Towards Computer-Automated Mechatronic Design and Optimization using Linear Graphs and Evolutionary Computing

by

Eric McCormick

A thesis submitted to the
School of Graduate and Postdoctoral Studies in partial
fulfillment of the requirements for the degree of

Master of Applied Science in Mechanical Engineering

Faculty of Engineering and Applied Science

University of Ontario Institute of Technology (Ontario Tech University)

Oshawa, Ontario, Canada

April 2020

THESIS EXAMINATION INFORMATION

Submitted by: **Eric McCormick**

Master of Applied Science in **Mechanical Engineering**

Thesis title: Towards Computer-Automated Mechatronic Design and Optimization using Linear Graphs and Evolutionary Computing

An oral defense of this thesis took place on April 21, 2020 in front of the following examining committee:

Examining Committee:

Chair of Examining Committee	Martin Agelin-Chaab
Research Supervisor	Haoxiang Lang
Examining Committee Member	Xianke Lin
Thesis Examiner	Dipal Patel, FEAS

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

ABSTRACT

A comparison of literature between the linear graph (LG) and bond graph (BG) approaches shows that the surpassing of BGs to LGs in applications related to the modeling of mechatronic systems is driven primarily by a lack of available LG-based software. As a result, a robust software toolbox called LGtheory has been developed for automating the evaluation of LG models in the MATLAB programming environment. This thesis details the development of LGtheory, and the algorithms and procedures employed for the evaluation of LG models. In addition, demonstrations of this toolbox to a process for automating the design of electronic filter circuits, as well as, the modeling and simulation of the dynamics of a mobile robotic system are presented. The results of these demonstrations validate the accuracy of the LGtheory toolbox for modeling complex multi-domain systems, and provides the methodological basis for the automated design of mechatronic systems using LGs.

Keywords: Linear Graph Modeling; Mechatronics; Automated Design; Genetic Programming; Genetic Algorithms;

AUTHOR'S DECLARATION

I hereby declare that this thesis consists of original work of which I have authored. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize the University of Ontario Institute of Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize the University of Ontario Institute of Technology to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my thesis will be made electronically available to the public.

[illegible]

YOUR NAME

STATEMENT OF CONTRIBUTIONS

Parts of this thesis have been published or submitted for publication in the following:

E. McCormick, H. Lang and C. W. de Silva, "Automated Multi-Domain Engineering Design through Linear Graphs and Genetic Programming," Genetic Programming and Evolvable Machines, 2020. (Submitted).

E. McCormick, H. Lang and C. W. de Silva, "Dynamic Modeling and Simulation of a Four-Wheel Skid-Steer Mobile Robot using Linear Graphs," IEEE/ASME Transactions on Mechatronics, 2020. (Submitted).

E. McCormick, H. Lang and C. W. de Silva, "Research and Development of a Linear Graph-based MATLAB Toolbox," in The 14th International Conference on Computer Science & Education (ICCSE 2019), Toronto, 2019, pp. 942-947.

E. McCormick, Y. Wang and H. Lang, "Optimization of a 3-RRR Delta Robot for a Desired Workspace with Real-Time Simulation in MATLAB," in The 14th International Conference on Computer Science & Education (ICCSE 2019), Toronto, 2019, pp. 935-941.

The writing of the manuscripts, data collection and research conducted in this work and the mentioned publication(s) was completed by the author. Co-authors reviewed and provided technical support when required.

ACKNOWLEDGEMENTS

Firstly, I would like to acknowledge my Master’s Degree supervisor, Dr. Haoxiang Lang, for his guidance, expertise, and support in the completion of both this thesis, and the research which went into its preparation.

In addition, I would like to express my gratitude to my mother and father for their love and financial support, as well as, my entire family, including my fiancée Lindsay and her family, for their support and encouragement which motivated me throughout the duration of my undergraduate and graduate degrees.

Finally, I would like to thank all of my colleagues in the GRASP Lab for all their help, guidance, and friendships over the last two years.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
A-Type	Across-Variable Element Type
BG	Bond Graph
DC	Direct Current
D-Type	Dissipative Element Type
GA	Genetic Algorithm
GP	Genetic Programming
GR	Gear Ratio
GY	Gyrator
LG	Linear Graph
MIT	Massachusetts Institute of Technology
ROS	Robot Operating System
SE	Source of Effort
SF	Source of Flow
TF	Transformer
T-Type	Through-Variable Element Type

NOMENCLATURE

P	Power Flow of Element
v	Generalized across-variable, linear translational velocity
f	Generalized through-variable
C	Generalized capacitance, electrical capacitance
R	Generalized resistance, electrical resistance
L	Generalized inductance, electrical inductance
V	Voltage
i	Current
F	Force
m	Mass
B	Damping constant
K	Spring constant
ω	Angular velocity
T	Torque
P	Pressure
Q	Volume flow rate
T	Temperature
q	Heat flow rate
$\boldsymbol{x}$	State vector
$\boldsymbol{u}$	Input vector
$\boldsymbol{y}$	Output Vector
$\boldsymbol{A}$	State matrix

$\boldsymbol{B}$	Input matrix
$\boldsymbol{C}$	Output matrix
$\boldsymbol{D}$	Feedforward matrix
$\boldsymbol{E}$	Input derivative matrix for state equation
$\boldsymbol{F}$	Input derivative matrix for output equation
e	Effort variable
f	Flow variable
0	Effort equalizing junction
1	Flow equalizing junction
C_e	Equivalent generalized capacitance
L_e	Equivalent generalized inductance
In	Incidence matrix
$\boldsymbol{A_{tr}}$	Reduced normal tree sub-matrix
$\boldsymbol{A_{co}}$	Reduced co-tree sub-matrix
$\boldsymbol{F_c}$	Fundamental cut-set matrix
$\boldsymbol{v_{tr}}, \boldsymbol{v_{co}}$	Across-variables of tree and co-tree elements
$\boldsymbol{f_{tr}}, \boldsymbol{f_{co}}$	Through-variables of tree and co-tree elements
$V_s, V_s(t)$	Voltage source of filter circuit models
R_s	Source resistance of filter circuit models
R_L	Load resistance of filter circuit models
T_{w_i}	Torque to wheel i
F_{w_i}	Force produced by wheel i

$F_{w_{i_t}}$	Tangential component of force produced by wheel i
F_{F_i}	Force of friction on wheel
r_w	Radius of wheels
r_{c_i}	Radius of contact point of wheel i to center of mass of mobile robot
θ_{w_i}	Angle of tangential wheel forces
obj	Objective function
$x_{data}(t)$	x component data from the trajectory of the mobile robot
$y_{data}(t)$	y component data from the trajectory of the mobile robot
$x_{LG}(t)$	x components of the linear graph model trajectory
$y_{LG}(t)$	y components of the linear graph model trajectory
v_t, v_r	Translational and rotational velocity commands to mobile robot
c	Coefficient of equation for estimating mobile robot motor voltages
$\lvert\bar{X}\rvert$	Maximum absolute tracking error in the x direction
$\lvert\bar{Y}\rvert$	Maximum absolute tracking error in the y direction

Chapter 1. Introduction

Engineering systems are becoming increasingly more advanced with the integration of multiple engineering disciplines into systems that may have traditionally only been comprised of a single energy domain. This field of multi-disciplinary engineering, known as mechatronics engineering, has brought with it a significant acceleration in technological advancement. While this advancement has come with the introduction of many robotic, electronic, computerized, and control systems; this has also resulted in an increase in complexity associated with designing, modeling, and controlling such multi-domain systems. In recent decades, many methods of system modeling have been introduced which address this complexity issue, but most of them lack the integrated and unified nature associated with the linear graph (LG) approach.

Graph theory, the basis in which LG theory is derived, was invented in 1736 by Leonhard Euler as a method for analyzing a problem known as the Seven Bridges of Königsberg [1]. This problem consists of 4 land masses that are connected to one another by seven bridges, with the question being whether it is possible to walk a route in which you cross over each of the seven bridges only once. By modeling the topology of the problem in abstract terms using graph theory, Euler ultimately proved that the problem has no solution.

