Research and Development of a Linear Graph-based MATLAB Toolbox

Eric McCormick, Haoxiang Lang
GRASP Laboratory
Faculty of Engineering and Applied Science
University of Ontario Institute of Technology
Oshawa, ON, Canada
eric.mccormick@uoit.net, haoxiang.lang@uoit.ca

Clarence W. de Silva
Industrial Automation Laboratory
Department of Mechanical Engineering
The University of British Columbia
Vancouver, BC, Canada
desilva@mech.ubc.ca

***Abstract*—This paper presents a new MATLAB-based software toolbox, called LGtheory, which provides a robust and automated method of evaluating Linear Graph (LG) models of multi-domain engineering systems for the primary energy domains (electrical, mechanical, hydraulic, and thermal). The necessary background on the major concepts of LG approach is presented along with a description of how the different LG processes are automated using MATLAB software. These processes are demonstrated through the extraction of the state-space model of an example system consisting of a DC motor with an inertial load. The results of this system output by the LGtheory Toolbox are simulated and compared against the same model as constructed in Simulink Simscape. This comparison demonstrates that LGtheory is capable of producing accurate state-space models of multi-domain dynamic systems.**

Keywords—linear graph approach, state-space modeling, multi-domain systems, multi-physics models, MATLAB toolbox, dynamic system modeling, mechatronic systems.

I. Introduction

Multi-energy domain engineering systems play an increasingly critical role in our daily lives. Systems that at one point may have been purely mechanical or electrical in nature may now span one or more additional energy domains. This can be seen in classical fields of engineering such as automotive, aviation, and power generation, but is also inherent in more modern fields of engineering such as robotics, mechatronics, and automation. While these multi-domain and multi-physics systems allow for additional accuracy and more robust control, they also add to the complexity of designing, modeling, and simulating such systems.

The Linear Graph (LG) approach is a robust method of multi-domain dynamic system modeling which is systematic, unique, unified, and integrated. This means that the LG approach provides a well-defined method across multiple energy domains which produces a unique model of the evaluated multi-domain system, using methodologies that are analogous across the domains while considering the entire system concurrently [1]. This method is derived from graph theory, invented in 1736 by Leonhard Euler in order to solve a problem known as the Seven Bridges of Königsberg [2]; however, the formal LG approach itself was born through the work of Paynter [3] in the 1950s and 1960s at the Massachusetts Institute of Technology (MIT), as a precursor to his developments in bond graphs. The first form of LG approach was applied to engineering systems for the purpose of modeling large electrical networks before extending these principles to other energy domains in the 1960s; before this unification, modeling of different energy domains would require vastly different approaches [4]. Traditionally, while the LG approach is often applied to the electrical, mechanical, hydraulic, and thermal domains [5, 6, 7], this modeling method has also proven to be suitable for a wide range of additional domains including multibody [8], electrochemical [9], hydrodynamic [10], and many more.

While the LG approach is relatively easy to perform manually for low-order systems, it is beneficial to automate this process in order to evaluate larger, more complex, multi-domain systems. In the past, there have been some examples of software packages with the purpose of evaluating LG models; these programs include Lgraph, developed at MIT in the 1990s [11], DynaFlexPro, developed at the University of Waterloo in the mid 2000s [2, 8], and LG2ss developed at the University of British Columbia in the 2010s [12]. Unfortunately, the Lgraph software was only available on MIT workstation computers, and was never released publically or further maintained for modern operating systems. Likewise, DynaFlexPro has since been incorporated into the MapleSim software package, and while the underlying technology is based on LG approach, the program does not represent systems in an LG format. LG2ss was an effort in generalizing the LG approach, but it too was not refined or made publicly available. The main goal of the MATLAB-based LGtheory Toolbox presented in this paper is to provide a tool for automated development and evaluation of LG models, particularly facilitating education and research.

II. State-Space Modeling by LG Approach

A. Background

The LG approach provides a method of simplifying complex dynamic systems in the form of minimalistic graphical representations in order to facilitate the creation of state-space models. State-space modeling methods, such as the LG approach, are often preferred by engineers over traditional mathematical techniques as they provide a relatively simple and algorithmic process which eliminates much of the complexity of deriving such mathematical models.

An LG model represents a dynamic system as a collection of interconnected lines and consists primarily of two main components: branches, which are directional line segments that represent either passive or source type system elements; and nodes, which represent physical connections between system elements.

