation derivation from the bond graph more complicated. The junction preferred causality is demonstrated in Figure 2.5. There are two rules to follow when assigning causality to junctions: 0-junctions must have exactly one effort-in causality, and 1-junctions must have exactly one effort-out causality.

2.2.3. Bond Graph Building Algorithm. Bond graphs can be universally understood across scientific disciplines with the unifying bond graph notation, which means they can be universally built as well. The construction of a bond graph from an ideal physical model, or IPM, follows a repetitive algorithm. Consequently, the builder does not need to understand all scientific domains in the IPM to create the bond graph and can simply follow the algorithm.

Table 2.3. Bond graph port elements and their preferred causality.

Port Element	Preferred Causality
Effort Source	S_e ⟶|
Flow Source	S_f |⟶
C-element	C ⟵|
I-element	I |⟵
R-element	R ⟵| or R |⟵
Transformer	|⟶ TF |⟶ or ⟶| TF ⟶|
Gyrator	|⟶ GY ⟶| or ⟶| GY |⟶

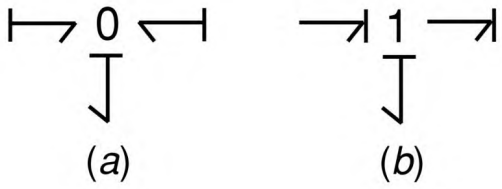

Figure 2.5. Preferred causality for (*a*) 0-junctions and (*b*) 1-junctions.

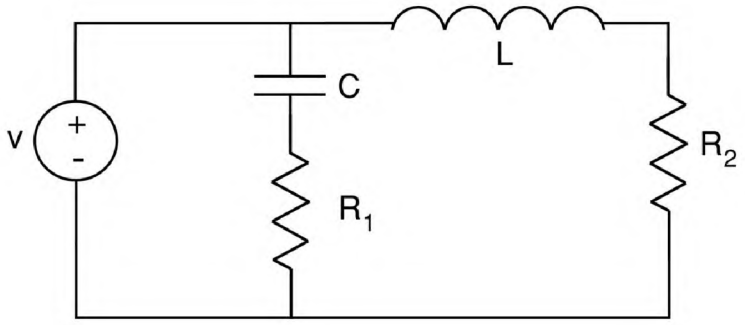

Figure 2.6. Ideal physical model of an RLC circuit.

The best way to explain the algorithm is with an example. The following example will be in the electrical domain, and the same procedure can be used in the hydraulic domain. For the linear and rotational mechanical domains, the steps will be the same with slight differences that will be noted in each step starting at the second step. Figure 2.6 is an IPM of an RLC circuit. The first step to translate an IPM into a bond graph is to identify all of the components in the IPM and determine which port elements they are. Figure 2.6 has a voltage source, v, capacitor C, inductor L, and two resistors R_1 and R_2. These IPM components will correlate to port elements as follows: voltage source to effort source, capacitor to C-element, inductor to I-element, and resistances to R-elements, as shown in table 2.4. Table 2.4 also shows how the port elements will be represented on the bond graph. The port element lies on the left side of the semi-colon and the IPM component name lies to the right. The semi-colon is an indicator that the system component is assumed to be linear, and this will be reflected in the equations derived from the bond graph later on.

The second step is to identify the voltage, or effort, reference on the IPM and the node voltages, or areas of equal effort. This step can be seen in Figure 2.7. For the mechanical domain, a velocity, or flow, reference will be indicated, along with velocity vectors, or areas of equal flow.

Table 2.4. Step 1: List of port elements.

$$
\begin{aligned}
S_e &: v \\
C &: C \\
I &: L \\
R &: R_1 \\
R &: R_2
\end{aligned}
$$

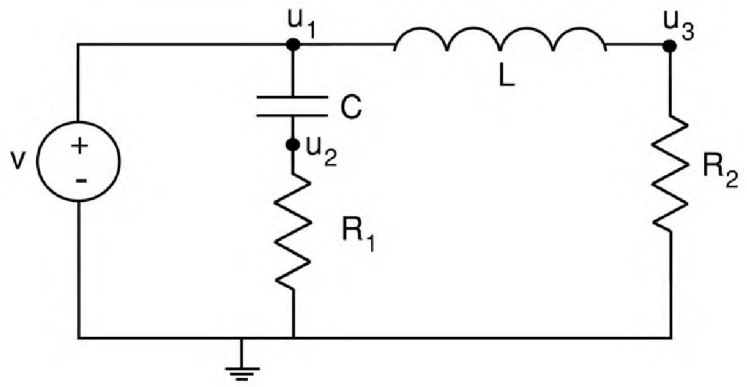

Figure 2.7. Step 2: Reference and node identification.

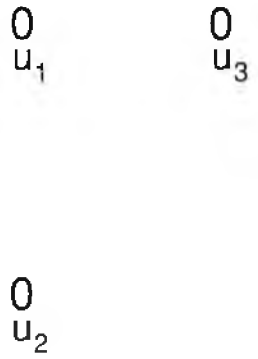

Figure 2.8. Step 3: 0-junction bond graph base.

The third step is to begin forming the base of the bond graph by placing a 0-junction for each node indicated in the previous step, as shown in Figure 2.8. For mechanical systems, a 1-junction will be placed for each velocity vector noted in the prior step.

The fourth step is to determine where voltage drops, or effort differences, occur and place them on the bond graph as 0-junctions as well. This step can be seen in Figure 2.9. Consequently, the velocity, or flow, differences will be shown as 1-junctions and added to the bond graph when in the mechanical domain.

The fifth step is to connect the 0-junctions with 1-junctions using bonds, as illustrated in Figure 2.10. For the mechanical domain, the 1-junctions will be connected using 0-junctions.

The sixth step is to add the port elements listed in table 2.4 to the bond graph by connecting them to the proper junctions using bonds, as seen in Figure2.11. The effort source is placed on a power bond pointing towards 0-junction u_1. This makes sense when looking back at Figure 2.7 in step two, as the voltage source lies along node u_1 and supplies power into the system. The C-element is connected to the u_{12} junction using a bond pointing away from the junction and into the C-element. Analyzing the second step again, it is clear that this bond graph placement is correct as the capacitor lies between nodes u_1 and u_2

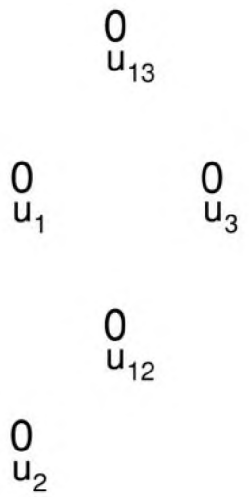

Figure 2.9. Step 4: Addition of effort difference 0-junctions.

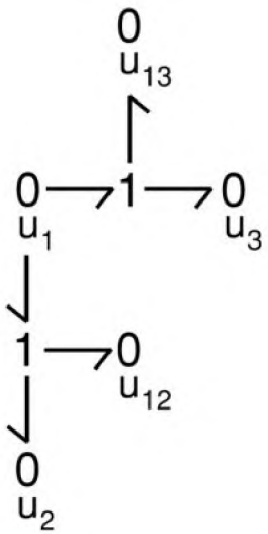

Figure 2.10. Step 5: Connecting 0-junctions to 1-junctions using bonds.

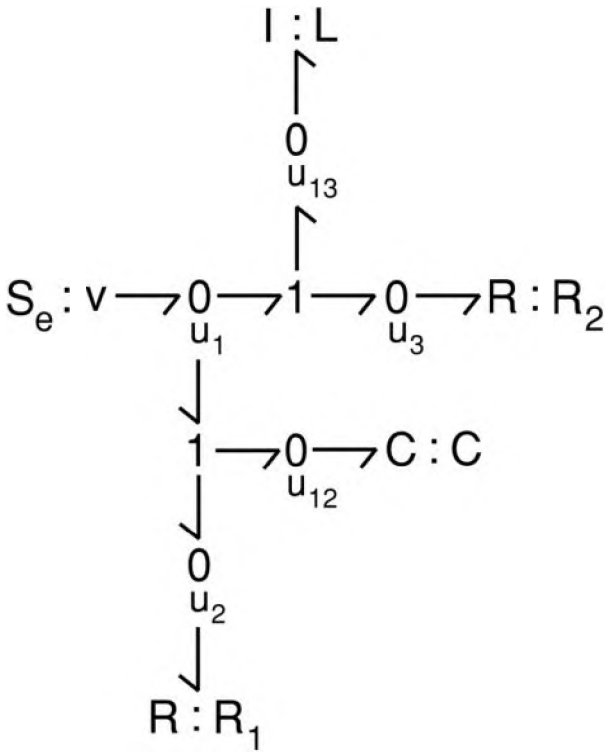

Figure 2.11. Step 6: Completion of the unsimplified bond graph.

where the voltage drop, u_{12}, occurs. The power bond points toward the C-element because power is entering the capacitor to be stored. The I-element is placed off of the 0-junction u_{13} with the power bond pointing into the I-element as it will be recieving power to store. The I-element is located off of the u_{13} junction because the inductor on the circuit in step two is between nodes u_1 and u_3 where the voltage drop u_{13} transpires. R-elements R_1 and R_2 are located off of 0-junctions u_2 and u_3 respectively. Both R-elements will have bonds pointing into them because the R-elements dissipate power. R_1 lies along node u_2 in Figure 2.7, and R_2 lies along node u_3, which are reflected in the bond graph accordingly. Once this is finished, the unsimplified version of the bond graph is complete.

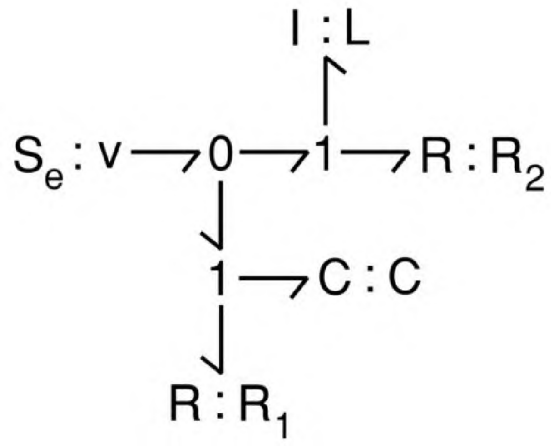

Figure 2.12. Step 7: Simplified bond graph.

The seventh step is to simplify the bond graph. Any time there is a 1-junction or a 0-junction with exactly two bonds connected to it, and both of the bonds are pointing in the same direction, the junction and two bonds can be replaced with a single bond. Applying this simplification rule to Figure 2.11 from step six removes 0-junctions u_2, u_{12}, u_3, and u_{13} and their connective bonds, and replaces them with a single bond. The simplified bond graph is displayed in Figure 2.12.

The eighth and final step of building a bond graph is to attach the causal strokes. There is an order of operations to follow when adding causal strokes to a bond graph. As seen in table 2.3 and Figure 2.5, the location of the bond is dependent upon the port-element or junction it is connected too. The preferred causality for sources and storage elements is fixed, so naturally the bonds connected to these elements will be the first to receive causal strokes. The rest of the port elements have some flexibility in the placement of their preferred causality, and can thus be implemented afterward. The causal strokes will be placed on bonds connected to port elements in this order: first effort and flow sources, then I- and C-elements, followed by transformers and gyrators, and lastly R-elements and junctions. Applying these rules to Figure 2.12 results in Figure 2.13. To get to this final bond graph,

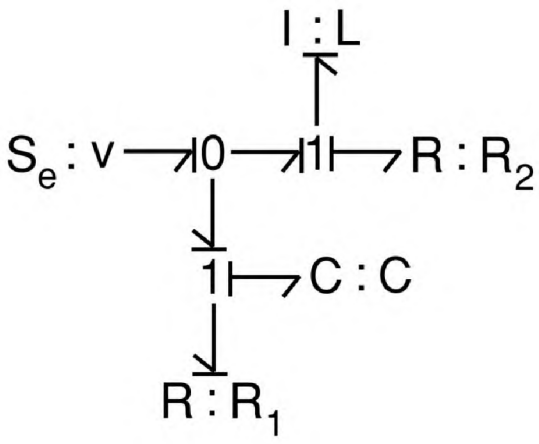

Figure 2.13. Step 8: Final simplified bond graph with causality.

the preferred causalities for bonds connecting to S_e:v, I:L, and C:C were attached. Since this example does not have any gyrators or transformers, the next pieces of the bond graph to examine are the junctions and R-elements. Once the source and storage causal strokes were added, one may note that this left the 0-junction with its preferred causality, as it has met the requirement for having exactly one effort-in causal stroke. Since the 0-junction was fully satisfied, the causal strokes on the bonds leaving the 0-junction and going into the 1-junctions are obvious. These bonds must have effort-out causality with respect to the 0-junction, and so these are the next causal strokes to be placed. All that remains are the bonds connecting to the R-elements. At this point, the bond entering R_2 still needs to satisfy the rule of having exactly one effort-out causal stroke on the 1-junction, thus an effort-in causal stroke with respect to said 1-junction was placed. Finally, analyzing the 1-junction connected to R_1 shows that the rule of having exactly one effort-out on the 1-junction has not yet been met. Hence, an effort-in causal stroke was placed on the bond going into R_1. This completes the final step of building a bond graph.

2.2.4. State Equations. After a bond graph is built, then a set of differential algebraic equations, or DAEs, can be derived from it. DAEs are comprised of differential equations and a set of algebraic constraints. The DAEs from a bond graph come in the form of state differential equations molded by the algebraic constraints determined by the underlying physics of the physical system. The state differential equations, or state equations for short, represent the *state* of the system's energy with respect to time. Our state equations are formulated to equate to the derivatives of the state variables, q and p, which are generalized displacement and generalized momentum, respectively. The final state equations should be in terms of state variables and external inputs, such as the values associated with port elements. These state variables lie on the storage I- and C-elements. Deriving these equations from bond graphs becomes simple with practice because it follows a repetitive algorithmic process. In order to demonstrate this algorithmic process, the example from Subsection 2.2.3 is continued here.

The first step to extracting equations from a bond graph is to assign a number to each bond on the bond graph, and then to indicate the location of state variables. The numerical assignments are arbitrary and are used to create unique variables. The C-elements always indicate a displacement state variable, or q. The I-elements always indicate a momentum state variable, or p. The result of step one on the example bond graph is shown in Figure 2.14.

The second step is to solve for the derivatives of the state variables. This step will be completed when there are no more effort or flow variables in the state equations. This step will be demonstrated over multiple equations. The initial state equations,

$$\dot{q}_6 = f_6 \tag{2.25}$$

and

$$\dot{p}_3 = e_3 \tag{2.26}$$

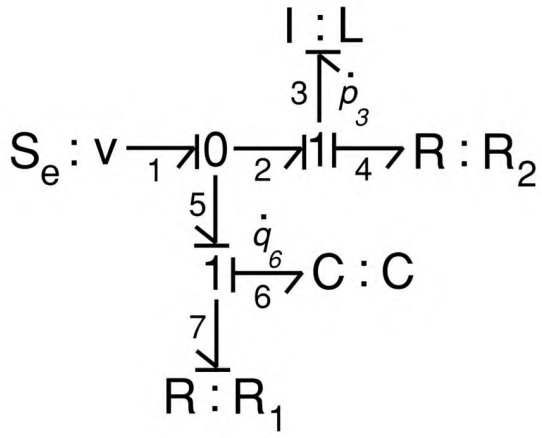

Figure 2.14. Step 1: Numerical assignment to bonds and state variable indication.

come from taking the derivative of (2.6) and (2.12). Since these state equations cannot have effort or flow variables in them, then f_6 and e_3 need to be substituted out. To begin this process, e_3 is the current focus, and because of the junction rules it is known that

$$e_2 - e_3 - e_4 = 0 \tag{2.27}$$

because at 1-junctions all efforts sum to zero, with bonds pointing into junctions taken as positive. Then, solving for e_3 gives

$$e_3 = e_2 - e_4. \tag{2.28}$$

Substituting (2.28) into (2.26) results in

$$\dot{p}_3 = e_2 - e_4. \tag{2.29}$$

Now, variables e_2 and e_4 must be replaced with expressions that only include sources and states. Creating an expression for e_2 is easy when evaluating the 0-junction, as all efforts at the 0-junction are equal. Therefore,

$$e_1 = e_2 = e_5, \tag{2.30}$$

with the effort source being equal to e_1 and the value of S_e:v being v, defines e_1 as

$$e_1 = v. \tag{2.31}$$

Substituting (2.31) into (2.29) for e_2 gives

$$\dot{p}_3 = v - e_4. \tag{2.32}$$

Now, the only effort variable in the state equation is e_4. Since e_4 lies on power bond 4, with power bond 4 going into the linear R:R_2 element, (2.16) can be used to evaluate for e_4. Using (2.16) results in

$$e_4 = f_4 R_2, \tag{2.33}$$

where R_2 is the value of that particular R-element. Substituting (2.33) into (2.32) results in

$$\dot{p}_3 = v - f_4 R_2. \tag{2.34}$$

Although the issue of e_4 was resolved, the new issue of f_2 has arisen. To solve for f_2, the 1-junction connected to bond 2 is analyzed. Considering the flows at 1-junctions are equal to each other, it is known that

$$f_2 = f_3 = f_4. \tag{2.35}$$

Thus, solving for either f_2 or f_3 results in f_4. For this example, the best variable to solve for next is f_3, as it is attached to I:L. With the semi-colon designating linearity, then (2.13) can be applied here to solve for f_3. This equation defines f_3 as

$$f_3 = \frac{p_3}{L}, \tag{2.36}$$

where L is the value of the I-element. Taking (2.36) and substituting it into (2.34) for f_4, as $f_3 = f_4$, results in the completed momentum state equation,

$$\dot{p}_3 = v - \frac{p_3}{L} R_2. \tag{2.37}$$

The second state variable to expand upon is $\dot{q}_6$. The initial equation,(2.25), defines $\dot{q}_6$ as the flow on bond 6, or f_6. Again, this variable cannot exist in the final state equation and will need to be solved for. The best way to begin solving for f_6 is to analyze the 1-junction connected to it. At this 1-junction the flows are all equal which means

$$f_5 = f_6 = f_7. \tag{2.38}$$

Variable f_7 is attached to the linear port element R_1, and (2.16) can be applied to f_7 as follows:

$$f_7 = \frac{e_7}{R_1}. \tag{2.39}$$

Then, f_6 can replace f_7 in (2.39) because they are equal to each other, and f_6 is substituted back into (2.25) resulting in

$$\dot{q}_6 = \frac{e_7}{R_1}. \tag{2.40}$$

Again, because effort and flow variables cannot remain in the final state equation the next variable to solve for is e_7. Looking at the same 1-junction, the efforts relative to it are

$$e_5 - e_6 - e_7 = 0 \tag{2.41}$$

and can be rewritten as

$$e_7 = e_5 - e_6. \tag{2.42}$$

Before substituting (2.42) back into (2.40), e_5 can be substituted out. The variable e_5 can be substituted out as it was already solved for in (2.30) and (2.31). So,

$$e_5 = v \tag{2.43}$$

and substituting it back into (2.42) gives

$$e_7 = v - e_6, \tag{2.44}$$

which can then be substituted back into (2.40) resulting in

$$\dot{q}_6 = \frac{v - e_6}{R_1}, \tag{2.45}$$

where R_1 is the value of that specific R-element. The effort variable e_6 must now be solved for. The variable e_6 is connected to port element C:C, so (2.7) can be used to solve for e_6. Doing so results in

$$e_6 = \frac{q_6}{C_6}. \tag{2.46}$$

Finally, substituting (2.46) into (2.45) gives

$$\dot{q}_6 = \frac{v - \frac{q_6}{C_6}}{R_1}, \tag{2.47}$$

which can be rewritten as

$$\dot{q}_6 = \frac{v}{R_1} - \frac{q_6}{C_6 R_1}, \tag{2.48}$$

which is the final form of the displacement state equation. From the start of this example in Subsection 2.2.3, an ideal physical model of a circuit has been built into a bond graph, and then state differential equations were derived from it. Being able to construct a bond graph and then derive equations from it is a useful tool set for real world applications, as will be discussed in the next section.