In the 1950s and 60s, graph theory was applied to engineering systems for the first time in the form of LG theory for the purpose of modeling large electrical networks, before being further developed and expanded to systems relating to other energy domains in the 1960s [2]. Before this development, traditional techniques for mathematically modeling systems of different energy domains varied greatly from one another; this advancement in LG theory allowed the simplified procedures and concepts for analyzing complex electrical networks to be extended to physical systems of other energy domains.

1.1 Scope and Objectives

The primary scope of this work is to design and develop a MATLAB-based software tool for automating the process of deriving state-space equations from LG models. The goal of developing this software tool is to facilitate further research into the field of LG theory, and to provide a means of evaluating larger, more complex multi-domain systems which may be impractical to evaluate manually. Demonstrations of this software tool in applications of evolutionary design are presented as a means of encouraging further research into the applications of LG theory in the field of automated system design. Additionally, the application of the toolbox to a complex electromechanical system, in the form of a mobile robot, is presented in order to provide further validation of software's capabilities. The expected outcome of this work is a fully functional LG-based MATLAB toolbox which is capable of evaluating dynamic systems of any size, encompassing any number of energy domains.

The detailed objectives of this work are as follows:

1.1.1 Design and Development

- Design and development of an LG-based MATLAB software toolbox for the automatic derivation of state-space models

1.1.2 Integration with Evolutionary Computing Techniques

- Integration of genetic programming (GP) with the LG-based software tool for the purpose of automatic LG model synthesis

- Integration of genetic algorithms (GA) with the LG-based software tool for the purpose of system calibration through model parameter estimation

1.1.3 Validation

- Validation of the LG-based software toolbox through a comparative simulation of an equivalent model using commercial software (Simulink Simscape)
- Validation of the combined LG and GP approach through a demonstration of the automatic design of low pass, high pass, and band pass electronic filter circuits
- Validation of the combined LG and GA approach through the LG modeling of a mobile robotic system with comparisons against a ROS Gazebo-based simulation and data collected from experiments with the physical system

1.2 Outline of the Thesis

Chapter 1 introduces the area of research of this thesis, highlights the scope and objectives of the presented work, and provides background on the major concepts of LG theory with an example of modeling a dynamic system using this approach.

Chapter 2 provides a detailed literature review of past and recent research related to LG theory and a closely related modeling method called Bond Graph (BG) theory, along with a comparison of these two approaches.

Chapter 3 discusses the design and development of an LG theory-based software tool, the LGtheory MATLAB toolbox, including the procedures and algorithms the program uses to convert LG models into their state-space representations.

Chapter 4 presents the combined LG and GP approach for the automated design and evaluation of dynamic systems, demonstrated through the automated synthesis of electronic filter circuits. This chapter details the various functions used to construct LG models using GP, and presents the system models constructed using this automated approach.

Chapter 5 proposes an LG model of a mobile robot system calibrated through a parameter estimation procedure conducted using GAs. Validation of this model is demonstrated through a comparison of the simulated model with data collected from a ROS/Gazebo simulation and experiments with the physical system.

Chapter 6 concludes this thesis by highlighting the contributions of the presented work, and proposes recommendations for possible future work in this field of research.

1.3 Background on Linear Graph Theory

The LG modeling approach is a method of evaluating complex dynamic systems through the use of a simplistic graphical representation in order to assist in the derivation of state-space models. This systematic, unique, unified, and integrated approach provides a robust modeling method that produces a single unique solution for any system through the analogous application of methodologies across multiple energy domains. This means that a multi-domain mechatronic system is evaluated as a single model, not as a series of separate models, while applying the same basic procedures for each of the system's energy domains [3]. Graphical methods, such as LGs, are often preferred by engineers over traditional mathematical techniques as they provide a relatively simple approach that is easier to visualize and understand.

1.3.1 Linear Graph Fundamentals

The LG modeling approach represents dynamic systems through two main components: branches, which are line segments used to represent the elements of the system; and nodes, which are points representing the physical connections between the system elements.

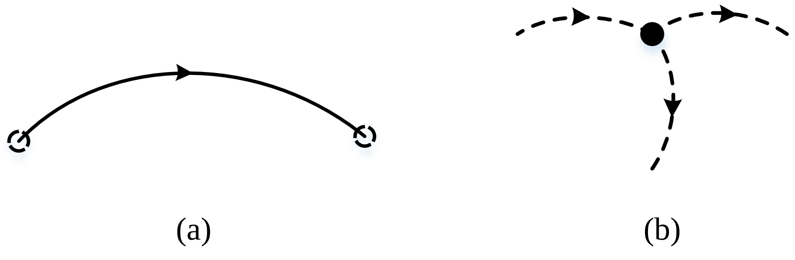


Figure 1-1: Example of a Branch (a) and a Node (b)

Similarly, there are two variable types which are considered in the LG modeling approach:

- Across-variables, generally denoted as v, are variables measured “across” an element. For example, in order to measure the voltage across an electrical component, a voltmeter must be set up in parallel with the element. The voltage reading from the voltmeter will be equal to the voltage drop across the component.
- Through-variables, generally denoted as f, are variables measured “through” an element. For example, in order to measure the current drawn by an electrical component, an ammeter must be set up in series with the element. The current reading from the ammeter will be equal to the current drawn through the component.

The power flow of an element can thus be determined by taking the product of the across- and through-variables of that element.

$$P = f \times v \tag{1}$$

There are three primary passive single-port element types represented in the LG modeling approach [4]:

- A-Type elements are a form of energy storage elements in which the energy storage of the element is expressed as a function in terms of the across-variable. Examples of A-Type elements include energy storage components such as capacitors for electrical systems, and mass and inertial elements for mechanical systems.
- T-Type elements are a form of energy storage elements in which the energy storage is expressed as a function in terms of the through-variable. Examples of T-Type elements include energy storage components such as inductors for electrical systems, and springs for mechanical systems.
- D-Type elements are energy dissipating elements and provide no energy storage. The amount of energy dissipated by these elements can be expressed as a function of either their across- or through-variables. Examples of D-Type elements include energy dissipative components such as resistors for electrical systems, and dampers for mechanical systems.

LG modeling also utilizes two additional non-passive single-port element types which act as a means of providing energy into the system. These elements are the across-variable and through-variable source elements. Examples of these elements include voltage sources (across) and current sources (through) for the electrical domain, and velocity sources, such as cams (across), and external force sources (through) for the mechanical translational domain [4].

The following table provides a summary of the variable-types and single-port passive elements for the five primary energy domains:

Table 1-1: Summary of Variable and Element Types for the Primary Energy Domains

Energy Domain		Across-Variable	Through-Variable	A-Type Element	D-Type Element	T-Type Element
Generalized	Symbol	$v(t)$	$f(t)$	C	R	L
Electrical	Name	Voltage	Current	Capacitance	Resistance	Inductance
	Parameter	$V(t)$	$i(t)$	C	R	L
	Units	V	A	F	Ω	H
Mechanical Translational	Name	Velocity	Force	Mass	Damper	Spring
	Parameter	$v(t)$	$F(t)$	m	B	$1/K$ *
	Units	m/s	N	kg	$N \cdot s/m$	m/N *
Mechanical Rotational	Name	Angular Velocity	Torque	Inertia	Rotational Damper	Torsional Spring
	Parameter	$\omega(t)$	$\mathrm{T}(t)$	J	B	$1/K$ *
	Units	rad/s	$N \cdot m$	$kg \cdot m^2$	$rad/(N \cdot m \cdot s)$	$rad/(N \cdot m)$ *
Hydraulic/ Fluid	Name	Pressure	Volume Flow Rate	Capacitance	Resistance	Inertance
	Parameter	$P(t)$	$Q(t)$	C_f	R_f	I_f
	Units	Pa	m^3/s	m^3/Pa	$Pa \cdot s/m^3$	$Pa \cdot s^2/m^3$
Thermal	Name	Temperature	Heat Flow Rate	Capacitance	Resistance	-
	Parameter	$T(t)$	$q(t)$	C_t	R_t	-
	Units	K	W	J/K	K/W	-

*T-Type element parameters and units for the mechanical translational and rotational domains are the inverse of their respective spring constants

Finally, there are two passive two-port element types which can be used either for converting power within a single domain, or for transferring power from one domain into another. These elements are called transformers and gyrators. The main distinction between transformers and gyrators are that transformers convert one variable type (across or through) at the first port to the same variable type at the second port; whereas, gyrators convert the variable type at the first port into the opposite variable type at the second port. Examples of each element type include DC motors for transformers (current into the motor is converted into torque), and pistons for gyrators (fluid pressure is converted into mechanical force) [5].