978-1-7281-1846-8/19/$31.00 ©2019 IEEE

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

Two variable types are considered in the LG approach: Across-variables, denoted generally as v, are defined as variables that can be measured "across" an element (e.g., voltage drop across a resister or pressure drop across a pipe segment); and through-variables, denoted generally as *f*, which are defined as variables that pass "through" an element unaltered (e.g., current passing through a resistor or fluid flow rate through a pipe). The product of the across- and through-variables gives the power flow through the element.

There are also three primary passive single-power-port element types, and two source element types. Specifically, A-type and T-type passive energy storage elements whose energy storage is expressed as a function of their across- and through-variable, respectively (e.g., electrical capacitors, and inertia elements are A-type elements; inductors, and springs are T-type elements). D-type elements are passive energy dissipative elements whose dissipation can be expressed as a function of either the across-variable or the through-variable (e.g., electrical resistors, mechanical damper). A-type and T-type source elements provide energy to the system in the form of their across-variable and through-variable, respectively. The constitutive equations of the three types of single-port passive elements are given in Table I.

TABLE I. CONSTITUTIVE EQUATIONS OF SINGLE-PORT ELEMENTS.

Element	Constitutive Equation	Energy/Power Equation
Generalized A-type	$f = C\frac{dv}{dt}$	$E = \frac{1}{2}Cv^2$
Generalized T-type	$v = L\frac{df}{dt}$	$E = \frac{1}{2}Lf^2$
Generalized D-type	$f = \frac{1}{R}v \qquad v = Rf$	$P = \frac{1}{R}v^2 = Rf^2$

Additionally, there two types of ideal two-power-port passive elements, Transformers and Gyrators. These are non-dissipative elements which convert their variable in magnitude (a single-domain element, e.g. ideal electrical transformer) or type (a two-domain element, e.g. DC Motor). In a transformer, the input variable type is related to the same output variable type; whereas, in a gyrator the input variable type is related to the opposite output variable type. The constitutive equations of two types of two-port elements are given in Table II.

TABLE II. CONSTITUTIVE EQUATIONS OF TWO-PORT ELEMENTS.

Element	Constitutive Equations	
Transformer	$v_1 = TFv_2$	$f_1 = -\left(\frac{1}{TF}\right)f_2$
Gyrator	$v_1 = GYf_2$	$f_1 = -\left(\frac{1}{GY}\right)v_2$

It is clear that these elements are not exclusive to just one energy domain, and are applicable analogously to other energy domains. Examples of these analogies for the five primary energy domains of focus are provided in Table III.

B. Linear Graph Methodology

In order to derive a state-space model from an LG model, the following procedure can be employed:

1. Construct the LG model of the system
2. Derive the independent differential and algebraic equations from the constitutive, continuity, and compatibility equations:
   a. Construct the normal tree
   b. Identify the state- ($\boldsymbol{x}$), input- ($\boldsymbol{u}$) and output-variables ($\boldsymbol{y}$), and the primary and secondary variables
   c. Produce the constitutive equations for each passive element
   d. Construct the continuity equations for each passive branch
   e. Construct the compatibility equations for each loop formed by including each passive link individually
3. Eliminate the secondary variables through substitution and construct the state-space model in the standard form:

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u} \quad (1)$$

$$\boldsymbol{y} = C\boldsymbol{x} + D\boldsymbol{u} \quad (2)$$

III. LGTHEORY MATLAB TOOLBOX

A. LG Model Input

In order to implement LG models in MATLAB, an incidence matrix representation is utilized. Incidence matrices, commonly used in graph theory, are sparse matrices used for representing relationships between two sets of objects. In the case of LG models, an incidence matrix is used to represent the relationship between the system elements (as columns) and the system nodes (as rows). Similarly, the directionality of the system elements is captured in this representation by a "-1" in the row corresponding to the node that the element is leaving, and a "1" in the row corresponding to the node that the element is entering.

Fig. 1 shows a DC motor with an inertial load and corresponding LG model. This system is a common example used to demonstrate LG modeling of multi-domain systems and will be referred to throughout this paper for demonstrating the algorithms and operations that are performed by the MATLAB program.

TABLE III. VARIABLES AND ELEMENT TYPES IN THE PRIMARY ENERGY DOMAINS.