2.2.5. Cyber Physical Systems Represented as Bond Graphs. The use of bond graphs to model a cyber physical system, or CPS, is not new. Bond graphs have many benefits that make building a system model, deriving equations from said model, and understanding the system as a whole much easier for the end user. These benefits emerge from the universal bond graph notation, and is the reason why many have adopted this modeling method to overcome the complexities that multi-domain cyber physical systems present. Bond graph modeling was used in [9] to be built concurrently alongside a virtual model of an airplane's hydraulic breaking system to give the system engineer a more holistic perspective. Individual parts of a wind turbine were modeled with bond graphs, and [10] combined those submodels to create a bond graph representative of an entire wind turbine for a cohesive view of how each submodel interacts with one and other. A steer-by-wire system, where the steering wheel and front wheel axle are connected by a controller in place of mechanical connections, was represented by bond graphs in [11] to validate the controller mechanism. Bond graphs were taken a step further in [12], as more complex port elements were algorithmically replaced and then transformed into a signal flow diagram to eliminate the DAEs produced by bond graphs to shorten simulation time. Bond graph modeling is versatile and widely used to represent CPSs and is used to represent a microgrid in this paper.

2.2.6. Microgrid Overview. The following is a summary of corroborated information about microgrids from [13], [14], and [15]. A microgrid consists of a one or more energy microsources connecting to the grid at a subtransmission or distribution level feeding some load. The microsources can be either renewable or non-renewable and range from photovoltaic panels, to wind turbines, to fuel cells. Since there are various microsources that can be connected to the grid with varying dynamics, a power electronics device along with internal energy storage is used to ensure a proper grid connection regardless of the type of microsource in place. This allows for a plug and play type of system that allows microgrids to be modular. Microgrids are beneficial to both the utilities and the customers. The utilities are able to benefit from the flexibility of microgrids, and can easily enable them if an issue were to arise on the grid. The customers are able to benefit from a more efficient power source, as often times the microsources produce heat that is able to be re-purposed to heat nearby buildings. Customers have more consistent power reliability because microgrids are able to disconnect from the grid should a fault or other malfunction occur and reconnect once the incident has passed. This is known as islanding the microgrid, with the separation from the microgrid to the grid facilitated by a circuit breaker or a disconnect switch. Connecting and disconnecting from the grid can cause power imbalances and illuminate frequency differences, which can be resolved using a power versus frequency droop controller. Another controller used within a microgrid is a voltage versus reactive power droop controller. This controller implements voltage control that aids with stability of the microgrid. Between the physical power producing aspects of a microgrid and the supervisory controls, a microgrid can easily be categorized as a cyber physical system.

2.2.7. Microgrids Represented as Bond Graphs. Microgrids are complex multidomain cyber physical systems. To represent microgrids in a digestible manner, researchers have been opting for the bond graph method. For instance, a solar panel and thermal collector system along with a thermoelectric device was modeled with the universal bond graph method in [16] to eliminate challenges caused by multidomain system modeling. In [17],

a hybrid microgrid and solid state transformer, or SST, system was modeled using bond graphs for a few reasons. One such reason is that SSTs have bidirectional power movement and bond graphs allow for bidirectional power representation. Equally important, the bond graph modeling method is able to connect submodels together, which is useful when illustrating a hybrid system containing subsystems. Another reason this system utilizes bond graph modeling is because this hybrid microgrid includes a PI controller, and thus it is a cyber physical system, or CPS, which benefits from this type of system modeling. The authors of [17] wanted to illustrate the advantages of bond graph modeling in multidomain systems while also establishing the validity and usefulness of them. Lastly, a a DC mesh microgrid is presented in [18]. The mesh microgrid topology consists of a solar panel and battery storage mesh, an electric vehicle charging mesh, and an arbitrary load mesh. These meshes were then modeled as bond graphs in order to derive equations to help with load balancing in the system. Then, in [19], the same authors implemented a supervisory control system amongst the same DC mesh microgrid to optimize the load balancing. Microgrid system comprehension and modeling thrives when the bond graph method is implemented because it creates a straight forward construction, derivation, and analysis process.

2.2.8. Role of Bond Graph Modeling in Cyber Physical System Security. As technology advances, it is readily available to be implemented and integrated into physical systems more often. Increasing or adding technology usage in physical systems opens up these systems to cyber attacks. Bond graph modeling is applicable to curtailing this concern due to the physics based equations derived from it. The physics based equations come from the law of conservation of energy — energy cannot be created or destroyed. Therefore, if a hacker were to try to execute a deception attack by altering data monitoring devices or sensors, then the bond graph equations could recognize the law of conservation of energy was broken and sense the cyber security breach. Researchers Zerdazi and Fezari proposed a method using bond graphs to detect deception of supervisory control and data acquisition, or SCADA devices, in [20]. The proposed approach is to incorporate the cyber attack into

the bond graph by altering the port elements representing data senors. The alterations to the sensor port elements are represented by a source element either adding or subtracting data on said sensor. The idea is to demonstrate the effects of altering data on the SCADA device, as well as the physical system, through these altered port elements. The altered data can then be compared to the state equations derived from the bond graph that is in agreeance with the law of conservation of energy to detect the attack. This proposed approach was then carried out in [21] on a RobuCar Vehicle. A unique feature to this approach is that the authors built the bond graph of the system and the attack with preferred, or integral, causality, and then they would identify the sensor port elements and put them into derivative causality. Putting the sensor port elements into derivative causality helped create equations for the analytical redundancy relations, or ARRS, which then produce residuals. These residuals are then put into a matrix where the deception attack can be detected.

The aim of this paper is to model a microgrid with the bond graph method in order to derive equations that can be verified against an ideal physical model simulation, so future work can utilize this modeling method and information to create a deception detection algorithm.

3. METHODOLOGY

The three phase ac microgrid studied in this paper consists of three defined sections: generation, distribution, and the load. The generation is comprised of a synchronous generator. The synchronous generator receives torque from a governed engine and field current from a controller. The load exists as an induction motor. The induction motor will output a torque that feeds a mechanical load. Both the synchronous generator and induction motor were modeled after examples given in [22, pp. 271–272, 376–379]. The distribution portion lies between the generator and motor, and includes two radials — power supplied lines branching off of the main bus. One radial contains a capacitor bank while the other is a small resistive load. The IPS model is shown as a one-line diagram in Figure 3.1. The goal of this paper is to construct a bond graph model of the microgrid, derive equations from this model, simulate this model, and then compare the outcome of the bond graph simulation to a simulation of the IPS model to validate the bond graph model. If accurate results are found, then the physics based equations derived from the bond graph can be used to detect cyber attacks on the microgrid.

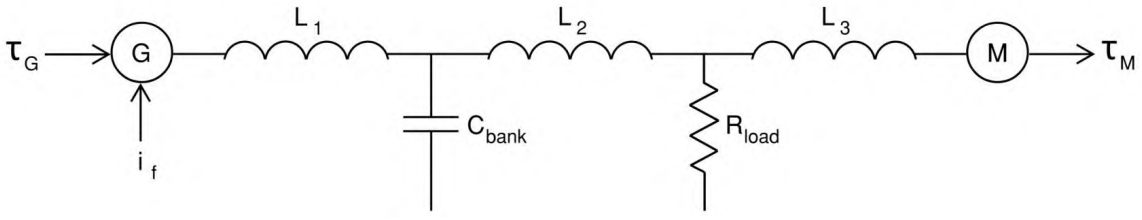

Figure 3.1. Microgrid one-line diagram.

3.1. PROCESS FOR BUILDING BOND GRAPH MODEL

In order to build a bond graph model of a microgrid, there needs to be an IPS, or ideal physical system, model to reference. The IPS considered in this paper is a three phase ac circuit with a synchronous generator feeding an induction motor. Line inductances, a capacitor bank, and a small resistive load are located between the two electrical machines. A governed engine supplies torque to enable power production on the generator, while a controller connected to the rotor regulates the field current. The motor's purpose is to provide torque to some mechanical load. Once this microgrid was constructed and defined, then the bond graph interpretation of this system could be built. This IPS representation of the microgrid is shown in Figure 3.1 as a one-line diagram.

3.2. BOND GRAPH MODEL: FIRST ATTEMPT

The first step to making the bond graph is to take each component in the IPS and relate them to their respective bond graph port element. This occurs through the categorization of the IPS elements at hand. The governed engine and load on the induction motor both classify as modulated effort sources, as both produce an effort and are overseen by controllers. Both the synchronous generator and the induction motor categorize as gyrators, as they are both relating effort to flow and flow to effort respectively. Additionally, the generator representation includes the field current controller, so no individual relationship was made for this controller. Each of the line inductances are I-type storage devices, and the capacitor bank is a C-type storage device. Lastly, the small resistive load is a dissipative device. The list of these bond graph elements and their appropriate nomenclature is located in table 3.1. After mapping the IPS components to their bond graph port elements, the bond graph can begin to take shape through the process explained in Subsection 2.2.3.

Table 3.1. List of bond graph elements for initial microgrid model.

$$
\begin{aligned}
MS_e &: T_G \\
MS_e &: T_M \\
GY &: G \\
GY &: M \\
I &: L_1 \\
I &: L_2 \\
I &: L_3 \\
C &: C_{bank} \\
R &: R_{load}
\end{aligned}
$$

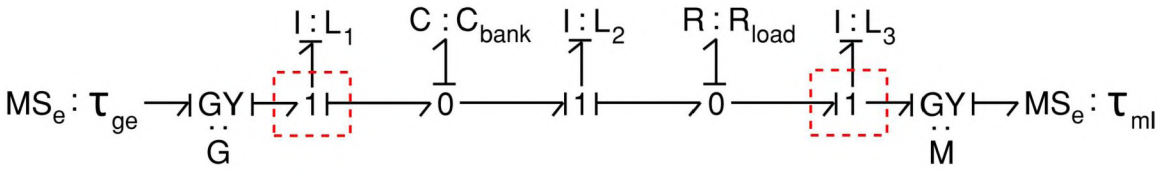

Figure 3.2. First attempt at microgrid bond graph model.

As can be seen in Figure 3.2, this bond graph has causality problems at the one-junctions placed immediately after and before the gyrators. Recalling information from Sub-subsection 2.2.2.4, this causality error is obvious because 1-junctions must have exactly one effort-out indication of causality, whereas both 1-junctions in question have two effort-out causality strokes. Finding causality issues indicates important details of the IPS were either left out of the bond graph model, left out of the IPS, or the translation from an IPS to a bond graph was done incorrectly.

After checking that the translation from the IPS to the bond graph was done correctly, the only other possible problems with the bond graph model surround the amount of detail involved in either the IPS or the bond graph. Initially, missing detail from the bond graph is examined. The only component of the IPS without a designated port element representation

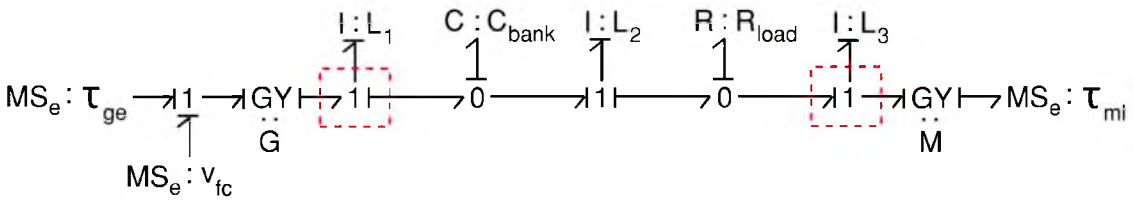

Figure 3.3. Second attempt at microgrid bond graph model with field current controller.

in Figure 3.2 is the field current controller, as it was incorporated with GY:G. A second attempt at the bond graph model including the field current controller is presented in Figure 3.3. The field current controller is represented as a modulated effort source because controlling the current also regulates the output voltage, or effort, from the controller and into the rotor of the synchronous generator. Although this new modulated effort source gives more detail to the bond graph, this information is not enough to fix the causality issues occurring in the same locations as the previous bond graph.

Therefore, there must be more detail added so the bond graph causality issue can resolve. This detail may be added to either the bond graph, or both the IPS and the bond graph. If detail is added to the IPS, then detail is simultaneously added to the bond graph, as the IPS components and bond graph elements are directly correlated. There are some instances where only the bond graph needs more detail added, as these details are not explicitly shown in the IPS but are implicitly shown. For instance, Figure 3.1 explicitly indicates there is a resistive load between line inductances L_2 and L_3 by using a resistor symbol. The resistor symbol represents the resistance in a circuit. There is more detail that *could* be added to the one-line diagram to depict each dissipative device located on the R_{load} radial; however, this level of detail is unnecessary because the end effect on the overall system remains the same: energy is dissipated through the resistive load. Using the same example, the generator and motor are represented implicitly. When looking at the round

circle used to depict these electrical machines, no information about the machines is given through this symbol other than the distinction of a generator, G, or motor, M. The idea that more is happening behind that round circle within the electric machines is *implied*, and additional detail may only need to be added to the bond graph to give more detail to this implication.

When analyzing both system models in Figures 3.1 and 3.3, the level of detail regarding the line inductances, capacitor bank, load resistance, and sources appears sufficient. However, the synchronous generator and induction motor detail could be expanded upon to illustrate the physics occurring within these electric machines. The need for additional detail of these electric machines is indicated on the bond graph, as the location of these causality issues are directly after and before the gyrator elements. Various ways to model the synchronous generator and induction motor in the IPS and the bond graph will be explored in the following Subsection to determine if detail needs to be added to both the IPS and bond graph or just the bond graph .

3.3. ELECTRIC MACHINES REPRESENTATION EXPLORATION

The goal of exploring electric machine representation is to add detail to the one-line diagram in Figure 3.1 to subsequently add detail to the bond graph through direct component translation. Electric machines convert electrical energy to mechanical energy or mechanical energy to electrical energy [22]. Throughout this energy conversion, the law of conservation of energy is upheld. On the electrical side of the machine, power is either produced or consumed by means of voltage and current, with some power dissipation through resistances in the coils of the machine. On the mechanical side of the machine, power is either produced or consumed by means of torque and angular velocity, with some power dissipation through friction and windage and some power storage through inertia. The conversion between these two scientific domains results from stored energy in the form of a magnetic field. In ac machines flux linkage occurs between the rotor and stator due to

flux flowing through coils. When the rotor, or rotating portion of the electrical machine, rotates relative to sets of coils on the stator, or stationary portion of the electrical machine, the flux linkage changes and therefore energy is transferred. Understanding how this energy conversion works is important to creating an accurate model of these electric machines.

Every model of an electromechanical machine undergoing energy conversion is built on the foundational magnetic field equation

$$\frac{dW_{fld}}{dt} = vi - P_{mech}, \tag{3.1}$$

Where W_{fld} is the stored magnetic energy, and $\frac{dW_{fld}}{dt}$ is how it changes with respect to time. P_{mech} is the mechanical power output, which may be the positive or negative depending on if the electric machine is motoring or generating power. The convention of 3.1 is for a motor, and it does not take into account losses from the core, coil resistances, or friction.

There are three main ways to portray this phenomenon in electric machines through either schematic diagrams, equivalent circuits, or an IPS model. Schematic diagrams show a cross-sectional view of electric machines. This view includes the stator topology, rotor topology, coil windings, and the direction of current in those windings. This type of model allows the end user to identify the number of poles present, the flux linkage path and distribution, the direction of magnetic axes, and the placement of those axes. Equivalent circuits are derived from and maintain the same characteristics of an IPS model. The characteristics are represented with circuit components such as voltage sources, inductors, capacitors, resistors, and transformers. Equivalent circuits are used for mathematical calculations of the IPS model because known concepts like Kirchhoff's laws can be easily applied to an equivalent circuit. IPS models are top-level idealized representations of complex devices and systems. For example, an IPS model of an electric machine can simply be a labeled circle as shown in Figure 3.1. IPS models are used to create equivalent circuits and bond graph models. While these three modeling methods are useful for understanding infor-

mation about electric machines from different perspectives, none of these perspectives are capable of detailing the energy conversion process. As none of these three options are of use, instead of trying to expand the IPS representation of the generator and motor in Figure 3.1 with one of the above modeling techniques, only the bond graph will be expanded upon.

Using the bond graph method to expand the generator and motor is the most logical approach, as bond graphs focus on modeling energy interactions like energy conversion. Port elements, as explained in Sub-subsection 2.2.2.2, identify the different ways energy is being exchanged throughout a given physical system. The first and second attempts at building a bond graph from the one-line diagram in Figure 3.1 resulted in Figures 3.2 and 3.3, where gyrators represent the synchronous generator and induction motor. For the generator, the gyrator represents the mechanical rate of change of rotation of the rotor $\dot{\theta}$ with respect to time, also known as *flow*, converting to the rate of change in magnetic flux linkage $\dot{\lambda}$ on the stator and rotor with respect to time, or *effort*. The inverse gyrator representation is true for the induction motor, where the change is going from an effort, $\dot{\lambda}$, to a flow, $\dot{\theta}$. To expand the gyrator components for these electromechanical devices to illustrate the actual conversion from $\dot{\theta}$ to $\dot{\lambda}$ and vice versa, a new port element is implemented.