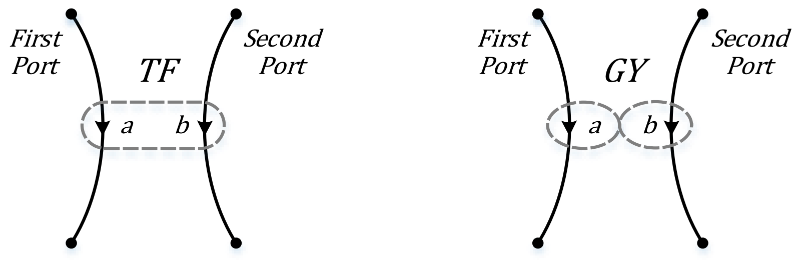


Figure 1-2: Examples of (a) Transformer and (b) Gyrator Element Representations

The following table shows the constitutive equations (also referred to as elemental equations) for each of the passive elements in LG theory [4, 5]:

Table 1-2: Constitutive Equations of Single- and Two-Port Passive Elements

Element	Constitutive Equations	
Generalized A-type	$f = C\frac{dv}{dt}$	
Generalized T-type	$v = L\frac{df}{dt}$	
Generalized D-type	$f = \frac{1}{R}v$ *or*	$v = Rf$
Transformer	$v_1 = TFv_2$	$f_1 = -\left(\frac{1}{TF}\right)f_2$
Gyrator	$v_1 = GYf_2$	$f_1 = -\left(\frac{1}{GY}\right)v_2$

1.3.2 Constructing a Linear Graph Model

The process of constructing an LG model starts by identifying the essential characteristics of the system. These characteristics include the energy domain(s) of the system, the source(s) of energy input and corresponding variable type(s), the components which provide energy storage and dissipation, and the desired system output(s). Once these characteristics are determined, the appropriate system elements representing each of the system inputs and components can be selected. The system nodes can thus be drawn by identifying the physical connections between system components within the system and placing a node at these locations. The LG elements can then be drawn by identifying the nodes between which the corresponding system components are located and drawing the elements between these nodes [6].

1.3.2.1 Example of Constructing a Linear Graph Model

The following is an example of the aforementioned process applied to the following electromechanical system of a DC motor with a compliant shaft and inertial load:

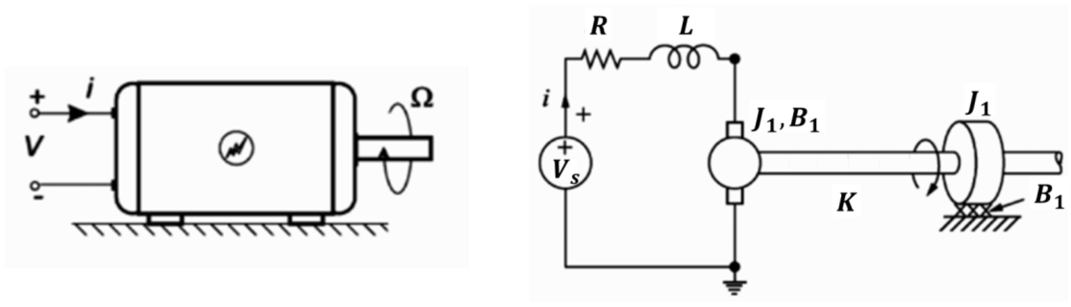


Figure 1-3: DC Motor System [7] with Equivalent Schematic Model

The energy domains of the system are electrical and mechanical rotational. The input source of the system is a voltage supply to the DC motor which can be represented in LG form with an across-variable source element. The system consists of the following components which can be represented with the corresponding passive element types:

- Internal resistance of the DC motor, represented as a D-Type element
- Internal inductance of the DC motor windings, represented as a T-Type element
- The electromechanical transducer mechanism which converts the current of the motor to torque, represented as a transformer element
- The inertia of the motor rotor, represented as an A-Type element
- The internal mechanical resistance of the motor bearings, represented as a D–Type element
- The stiffness of the compliant motor shaft, represented as a T-Type element

- The inertia of the load on the motor shaft, represented as an A-Type element
- The resistance on the load of the motor shaft, represented as a D-Type element

Similar to the schematic diagram for the electrical side of the system, the electrical side of the LG model is drawn with the across-variable voltage source providing energy to a series circuit consisting of the resistive D-Type element, the inductive T-Type element, and the first port of the transformer element. On the mechanical side, the top node of the transformer's second port splits into three paths. The first two of these paths consist of an inertial A-Type element for the rotor and a resistive D-Type element for the motor bearing, connected in parallel with one another to ground. The third path consists of the T-Type element, representing the compliant motor shaft. This path splits once again at the far node of the T-Type element into another set of parallel A- and D-Type elements, representing the inertia and resistance of the load. The following figure illustrates this linear graph model:

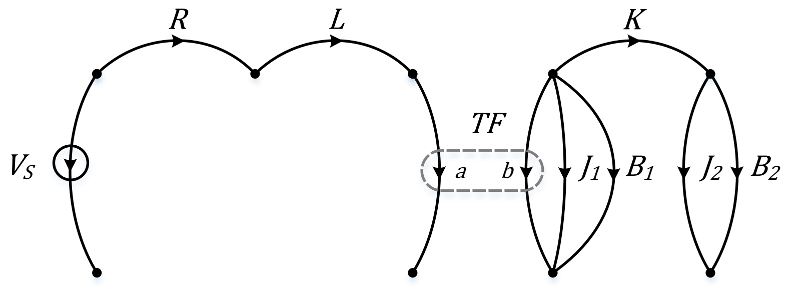


Figure 1-4: LG Model of Electromechanical System

1.3.3 Conversion of Linear Graphs to State-Space Models

The next step after the creation of the linear graph is to extract the state-space model of the system. State-space models are a set of first-order differential equations expressed in terms of the state- and input-variables which describe how the system responses to external inputs. The first step in this process is to identify the state variables, input variables, and output variables of the system:

- State variables, denoted as a vector $\boldsymbol{x}$, are a set of variables that can be used to determine the total status of a system. The number of state variables in a system is equal to the number of independent energy storage elements in the system (and equal to the system order).
- Input variables, denoted as a vector $\boldsymbol{u}$, are the variables representing the system's across- and through-variable inputs as a function of time.
- Output variables, denoted as a vector $\boldsymbol{y}$, are the selected variables that will ultimately be analyzed and represented in terms of the state and input variables.