Energy Domain	Source Elements		Storage Elements		Dissipating Elements
	Across-Variables	Through-Variables	A-Type	T-Type	D-Type
Electrical	Voltage	Current	Capacitor	Inductor	Resistor
Mechanical (Translational)	Rectilinear Velocity	Force	Mass	Rectilinear Spring	Rectilinear Damper
Mechanical (Rotational)	Angular Velocity	Torque	Rotary Inertia	Torsional Spring	Torsional Damper
Hydraulic/Fluid	Pressure	Flow Rate	Fluid Capacitor	Inertor	Fluid Resistor
Thermal	Temperature	Heat Transfer Rate	Thermal Capacitor		Thermal Resistor

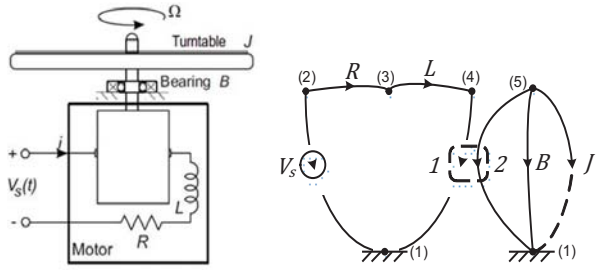


Fig. 1. Schematic [6] and LG Model of DC Motor with Inertial Load System.

From the above LG model, the following incidence matrix can be formed:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccccc} Vs & R & L & TF_1 & TF_2 & B & J \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & -1 \end{bmatrix} \end{array}$$

This matrix is input by the user into MATLAB along with the type, domain, variable names, and output arrays. These arrays are used to provide additional information about the system required to perform the LG operations: the type and domain arrays use indexing values (see Table IV) to specify the element type and energy domain corresponding column-wise to the elements of the incidence matrix; the variable names and output arrays use symbolic variables via MATLAB's Symbolic Toolbox in order to identify the element parameters corresponding to each column, and to identify the output variables of interest.

TABLE IV. INDEXES OF ELEMENT TYPES AND ENERGY DOMAINS IN LGTHEORY.

Index	Element Type
1	A-Source
2	A-Type Element
3	Transformer
4	Gyrator
5	D-Type Element
6	T-Type Element
7	T-Source

Index	Energy Domain
1	Electrical
2	Mech. Translational
3	Mech. Rotational
4	Hydraulic/Fluid
5	Thermal

B. Building the Normal Tree

The normal tree is a sub-graph of the LG model which connects all nodes of the LG while forming no loops. The normal tree is important in the LG approach as it allows for classification of the primary and secondary variables, as well as for providing a systematic process of identifying independent (A-types on tree, T-types not on tree) and dependent (A-types not on tree, T-types on tree) energy storage elements. Elements that belong to the tree are referred to as branches, while those excluded from the tree are called links. The algorithm for constructing the normal tree is provided:

1. Include all across-variable source elements
2. Include as many A-type elements as possible without forming loops
3. Include branches for two-port elements (transformers and gyrators): for a transformer, one branch is included; for a gyrator, either both or no branches are included
4. Include as many D-type elements as possible without forming loops
5. Include as many T-type elements as possible without forming loops.

The normal tree resulting from this process for the example system can be seen in Fig. 2, where the solid and half dashed lines represent branches and the full dashed lines represent links.

In MATLAB, each element is added to an empty incidence matrix, representing the normal tree, one at a time and evaluated to determine whether a loop is formed. In order to detect if a loop is created, a depth-first search algorithm is performed which starts at the ground node and searches through the tree; if the algorithm determines that the same node has been visited more than once, it is then known that a loop has been detected in the tree. If a loop is detected, the last element added to the tree is removed and the process is continued for the remaining elements.

In the case where the system contains one or more two-port elements (transformers or gyrators), the number of possible normal trees that can be created is 2^n, where n is the number of two-port elements in the system. In this case, LGtheory selects the final tree as the tree with the least number of T-type branches in order to minimize dependent T-type elements.