The IC port element was introduced by Dean Karnopp in [23] to represent electric machines. The IC-element is a combination of the energy storing I- and C-elements, and it is considered a multi-port element with as many ports on the I side as are needed and a single port on the C side. The I portion of the IC-element is representative of magnetic flux energy storage from the stator windings, and the C portion is representative of mechanical energy storage from rotational displacement of the rotor. To gain a better understanding of how the IC-element works, an example of a two pole, four winding electromechanical motor from [23] modeled with the IC-element is shown in Figure 3.4, and the analysis of [23] is summarized here, with updates to match the microgrid model of Figure 3.6. The I side of Figure 3.4 shows effort sources $S_e : v_1$ and $S_e : v_2$ providing voltage to the rotor, or field of the motor, and effort sources $S_e : v_a$ and $S_e : v_b$ providing voltage to the stator, or

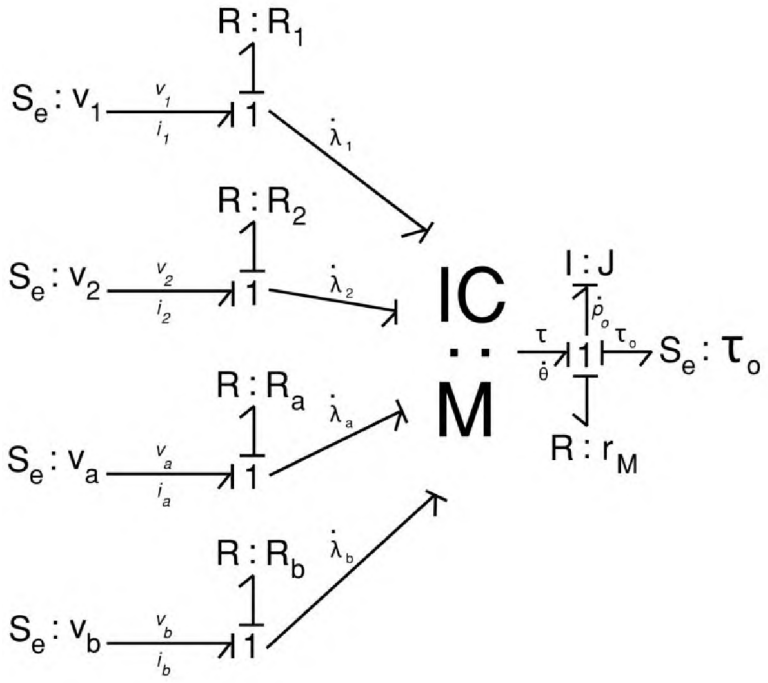

Figure 3.4. Bond graph of a two pole, four winding electromechanical motor with IC-element implementation.

armature. The resistive losses on the windings are modeled with $R : R_1$ through $R : R_b$, and $\dot{\lambda_1}$ through $\dot{\lambda_b}$ represent the induced voltage on the windings. The C side inputs rotational displacement at a rate of $\dot{\theta}$ to the mechanical portion of the motor. Power loss due to friction is shown as $R : r_M$, and energy storage due to inertia is indicated by $I : J$ which changes due to momentum at a rate of $\dot{p_o}$. The output of the mechanical side of the motor, $S_e : \tau_o$, is represented as an effort source because it is providing torque to some mechanical load.

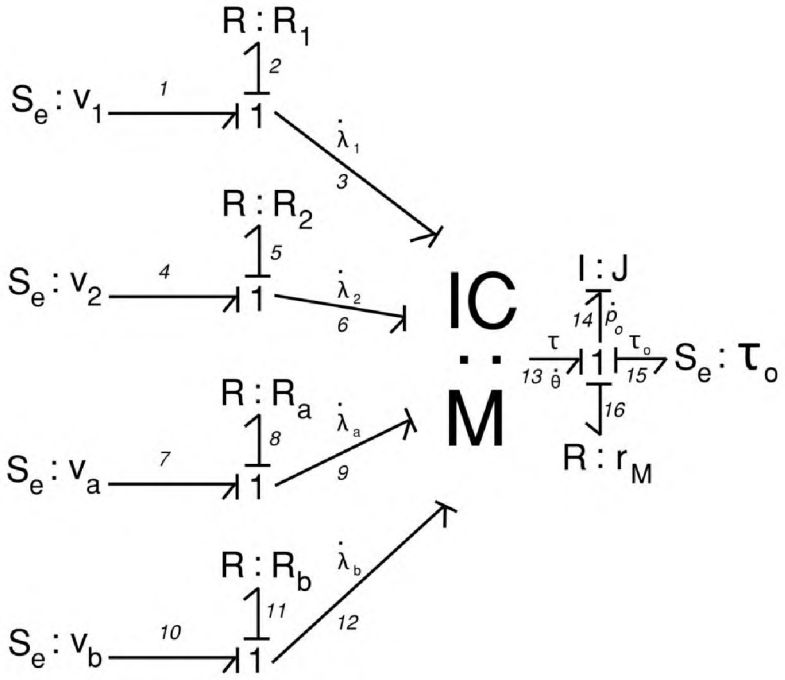

Figure 3.5. Figure 3.4 numbered for equation derivation.

To derive equations from Figure 3.4, each of the bonds were labeled numerically as shown in Figure 3.5. The I side of the motor will be calculated using vectors and matrices. The effort vectors are

$$\overrightarrow{e_{S_e}} = \begin{bmatrix} e_1 \\ e_4 \\ e_7 \\ e_{10} \end{bmatrix}, \quad \overrightarrow{e_R} = \begin{bmatrix} e_2 \\ e_5 \\ e_8 \\ e_{11} \end{bmatrix}, \quad \overrightarrow{e_\lambda} = \begin{bmatrix} e_3 \\ e_6 \\ e_9 \\ e_{12} \end{bmatrix}, \tag{3.2}$$

where $\overrightarrow{e_{S_e}}$ is the effort vector from the bonds connected to the effort sources, $\overrightarrow{e_R}$ is the effort vector from the bonds connected to the dissipative R-elements, and $\overrightarrow{e_\lambda}$ is the effort vector from the bonds connected to the IC-element. The flow vectors are

$$\overrightarrow{f_{S_e}} = \begin{bmatrix} f_1 \\ f_4 \\ f_7 \\ f_{10} \end{bmatrix}, \quad \overrightarrow{f_R} = \begin{bmatrix} f_2 \\ f_5 \\ f_8 \\ f_{11} \end{bmatrix}, \quad \overrightarrow{f_\lambda} = \begin{bmatrix} f_3 \\ f_6 \\ f_9 \\ f_{12} \end{bmatrix}, \tag{3.3}$$

where $\overrightarrow{f_{S_e}}$ is the flow vector from the bonds connected to the effort sources, $\overrightarrow{f_R}$ is the flow vector from the bonds connected to the dissipative R-elements, and $\overrightarrow{f_\lambda}$ is the flow vector from the bonds connected to the IC-element. The effort source, state variable λ and $\dot{\lambda}$ vectors are

$$\overrightarrow{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_a \\ v_b \end{bmatrix}, \quad \overrightarrow{\lambda} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_a \\ \lambda_b \end{bmatrix} \quad and \quad \overrightarrow{\dot{\lambda}} = \begin{bmatrix} \dot{\lambda}_1 \\ \dot{\lambda}_2 \\ \dot{\lambda}_a \\ \dot{\lambda}_b \end{bmatrix}, \tag{3.4}$$

respectively. The values of the R-elements are represented by the matrix

$$\mathbf{R} = \begin{bmatrix} R_1 & 0 & 0 & 0 \\ 0 & R_2 & 0 & 0 \\ 0 & 0 & R_a & 0 \\ 0 & 0 & 0 & R_b \end{bmatrix}. \tag{3.5}$$

There is also a matrix to account for the inductances on the field and armature of the motor which is

$$\mathbf{L}(\theta) = \begin{bmatrix} L_f & 0 & L\cos(\theta) & L\sin(\theta) \\ 0 & L_f & -L\sin(\theta) & L\cos(\theta) \\ L\cos(\theta) & -L\sin(\theta) & L_a & 0 \\ L\sin(\theta) & L\cos(\theta) & 0 & L_a \end{bmatrix}, \tag{3.6}$$

where L_f and L_a are the self inductances of the field and armature, and L is the value of the mutual inductance. The self inductances are defined as

$$L_f = \frac{1}{n}L + l_f \tag{3.7}$$

and

$$L_a = nL + l_a, \tag{3.8}$$

where n is the turns ratio between the windings on the rotor and stator, and l_f and l_a are the leakage inductances on the field and armature, respectively. Now that the vectors and matrices are set up, the equation derivation can take place. Defining v from (3.1) as

$$v = \frac{d\lambda}{dt} \tag{3.9}$$

and P_{mech} as

$$P_{mech} = \tau\frac{d\theta}{dt} \tag{3.10}$$

results in

$$\frac{dW_{fld}}{dt} = \frac{d\lambda}{dt}i - \tau\frac{d\theta}{dt}, \tag{3.11}$$

which will be implemented later. From (3.9), the beginning of the $\dot{\lambda}$ state equation can take place as voltage equals effort; therefore

$$\overrightarrow{\dot{\lambda}} = \overrightarrow{e_\lambda}, \tag{3.12}$$

which is also validated from (2.8) and (2.9). Then efforts at the 1-junctions on the I side are summed to zero in order to solve for $\overrightarrow{e_\lambda}$ which is

$$\overrightarrow{e_\lambda} = \overrightarrow{e_{S_e}} - \overrightarrow{e_R}. \tag{3.13}$$

The vector $\overrightarrow{e_{S_e}}$ can be substituted out for $\overrightarrow{v}$ in (3.13) because $\overrightarrow{e_{S_e}} = \overrightarrow{v}$ resulting in

$$\overrightarrow{e_\lambda} = \overrightarrow{v} - \overrightarrow{e_R}. \tag{3.14}$$

Next, solving for $\overrightarrow{e_R}$ using (2.16) gives

$$\overrightarrow{e_\lambda} = \overrightarrow{v} - \mathbf{R}\overrightarrow{f_R} \tag{3.15}$$

The last term in (3.15) to solve for is $\overrightarrow{f_R}$. Knowing that flow is equal to current, and that $\lambda = Li$, gives

$$\overrightarrow{\lambda} = \mathbf{L}(\theta)\overrightarrow{f_R}. \tag{3.16}$$

Rewriting (3.16) to isolate the flow variable, with attention to matrix operations, results in

$$\overrightarrow{f_R} = \mathbf{\Gamma}(\theta)\overrightarrow{\lambda}, \tag{3.17}$$

where $\mathbf{\Gamma}(\theta)$ is the inverse of $\mathbf{L}(\theta)$, which is

$$\mathbf{\Gamma}(\theta) = \begin{bmatrix} L_a & 0 & -L\cos(\theta) & -L\sin(\theta) \\ 0 & L_a & L\sin(\theta) & -L\cos(\theta) \\ -L\cos(\theta) & L\sin(\theta) & L_f & 0 \\ -L\sin(\theta) & -L\cos(\theta) & 0 & L_f \end{bmatrix} \frac{1}{L_f L_a - L^2}. \tag{3.18}$$

Finally, substituting in (3.17) into (3.15) gives the final state equation

$$\dot{\vec{\lambda}} = \vec{v}(t) - \mathbf{R\Gamma}(\theta)\vec{\lambda} \tag{3.19}$$

Now that the I side equations have been found, it is time to solve for the C side equations. Taking the derivative of (2.12) with respect to bond 14 gives

$$\dot{p}_o = e_{14}. \tag{3.20}$$

Then, efforts on the C side 1-junction are summed to zero, and e_{14} is solved for as

$$e_{14} = e_{13} + e_{15} - e_{16}, \tag{3.21}$$

where output torque, τ_o of the motor is taken as positive. Next, e_{13} is defined as the torque produced from the motor, or

$$e_{13} = \tau(\theta, \lambda), \tag{3.22}$$

where $\tau(\theta, \lambda)$ is defined as

$$\tau(\theta, \lambda) = -\frac{\partial W_{fld}}{\partial \theta}, \tag{3.23}$$

which comes from (3.11) through integration of the independent paths of flux linkage and rotational displacement. The integration of (3.11) results in

$$\int_{t_o}^{t} \left(\frac{dW_{fld}}{dt}\right) dt = \int_{t_o}^{t} \left(\frac{d\lambda}{dt} i\right) dt - \int_{t_o}^{t} \left(\tau \frac{d\theta}{dt}\right) dt. \tag{3.24}$$

Taking advantage of the path independence allows (3.24) to become

$$W_{fld}(\theta, \lambda) = \int_{\lambda_o}^{\lambda} i d\lambda - \int_{\theta_o}^{\theta} \tau d\theta, \tag{3.25}$$

where the bounds on $id\lambda$ are chosen such that the magnetic field is zero, which is where the electromechanical machine stores energy during the energy conversion process. If there is is no magnetic stored energy, then $W_{fld}(\theta, \lambda) = 0$, then there will be no mechanical force produced due to rotational inertia, and that leaves

$$W_{fld}(\theta, \lambda) = \int_{\lambda_o}^{\lambda} id\lambda, \tag{3.26}$$

where i can be substituted with (3.17) which is

$$W_{fld}(\theta, \lambda) = \int_{\lambda_o}^{\lambda} \mathbf{\Gamma}(\theta)\overrightarrow{\lambda}\, d\lambda. \tag{3.27}$$

Integrating (3.27) once more results in

$$W_{fld}(\theta, \lambda) = \frac{1}{2}\overrightarrow{\lambda}^t \mathbf{\Gamma}(\theta)\overrightarrow{\lambda}. \tag{3.28}$$

Once $W_{fld}(\theta, \lambda)$ is known, then the partial derivative of the magnetic field with respect to θ can be done, which is how (3.23) was derived, where

$$\frac{\partial W_{fld}}{\partial \theta} = -\frac{1}{2}\overrightarrow{\lambda}^t \frac{d\mathbf{\Gamma}(\theta)}{d\theta}\overrightarrow{\lambda}. \tag{3.29}$$

Now that the derivation of $\tau(\theta, \lambda)$ is known for context, the bond graph equations can continue to be solved for by substituting (3.22) into (3.21), which gives

$$e_{14} = \tau(\theta, \lambda) + e_{15} - e_{16}. \tag{3.30}$$

Since the output torque, τ_o, is the value of the effort source $S_e : \tau_o$, then $e_{15} = \tau_o$. Substituting in τ_o for e_{15} in (3.30) results in

$$e_{14} = \tau(\theta, \lambda) + \tau_o(t) - e_{16}, \tag{3.31}$$

where $e_{16} = f_{16} r_M$. Solving for f_{16} using the knowledge that bonds on 1-junctions have equal flow, $f_{13} = f_{14} = f_{15} = f16$, and 2.13 gives

$$f_{16} = f_{14} = \frac{p_o}{J}. \tag{3.32}$$

Then, substituting in (3.32) into (3.31), and then substituting that result into (3.20) gives the final state equation as

$$\dot{p_o} = \tau(\theta, \lambda) + \tau_o(t) - \frac{p_o r_M}{J}. \tag{3.33}$$

The last state equation,

$$\dot{\theta} = f_{13}, \tag{3.34}$$

is easily solved for as $f_{13} = f_{16} = \frac{p_o}{J}$; therefore,

$$\dot{\theta} = \frac{p_o}{J}. \tag{3.35}$$

The bond graph model of the IC-element depicted in Figure 3.5, and the state equations derived from it are all with respect to an electric machine that is motoring. For an electric machine that is generating, the IC-element would be reversed with τ_o being an input torque to the system, and the induced voltages on the windings, $\overrightarrow{\lambda}$, would be an output of the system. The equations derived from a generating electric machine would also have reversed polarity from those of the motoring machine, with the base equation for magnetic field storage being

$$\frac{dW_{fld}}{dt} = P_{mech} - vi. \tag{3.36}$$

Now that the IC-element as been explained in depth, it can be implemented for a more accurate representation of the microgrid proposed in this paper to resolve the causality problems presented in Subsection 3.2 .

3.4. IC-ELEMENT IMPLEMENTATION

With the addition of the multiport IC-element comes the addition of ports connected to the I side of the IC-element. Therefore, a new table of port elements in the microgrid is necessary. Furthermore, the IC-element allows for representation of the $d-q$ reference frame [23] which also requires more port elements by showing both the d and q reference frames. For this paper, the $\alpha-\beta$ reference frame is considered. The use of the $\alpha-\beta$ reference frame, or Clarke transformation, allows for imaginary power to be calculated in the time domain and simplifies subsequent calculations by transforming a three phase system to a two phase system [24]. There are two forms of the Clarke transformation. This paper will utilize the power conserving Clarke transform defined as

$$\begin{bmatrix} v_0 \\ v_\alpha \\ v_\beta \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} v_a \\ v_b \\ v_c \end{bmatrix}, \tag{3.37}$$

where v_a, v_b, and v_c represent the a-, b-, and c-phases of a three phase ac system, and v_0, v_α, and v_β represent the three phase system *transformed* onto the $\alpha-\beta$ reference frame. With the implementation of the IC-element in place of the gyrator elements for the synchronous generator and induction motor, as well as the use of the power conserving Clarke transform, a new list of port elements is shown in Table 3.2. A subscript key for the list of port elements is detailed in Table 3.3.

Table 3.2. List of bond graph elements.

$$
\begin{aligned}
MS_e &: T_{ge} \\
MS_e &: v_{fc} \\
MS_e &: T_{ml} \\
S_e &: v_{short,Gf2} \\
S_e &: v_{short,Mf1} \\
S_e &: v_{short,Mf2} \\
R &: r_G \\
R &: R_{Gf1} \\
R &: R_{Gf2} \\
R &: R_{Ga\alpha} \\
R &: R_{Ga\beta} \\
R &: R_{load\alpha} \\
R &: R_{load\beta} \\
R &: R_{Mf1} \\
R &: R_{Mf2} \\
R &: R_{Ma\alpha} \\
R &: R_{Ma\beta} \\
R &: r_M \\
I &: J_G \\
I &: L_{2\alpha} \\
I &: L_{2\beta} \\
I &: J_M \\
C &: C_{bank\alpha} \\
C &: C_{bank\beta} \\
IC &: G \\
IC &: M
\end{aligned}
$$

Table 3.3. Subscript key.

Subscript	Definition
G	Generator
M	Motor
α	Alpha axis
β	Beta axis
θ	Angle of rotation, theta
f	Field
a	Armature
ge	Governed engine
fc	Field current controller
ml	Mechanical load

Using the port elements in Table 3.2, the information on IC-elements in Subsection 3.3, and following the bond graph building algorithm from Subsection 2.2.3, a bond graph of the microgrid one-line diagram proposed in this paper was able to be constructed with preferred causality. The completed bond graph demonstrating the preferred, or integral, causality is depicted in Figure 3.6.

Then, using the state equation derivations of the IC-element from Subsection 3.3 and following the steps from Subsection 2.2.4, state equations were derived from the numbered bond graph in Figure 3.7. All of the equations and their derivations for this bond graph can be found in Appendix 1 and Appendix 2.