The state, input, and output vectors are related through the state and output equations. These equations make up what is called the state-space model of a physical system:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} \end{aligned} \tag{2}$$

Where the state-space matrixes of the equation are as follows:

- The system matrix, denoted as $\boldsymbol{A}$, is an $n \times n$ matrix (where n is the order of the system) which describes how the current states of the system affect the rate of change of the states
- The input matrix, denoted as $\boldsymbol{B}$, is an $n \times r$ matrix (where r is the number of inputs into the system) which describes how the inputs to the system affect the rate of change of the states

- The output matrix, denoted as $\boldsymbol{C}$, is an $m \times n$ matrix (where m is the number of outputs specified for the system) which describes how the current states of the system affect the outputs
- The feedforward matrix, denoted as $\boldsymbol{D}$, is an $m \times r$ matrix which describes how the inputs to the system affect the outputs

Before the state-space models can be derived from the LG, a normal tree model must be constructed. A normal tree is a subgraph of the LG which contains every node of the model but no closed loops. In order to construct the normal tree, the following features from the LG model must be included [8]:

1) all system nodes
2) all across-variable sources (if an across-variable source cannot be included, the model is not valid)
3) the maximum number of A-Type elements as possible without creating any loops (or though its inclusion, would require the inclusion or exclusion of both transformer branches or the inclusion of only one branch of a gyrator, thus signifying an invalid model)
4) one branch of each transformer element, and both or neither branches of each gyrator element in such a manner as to minimize the number of T-Type elements contained within the normal tree.
5) as many D-Type elements as possible without creating any loops
6) as many T-Type elements as possible without creating any loops
7) none of the through-variable sources (if a through-variable source is required to complete a tree, the model is not valid)

All elements contained within the normal tree are called branches, while the elements not in the tree, which make up the co-tree, are called links. A-Type elements not in the tree and T-Type elements in the tree are dependant energy storage elements, while A-Types in the tree and T-Types not in the tree are independent energy storage elements. The state-variables of the system can thus be defined as the across-variables of A-Type branches and the through-variables of T-Type links.

Due to the nature of the algorithm for creating the normal tree (specifically due to step 4), an LG model consisting of T number of two-port elements have a possible 2^T normal tree configurations. In these cases, the normal tree configuration which must be selected is the one which maximizes the number of A-Type elements, while minimizing the number of T-Type elements contained within the tree.

In order to derive the state-space model of a multi-domain engineering system using the LG modeling approach, the following procedure must be applied [8]:

1) Construct the LG model of the system using the procedure detailed in Section 1.3.2
2) Derive the independent differential and algebraic equations of the system from the constitutive, continuity, and compatibility equations:
    a) Construct the normal tree in order to determine if there are any dependent energy storage elements:
    b) Identify the state- ($\boldsymbol{x}$), input- ($\boldsymbol{u}$), and output-variables ($\boldsymbol{y}$), as well as the primary- (across-variables of branches and through-variables of links) and secondary-variables (through-variables of branches and across-variables of links) of the system

c) Construct the constitutive equations for every passive element in the system by referring to Table 1-2

d) Construct the continuity (node) equations by creating a contour around one node of each passive branch and solve for that element's through-variable

e) Construct the compatibility (loop) equations for the loops formed by temporarily including each passive link and solve for that element's across-variable

3) Construct the state-space model in standard form through substitution of the continuity and compatibility equations into the constitutive equations and reduce the number of equations to that of the system order, or use the linear algebra method described in [8].

1.3.4 Example of State-Space Derivation from a Linear Graph Model

Referring to the LG model produced for the electromechanical system in Figure 1-4, the process of deriving the independent differential and algebraic equations of the system begins by producing the normal tree.

While producing the normal tree following the directions in Section 1.3.3, it must be noted that the model contains a transformer element; therefore, this model has two possible normal tree configurations, both of which are pictured below:

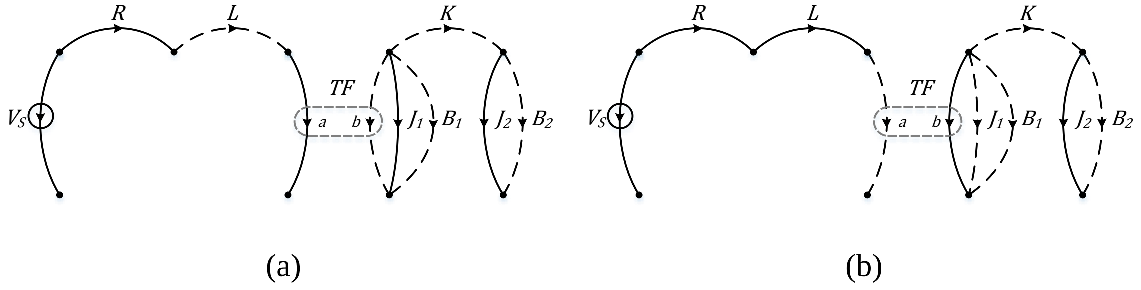


Figure 1-5: Possible Normal Tree Configurations of the Electromechanical System

Based on the requirement to maximize the A-Type elements, and minimize the T-Type elements contained within the tree, configuration (a) which includes the first-port of the transformer element shall be chosen as it results in no dependent energy storage elements.

Now that the normal tree has been constructed, the state-, input-, primary-, and secondary-variables can be identified.

$$\boldsymbol{x} = [i_L \quad \omega_{J_2} \quad T_K \quad \omega_{J_2}]^T \tag{3}$$

$$\boldsymbol{u} = V_s \tag{4}$$

Primary variables: V_S, V_R, i_L, V_{TF}, T_{TF}, ω_{J_1}, T_{B_1}, T_K, ω_{J_2}, and T_{B_2}

Secondary variables: i_S, i_R, V_L, i_{TF}, ω_{TF}, T_{J_1}, ω_{B_1}, ω_K, T_{J_2}, and ω_{B_2}

Referring to Table 1-2, the constitutive equations of each passive element are constructed and isolated for the primary variables and their derivatives:

$$\begin{gathered}
J_1 \to \frac{d\omega_{J_1}}{dt} = \frac{1}{J_1} T_{J_1} \\
J_2 \to \frac{d\omega_{J_2}}{dt} = \frac{1}{J_2} T_{J_2} \\
L \to \frac{di_L}{dt} = \frac{1}{L} V_L \\
K \to \frac{dT_K}{dt} = K \cdot \omega_K \\
R \to V_R = R \cdot i_R \\
B_1 \to T_{B_1} = B_1 \cdot \omega_{B_1} \\
B_2 \to T_{B_2} = B_2 \cdot \omega_{B_2} \\
TF \to V_{TF} = TF \cdot \omega_{TF} \\
\text{and} \\
T_{TF} = -TF \cdot i_{TF}
\end{gathered} \tag{5}$$

Next, contours must be created for each passive branch which intersect only that branch. This can be seen in the following figure:

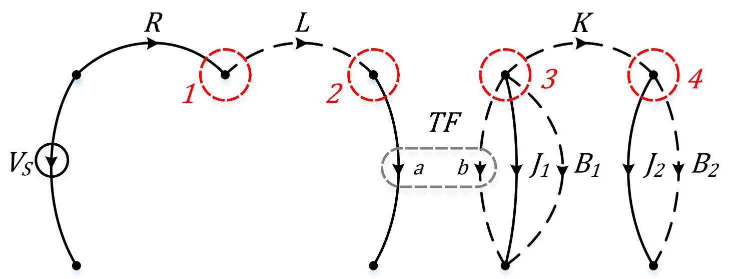


Figure 1-6: Contours Intersecting Passive Branches of LG Model

Producing the continuity equations for these branches is done by equating the sum of all the through-variables entering the contours to those leaving (similar to Kirchhoff's Current Law), then isolating for the through-variable of the branch. The continuity equations of this system are as follows:

$$\begin{gathered}(1) \rightarrow i_R = i_L \\ (2) \rightarrow i_{TF} = i_L \\ (3) \rightarrow T_{J_1} = -T_{B_1} - T_K - T_{TF} \\ (4) \rightarrow T_{J_2} = T_K - T_{B_2}\end{gathered} \tag{6}$$

Next, the compatibility equations are produced by temporarily including each link of the co-tree in the normal tree, evaluating the loops of the resulting circuits for the across-variables (similar to Kirchhoff's Voltage Law), and isolating for the across-variable of the links. The compatibility equations of this system are as follows:

$$\begin{gathered}V_L = V_s - V_{TF} - V_R \\ \omega_{TF} = \omega_{J_1} \\ \omega_{B_1} = \omega_{J_1} \\ \omega_K = \omega_{J_1} - \omega_{J_2}\end{gathered} \tag{7}$$

$$\omega_{B_2} = \omega_{J_2}$$

Now that all independent differential and algebraic equations of the system have been produced, the elimination through substitution method will be employed to reduce the set of equations to the order of the system, and produce the state-space model.