Applying this process to the example of DC motor with inertial load results in the following normal tree matrix:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccccc} Vs & R & L & TF_1 & TF_2 & B & J \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \end{array}$$

C. Variable Classification

Once the normal tree of the LG has been constructed, it can be utilized to assist in the process of variable classification. First, MATLAB creates two arrays which contain the across- and through-variables of all the system elements in symbolic form:

$$A_{vars} = [V_s(t)\ V_R\ V_L\ V_1\ \Omega_2\ \Omega_B\ \Omega_J] \quad (3)$$

$$T_{vars} = [i_s\ i_R\ i_L\ i_1\ T_2\ T_B\ T_J] \quad (4)$$

Similarly, the primary variables are classified as the across-variables of the branches and the through-variables of the links, while the secondary variables are classified as the through-variables of the branches and the across-variables of the links:

$$Primary = [V_s(t)\ V_R\ i_L\ V_1\ T_2\ T_B\ \Omega_J] \quad (5)$$

$$Secondary = [i_s\ i_R\ V_L\ i_1\ \Omega_2\ \Omega_B\ T_J] \quad (6)$$

The program then extracts the state variables as a vector containing the across-variables of the A-type branches, and the through-variables of the T-type links:

$$\boldsymbol{x} = \left[\Omega_J \;\; i_L\right]^T \tag{7}$$

The input variables are also extracted in the vector form from their respective source elements:

$$\boldsymbol{u} = [V_s(t)]^T \tag{8}$$

D. Constitutive Equations

The constitutive equations of the system are created for all passive elements. Referring to Table I and Table II, the constitutive equations of each passive element can be formed. Once formed, the program rearranges each equation to isolate for the primary variable associated with that element or its derivative.

For the example of the DC motor with inertial load, the constitutive equations are found to be:

$$\begin{aligned} \frac{d\Omega_J}{dt} &= \frac{1}{J}\, T_J \\ \frac{di_L}{dt} &= \frac{1}{L}\, V_L \\ V_R &= R \cdot i_R \\ T_B &= B \cdot \Omega_B \\ V_1 &= TF \cdot \Omega_2 \\ T_2 &= -TF \cdot i_1 \end{aligned} \tag{9}$$

E. Continuity Equations

The continuity equations of an LG model are formed using the contouring method. This method involves "cutting" around a node or set of nodes in such a way that only a single branch is intersected by the contour. This contour can thus be treated in a similar manner as a junction in Kirchhoff's Current Law, where the sum of all through-variables entering and exiting the contour are equal to zero. A continuity equation is constructed for each passive branch of the normal tree, were each equation is rearranged to isolate for the secondary variable of the passive branch.

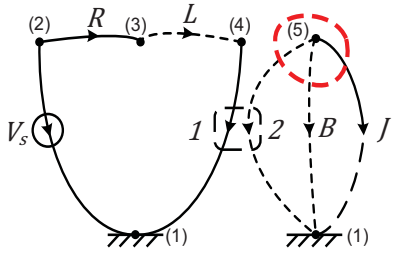


Fig. 2. Normal Tree of LG Model with a Node Contour.

This method is accomplished in MATLAB through the use of a depth-first search algorithm performed on the incidence matrix of the normal tree. Starting at either one of the nodes attached to the element in question, the algorithm checks for any other branches connected to the node, and follows each of these other branches to the next nodes which they are connected to. This process is repeated until there are no more branches that can be followed. The nodes that were visited during this process are now considered to be the nodes contained within the contour.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccccc} Vs & R & L & TF_1 & TF_2 & B & J \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & -1 \end{bmatrix} \end{array}$$

These nodes, identified in MATLAB as their respective incidence matrix rows, can thus be used to form the continuity equation. To do this, the rows of the LG incidence matrix identified as part of the contour are multiplied by the transpose of (4). The following is an example of this calculation for the inertial element J of the example LG model. The contour for this element consists of only node 5; therefore, row 5 of the LG incidence matrix is used:

$$0 = \begin{bmatrix} 0 & 0 & 0 & 0 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} i_s \\ i_R \\ i_L \\ i_1 \\ T_2 \\ T_B \\ T_J \end{bmatrix} \tag{10}$$

Once solved and rearranged for T_J, the continuity equation for inertial element J is:

$$T_J = -T_2 - T_B \tag{11}$$

Using the same process, the remaining continuity equations are written as:

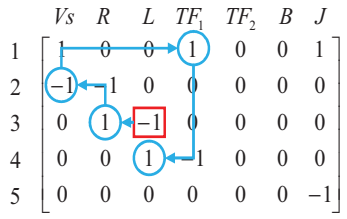
$$i_R = i_L \tag{12}$$

$$i_1 = i_L \tag{13}$$

F. Compatibility Equations

The compatibility equations of an LG model are constructed by temporarily including each passive link into the normal tree and writing the equation of the resulting loop. This method is treated in a similar manner as a loop in Kirchhoff's Voltage Law, where the sum of all across-variables in the loop is equal to zero. A compatibility equation is constructed for each passive link not contained in the normal tree, where each equation is rearranged to isolate for the passive link's secondary variable.