3.5. MICROGRID COMPONENT INFORMATION AND NUMERICAL ANALYSIS

Up until this point, the different components of the microgrid have been left in generic forms while the causality issues were resolved. The bond graph is now complete with the IC-element implementation in place, allowing microgrid components to be described in

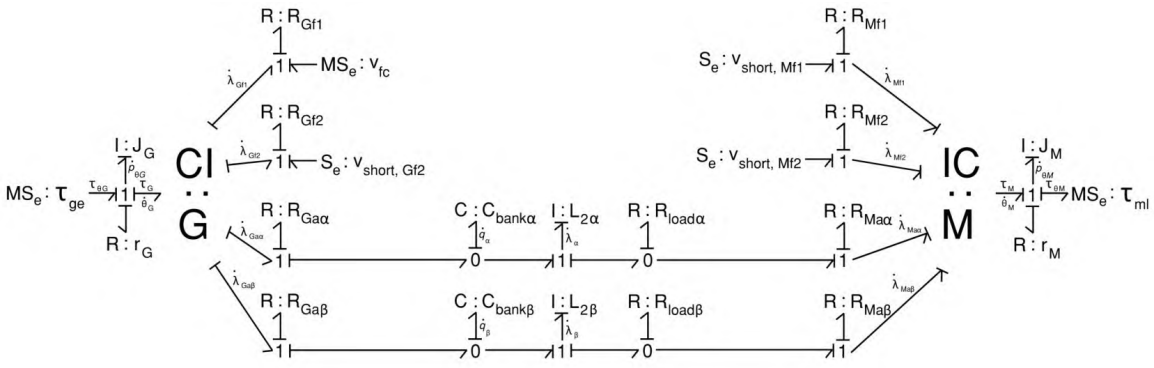

Figure 3.6. Microgrid bond graph with IC-element implementation.

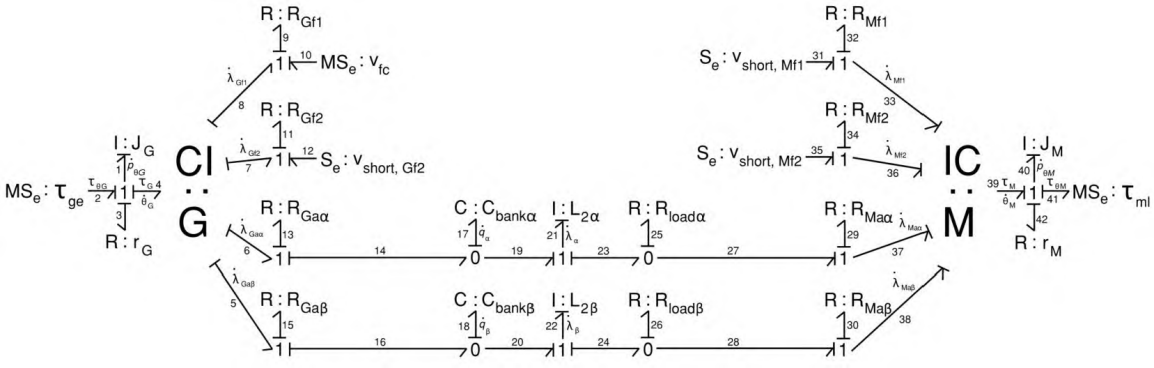

Figure 3.7. Microgrid numbered bond graph with IC-element implementation.

Table 3.4. Synchronous generator values.

Electric Machine Parameters	MATLAB or Numbered Variable	Bond Graph Variable	Value
Field Leakage Inductances in H	lfGen	N/A	$2.25e-3$
Armature Leakage Inductances in H	laGen	N/A	$8e-5$
Mutual Inductance in H	LGen	N/A	$4.42e-3$
Winding Ratio	n	G	1
Winding Resistance on Axis 1 of Field in Ω	R9	R_{Gf1}	1.5
Winding Resistance on Axis 2 of Field in Ω	R11	R_{Gf2}	0.06
Winding Resistance on α Axis of Armature in Ω	R13	$R_{Ga\alpha}$	$6e-4$
Winding Resistance on β Axis of Armature in Ω	R15	$R_{Ga\beta}$	$6e-4$
Friction in N	R3	r_G	0.007
Inertia in kgm^2	J1	J_G	0.42

detail. Furthermore, the equations found from Figure 3.7 were all formed symbolically. Merging the component details with the symbolic equations will result in a finalized bond graph with numerical results.

The synchronous generator and induction motor in the proposed microgrid are modeled after examples from [22, pp. 271–272, 376–379]. The microgrid as a whole is nominally a 3-phase, 60 Hz, 480 V line-to-line, 100 kW system. Within this system there exists a 2-pole round rotor synchronous generator and a 2-pole squirrel cage induction motor; both of which are rated for the system nominal values. Between the generator and motor lies the distribution grid, which includes 0.02 per-unit line inductances, an arbitrary 15 kW resistive load, and a capacitor bank that satisfies a 0.95 power factor. The specific system values are depicted in Tables 3.4, 3.5, and 3.6 for the synchronous generator, induction motor, and grid respectively.

Table 3.5. Induction motor values.

Electric Machine Parameters	MATLAB or Numbered Variable	Bond Graph Variable	Value
Field Leakage Inductances in H	lfMot	N/A	$0.729e-3$
Armature Leakage Inductances in H	laMot	N/A	$0.729e-3$
Mutual Inductance in H	LMot	N/A	$19.9e-3$
Winding Ratio	n	M	1
Winding Resistance on Axis 1 of Field in Ω	R32	R_{Mf1}	$23.1e-3$
Winding Resistance on Axis 2 of Field in Ω	R34	R_{Mf2}	$23.1e-3$
Winding Resistance on α Axis of Armature in Ω	R29	$R_{Ma\alpha}$	$30.3e-3$
Winding Resistance on β Axis of Armature in Ω	R30	$R_{Ma\beta}$	$30.3e-3$
Friction in N	R42	r_M	$83.3e-3$
Inertia in kgm^2	J40	J_M	2.5

Table 3.6. Distribution grid values.

Grid Parameters	MATLAB or Numbered Variable	Bond Graph Variable	Value
α Axis Bank Capacitance in F	C17	$C_{bank\alpha}$	$2.75e-4$
β Axis Bank Capacitance in F	C18	$C_{bank\beta}$	$2.75e-4$
α Axis Line Inductance in H	L21	$L_{2\alpha}$	$122.2e-6$
β Axis Line Inductance in H	L22	$L_{2\beta}$	$122.2e-6$
α Axis Load Resistance in Ω	R25	$R_{load\alpha}$	15.36
β Axis Load Resistance in Ω	R26	$R_{load\beta}$	15.36

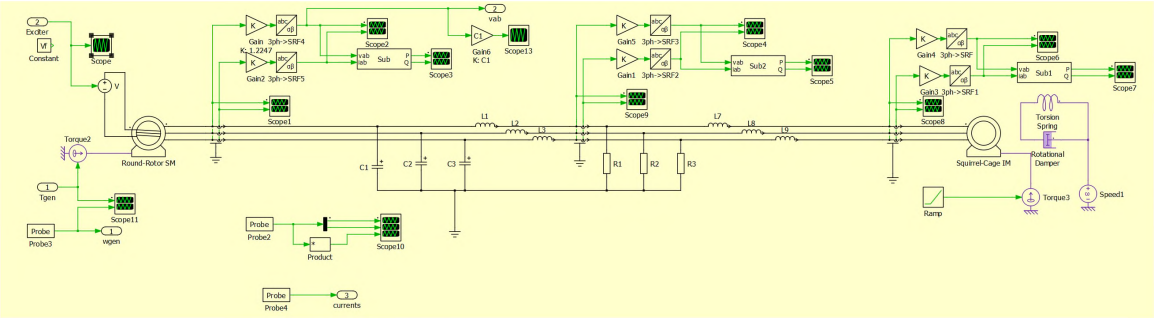

Figure 3.8. Ideal physical system model of microgrid in PLECS software.

3.6. IDEAL PHYSICAL SYSTEM SIMULATION

The proposed bond graph in this paper must be verified for accuracy and correctness. To do so, a simulation of the ideal physical system model of the microgrid was constructed in PLECS, and then the PLECS model was placed into Simulink and connected to the governed engine and field current controllers. The PLECS model is displayed in Figure 3.8, and the application of the PLECS circuit in Simulink is laid out in Figure 3.9. The initialization MATLAB code and equations for the Simulink and PLECS software is located in 3, and the PLECS variables relationships to MATLAB variables is listed in Table 3.7. After some trial and error, the synchronous generator needed to be increased in rating to properly feed the induction motor.

There is now a finalized microgrid bond graph with numerical data, and a completed ideal physical model simulation of the microgrid. The two systems can now be compared to see if bond graph modeling is an accurate way to represent critical cyber-physical infrastructure.

Table 3.7. MATLAB variable's and their correlated PLECS variables.

MATLAB or Numbered Variable	PLECS Variable
lfGen	Llf
laGen	Lls
LGen	Lm0
R9	Rf
R11	Rf
R13	Rs
R15	Rs
R3	F
J1	J
lfMot	LlrIM
laMot	LlsIM
LMot	LmIM
R32	RrIM
R34	RrIM
R29	RsIM
R30	RsIM
R42	FIM
J40	JIM
C17	C1
C18	C1
L21	L1
L22	L1
R25	R1
R26	R1

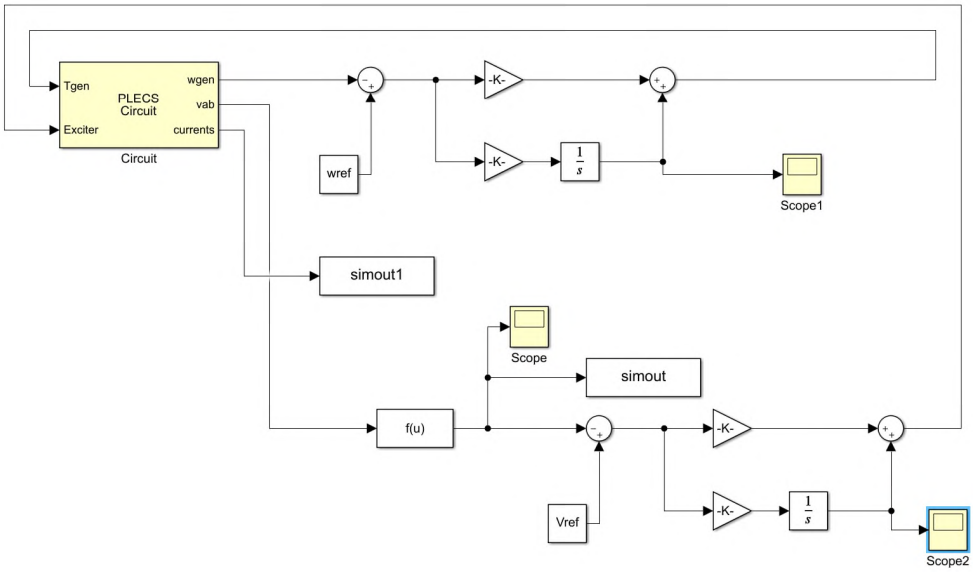

Figure 3.9. Governed engine and field current controller acting on PLECS circuit in Simulink software.

4. RESULTS AND DISCUSSION

4.1. GOVERNED ENGINE RESULTS

The integral torque of the PI controller for the bond graph is shown in Figure 4.1. The graph starts at a high torque that decreases until just after the three second mark when it begins to increase. This matches the results from the ideal physical system, or IPS, as demonstrated in Figure 4.2. These results make sense, as the governed engine is assumed to start with some load that needs to be met with a high torque. Then, between the time when the initial load is assumed and when the induction motor turns on the governed engine is not seeing any load causing a decrease in torque. Once the induction motor turns on, there is now a load for the governed engine to upkeep resulting in an increase in torque from the three to five second time range.

The next variables on the governed engine to be examined are the output torque and output speed. The output torque of the governed engine with respect to the bond graph and IPS are displayed in Figures 4.3 and 4.5, respectively. Both graphs follow the same trend of an increase in torque as time increases. This is an accurate representation of the output torque of the governed engine because there is no torque to output initially, and again, once the induction motor is energized the output toque increases to meet the needs of the load. The output speed of the governed engine from the bond graph and IPS are depicted in Figures 4.4 and 4.5. Both figures follow a trend starting with a high angular velocity that decreases until about the 4.2 second mark. Then, an increase in angular velocity occurs. This may seem incorrect at first, as a higher torque will cause the speed, or angular velocity, to increase, but when comparing the torque in Figure 4.3 to the speed in Figure 4.4 the opposite *seems* to be happening. However, it is important to notice the short range on the y-axis in Figure 4.4. This range goes from 375 rad/sec to roughly 381.5 rad/sec. So, while

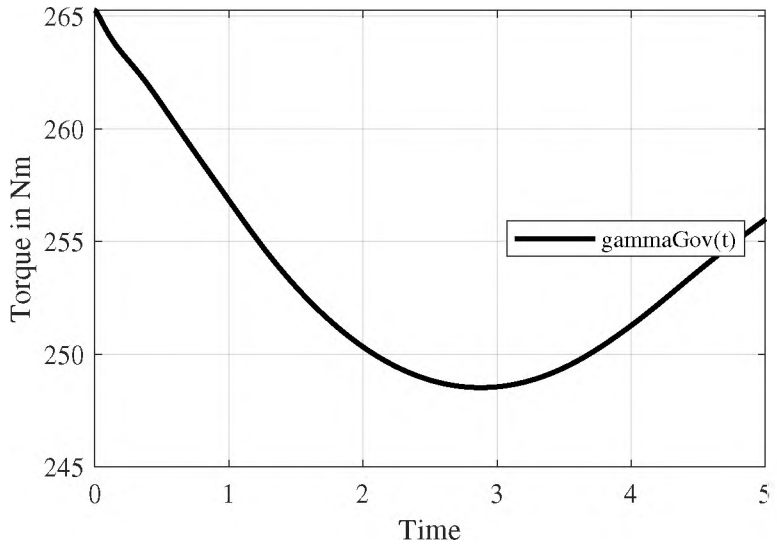

Figure 4.1. Bond graph integral term of PI controller for governed engine.

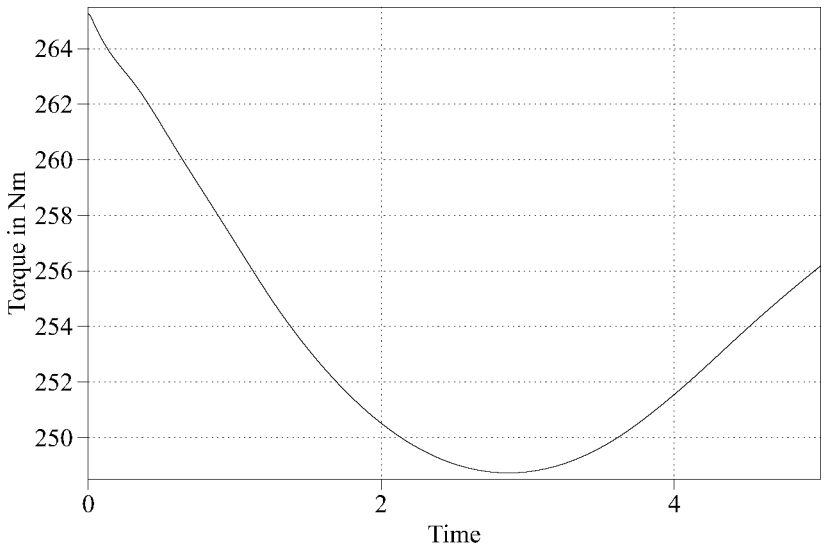

Figure 4.2. IPS integral term of PI controller for governed engine.

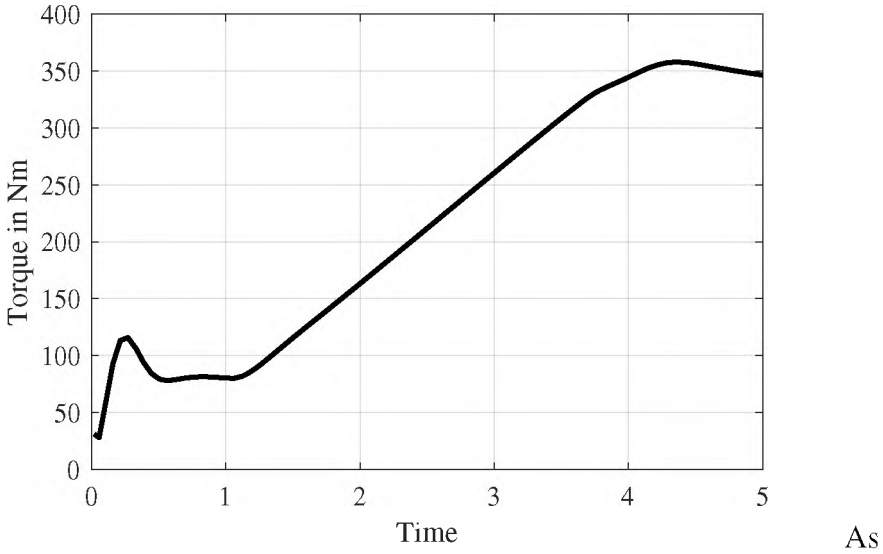

Figure 4.3. Bond graph output torque of the governed engine.

this result seems incorrect at first glance, that is not the case, and this is the outcome of a delay of information within the governed engine. When the speed is increasing at the end, this is the controller recognizing the issue and trying to regulate the speed.

4.2. FIELD CURRENT CONTROLLER RESULTS

The bond graph result of the integral term of the PI controller for the field current controller, or exciter, is illustrated in Figure 4.6. This graph matches the result from the IPS, and is exhibited in Figure 4.7. These graphs show the field current controller is trying to regulate the synchronous generator voltage. A very similar scene is depicted in Figures 4.8 and 4.9, where the controller corrected output voltage for both the bond graph and IPS is shown. A longer run time for the IPS simulation indicates that the voltage regulates at around 580 V which can be seen in Figure 4.11. Doing the same for the bond graph results ends up with voltage regulating around 700 V as shown in Figure 4.10. A running theme

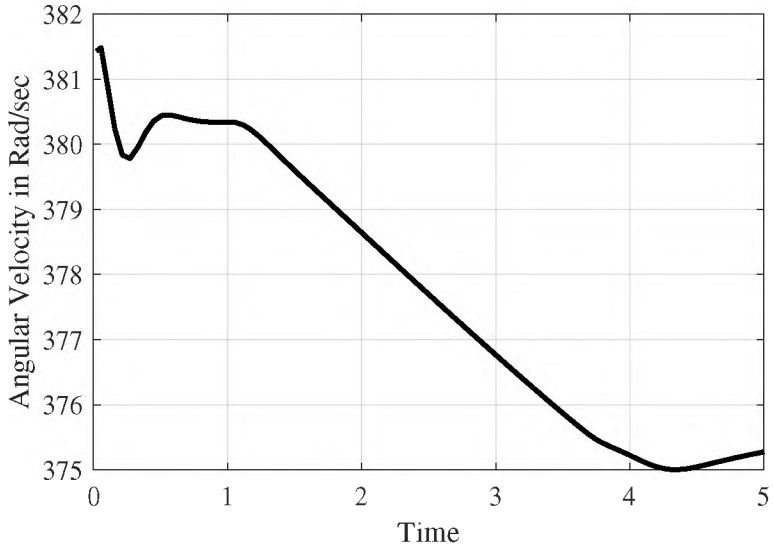

Figure 4.4. Bond graph output speed of the governed engine.

with the results from these two simulations is that the bond graph results consistently have higher values than the IPS results. This is most likely due to human error when applying the power conserving Clarke transform to both models. Although there is a significant quantitative difference, the qualitative behavior matches.