First, the continuity and compatibility equations, (6) and (7), must be substituted into the continuity equations, (5), in order to eliminate secondary-variables:

$$\begin{gathered}
\frac{d\omega_{J_1}}{dt} = \frac{1}{J_1}\left(-T_{B_1} - T_K - T_{TF}\right) \\
\frac{d\omega_{J_2}}{dt} = \frac{1}{J_2}\left(T_K - T_{B_2}\right) \\
\frac{di_L}{dt} = \frac{1}{L}\left(V_s - V_{TF} - V_R\right) \\
\frac{dT_K}{dt} = K \cdot \left(\omega_{J_1} - \omega_{J_2}\right) \\
V_R = R \cdot i_L \\
T_{B_1} = B_1 \cdot \omega_{J_1} \\
T_{B_2} = B_2 \cdot \omega_{J_2} \\
V_{TF} = TF \cdot \omega_{J_1} \\
T_{TF} = -TF \cdot i_L
\end{gathered} \tag{8}$$

Now, the equations for dependent energy storage (if present) and dissipative elements must be substituted into the equations for independent energy storage elements in order to eliminate all but the state- and input variables:

$$
\begin{aligned}
\frac{d\omega_{J_1}}{dt} &= \frac{1}{J_1}\left(-B_1 \cdot \omega_{J_1} - T_K + TF \cdot i_L\right) \\
\frac{d\omega_{J_2}}{dt} &= \frac{1}{J_2}\left(T_K - B_2 \cdot \omega_{J_2}\right) \\
\frac{di_L}{dt} &= \frac{1}{L}\left(V_s - TF \cdot \omega_{J_1} - R \cdot i_L\right) \\
\frac{dT_K}{dt} &= K \cdot \left(\omega_{J_1} - \omega_{J_2}\right)
\end{aligned}
\tag{9}
$$

Finally, the produced set of equations in (9) can thus be rewritten in the standard state-space form as shown in (2):

$$
\begin{bmatrix} \frac{d\omega_{J_1}}{dt} \\ \frac{d\omega_{J_2}}{dt} \\ \frac{di_L}{dt} \\ \frac{dT_K}{dt} \end{bmatrix}
=
\begin{bmatrix}
-\frac{B_1}{J_1} & 0 & \frac{TF}{J_1} & -\frac{1}{J_1} \\
0 & -\frac{B_2}{J_2} & 0 & \frac{1}{J_2} \\
-\frac{TF}{L} & 0 & -\frac{R}{L} & 0 \\
K & -K & 0 & 0
\end{bmatrix}
\begin{bmatrix} \omega_{J_1} \\ \omega_{J_2} \\ i_L \\ T_K \end{bmatrix}
+
\begin{bmatrix} 0 \\ 0 \\ \frac{1}{L} \\ 0 \end{bmatrix} V_s
\tag{10}
$$

1.4 Summary

This chapter introduced the key background and concepts of the LG modeling approach with an example of the derivation of the state-space model presented for an electromechanical system using LG theory.

In Chapter 2, a literature review shall be presented on recent research related to two graph based modeling methods, LG theory and BG theory, along with a comparison of these two approaches. Justification as to why the LG approach was chosen for this work will also be presented.

Chapter 2. Literature Review

2.1 Introduction

The literature review presented in this chapter will highlight applications and advancements of the LG Theory approach, as well as, the closely related BG Theory approach, for the modeling of dynamic systems encompassing various energy domains. Due to the apparent lack of recent literature related to LG theory, a look into the recent developments of BG theory was seen as a good indication of possible avenues for pursuing research in the field of LGs. Included with the review of BG related literature will be an example of this approach on a system mirroring the example provided in Chapter 1 of this thesis. A comparison of these two approaches will be provided, along with a justification as to why the LG modeling approach was chosen as the basis for the work presented in this thesis.

2.2 Review of Linear Graph Theory Literature

In [9], the authors developed an interactive software tool for dynamic system modeling which is based on the LG approach. This program was developed at MIT in the early 90s for their internal Unix-based engineering workstations as a means of assisting undergraduate students in learning LG modeling. Similarly, in the late 90s, the authors of [10] developed a Mathematica-based software tool for automatically constructing and evaluating LG models based on a questionnaire of user inputs. Beyond these papers, there is no evidence to suggest that these programs were further developed, maintained, or ever released publically.

The authors of the following papers present the application of LG theory for the modeling of multibody system dynamics. In [11], the author utilizes LG methods for representing the topology of a four-bar mechanism and for constructing topology matrices which are thus used for formulating the kinematic relationships for the system. In [12], the author and his associates extend the principles of the first paper by combining LG methods with symbolic programming in order to develop a Maple program called DynaFlex for real-time simulations of flexible multibody mechatronic systems. In [13], the authors incorporate a new pneumatic tire component, represented in LG form, into the DynaFlex program (now called DynaFlexPro) and use this new component in the simulation of the dynamics of a heavy-duty articulated vehicle called the Timberjack. The resulting simulation is then validated through a comparison with the ADAMS commercial software for a simulation of the same vehicle, where agreement between the two simulations is observed.

The authors of the following set of papers demonstrate how the LG method can be employed for the extraction of state-space models for thermohydraulic systems. The goal of this series of papers is to derive a methodology for determining the state-space models of thermohydraulic systems through the use of LG theory for the purpose of system control. In [14], the authors demonstrate how the LG method can be applied to systems in this domain and describe the advantages of this approach as producing models which are tangible and transparent with the resulting state-space representations being automatically reduced to contain only independent state variables (which requires additional work with other methods). In [15], the authors validate their methodology by modeling a Brayton cycle-based power conversion unit of a pebble bed modular reactor in order to study how the method can handle more complex thermohydraulic systems. When the extracted state-space model is compared to other previously validated simulation packages, the results demonstrate that the state-space model closely resembles the outcome of the simulated models.

In [16], the authors introduce an approach to developing LG models and associated governing equations for electrochemical systems. Models for a nickel-metal hydride battery in both the chemical and thermal domains are presented and the governing equations are derived. The developed model is then simulated and compared with other simulated and experimental results. The results of this comparison validate the accuracy of the LG model with this method shown to simulate 30% faster when compared to the previous simulation models due to the equations being greatly reduced. The authors of this paper note that future applications of this modeling technique can be applied to electric car batteries and their interactions with components of various domains within a vehicle.

The authors of [17] expand the use of LG modeling in applications related to hydrodynamic machines, such as propellers, impellers, and axial turbines. In this article, the authors use the LG method to represent the topology of such systems, as well as, the physics of the fluid and machine interaction in the form of the governing equations for both axial and radial flow devices. These models are then applied to the impeller of an automotive torque converter, with the results of this model being validated against experimental results of a real torque converter for different modes of operation.

In [18], the author presents paradigms for facilitating LG modeling of mechatronic systems. In this paper, the methods and procedures for constructing LG models are applied to mechatronic systems and numerous examples of system models are provided.

The authors of [7] present a novel method of physically modelling piezoelectric transducers and actuators as part of mechatronic systems. This article presents LG model structures for a variety of standard piezoelectric components, which account for both the energy storage and dissipation properties of the material. These predetermined model structures can thus be adapted to suit the needs of the user in numerous applications such as piezoelectric accelerometers, stack actuators, hybrid actuators, and transducers.