This method is accomplished in MATLAB by cycling through each passive link and temporarily adding its corresponding elemental column into the normal tree matrix. The same depth-first search algorithm used in constructing the normal tree is employed in order to find the loop created by the addition of the link. A vector is constructed which represents the directionality of the elements leaving each node (1 or -1) while traveling in a direction around the loop. This vector is multiplied by a column vector of the across-variables of the elements contained within the loop and equating the result to zero in order to form the compatibility equation for the loop. The following is an example of this process for the inductance element L of the linear graph; starting at node 3 and following the loop formed by this link in the counterclockwise direction results in:



$$0 = \begin{bmatrix} 1 & -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} V_R \\ V_s(t) \\ V_1 \\ V_2 \end{bmatrix} \tag{14}$$

Once solved and rearranged for V_L, the continuity equation for inductance element L is:

$$V_L = V_s(t) - V_R - V_1 \tag{15}$$

Using the same process, the remaining compatibility equations are found to be:

$$\Omega_2 = \Omega_J \tag{16}$$

$$\Omega_B = \Omega_J \tag{17}$$

G. Creating the State-Space Matrices

1) State Equation Matrices

With the construction of the constitutive, continuity, and compatibility equations, a symbolic substitution of the continuity and compatibility equations into the constitutive equations is performed in order to reduce the set of equations and eliminate all the secondary variables.

The substituted constitutive equation are classified into one of three column vectors depending on the isolated primary variable associated with the element: vector $\boldsymbol{x}$ for primary variables of independent storage elements (state variables); vector $\boldsymbol{d}$ for primary variables of dependent storage elements; and vector $\boldsymbol{p}$ for primary variables of non-energy storage elements. These vectors can thus be written as the following matrix equations:

$$\dot{\boldsymbol{x}} = \boldsymbol{P}\boldsymbol{x} + \boldsymbol{Q}\boldsymbol{p} + \boldsymbol{R}\boldsymbol{d} + \boldsymbol{S}\boldsymbol{u} \tag{18}$$

$$\boldsymbol{d} = \boldsymbol{M}\dot{\boldsymbol{x}} + \boldsymbol{N}\dot{\boldsymbol{u}} \tag{19}$$

$$\boldsymbol{p} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{J}\boldsymbol{p} + \boldsymbol{K}\boldsymbol{d} + \boldsymbol{L}\boldsymbol{u} \tag{20}$$

The general solution to the state-space equation is formed by isolating $\boldsymbol{d}$ in (19) and $\boldsymbol{p}$ in (20), and substituting the results into (18). Once simplified, this process results in the following general formulation of the state-space model:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{E}\dot{\boldsymbol{u}} \tag{21}$$

where,

$$\boldsymbol{A} = [\boldsymbol{I} - (\boldsymbol{Q}\boldsymbol{K}' + \boldsymbol{R})\boldsymbol{M}]^{-1}(\boldsymbol{P} + \boldsymbol{Q}\boldsymbol{H}') \tag{22}$$

$$\boldsymbol{B} = [\boldsymbol{I} - (\boldsymbol{Q}\boldsymbol{K}' + \boldsymbol{R})\boldsymbol{M}]^{-1}(\boldsymbol{S} + \boldsymbol{Q}\boldsymbol{L}') \tag{23}$$

$$\boldsymbol{E} = [\boldsymbol{I} - (\boldsymbol{Q}\boldsymbol{K}' + \boldsymbol{R})\boldsymbol{M}]^{-1}(\boldsymbol{R} + \boldsymbol{Q}\boldsymbol{K}')\boldsymbol{N} \tag{24}$$

and,

$$\boldsymbol{K}' = [\boldsymbol{I} - \boldsymbol{J}]^{-1}\boldsymbol{K} \tag{25}$$

$$\boldsymbol{H}' = [\boldsymbol{I} - \boldsymbol{J}]^{-1}\boldsymbol{H} \tag{26}$$

$$\boldsymbol{L}' = [\boldsymbol{I} - \boldsymbol{J}]^{-1}\boldsymbol{L} \tag{27}$$