4.3. SYNCHRONOUS GENERATOR RESULTS

The magnitude of the synchronous generator RMS line-to-line $\alpha - \beta$ voltage for the bond graph model is presented in Figure 4.12 and for the IPS model in Figure 4.13. The nominal line-to-line voltage of the microgrid is 480 Volts, and these Figures show the bond graph and IPS model working correctly as they are hovering around that range.

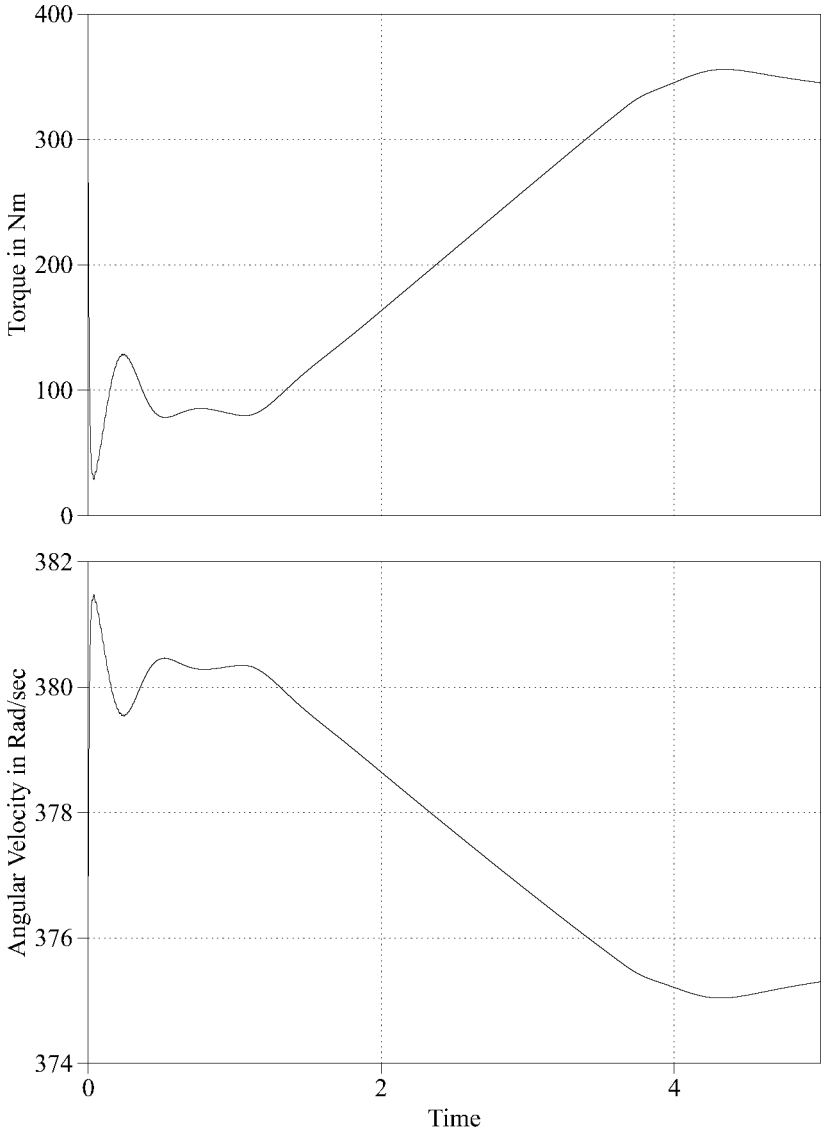

Figure 4.5. IPS output torque and speed of the governed engine.

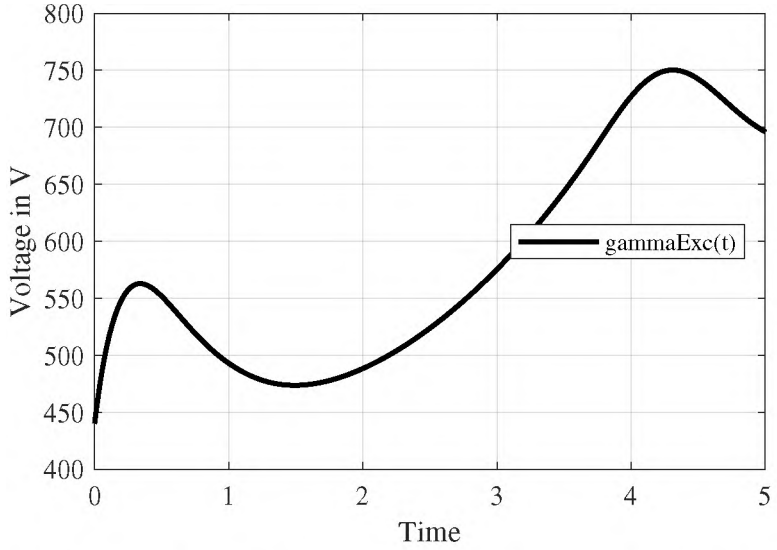

Figure 4.6. Bond graph integral term of PI controller for exciter.

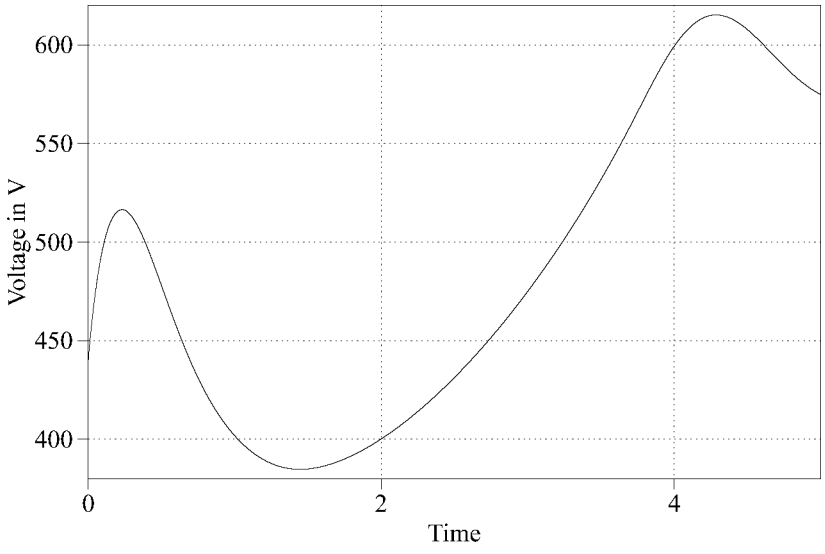

Figure 4.7. IPS integral term of PI controller for exciter.

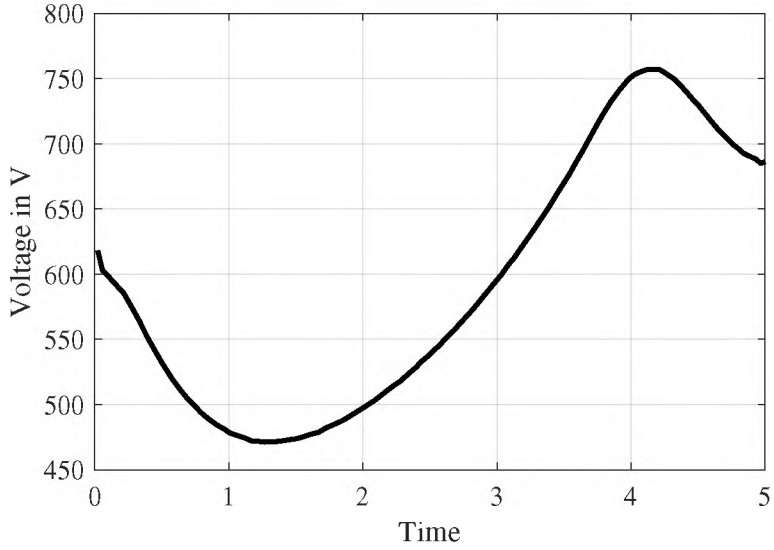

Figure 4.8. Bond graph output voltage from field current controller.

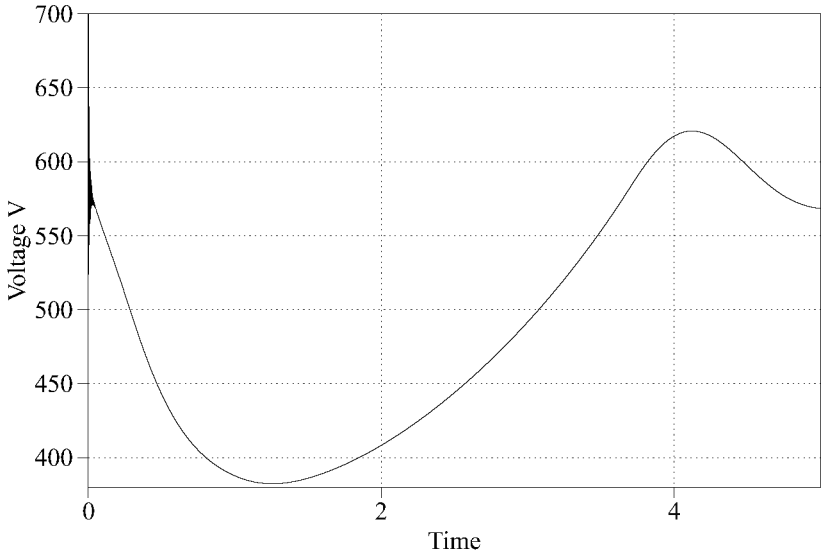

Figure 4.9. IPS output voltage from field current controller.

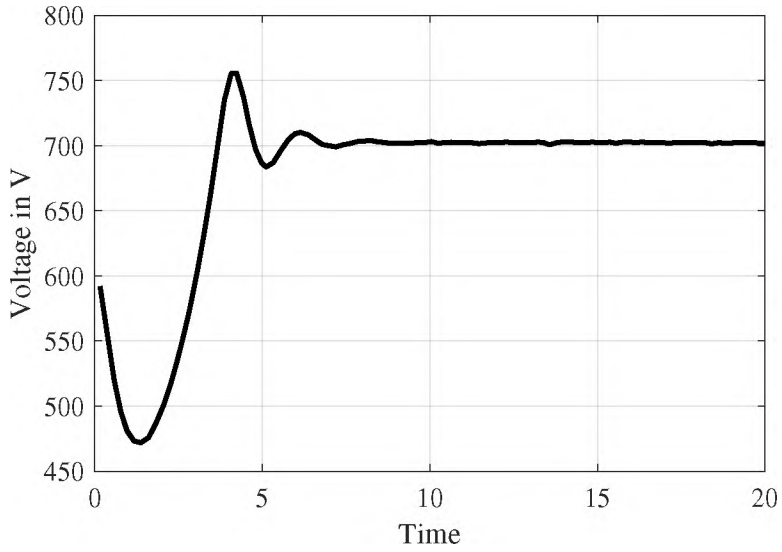

Figure 4.10. Bond graph output voltage from field current controller with extended time range.

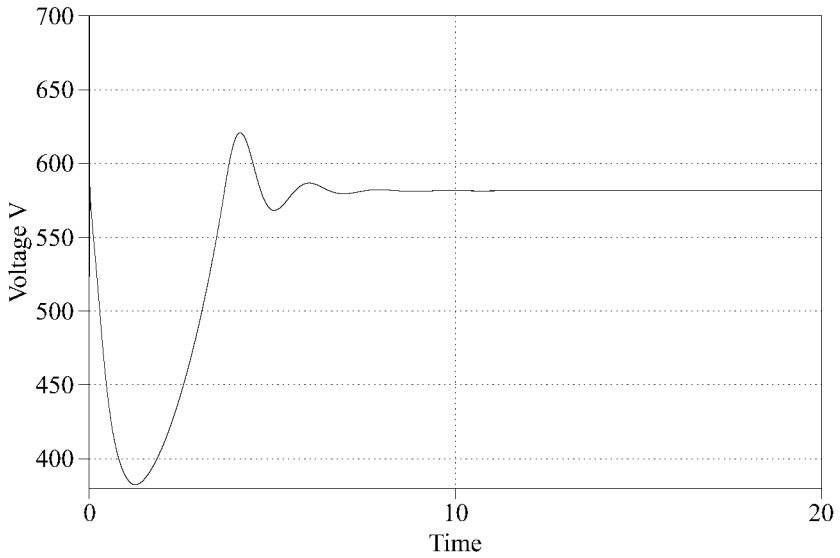

Figure 4.11. IPS output voltage from field current controller with extended time range.

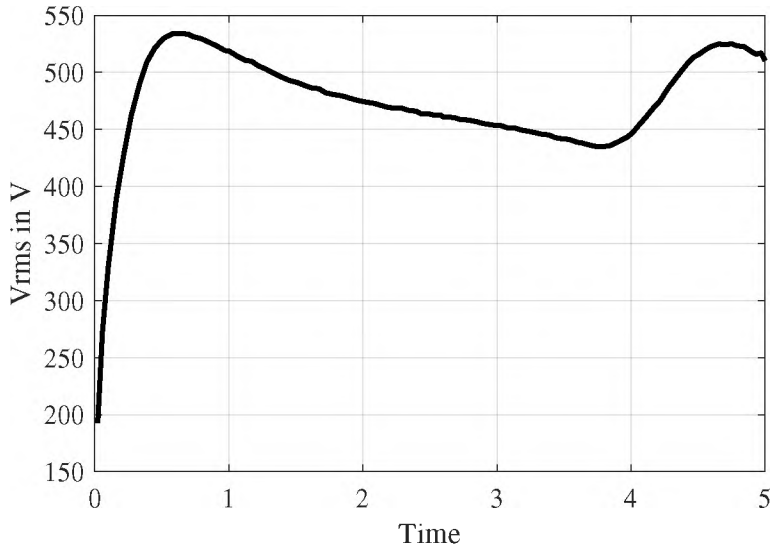

Figure 4.12. Bond graph magnitude of RMS line-to-line $\alpha - \beta$ voltage at the terminals of the synchronous generator.

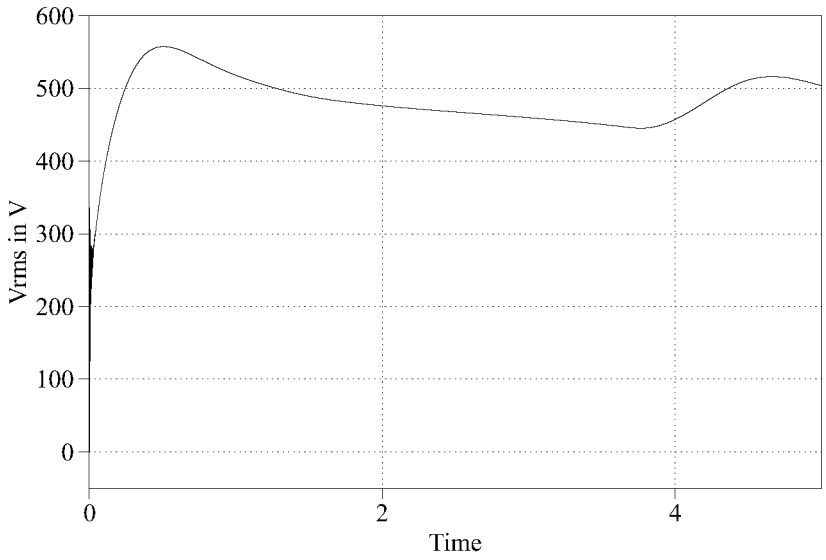

Figure 4.13. IPS magnitude of RMS line-to-line $\alpha - \beta$ voltage at the terminals of the synchronous generator.

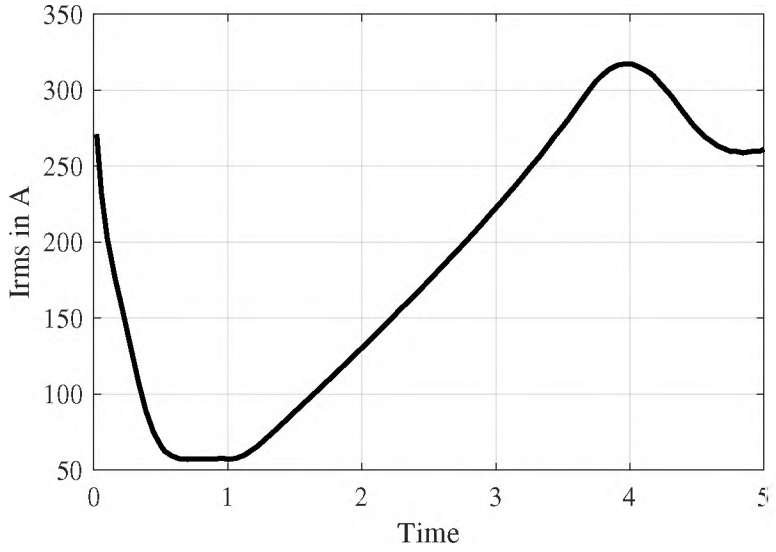

Figure 4.14. Bond graph synchronous generator $\alpha - \beta$ output RMS current.

The bond graph model RMS output current for the synchronous generator is depicted in Figure 4.14. The IPS model instantaneous $\alpha - \beta$ current for the synchronous generator is displayed in Figure 4.15. The envelope of the instantaneous current in Figure 4.15 is equal to the RMS current in Figure 4.14.

4.4. INDUCTION MOTOR RESULTS

Figures 4.16 and 4.17 demonstrate the output speed, or angular velocity, of the induction motor from the bond graph model. Figures 4.18 and 4.19 illustrate the output torque of the induction motor from the bond graph model. As the generator unloads onto the motor, the torque increases which causes the speed to increase as well. This is also indicated in the IPS Figures 4.20 and 4.21.

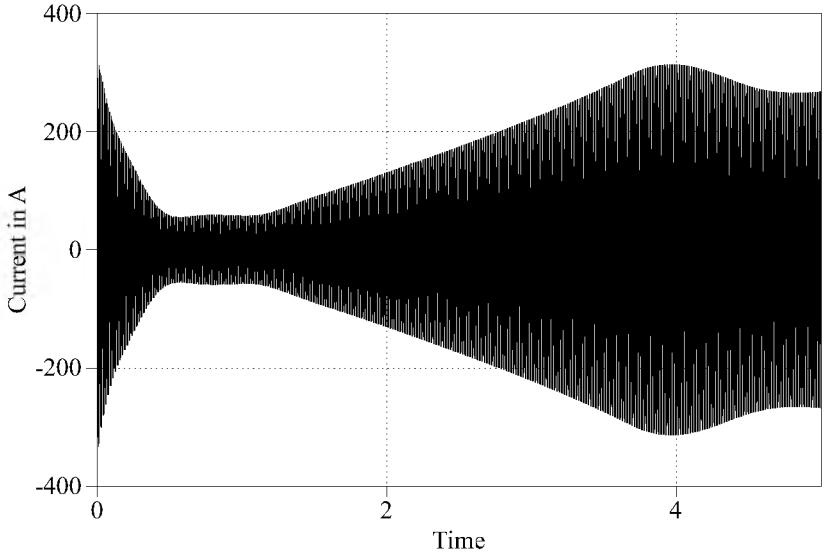

Figure 4.15. IPS synchronous generator $\alpha - \beta$ output current.