The authors of [19] proposed a combined LG and GA approach to the modeling and optimization of a 3-degree-of-freedom manipulator robot. In the paper, the authors develop the state-space model of the robot using the LG approach, then apply a GA to the various component parameters of the robot in order to obtain the desired system performance. The results of this work demonstrate that this combined approach is an adequate method of conducting model optimization. Similarly, the authors of [20] apply the LG approach for determining the dynamic model of a parallel manipulator robot.

In [21], the authors apply the LG modeling technique to a multi-domain mechatronics system in order to facilitate the design evolution of the system. The system in question is an automated fish cutting machine called the Intelligent Iron Butcher, which utilizes sensors and actuators spanning various energy domains in order to process fish with minimal waste. The LG model of the system is presented and the state-space model is derived. This model is then converted into a Simscape model and subsequently evolved and optimized through the use of GP in order to obtain a design solution which meets the performance requirements of the system.

2.3 Review of Bond Graph Theory Literature

2.3.1 History of Bond Graph Theory

BG Modeling was formally introduced by Henry M. Paynter in 1961, with his book entitled *Analysis and Design of Engineering Systems* [22], which was comprised of a collection of lectures which he first presented to his Systems Engineering students at MIT. Although Paynter published and presented various papers and conferences relating to his work in BG theory, his method would not be widely recognized or researched until decades later when his students would adopt and expand his theory, thus beginning the vast research into the topic [23].

During the 1960s and 1970s, two of Paynter's students, Dean Karnopp and Roland Rosenberg, would write numerous publications in which they would apply Paynter's theory to various engineering applications. A brief summary of some of these publications as well as their contributions to the advancement of the theory are provided.

In 1968, Karnopp and Rosenberg published a book entitled *Analysis and Simulation of Multiport Systems* [24] which described the processes relating to the systematic analysis of BGs and is credited with introducing these concepts to the application of computer simulation. A paper published by Karnopp and Rosenberg in 1970 shows the application of BG theory to vehicle dynamics through the modeling and subsequent simulation of a vehicle driveline [25]. This paper demonstrates the modeling of such an application, and the technique for obtaining the state-space equations from the BG model for the purpose of computer simulation. A paper published by Karnopp and Rosenberg in 1972, entitled *Definition of the BG Language* provides formal definitions for the key components of a BG model in general terms, with the goal of this paper being to provide a standardized basis for which all BG modeling shall be constructed [26]. In 1973, a paper by Karnopp applied BG theory into the fluid dynamics energy domain [27]. This paper demonstrated that although a fluid system can be described using BG techniques, there exist some unfavorable non-linear behaviors which are rarely encountered with systems of other energy domains. In 1975, Rosenberg published a paper which details how BGs can be used to develop a unified database of models for engineering systems in the form of multi-port engineering elements, which is a useful tool for evaluating large-scale, non-linear, multi-domain systems [28].

Though Paynter himself credits his students with popularizing and further developing BG theory [23], other researchers have also made valuable contributions to the field at this time. Examples of such include Auslander, Tsai, and Farazian who applied BG theory to the modeling and analysis of the dynamic behaviors of torsional energy transmission systems [29]; Auslander, Lobdell, and Chong who modeled and simulated the human cardiovascular system using delay-BG modeling [30]; Ort and Martens who devised a procedure for converting BGs into LGs [31]; and many others.

2.3.2 Recent Research in Bond Graph Theory

More recently, research into the uses and benefits of this theory continues to emerge, such as, with the ever-growing interest in fields such as mechatronics, robotics, and automation. Examples of publications which demonstrate the growing connection between the mechatronics field and BG theory are [32, 33, 34, 35], and numerous others. Each of these examples provides details and methodologies on how BG theory can be applied to the modeling, simulation, and design of mechatronics systems, and describe the benefits of the multi-domain nature of BGs on representing these complex systems. These papers also mention and make use of software packages, such as 20-Sim and Simulink, for the purpose of evaluating BG models.

One example of an application of BGs in the field of mechatronics and aerospace technology is the design and optimization of an intelligent controller for a satellite controlled by magnetic actuators [36]. Magnetic actuators are electromagnets which produce a mechanical torque on the satellite through interaction with the Earth's magnetic field, allowing for the satellite to be positioned at the desired angle. Through the use of BG modeling, a model of a satellite consisting of three magnetic actuators was constructed and the state-space equations were derived in order to facilitate the creation of an intelligent control system which the authors to be able to direct and stabilize a small satellite. While the authors note that the use of BGs in the creation of the control system was beneficial, there were draw backs to the use of the magnetic actuators which had to be accounted for and overcome.

In the field of mobile robotics, the authors of [37] demonstrate how to apply the BG modeling approach to a four-wheel skid steer mobile robotic vehicle. The purpose of this application was to describe the dynamics of the complex mechatronic, multi-domain robotic vehicle and to validate the accuracy of the model through simulation against the traditional Newton-Euler method. In this paper, the authors construct a unified, multi-domain BG model comprising the entire mobile vehicle and present the corresponding MATLAB Simulink simulation results. An analysis of the results between the BG model and Newton-Euler method simulations show that for the robot travelling in a simple straight line or circular path, a similar accuracy is observed between both methods, thus validating the effective implementation of the BG approach on the mobile robotic system. Similar research is conducted by the same authors in their paper modelling the dynamics of a three-wheeled omnidirectional mobile robot in [38].

The authors of [39] present a novel design for a mobile robot which is able to navigate over obstacles while reducing the tilt angle of the robot through the use of a pose deformation system. The proposed pose deformation system, which utilizes push rod mechanisms to adjust the position of moveable wheels, was analyzed using BG modeling and simulated in Simulink in order to evaluate the response of the design to a step disturbance and validate the robustness of the system under varying failure modes. In this paper, the authors explain that future work on this topic can comprise replacing the PID controller with an intelligent control system that can be adaptable and react to various types of obstacles and failure modes, as well as, re-evaluation of the design in three-dimensional space.

In [40] and [41], the authors relate BG theory to applications with autonomous vehicles in order to make use of the theory's behavioural, structural, and causal properties to measure uncertainties, conduct system diagnosis, and to determine the feasibility of system recovery. Since autonomous vehicle dynamics rely on many actuators, sensors, and plants which are susceptible to faults, it is essential that a system is in place to diagnose these faults and determine whether the vehicle will remain safely operable. In these papers, the authors propose a calculation-free algorithm based on BG models which perform the diagnosis of system faults and determines whether the system will be able to complete its objectives.

BG models have been applied to a variety of robotic leg mechanisms for the purposes of control and locomotion. In [42], the authors model the kinematics and dynamics of the Jansen leg, a 12-link planar mechanism, using a BG approach. This model provides a useful tool in the design process of such mechanisms as it is useful in determining the required motor torque for operating the mechanism, and allows for designing the associated control system. In [43], a two degree of freedom robotic leg is modeled using BGs which, when combined with fuzzy control logic and a dynamic sliding position control method, provides for smoother control of robot walking movement on uneven surfaces. Similarly, various papers including [44, 45, 46] demonstrate the use of BG models for the simulation, analysis, control, fault detection and tolerance control of a compliant legged quadruped robot.

The authors of [47] provide an example of the application of BG modeling to the field of bioengineering, specifically as a teaching method for the simulation of complex biotechnological processes. The authors utilize the modular nature of BGs to construct a library of biotechnological processes within the 20-Sim interactive software environment which students can use to investigate fields related to bioengineering. This paper demonstrates not only the versatility of BGs with regards to the domains in which it encompasses but also the benefits of the BG method's ability to create unified databases of models, similar to what Rosenberg describes in [28].

Similarly, in [48] the authors present a library of BG-based models for applications related to turbocharged diesel engines. These models can be adopted and adapted by users to suit their unique needs. The developed models were also validated through the creation of a specific diesel engine model using the developed model library and comparing the simulation results for this model against experimental data for the real engine.