Depending on the system being evaluated, this general solution can be simplified in the following two scenarios:

1. If the system contains no dependent energy storage elements (i.e. $\boldsymbol{d} = \boldsymbol{0}$), the general solution can be simplified by eliminating $\boldsymbol{R}$, $\boldsymbol{M}$, $\boldsymbol{N}$ and $\boldsymbol{K}$. This results in the following state-space model matrices:

$$\boldsymbol{A} = \boldsymbol{P} + \boldsymbol{Q}\boldsymbol{H}' \tag{28}$$

$$\boldsymbol{B} = \boldsymbol{S} + \boldsymbol{Q}\boldsymbol{L}' \tag{29}$$

2. If the system contains dependent energy storage elements (i.e. $\boldsymbol{d} \neq \boldsymbol{0}$) but contains no input derivatives (i.e. $\dot{\boldsymbol{u}} = \boldsymbol{0}$), the general solution can be simplified by eliminating $\boldsymbol{N}$. This results in the elimination of the $\boldsymbol{E}$ matrix, while $\boldsymbol{A}$ and $\boldsymbol{B}$ are calculated using (22) and (23), respectively.

For the example system, the MATLAB program determines from the normal tree that there are no dependent energy storage elements ($d = 0$), meaning that this system falls into scenario 1 described above. The MATLAB program subsequently extracts the necessary matrices and performs calculations for the state-space matrices using (28) and (29), to obtain:

$$\boldsymbol{A} = \begin{bmatrix} -\frac{B}{J} & \frac{TF}{J} \\ -\frac{TF}{L} & -\frac{R}{L} \end{bmatrix} \qquad \boldsymbol{B} = \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} \tag{30}$$

2) Output Equation Matricies

The output equations are constructed as an algebraic relationship between the variables of interest, as defined by the user in the output array, and the state and input variables. This is achieved in MATLAB by examining the continuity and compatibility equations, as well as, the substituted constitutive equations from the previous section, and selecting equations that can isolate the desired output variables. Once these equations are identified, substitution and manipulation operations are conducted in order to express the output variables exclusively in terms of the state and input variables; the $\boldsymbol{C}$ and $\boldsymbol{D}$ matrices are thus extracted from these equations.

For the example system, the variables of interest will be the current supplied to the motor, the torque output by the motor, and the rotational velocity of the inertial load. A corresponding output array is defined as:

$$\boldsymbol{y} = \begin{bmatrix} i_1 & T_2 & \Omega_J \end{bmatrix}^T \tag{31}$$

For these output variables, the following $\boldsymbol{C}$ and $\boldsymbol{D}$ matrices are determined:

$$\boldsymbol{C} = \begin{bmatrix} 0 & 1 \\ 0 & -TF \\ 1 & 0 \end{bmatrix} \qquad \boldsymbol{D} = 0 \tag{32}$$

IV. Results and Discussion

In order to validate the results generated by the LGtheory Toolbox, the state-space matrices produced by the program were simulated in MATLAB using commands from the Control System Toolbox. An identical system of a DC motor with an inertial load was modeled and simulated in Simscape, a dynamic system modeling library within the Simulink environment. Both simulations were conducted with reference to the parameter values obtained from a similar system in [13] and a step input of 12V for the voltage source.

From these graphs, a strong conformance between the two results can be observed, as seen in Fig. 3. Likewise, calculations of the error between the two results show that the difference between data points of each simulation is negligible. These observations demonstrate that the LGtheory Toolbox is capable of producing accurate and reliable state-space models of multi-energy domain dynamic systems.

V. Conclusion

The Linear Graph (LG) approach is a powerful tool for modeling complex, multi-physics dynamic systems spanning multiple energy domains. While in the past there have been examples of software tools capable of formulating LG models, most of these programs are no longer available for the purposes of direct research and education related to the LG approach. The LGtheory MATLAB Toolbox fills this gap by providing a complete and robust method of formulating LG models in the MATLAB software environment. For the example system of the DC motor with inertial load presented in this paper, LGtheory was able to produce an accurate state-space model which was validated via a comparative simulation with Simscape. The results of this comparison demonstrated strong conformance between the LGtheory and Simscape methods.