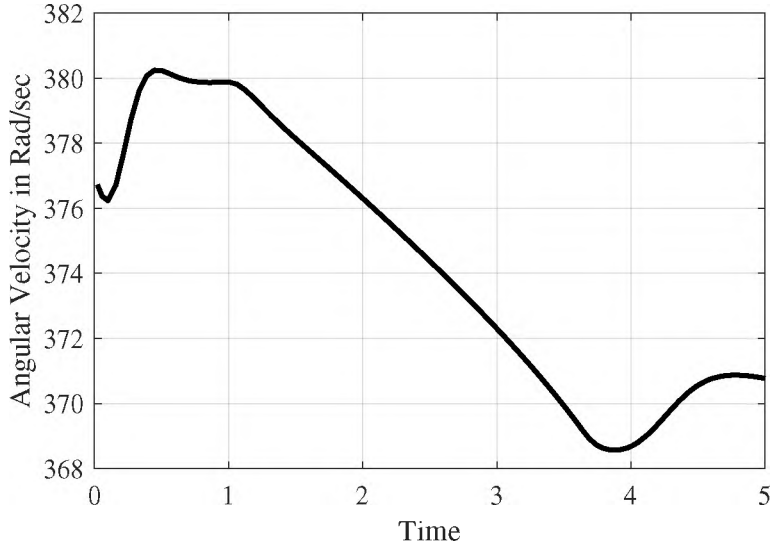

Figure 4.16. Bond graph induction motor output speed.

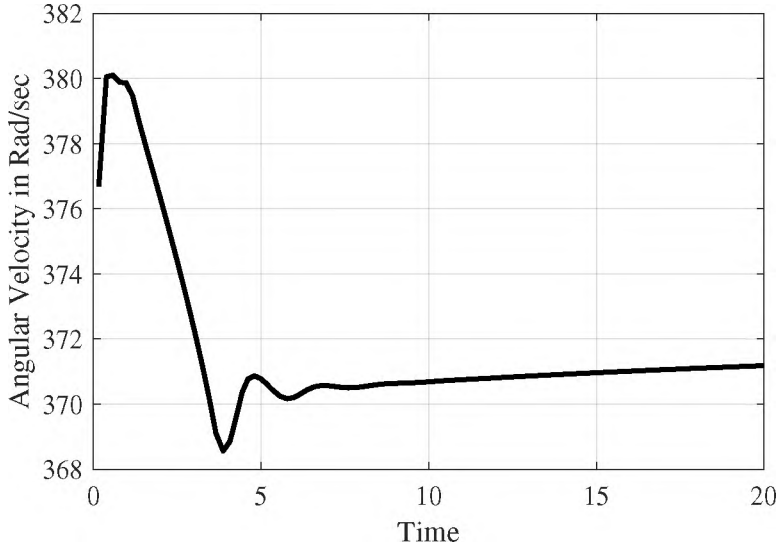

Figure 4.17. Bond graph induction motor output speed with extended time range.

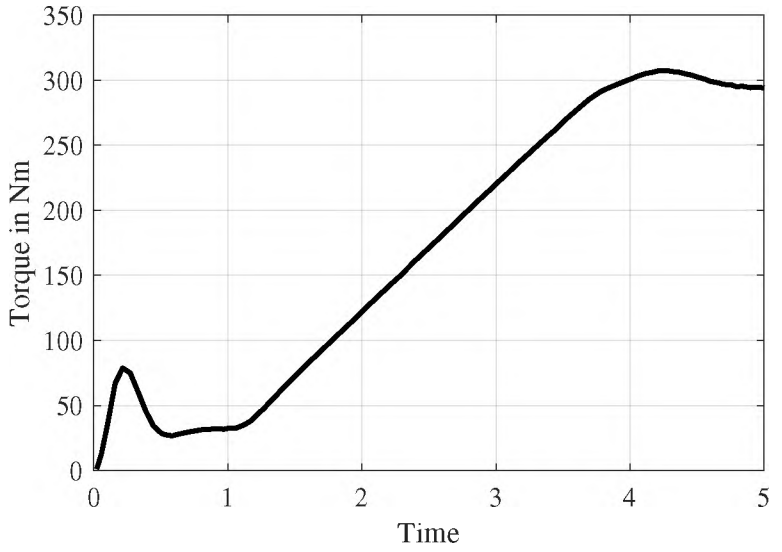

Figure 4.18. Bond graph induction motor torque.

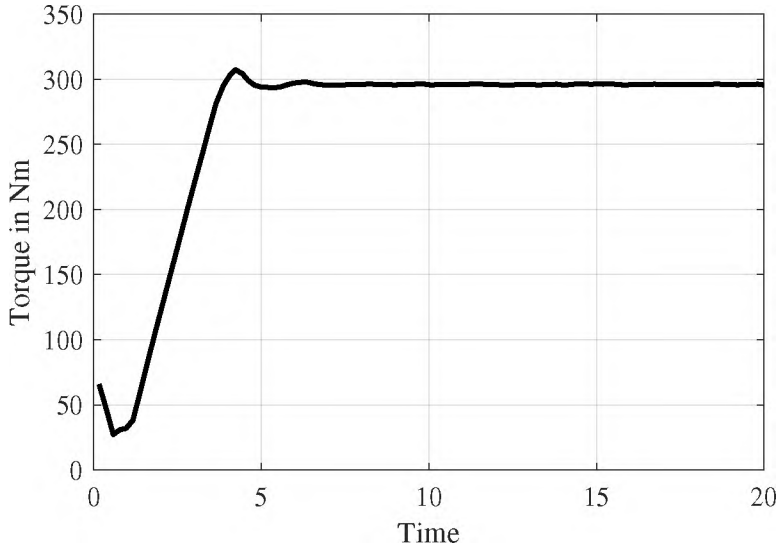

Figure 4.19. Bond graph induction motor torque with extended time range.

4.5. DISTRIBUTION GRID RESULTS

Figure 4.22 displays the charges on the capacitors located on the capacitor bank from the bond graph model, and Figure 4.23 shows this for the IPS model. Both of these models show the same end results.

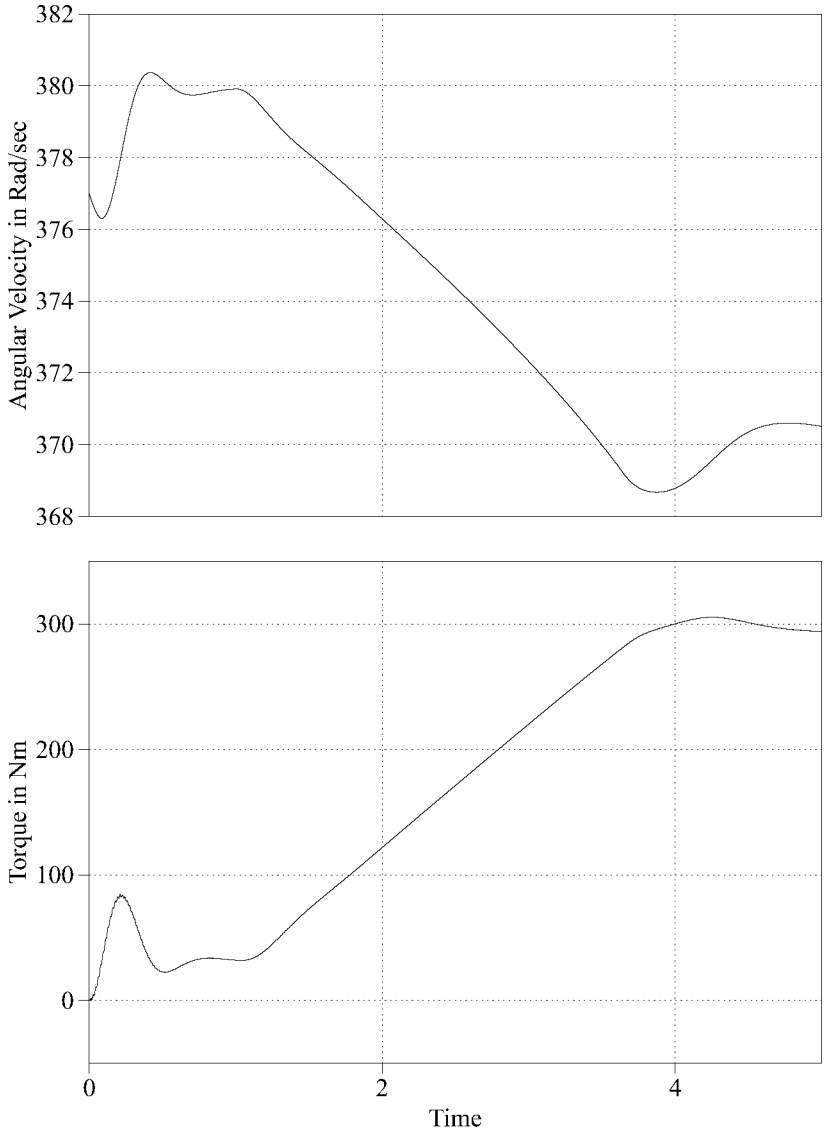

Figure 4.20. IPS induction motor torque and output speed.

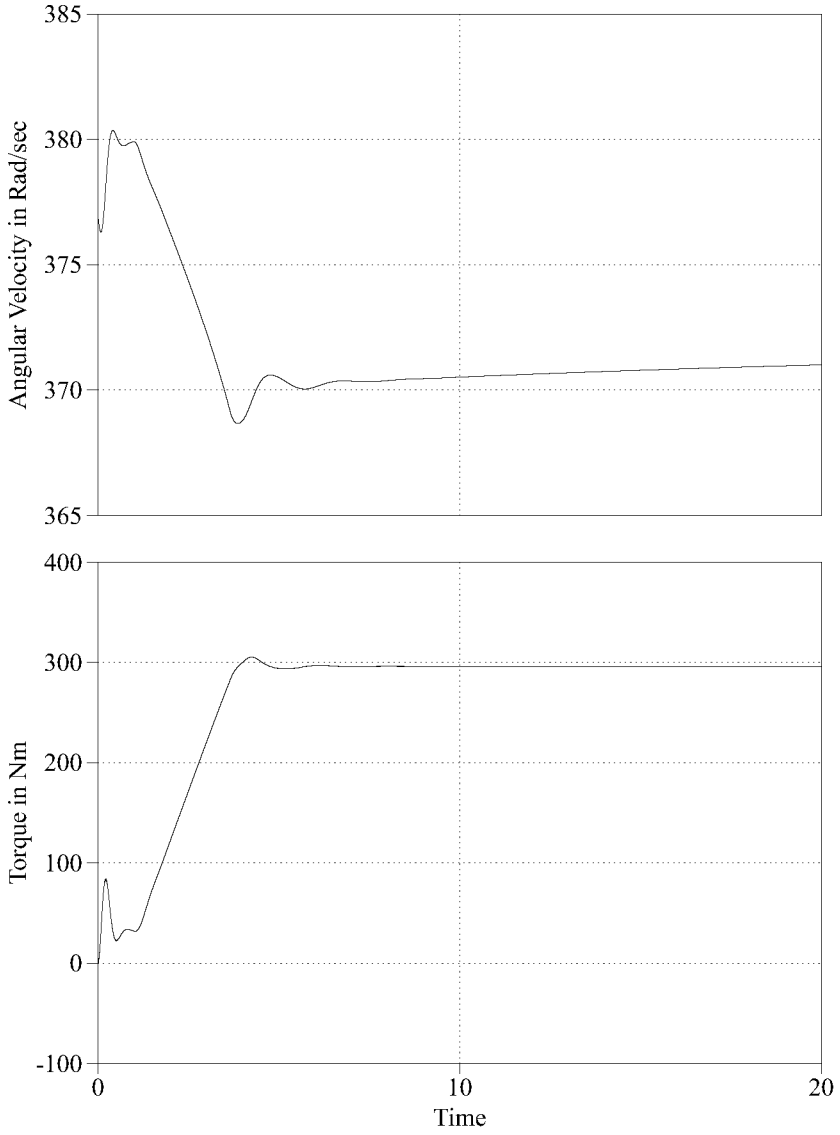

Figure 4.21. IPS induction motor torque and output speed with extended time range.

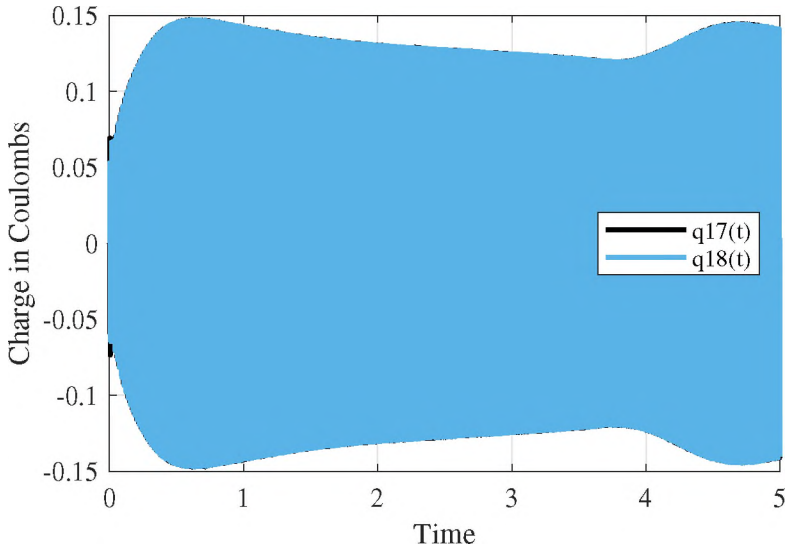

Figure 4.22. Bond graph charges on capacitor bank capacitors.

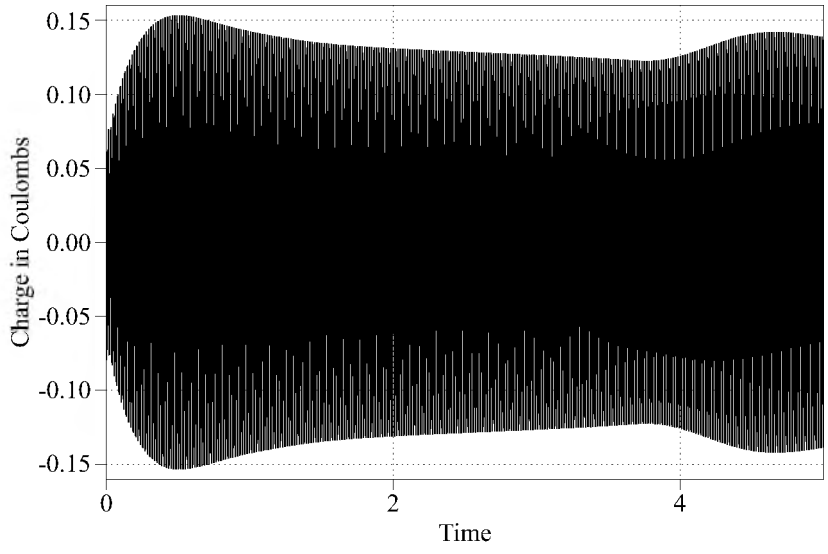

Figure 4.23. IPS charges on capacitor bank capacitors.

5. CONCLUSIONS

As technology advances and evolves, it becomes more available and regularly used. Specifically, advances in monitoring and control devices has resulted in prolific use of them in critical physical infrastructures. Combining monitoring and control devices with a physical system results in a cyber-physical system. These cyber-physical systems are prone to cyber attacks because any time a system is transferring communications between devices, that communication is vulnerable to be intercepted and changed. Therefore, a cyber-physical system protection software is needed to detect such events. However, the interdisciplinary nature of these systems makes creating such a software difficult, as each scientific domain involved uses different notations, system models, and equations. The solution to modeling cyber-physical systems proposed in this paper is the bond graph modeling method. This method takes advantage the physics commonalities found in all scientific domains, and encompasses the law of conservation of energy to model energy interactions throughout a system. Through this method, a multidomain system can be described using a singular universal notation, modeled using the bond graph building algorithm, and numerically analyzed through equations derived from the bond graph. To demonstrate the use of the bond graph method on a cyber-physical system, this paper presented a microgrid to be modeled. This microgrid consists of a synchronous generator receiving torque from a governed engine and voltage regulation from a field current controller, a distribution grid including a capacitor bank, line inductances, and a resistive load, and an induction motor feeding a mechanical load. Upon the first attempt to create the bond graph, a causality issue occurred, prompting the use of the IC port element. Once the IC-element was implemented, then equation derivation and numerical analysis was completed. To check the validity of the numerical results, a simulation of the microgrid's ideal physical system model was made. After comparing the bond graph model to the ideal physical system model, the conclusion

is that this is a viable option to accurately model a cyber-physical system. However, future work will need to be done to ensure the quantitative results match as well as the qualitative physics based characteristics matched in this paper.