In [49], a method of evolving systems through the use of BG theory combined with a hybrid GA and GP model exploration method is proposed. The paper states that the proposed hybrid method allows the GP to evolve the topology of the system, while the GA optimizes the parameters within the created topology. This method was applied to the automatic synthesis of electronic filters, as well as, the electrohydraulic actuator system of the Iron Butcher fish cutting machine mentioned in the LG literature review. A similar work, presented in [50], applies the BG and GP approach to the evolution of the Iron Butcher's mechanical mechanism model.

2.3.3 Fundamentals of Bond Graph Theory

Before the BG model can be drawn, the basic components of the BG approach must be defined. Instead of the network-like approach of drawing LG models, BG models more closely resemble block diagram models. The primary components of a BG model are as follows:

- Bonds, drawn as line segments consisting of:
    - a half arrowhead, signifying the positive power transfer direction
    - a perpendicular bar, signifying the direction of flow transfer between system elements
- Elements, drawn as the letters SE, SF, I, C, R, TF, and GY, which represent the physical components of the system in a generalized form:
    - SE/SF: single-port source elements for effort and flow to the system, respectively
    - I/C/R: generalized single-port elements for inertance, compliance, and resistance, respectively

  - TF/GY: generalized two-port elements for transformers and gyrators, respectively (similar to LG, TFs relate the input type to the same output type; whereas, GYs relate the input type to the opposite output type)
- Junctions, drawn as either 0 or 1, which describe how flow and effort information is equalized in the system
  - 0: multi-port effort equalizing junction, where only one bond can transfer effort into the junction

$$\begin{gathered} e_1 = e_2 = \cdots = e_n \\ f_1 = f_2 + \cdots + f_n \end{gathered} \tag{11}$$

  - 1: multi-port flow equalizing junction, where only one bond can transfer flow into the junction

$$\begin{gathered} e_1 = e_2 + \cdots + e_n \\ f_1 = f_2 = \cdots = f_n \end{gathered} \tag{12}$$

A summary of the effort and flow, and elements types for each of the primary energy domains can be seen in the following table:

Table 2-1: Summary of Effort and Flow, and Element Types for the Primary Energy Domains

Energy Domain	Source Elements		Storage Elements		Dissipating Elements
	Effort	Flow	Inertance	Compliance	Resistance
Electrical	Voltage	Current	Inductor	Capacitor	Resistor
Mech. Translational	Force	Velocity	Mass	Spring	Damper
Mech. Rotational	Torque	Angular Velocity	Inertia	Spring	Damper
Hydraulic/Fluid	Pressure	Volume Flow Rate	Inertance	Capacitance	Resistance
Thermal	Temperature	Heat Flow Rate	--	Capacitance	Resistance

2.3.4 Constructing a Bond Graph Model

The process for drawing a BG model is more abstract than the process for drawing an LG model, and differs between various domains. The general process is as follows: start with identifying the energy domain(s) of the system, the input(s) to the system, and the key elements of the system. Then, identify the points of the system which have a distinct flow (i.e. mass objects or zero velocity boundaries of a mechanical system, or at the location of components for electrical systems) and draw "1" junctions at these locations. Next, identify the points of the system which have a distinct effort (i.e. spring and/or damper locations for mechanical systems, nodes for electrical systems) and draw "0" junctions at these locations. The effort and flow source elements can then be added to the system at the locations for which they act. The I, R, C, TF, and GY elements can then be attached to the junctions for which they act upon/between as well. Simplify the model by combining any similar junctions attached to one another and eliminate any junctions with less than 3 bonds. Finally, the power transfer and flow directions can then be determined by adding the half arrow heads (generally pointing away from source elements, and towards passive elements) and bars (generally on the side of the bond coming from I and SF elements) to the bonds of the model. The position of the bar on the bond defines the causality of the bond, and is used to determine the independent and dependent energy storage elements of the system (similarly to the normal tree approach of LG).

2.3.5 Example using Bond Graph Modeling Approach

Applying this approach to the same electromechanical system which was evaluated with the LG method in Chapter 1 (as shown in Figure 1-3), the following BG model of the system can be produced:

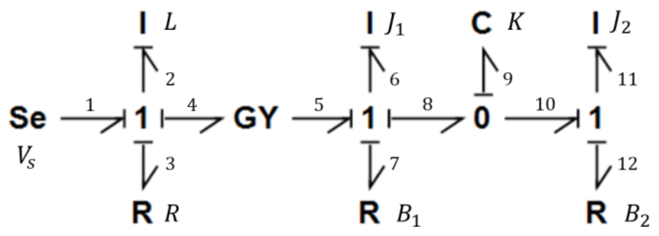


Figure 2-1:BG Model of the DC Motor Electromechanical System

The model can thus be converted into a state-space model through the following steps:

Step 1: Identify the state variables. There is one state variable for each independent energy storage element:

For the independent I and C elements, the state variables are the generalized momentum (p) and the generalized displacement (q), respectively. Therefore, the state variables of this system are:

$$x = [p_2 \quad p_6 \quad q_9 \quad p_{11}]^T \tag{13}$$

Step 2: Determine what information (effort or flow) each element gives to the system, expressing it in terms of the state and input variables:

The source element SE gives effort in the form of voltage to the system. Therefore:

$$e_1 = V_s \tag{14}$$

For each of the I elements, because they all have their preferred causality (independent), they all provide flow to the system. Therefore:

$$\begin{aligned} f_2 &= \frac{p_2}{L} \\ f_6 &= \frac{p_6}{J_1} \\ f_{11} &= \frac{p_{11}}{J_2} \end{aligned} \tag{15}$$

For the C element, because it has it's preferred causality (independent), it provides effort to the system. Therefore:

$$e_9 = K \cdot q_9 \tag{16}$$

For the R and GY elements, the information being provided to the system is dependent on the respective causality of each element. These equations must then be written using the other known equations so that they are expressed only in terms of the state- and input-variables. Therefore, for the R element:

$$e_3 = R \cdot f_3 \tag{17}$$

Since bond 3 connects the R element to a flow equalizing junction with bond 2 of the L element, the equation can be rewritten as follows:

$$e_3 = R \cdot f_2 = R \cdot \frac{p_2}{L} \tag{18}$$

For the B_1 element:

$$e_7 = B_1 \cdot f_7 \tag{19}$$

Similarly, since B_1 is connected to J_1 through a flow equalizing junction with bonds 6 and 7, the equation can be rewritten as follows:

$$e_7 = B_1 \cdot f_6 = B_1 \cdot \frac{p_6}{J_1} \tag{20}$$

For the B_2 element:

$$e_{12} = B_2 \cdot f_{12} \tag{21}$$

Again, since B_2 is connected to J_2 through a flow equalizing junction with bonds 11 and 12, the equation can be rewritten as follows:

$$e_{12} = B_2 \cdot f_{11} = B_2 \cdot \frac{p_{11}}{J_2} \tag{22}$$

For the GY element:

$$\begin{aligned} e_4 &= GY \cdot f_5 \\ e_5 &= GY \cdot f_4 \end{aligned} \tag{23}$$

Finally, since GY is connected to independent storage elements on either side by flow equalizing junctions, these equations can be rewritten as follows:

$$\begin{aligned} e_4 &= GY \cdot f_6 = GY \cdot \frac{p_6}{J_1} \\ e_5 &= GY \cdot f_2 = GY \cdot \frac{p_2}{L} \end{aligned} \tag{24}$$

Step 3: Determine what information (effort or flow) each independent energy storage element receives from the system, expressing it in terms of the state and input variables:

For element I:

$$\begin{aligned} \dot{p_2} &= e_2 = e_1 - e_3 - e_4 \\ &= V_s - R \cdot \frac{p_2}{L} - GY \cdot \frac{p_6}{J_1} \end{aligned} \tag{25}$$