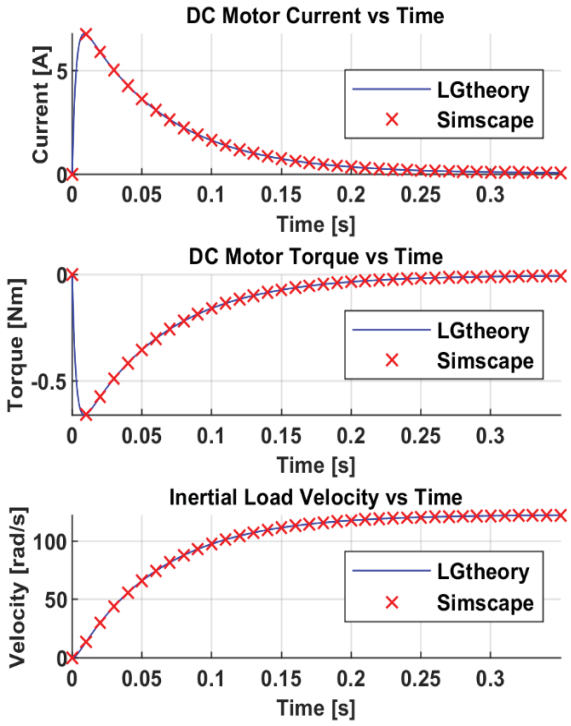


Fig. 3. MATLAB Simulation vs. Simscape Simulation Results.

Acknowledgment

This research was partially funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) through the Discovery Grant Program (RGPIN-2017-05762). The authors appreciate the support of their sponsors.

References

[1] C. W. de Silva, "Linear Graphs," in Modeling of Dynamic Systems - With Engineering Applications, Boca Raton, Taylor & Francis, CRC Press, 2018, pp. 199-391.

[2] C. Schemike, K. Morency and J. McPhee, "Using Graph Theory and Symbolic Computing to Generate Efficient Models for Multi-Body Vehicle Dynamics," Proceedings of the Institution of Mechanical Engineers. Part K, Journal of multi-body dynamics, vol. 222, no. 4, pp. 339-352, 2008.

[3] H. M. Paynter, Analysis and Design of Engineering Systems, Boston: The M.I.T. Press, 1961.

[4] H. E. Koenig and W. A. Blackwell, "Linear Graph Theory - A Fundamental Engineering Discipline," IRE Transactions on Education, vol. 3, no. 2, pp. 42-49, 1960.

[5] Massachusetts Institute of Technology, Department of Mechanical Engineering, "Linear Graph Modeling: One-Port Elements," 2004.

[6] Massachusetts Institute of Technology, Department of Mechanical Engineering, "Linear Graph Modeling: Two-Port Energy Transducing Elements," 2003.

[7] Massachusetts Institute of Technology, Department of Mechanical Engineering, "Linear Graph Modeling: State Equation Formulation," 2004.

[8] J. McPhee, C. Schmitke and S. Redmond, "Dynamic modelling of mechatronic multibody systems with symbolic computing and linear graph theory," Mathematical and Computer Modelling of Dynamical Systems, vol. 10, no. 1, pp. 1-23, 2004.

[9] T. -S. Dao and J. McPhee, "Dynamic modeling of electrochemical systems using linear graph theory," Journal of Power Sources, vol. 196, no. 23, pp. 104442-10454, 2011.

[10] J. Banerjee and J. McPhee, "System dynamic modelling and simulation of hydrodynamic machines," Mathematical and Computer Modelling of Dynamical Systems, vol. 22, no. 1, pp. 54-86, 2015.

[11] W. K. Durfee, M. B. Wall, D. Rowell and F. K. Abbott, "Interactive Software for Dynamic System Modeling Using Linear Graphs," IEEE Control Systems Magazine, vol. 11, no. 4, pp. 60-66, 1991.

[12] C. de Silva, "Some Generalisations of Linear-Graph Modelling for Dynamic Systems," International Journal of Control, vol. 86, no. 11, pp. 1990-2005, 2013.

[13] R. A. R. Picone, "Dynamic Systems: An Introduction," 2018. [Online]. Available: http://ricopic.one/dynamic_systems/dynamic_systems_partial.pdf. [Accessed August 2018].