APPENDIX

MICROGRID MATLAB CODE

1. MAIN MATLAB MICROGRID BOND GRAPH CODE

```
% close all
% clear all
% clc

%creating symbolic variables
% n is the turns ratio between armerature and field coils
% lf and la are leakage inductances
% L is mutual inductance
syms nGen LGen lfGen laGen

%Self Inductance of generator field
LfGen = (1/nGen)*LGen + lfGen;
%Self Inductance of generator armature
LaGen = nGen*LGen + laGen;

%% creating symbolic variables for the flux linkage on elec
   side of bond
%graph on generator IC port
syms lambda_8(t) lambda_7(t) lambda_6(t) lambda_5(t) theta_4
   (t)
```

```
%generator vector of state variable lambda
lambdaGen = [lambda_8(t) ; lambda_7(t); lambda_6(t);
   lambda_5(t)];

%Mutual inductace equations for generator
LcGen = LGen*cos(theta_4(t));
LsGen = LGen*sin(theta_4(t));

%Generator field and armature inductance matrix
Lambda_theta_Gen = [LfGen    0        LcGen    LsGen;
                    0        LfGen   -LsGen    LcGen;
                    LcGen   -LsGen    LaGen    0;
                    LsGen    LcGen    0        LaGen];

%Generator Gamma(theta), which is the inverse of Lambda(
   theta), the field
%and armature inductance matrix
Gamma_theta_Gen = inv(Lambda_theta_Gen);

%Generator side mechanical torque output from synchronous
   generator
%1-junction into IC:G element,\tau_G,in terms of theta and
   lambda
tau_4 = (LGen/(LfGen*LaGen - LGen^2))*((-lambda_8(t)*
   lambda_6(t) ...
```

```
- lambda_7(t)*lambda_5(t))*sin(theta_4(t))+ (lambda_8(t)*
   lambda_5(t)...
- lambda_7(t)*lambda_6(t))*cos(theta_4(t)));

syms e1 e2 e3 e4 f1 f2 f3 f4 R3 J1 p1(t) Se2 gammaGovydot
syms f5 f6 f7 f8


%% governed generator controller for MSe:Tau_ge
syms gammaGov(t)
%goverened engine controller
%f = frequency in Hz
%wref = angular frequency in rad/sec
%wmeasured = angular frequency measured on bond 4
%kiGov and kpGov are the the PI controller coefficients
%uGov = torque output of governed engine and input to
   generator
%gammaGov(t) =
f = 60;
wref = 2*pi*f;
wmeasured = f4;
werror = wref - wmeasured;
kiGov = 0.1253*20;
kpGov = 5.2706*10;
eqn101 = gammaGovydot == kiGov * werror;
uGov = kpGov * werror + gammaGov(t);
%NOT A STATE EQ
```

```
eqn100 = Se2 == uGov;

%Algebraic constraint equations from C side of bond graph
   for synchronous
%generator
eqn1 = f1 == f2;
eqn2 = f2 == f3;
eqn3 = f3 == f4;
eqn4 = e2 - e1 - e3 - e4 == 0;
eqn5 = tau_4 == -e4;
eqn6 = f1 == p1(t)/J1;
eqn7 = e2 == Se2;
eqn8 = e3 == f3*R3;

%Using a solver to find the state equations of the
   synchronous generator by
%removing all effort and flow terms
ansMechGen = solve(eqn1, eqn2, eqn3, eqn4, eqn5, eqn6, eqn7,
    eqn8, ...
    eqn100, eqn101, e1, e2, e3, e4, f1, f2, f3, f4, Se2,
       gammaGovydot);


gen_p1dot = ansMechGen.e1; %generator state equation for
   momentum rate of
%change with respect to time, \dot{p_{\theta_G}}
```

```
gen_theta4dot = ansMechGen.f4; %generator state equation for
    theta rate of
%change with repsect to time, \dot{\theta_G}
gammaGovdot = ansMechGen.gammaGovydot; %

%% Similarly for motor
syms lambda_33(t) lambda_36(t) lambda_37(t) lambda_38(t)
   theta_39(t)

%motor vector of state variable lambda
lambdaMot = [lambda_33(t) ; lambda_36(t); lambda_37(t);
   lambda_38(t)];

syms e5 e6 e7 e8 e9 e10 e11 e12 e13 e14 e15 e16 e17 e18 e19
   e20 e21 e22 ...
    e23 e24 e25 e26 e27 e28 e29 e30 e31 e32 e33 e34 e35 e36
       e37 e38

syms f5 f6 f7 f8 f9 f10 f11 f12 f13 f14 f15 f16 f17 f18 f19
   f20 f21 f22 ...
    f23 f24 f25 f26 f27 f28 f29 f30 f31 f32 f33 f34 f35 f36
       f37 f38

syms R9 R11 R13 R15 R25 R26 R29 R30 R32 R34

syms  Se12(t) Se31(t) Se35(t) C17 C18 L21 L22 q17(t) q18(t)
   lambda_21(t)...
```

```
    lambda_22(t)

syms Sf39(t)

syms nMot LMot lfMot laMot

%Self Inductance of motor field
LfMot = (1/nMot)*LMot + lfMot;
%Self Inductance of motor armature
LaMot = nMot*LMot + laMot;

%Mutual inductace equations for motor
LcMot = LMot*cos(theta_39(t));
LsMot = LMot*sin(theta_39(t));

%Motor field and armature inductance matrix
Lambda_theta_Mot = [LfMot      0       LcMot    LsMot;
                    0        LfMot    -LsMot    LcMot;
                    LcMot   -LsMot     LaMot      0;
                    LsMot    LcMot     0        LaMot];
%Motor Gamma(theta), which is the inverse of Lambda(theta),
   the field
%and armature inductance matrix
Gamma_theta_Mot = inv(Lambda_theta_Mot);

%performing eq i = Gamma(theta)*lambda on gen side
```

```
f_8_7_6_5 = -Gamma_theta_Gen*lambdaGen; % sign change to
    change from motor
%to gen convention

%performing eq i = Gamma(theta)*lambda on motor side
f_33_36_37_38 = Gamma_theta_Mot*lambdaMot;

%% Microgrid (lines, loads)
%%field current controlled exciter
syms gammaExc(t) gammaExcydot uExc Se10

% Controlling the field based on the stator voltage
%%kiExc and kpExc are the the PI controller coefficients
Vref = 491.5472;
Vmeasured = sqrt(e17^2 + e18^2);
Verror = Vref - Vmeasured;
kiExc = 0.1253*25; %same Ki as for Gov but w/o factor of 10
kpExc = 5.2706/10; %same Kp as for Gov but w/o factor of 10
eqn102 = gammaExcydot == kiExc * Verror;
uExc = kpExc * Verror + gammaExc(t);
%NOT A STATE EQ
eqn103 = Se10 == uExc;

%%Algebraic constraint equations from bond graph for entire
    distribution
%%section from the right side of IC:G to the left side of IC
    :M
```

```
eqn7 = f8 == f9;
eqn8 = f9 == f10;
eqn9 = f7 == f11;
eqn10 = f11 == f12;
eqn11 = f6 == f13;
eqn12 = f13 == f14;
eqn13 = f5 == f15;
eqn14 = f15 == f16;
eqn15 = f19 == f21;
eqn16 = f21 == f23;
eqn17 = f20 == f22;
eqn18 = f22 == f24;
eqn19 = f31 == f32;
eqn20 = f32 == f33;
eqn21 = f35 == f34;
eqn22 = f34 == f36;
eqn23 = f27 == f29;
eqn24 = f29 == f37;
eqn25 = f28 == f30;
eqn26 = f30 == f38;
eqn27 = e14 == e17;
eqn28 = e17 == e19;
eqn29 = e16 == e18;
eqn30 = e18 == e20;
eqn31 = e23 == e25;
eqn32 = e25 == e27;
eqn33 = e24 == e26;
```

```
eqn34 = e26 == e28;
eqn35 = e8 - e10 - e9 == 0;
eqn36 = e7 - e11 - e12 == 0;
eqn37 = e6 - e13 - e14 == 0;
eqn38 = e5 - e15 - e16 == 0;
eqn39 = e19 - e21 - e23 == 0;
eqn40 = e20 - e22 - e24 == 0;
eqn41 = e31 - e32 - e33 == 0;
eqn42 = e35 - e34 - e36 == 0;
eqn43 = e27 - e29 - e37 == 0;
eqn44 = e28 - e30 - e38 == 0;
eqn45 = f14 - f17 - f19 == 0;
eqn46 = f16 - f18 - f20 == 0;
eqn47 = f23 - f25 - f27 == 0;
eqn48 = f24 - f26 - f28 == 0;
eqn49 = e9 == f9*R9;
eqn50 = e11 == f11*R11;
eqn51 = e13 == f13*R13;
eqn52 = e15 == f15*R15;
eqn53 = e25 == f25*R25;
eqn54 = e26 == f26*R26;
eqn55 = e32 == f32*R32;
eqn56 = e34 == f34*R34;
eqn57 = e29 == f29*R29;
eqn58 = e30 == f30*R30;
eqn59 = e17 == q17(t)/C17;
eqn60 = e18 == q18(t)/C18;
```

```
eqn61 = f21 == lambda_21(t)/L21;
eqn62 = f22 == lambda_22(t)/L22;
eqn63 = e10 == Se10;
eqn64 = e12 == Se12(t);
eqn65 = e31 == Se31(t);
eqn66 = e35 == Se35(t);
eqn67 = f8 == f_8_7_6_5(1,1);
eqn68 = f7 == f_8_7_6_5(2,1);
eqn69 = f6 == f_8_7_6_5(3,1);
eqn70 = f5 == f_8_7_6_5(4,1);
eqn71 = f33 == f_33_36_37_38(1,1);
eqn72 = f36 == f_33_36_37_38(2,1);
eqn73 = f37 == f_33_36_37_38(3,1);
eqn74 = f38 == f_33_36_37_38(4,1);

%% Electrical state equations

ansElecGen = solve(eqn102, eqn103, eqn7, eqn8, eqn9, eqn10,
   eqn11, ...
    eqn12, eqn13, eqn14, eqn15, eqn16, eqn17, eqn18, eqn19,
       eqn20, ...
    eqn21, eqn22, eqn23, eqn24, eqn25, eqn26, eqn27, eqn28,
       eqn29, ...
    eqn30, eqn31, eqn32, eqn33, eqn34, eqn35, eqn36, eqn37,
       eqn38, ...
    eqn39, eqn40, eqn41, eqn42, eqn43, eqn44, eqn45, eqn46,
       eqn47, ...
```

```
    eqn48, eqn49, eqn50, eqn51, eqn52, eqn53, eqn54, eqn55,
       eqn56, ...
    eqn57, eqn58, eqn59, eqn60, eqn61, eqn62, eqn63, eqn64,
       eqn65, ...
    eqn66, eqn67, eqn68, eqn69, eqn70, eqn71, eqn72, eqn73,
       eqn74, e5, ...
    e6, e7, e8, e9, e10, e11, e12, e13, e14, e15, e16, e17,
       e18, e19, ...
    e20, e21, e22, e23, e24, e25, e26, e27, e28, e29, e30,
       e31, e32, ...
    e33, e34, e35, e36, e37, e38, f5, f6, f7, f8, f9, f10,
       f11, f12, ...
    f13, f14, f15, f16, f17, f18, f19, f20, f21, f22, f23,
       f24, f25, ...
    f26, f27, f28, f29, f30, f31, f32, f33, f34, f35, f36,
       f37, f38, ...
    gammaExcydot, Se10);
% Because of the square root, there are two solutions.
   Assume the first one
% is correct.

%state equations for the generator, motor, and distribution
   lines changes
%in flux linkage with respect to time
gen_lambda8dot = ansElecGen.e8(1);
gen_lambda7dot = ansElecGen.e7(1);
gen_lambda6dot = ansElecGen.e6(1);
```

```
gen_lambda5dot = ansElecGen.e5(1);
mot_lambda33dot = ansElecGen.e33(1);
mot_lambda36dot = ansElecGen.e36(1);
mot_lambda37dot = ansElecGen.e37(1);
mot_lambda38dot = ansElecGen.e38(1);
grid_lambda21dot = ansElecGen.e21(1);
grid_lambda22dot = ansElecGen.e22(1);
grid_q17dot = ansElecGen.f17(1);
grid_q18dot = ansElecGen.f18(1);

%Vf excitor
gammaExcydot = ansElecGen.gammaExcydot(1);

%% Motor mechanical state equations
%Eq 20 - motor side mechanical torque
pp = 1; % pole pairs
%Motor side mechanical torque output from induction motor
   element IC:G,
%\tau_G, into 1-junction in terms of theta and lambda
tau_39 = (LMot/(LfMot*LaMot - LMot^2))*((-lambda_33(t)*
   lambda_37(t) ...
     - lambda_36(t)*lambda_38(t))*sin(pp*theta_39(t))+ ...
     (lambda_33(t)*lambda_38(t) ...
     - lambda_36(t)*lambda_37(t))*cos(pp*theta_39(t)));

syms e39 e40 e41 e42 f39 f40 f41 f42 R42 J40 p40(t) Se41(t)
```

```
%Algebraic constraint equations from the C side of the bond
    graph for
%induction motor
eqn75 = e39 - e40 - e41 - e42 == 0;
eqn76 = f39 == f40;
eqn77 = f40 == f41;
eqn78 = f41 == f42;
eqn79 = e42 == f42*R42;
eqn80 = f40 == p40(t)/J40;
eqn81 = e39 == tau_39;
eqn82 = e41 == Se41(t);

%Using a solver to find the state equations of the induction
     motor by
%removing all effort and flow terms
ansMechMot = solve(eqn75, eqn76, eqn77, eqn78, eqn79, eqn80,
     eqn81, ...
     eqn82, e39, e40, e41, e42, f39, f40, f41, f42);

mot_p40dot = ansMechMot.e40; %Motor state equation for
    momentum rate of
%change with respect to time, \dot{p_{\theta_M}}
mot_theta39dot = ansMechMot.f39; %%Motor state equation for
    theta rate of
%change with repsect to time, \dot{\theta_M}

%% DAE solver
```

```
%Step 1

eqn83 = diff(lambda_8(t), 1) == gen_lambda8dot;
eqn84 = diff(lambda_7(t), 1) == gen_lambda7dot;
eqn85 = diff(lambda_6(t), 1) == gen_lambda6dot;
eqn86 = diff(lambda_5(t), 1) == gen_lambda5dot;
eqn87 = diff(lambda_33(t), 1) == mot_lambda33dot;
eqn88 = diff(lambda_36(t), 1) == mot_lambda36dot;
eqn89 = diff(lambda_37(t), 1) == mot_lambda37dot;
eqn90 = diff(lambda_38(t), 1) == mot_lambda38dot;
eqn91 = diff(theta_4(t), 1) == gen_theta4dot;
eqn92 = diff(theta_39(t), 1) == mot_theta39dot;
eqn93 = diff(p1(t), 1) == gen_p1dot;
eqn94 = diff(p40(t), 1) == mot_p40dot;
eqn95 = diff(q17(t), 1) == grid_q17dot;
eqn96 = diff(q18(t), 1) == grid_q18dot;
eqn97 = diff(lambda_21(t), 1) == grid_lambda21dot;
eqn98 = diff(lambda_22(t), 1) == grid_lambda22dot;
%controller eq
eqn99 = diff(gammaGov(t), 1) == gammaGovdot;
eqn201 = diff(gammaExc(t), 1) == gammaExcydot;

eqns = [eqn83 eqn84 eqn85 eqn86 eqn87 eqn88 eqn89 eqn90
   eqn91 eqn92 ...
    eqn93 eqn94 eqn95 eqn96 eqn97 eqn98, eqn99, eqn201];
```

```
state_vars = [lambda_8(t); lambda_7(t); lambda_6(t);
   lambda_5(t); ...
    lambda_33(t); lambda_36(t); lambda_37(t); lambda_38(t);
       theta_4(t); ...
    theta_39(t); p1(t); p40(t); q17(t); q18(t); lambda_21(t)
       ; ...
    lambda_22(t); gammaGov(t); gammaExc(t)];
orig_state_vars = length(state_vars);

%step 2
%step 2.1: checking for unused variables
check_incidence = incidenceMatrix(eqns, state_vars);
%found no unused variables

%skipped step 2.2 as all diff are order 1

%step 3: checking differential index of system
isLowIndexDAE(eqns, state_vars)
%returns logic 1 => index is 0 or 1, and step 3.2 can be
   skipped

%step 4: conver DAE systems to MATLAB Function Handles (oof)
%setting up DAE to be useable in the ode15i solver
parameters_eqns_DAE = symvar(eqns);
parameters_state_vars_DAE = symvar(state_vars);

%finding constants that need assigning
```

```
extra_parameters = setdiff(parameters_eqns_DAE,
    parameters_state_vars_DAE)

%evaluating DAEs
usable_parameters = daeFunction(eqns, state_vars, C17, C18,
    J1, J40, ...
     L21, L22, LGen, LMot, R3, R9, R11, R13, R15, R25, R26,
        R29, R30, ...
     R32, R34, R42, laGen, laMot, lfGen, lfMot, nGen, nMot,
        Se12(t), ...
     Se31(t), Se35(t), Se41(t), Sf39(t));

%% assigning constants (found from example 5.1 on page
    271-272 of electric
%machinery fitzgerald and kingsley book)
C17 = 2.749718334511574e-04;
C18 = 2.749718334511574e-04;
J1 = 0.42;
J40 = 2.5;
L21 = 122.223 * 10^-6;
L22 = 122.223 * 10^-6;

KK = 1;
nGen = 1;%match*
nMot = 1;
LGen = KK *  22.1e-3 / 5;
LMot = KK *  19.9 * 10^-3;
```

```
laGen = KK *  8e-5;
laMot = KK *  0.729 * 10^-3;
lfGen = KK *  1.5e-3 * 1.5;
lfMot = KK *  0.729 * 10^-3;

R3 = 0.007;
R9 = KK *1.5;
R11 = KK *0.06;
R13 = KK *6e-4;
R15 = KK *6e-4;
R25 = 15.359;
R26 = 15.359;
R29 = KK *30.3e-3;
R30 = KK *30.3e-3;
R32 = KK *23.1e-3;
R34 = KK *23.1e-3;
R42 = .08333;

%assigning time domain functions for effort sources
Se12 = @(t) 0;
Se31 = @(t) 0;
Se35 = @(t) 0;
Se41 = @(t) ramp(t, 1, 100, 265); %function ramp.m

F = @(t, variables, derivative_of_variables)
   usable_parameters(t, ...
```

```
    variables, derivative_of_variables, C17, C18, J1, J40,
       L21, L22, ...
    LGen, LMot, R3, R9, R11, R13, R15, R25, R26, R29, R30,
       R32, R34, ...
    R42, laGen, laMot, lfGen, lfMot, nGen, nMot, Se12(t),
       Se31(t), ...
    Se35(t), Se41(t), Sf39(t));

%% step 5: find initial conditions for solvers
Tnominal = 265.2582;
initial_value_guess = zeros(length(state_vars), 1);
initial_derivative_value_guess = zeros(length(state_vars),
   1);
initial_value_guess(9) = 0; %theta
initial_value_guess(10) = 0; %theta
initial_value_guess(11) = wref*J1;
initial_value_guess(12) = wref*J40;
initial_value_guess(17) = Tnominal;
initial_value_guess(18) = 440.5;
initial_derivative_value_guess(9) = wref;
initial_derivative_value_guess(10) = (wref);

%% Solve
opt = odeset('RelTol', 1e-3,'AbsTol',1e-7,'Stats','on');

[initial_value, initial_derivative_value] = decic(F, 0, ...
```

```
    initial_value_guess, [], initial_derivative_value_guess,
        [], opt);

[tSol,y] = ode15i(F,[0 5],initial_value,
    initial_derivative_value,opt);

%% creating plot structure
for k = 1:orig_state_vars
  SS{k} = char(state_vars(k));
end
%subplot 911 = synchronous generator flux linkages \lambda_5
    (t),
%\lambda_6(t), \lambda_7(t), \lambda_8(t)
figure(1)
subplot(911);
plot(tSol,y(:,1:4),'LineWidth',2)
legend(SS(1:4),'Location','East')
xlabel('Time')
ylabel('Flux Linkage in Wb turns')
grid on

%subplot 912 = induction motor flux linkages \lambda_{33}(t)
    ,
%\lambda_{36}(t), \lambda_{37}(t), \lambda_{38}(t)
subplot(912);
plot(tSol,y(:,5:8),'LineWidth',2)
legend(SS(5:8),'Location','East')
```

```
xlabel('Time')
ylabel('Flux Linkage in Wb turns')
grid on

%subplot 913 = synchronous generator angle of rotation \
    theta_4(t)
subplot(913);
plot(tSol,y(:,9),'LineWidth',2)
legend(SS(9),'Location','East')
xlabel('Time')
ylabel('Angular Displacement in Radians')
grid on

%subplot 914 = induction motor angle of rotation \theta_
    {39}(t)
subplot(914);
plot(tSol,y(:,10),'LineWidth',2)
legend(SS(10),'Location','East')
xlabel('Time')
ylabel('Angular Displacement in Radians')
grid on

%subplot 915 = synchronous generator momentum p_1(t)
subplot(915);
plot(tSol,y(:,11),'LineWidth',2)
legend(SS(11),'Location','East')
xlabel('Time')
```

```
ylabel('Momentum in (kg*m)/sec')
grid on

%subplot 916 = induction motor momentum p_40(t)
subplot(916);
plot(tSol,y(:,12),'LineWidth',2)
legend(SS(12),'Location','East')
xlabel('Time')
ylabel('Momentum in (kg*m)/sec')
grid on

%subplot 917 = charge on capcitor bank capacitors q_17(t)
   and q_18(t)
subplot(917);
plot(tSol,y(:,13:14),'LineWidth',2)
legend(SS(13:14),'Location','East')
xlabel('Time')
ylabel('Charge in Coulombs')
grid on

%subplot 918 = flux linkage on line inductances \lambda_
   {21}(t) and
%\lambda_{22}(t)
subplot(918);
plot(tSol,y(:,15:16),'LineWidth',2)
legend(SS(15:16),'Location','East')
xlabel('Time')
```

```
ylabel('Flux Linkage in Wb turns')
grid on

%subplot 919 = integral term of PI controller, \gammaGov(t)
subplot(919);
plot(tSol,y(:,17),'LineWidth',2)
legend(SS(17),'Location','East')
xlabel('Time')
ylabel('Torque in Nm')
grid on

%% Plot on separate figures
figure(911);
plot(tSol,y(:,1:4),'LineWidth',2)
legend(SS(1:4),'Location','East')
xlabel('Time')
ylabel('Flux Linkage in Wb turns')
grid on

figure(912);
plot(tSol,y(:,5:8),'LineWidth',2)
legend(SS(5:8),'Location','East')
xlabel('Time')
ylabel('Flux Linkage in Wb turns')
grid on

figure(913);
```

```
plot(tSol,y(:,9),'LineWidth',2)
legend(SS(9),'Location','East')
xlabel('Time')
ylabel('Angular Displacement in Radians')
grid on

figure(914);
plot(tSol,y(:,10),'LineWidth',2)
legend(SS(10),'Location','East')
xlabel('Time')
ylabel('Angular Displacement in Radians')
grid on

figure(915);
plot(tSol,y(:,11),'LineWidth',2)
legend(SS(11),'Location','East')
xlabel('Time')
ylabel('Momentum in (kg*m)/sec')
grid on

figure(916);
plot(tSol,y(:,12),'LineWidth',2)
legend(SS(12),'Location','East')
xlabel('Time')
ylabel('Momentum in (kg*m)/sec')
grid on
```

```
figure(917);
plot(tSol,y(:,13:14),'LineWidth',2)
legend(SS(13:14),'Location','East')
xlabel('Time')
ylabel('Charge in Coulombs')
grid on

figure(918);
plot(tSol,y(:,15:16),'LineWidth',2)
legend(SS(15:16),'Location','East')
xlabel('Time')
ylabel('Flux Linkage in Wb turns')
grid on

figure(919);
plot(tSol,y(:,17),'LineWidth',2)
legend(SS(17),'Location','East')
xlabel('Time')
ylabel('Torque in Nm')
grid on

figure(920);
plot(tSol,y(:,18),'LineWidth',2)
legend(SS(18),'Location','East')
xlabel('Time')
ylabel('Voltage in V')
grid on
```

```
%%

PlotMicrogrid_20210324;
```

2. SUPPLEMENTAL MATLAB MICROGRID BOND GRAPH CODE

```
% First do whatever substitutions are possible from
    analytical solutions
uGov1 = subs(subs(uGov, ansMechGen));
e17x = subs(e17, ansElecGen);
e17x1 = subs(e17x(1));
e18x = subs(e18, ansElecGen);
e18x1 = subs(e18x(1));
e39x1 = subs(subs(e39, ansMechMot));
f41x1 = subs(subs(f41, ansMechMot));
f4x1 = subs(subs(f4, ansMechGen));
e10x1 = subs(subs(e10, ansElecGen));
f14x1 = subs(subs(f14, ansElecGen));
f16x1 = subs(subs(f16, ansElecGen));

% Down-sample solution to make post-processing feasible
tsize = length(tSol);
pts = 100;
% Initialize variables
tx = zeros(pts,1);
uGov2 = zeros(pts,1);
e17x2 = zeros(pts,1);
e18x2 = zeros(pts,1);
```

```
e39x2 = zeros(pts,1);
f41x2 = zeros(pts,1);
f4x2 = zeros(pts,1);
e10x2 = zeros(pts,1);
f14x2 = zeros(pts,1);
f16x2 = zeros(pts,1);

for i = 1:pts
    if mod(i,10) == 0
        i
    end
    tx(i) = tSol(ceil(i*tsize/pts),1);
    yvec = y(ceil(i*tsize/pts),:);
    uGov2(i) = subs((uGov1),{gammaGov(t),p1(t)},{yvec(17),
       yvec(11)});
    e17x2(i) = subs((e17x1),q17,yvec(13));
    e18x2(i) = subs((e18x1),q18,yvec(14));
    e39x2(i) = subs((e39x1),{theta_39(t), lambda_36(t),
       lambda_37(t), lambda_33(t), lambda_38(t)},{yvec(10),
       yvec(6),yvec(7),yvec(5),yvec(8)});
    f41x2(i) = subs((f41x1), p40(t), yvec(12));
    f4x2(i) = subs((f4x1), p1(t), yvec(11));
    e10x2(i) = subs((e10x1),{q17,q18,gammaExc(t)},{yvec(13),
       yvec(14),yvec(18)});
    f14x2(i) = subs((f14x1),{theta_4(t),lambda_5(t),lambda_6
       (t),lambda_7(t),lambda_8(t)},{yvec(9),yvec(4),yvec(3)
       ,yvec(2),yvec(1)});
```

```
    f16x2(i) = subs((f16x1),{theta_4(t),lambda_5(t),lambda_6
        (t),lambda_7(t),lambda_8(t)},{yvec(9),yvec(4),yvec(3)
        ,yvec(2),yvec(1)});

end

Vrms = sqrt(e17x2.^2 + e18x2.^2); %rms line to line voltage
    at that bus, which is generator terminals which is fig
    802
Irms = sqrt(f14x2.^2 + f16x2.^2);

%figure 801 = output torque of the governed engine, uGov
figure(801)
plot(tx,uGov2)
xlabel('Time')
ylabel('Torque in Nm')

%figure 802 = magnitude of the RMS line-to-line \alpha-\beta
    voltage at the
% terminals of the synchronous generator
figure(802)
% plot(tx,e17x2,tx,e18x2,tx,Vrms)
plot(tx,Vrms)
xlabel('Time')
% ylabel('e17, e18, Vrms')
ylabel('Vrms in V')
```

```
%figure 803 = induction motor torque output from IC-element
    into 1-junction
%\tau_{M}, also known as e_{39}
figure(803)
plot(tx,e39x2)
xlabel('Time')
ylabel('Torque in Nm')

%figure 804 = induction motor output speed, f_{41}, to
    mechanical load
figure(804)
plot(tx,f41x2)
xlabel('Time')
ylabel('Angular Velocity in Rad/sec')

%figure 805 = speed from governed engine, f_4 = \dot{\
    theta_G}
figure(805)
plot(tx,f4x2)
xlabel('Time')
ylabel('Angular Velocity in Rad/sec')

%figure 806 = synchronous generator controlled output
    voltage to the field
%from the exciter
figure(806)
plot(tx,e10x2)
```

```
xlabel('Time')
ylabel('Voltage in V')

%figure 807 = synchronous generator RMS \alpha-\beta current
    on windings
%f_14 and f_16
figure(807)
plot(tx,f14x2,tx,f16x2);
xlabel('Time')
ylabel('Irms in A, Irms in A')

%figure 808 = synchronous generator \alpha-\beta RMS current
    magnitude on
%windings f_14 and f_16
figure(808)
plot(tx,Irms)
xlabel('Time');
ylabel('Irms in A')

function myramp = ramp(t, startTime, slope, limit)

if t < startTime
    myramp = 0;
elseif t < startTime+limit/slope
    myramp = slope*(t-startTime);
else
    myramp = limit;
end
```

3. PLECS AND SIMULINK INITIALZATION FILE

```
close all
clear all
clc

VLL = 480; % nominal system rating
wref = 2*pi*60;
K = (sqrt(6))/2; % scale factor to make Clarke
    transformation power-invariant
% Note, these gains end up being scaled in Simulink.
Ki = 0.1253; % integral gain for governor
Kp = 5.2706; % proportional gain for governor
Pnominal = 100e3;
Tnominal = (Pnominal)/wref; % synchronous generator torque
    for nominal power


%% Induction Machine Parameters
RsIM = 30.3e-3; %R29 and R30
LlsIM = 0.729e-3; % laMot
RrIM = 23.1e-3;%R32 and R34
LlrIM = LlsIM; % lfMot
LmIM = 19.9e-3;% LMot
JIM = 2.5;%J40
FIM = 0.08333;%R42
```

```
% Calculate its torque-speed curve at nominal voltage
poles = 2;
V1 = VLL/sqrt(3); %calculating VLN RMS
R1 = RsIM;
R2 = RrIM;
X1 = LlsIM * wref;
X2 = LlrIM * wref;
Xm = LmIM * wref;
ns = 120*60/poles;
ws = wref*2/poles;

s = 1e-4:1e-4:1;
V1eq = abs(V1*(1i*Xm)/(R1+1i*(X1+Xm)));
Z1eq = 1i*Xm*(R1+1i*X1)/(R1 + 1i*(X1+Xm));
R1eq = real(Z1eq);
X1eq = imag(Z1eq);
Tmech = (3*V1eq^2/ws)*(R2 ./ s) ./ ((R1eq + (R2 ./ s)).^2 +
    (X1eq + X2)^2);
n = (1-s)*ns;
wm = n*pi/30;
Pmech = wm.*Tmech;
T0 = Pnominal/ws;

% figure(1)
% plot(n,Tmech,'-',n,T0*ones(size(n)),':')
```

```
% figure(2)
% plot(n,Pmech,'-',n,Pnominal*ones(size(n)),':');

Zmot = R1 + 1i*X1 + 1i*Xm*(1i*X2 + R2./s) ./ (R2./s + 1i*X2
    + 1i*Xm);
Ia = (V1./Zmot);
% figure(3)
% plot(n,real(Ia))

% Find operating point for Pmech = Pnominal
ind = (find(Pmech>=Pnominal,1)); % index of point close to
    correct power
s0 = s(ind);
I1 = Ia(ind);
% [Iam,Iaf] = r2p(I1) % returns magnitude and phase in
    degrees

%% Now solve the circuit, to determine the var support
    needed
% V1 is the voltage L-N at the induction motor terminals
% V2 is to the left of one set of inductors, where resistors
     connect
% V3 is to the left of another set of inductors, where
    capacitors connect
% V3 is also the terminal voltage of the synchronous machine
% I1, I2, I3 are currents flowing from the left into nodes
    V1, V2, V3
```

```
% We already know V1, I1 from above; work right-to-left
    through circuit
L1 = 122.223 * 10^-6; % inductance of the lines, L21 and L22
loadbank = 15e3; %total power lost by R25 and R26
R1 = VLL^2 / loadbank; % Rload Resistance?

XL1 = wref*L1;
V2 = V1 + I1 * 1i*XL1;
I2 = I1 + V2/R1;

V3 = V2 + I2 * 1i*XL1;

S3 = V3 * conj(I2); % complex power consumed by elements
    considered up to this point; single-phase
P3 = real(S3);
pf = P3/abs(S3);
pf_target = 0.95; % desired power factor of circuit that
    loads the generator
theta_target = acos(pf_target);
S_target = P3/pf_target;
Q_target = S_target * sin(theta_target);
Q_actual = imag(S3);
Q_cap = Q_actual - Q_target;
C1 = Q_cap / (wref * abs(V3)^2);%C17 and C18
XC1 = -1/(wref*C1);
I3 = I2 + V3*1i/XC1;
```

```
%% Synchronous Machine Parameters
%a = 1 % Ns/Nf
a=1;
XXX = 5; % scale everything to make the SM bigger
Rs = 3e-3 / XXX;%R13 abd R15
Lls = 0.4e-3 / XXX; %laGen
Lm0 = 22.1e-3 * a / XXX;%LGen
Lmsat = Lm0; %saturated mutual inductance
PsiT = 1e6;
fT = 1;
Rf = 1 * a^2 * 1.5; % R9 and R11
Llf = 1.5e-3 * a^2 * 1.5; %lfGen
Rkdqq = [1e6 Rf 1e6];
Llkdqq = [1e6 Llf 1e6]; %damper leakage inductance
J = 0.42; %J1
F = 0.007; %R3

Xls = wref*(Lls+Lm0);
EaZ = V3 + 1i*Xls * I3;
Ea = abs(EaZ);
If = sqrt(2) * Ea / (wref*Lm0);
Vf = If * Rf/(1.5);
Vref = abs(V3) * sqrt(3); % reference voltage at generator
```

REFERENCES

[1] F. A. Firestone, "A NEW ANALOGY BETWEEN MECHANICAL AND ELECTRICAL SYSTEMS," p. 20.

[2] D. Preis, "Block Diagrams: A Tutorial Alternative to Dynamical Analogies," *IEEE Transactions on Education*, vol. 19, pp. 143–148, Nov. 1976. Conference Name: IEEE Transactions on Education.

[3] C. Suggs and C. Abrams, Jr., "Mechanical impedance techniques for evaluating the dynamic characteristics of biological materials," *Journal of Agricultural Engineering Research*, vol. 16, no. 3, pp. 307–315, 1971.

[4] M. Hanief and M. F. Wani, "Wear Modeling Revisited Using Electrical Analogy," *Journal of Tribology*, vol. 139, July 2017.

[5] F. Katiraei, M. R. Iravani, and P. W. Lehn, "Small-signal dynamic model of a microgrid including conventional and electronically interfaced distributed resources," *Transmission Distribution IET Generation*, vol. 1, pp. 369–378, May 2007. Conference Name: Transmission Distribution IET Generation.

[6] M. S. Mahmoud, S. Azher Hussain, and M. A. Abido, "Modeling and control of microgrid: An overview," *Journal of the Franklin Institute*, vol. 351, pp. 2822–2859, May 2014.

[7] D. C. Karnopp, D. L. Margolis, and R. C. Rosenberg, *System Dynamics: Modeling, Simulation, and Control of Mechatronic Systems*. Hoboken, UNITED STATES: John Wiley & Sons, Incorporated, 2012.

[8] J. F. Broenink, "Introduction to Physical Systems Modelling with Bond Graphs," p. 31, 1999.

[9] M. Maia Neto and L. C. S. Góes, "Use of LMS Amesim®model and a bond graph support to predict behavior impacts of typical failures in anaircraft hydraulic brake system," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 40, p. 414, Aug. 2018.

[10] R. Sanchez and A. Medina, "Wind turbine model simulation: A bond graph approach," *Simulation Modelling Practice and Theory*, vol. 41, pp. 28–45, Feb. 2014.

[11] S.-W. Oh, H.-C. Chae, S.-C. Yun, and C.-S. Han, "The Design of a Controller for the Steer-by-Wire System," *JSME International Journal Series C*, vol. 47, no. 3, pp. 896–907, 2004.

[12] T. Ishigooka, H. Saissi, T. Piper, S. Winter, and N. Suri, "Practical Formal Verification for Model Based Development of Cyber-Physical Systems," in *2016 IEEE Int. Conf. on Computational Science and Engineering (CSE) and IEEE Int. Conf. on Embedded and Ubiquitous Computing (EUC)*, (Paris), pp. 1–8, IEEE, Aug. 2016.

[13] "MicroGrids," vol. 1, pp. 305–308 vol.1, Jan. 2002.

[14] B. Kroposki, R. Lasseter, T. Ise, S. Morozumi, S. Papathanassiou, and N. Hatziargyriou, "Making microgrids work," *IEEE Power and Energy Magazine*, vol. 6, pp. 40–53, May 2008. Conference Name: IEEE Power and Energy Magazine.

[15] H. Nikkhajoei and R. H. Lasseter, "Distributed Generation Interface to the CERTS Microgrid," *IEEE Transactions on Power Delivery*, vol. 24, pp. 1598–1608, July 2009. Conference Name: IEEE Transactions on Power Delivery.

[16] G. Raiker, L. Umanand, and B. Subba Reddy, "Bond Graph Approach for Modelling Solar PV-Thermal and Thermoelectric Devices," 2020.

[17] R. Meshram, S. Khade, S. Wagh, N. Singh, and A. Stanković, "Bond graph approach for port-controlled Hamiltonian modeling for SST," *Electric Power Systems Research*, vol. 158, pp. 105–114, 2018.

[18] I. Zafeiratou, I. Prodan, L. Lefèvre, and L. Piétrac, "Dynamical modelling of a DC microgrid using a port-Hamiltonian formalism," *IFAC-PapersOnLine*, vol. 51, pp. 469–474, Jan. 2018.

[19] I. Zafeiratou, D. Nguyen, I. Prodan, L. Lefèvre, and L. Piétrac, "Flatness-based hierarchical control of a meshed DC microgrid," *IFAC-PapersOnLine*, vol. 51, pp. 222–227, 2018.

[20] "SCADA Attack Modeling Using Bond Graph," pp. 1–2, Dec. 2019. ISSN: 2643-6868.

[21] I. Zerdazi, M. Fezari, and M. Ouziala, "Detection of Deception Attacks in Supervisory Control Systems Using Bond Graph," *Automatic Control and Computer Sciences*, vol. 54, no. 2, pp. 156–167, 2020.

[22] S. D. Umans, *Fitzgerald & Kingsley's Electric Machinery*. New York, New York USA: McGraw-Hill, seventh edition ed., 2014.

[23] D. Karnopp, "State functions and bond graph dynamic models for rotary, multiwinding electrical machines," *Journal of the Franklin Institute*, vol. 328, pp. 45–54, Jan. 1991.

[24] H. Akagi, E. H. Watanabe, and M. Aredes, *Instantaneous power theory and applications to power conditioning*. IEEE Press series on power engineering, Hoboken, N.J. : Piscataway, N.J: Wiley ; IEEE Press, 2007. OCLC: 76798166.

VITA

Michele Jane White received her bachelors degree in Electrical Engineering with an emphasis in power and energy from Missouri University of Science and Technology (formerly University of Missouri Rolla) in August of 2019. Then she began work on her masters degree in Electrical Engineering from the same university in the Fall of 2019, and she completed her degree in July of 2021.

During this time, Michele has worked at the university as a math and circuits tutor from 2017 to 2019 and as a Graduate Research Assistant from 2019 to 2020. Additionally, she worked as an Electrical Engineering intern over the Summer in 2019 and 2020 for the engineering consulting firm Burns McDonnell in the Transmission and Distribution department. Specifically, Michele worked on the protection and control schemes for substations. In 2021 she became a full time employee for Burns McDonnell in the Denver, Colorado office continuing work with substations.