For element J_1:

$$\begin{aligned} \dot{p_6} &= e_6 = e_5 - e_7 - e_8 = e_5 - e_7 - e_9 \\ &= GY \cdot \frac{p_2}{L} - B_1 \cdot \frac{p_6}{J_1} - K \cdot q_9 \end{aligned} \tag{26}$$

For element K:

$$\begin{aligned} \dot{q_9} = f_9 = f_8 - f_{10} = f_6 - f_{11} \\ = \frac{p_6}{J_1} - \frac{p_{11}}{J_1} \end{aligned} \tag{27}$$

For element J_2:

$$\begin{aligned} \dot{p_{11}} = e_{11} = e_{10} - e_{12} = e_9 - e_{12} \\ = K \cdot q_9 - B_2 \cdot \frac{p_{11}}{J_2} \end{aligned} \tag{28}$$

Step 4: Express the obtained equations in the standard state-space form:

$$\begin{bmatrix} \dot{p_2} \\ \dot{p_6} \\ \dot{q_9} \\ \dot{p_{11}} \end{bmatrix} = \begin{bmatrix} -\frac{R}{L} & \frac{GY}{J_1} & 0 & 0 \\ \frac{GY}{L} & -\frac{B_1}{J_1} & -K & 0 \\ 0 & \frac{1}{J_1} & 0 & -\frac{1}{J_1} \\ 0 & 0 & K & -\frac{B_2}{J_2} \end{bmatrix} \begin{bmatrix} p_2 \\ p_6 \\ q_9 \\ p_{11} \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} [V_s] \tag{29}$$

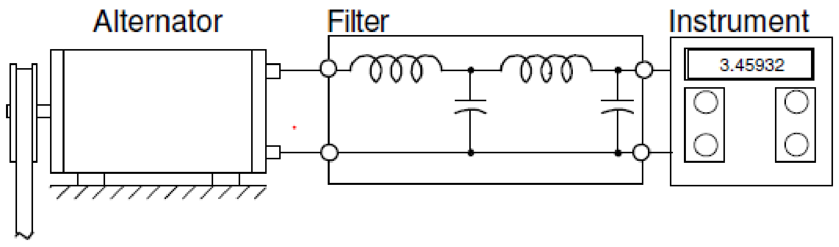
2.4 Comparison of Linear Graph and Bond Graph Modeling Approaches

While the LG and BG modeling approaches are both adequate methods of modeling dynamic systems (as can be seen from the results of the previous section and Section 1.3.4), a few key benefits and drawbacks between these approaches can be observed.

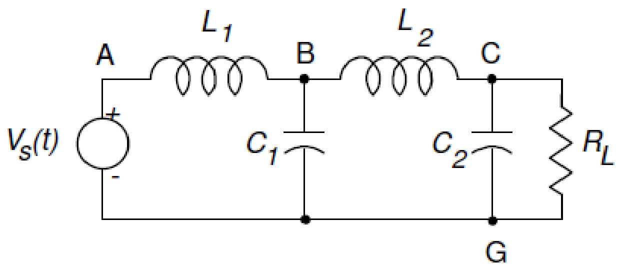
2.4.1 Graphical Representation

Firstly, the methods of graphically representing the system in each approach differ greatly. While LG modeling takes on a more traditional network representation, reminiscent of that used by Euler at the inception of graph theory, BG modeling uses a unique graphical representation resembling a block diagram. A benefit of the LG representation is that while modeling systems in the electrical and hydraulic/fluid domains, the topological layout of the system and the LG model will be nearly identical.

The following figure is an example of a system containing a signal filter between the voltage output of the alternator and the input to a measurement instrument.



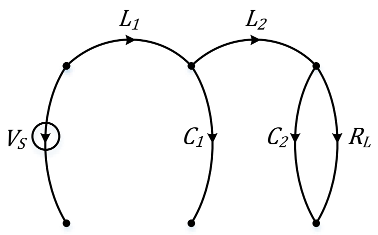
(a)



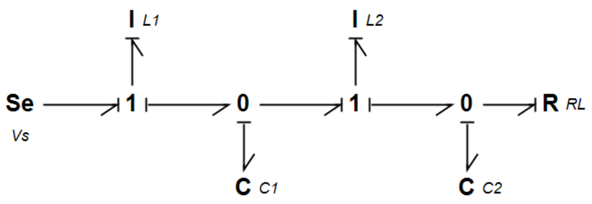
(b)

Figure 2-2: Example of the (a) Physical System Model and (b) Schematic Model of a Filter Circuit [4]

The corresponding LG and BG representations are shown in the figure below.



(a)



(b)

Figure 2-3: Equivalent (a) LG Model and (b) BG Model of the Filter Circuit

As can be observed by this example, the topology of the LG representation more closely resembles the system's schematic model; whereas, for the BG model, while many of the system components are located in similar locations as the schematic model, the overall appearance of the model and the system differs greatly.

This characteristic of resemblance to the physical system for LG models extends somewhat to the other domains as well, making the process of constructing and understanding LG models much more intuitive.

The following figure is an example of a mechanical translational system consisting of a velocity source which provides input energy to a spring attached to a mass element, then to a spring-damper combination to a second mass element which is attached to ground through another damper:

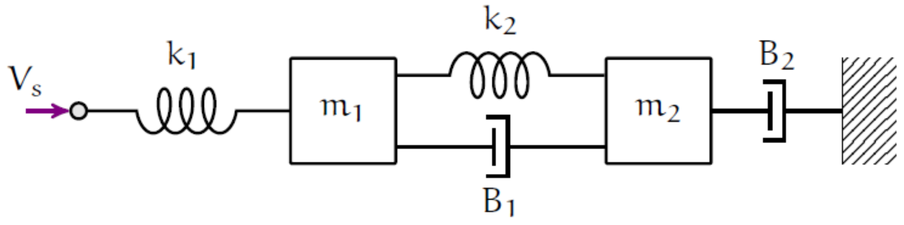
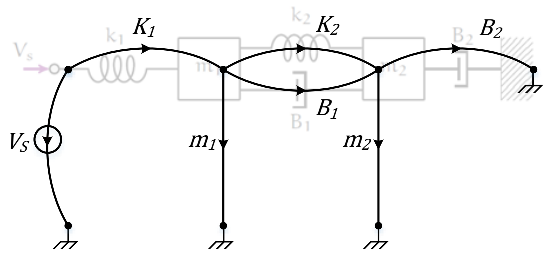
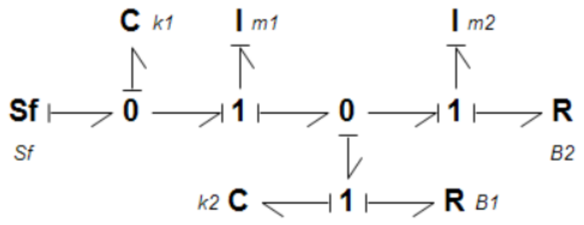


Figure 2-4:Schematic Model of a Mass-Spring-Damper System [6]

The corresponding LG and BG representations are shown in the figure below.



(a)



(b)

Figure 2-5: Equivalent (a) LG Model and (b) BG Model of the Mass-Spring-Damper System

Again, the topology of the LG representation more closely resembles the system's schematic model; whereas the BG model is a more abstract representation of the system. While both techniques take some experience to recognize by sight, the LG model provides a much more intuitive representation of the system than the BG model.

Building upon this difference in visual representation, a benefit of using of the electrical network-like layout of the LG approach is that it allows for the analogous application of electrical node and loop equations (Kirchhoff's Laws) to systems consisting of multiple energy domains. BG modeling, on the other hand, uses a more abstract representation of the system, which can result in a less intuitive method of producing and evaluating the graphical model.

2.4.2 Multi-Domain Analogies and System Variables

Secondly, a key difference between the two approaches is the main variable and element types which are utilized by each method. The LG approach uses the concepts and variables of the across and through analogy, whereas the BG approach uses the concepts and variables of the effort and flow analogy. This difference means that while the LG approach represents mass and spring elements as analogous to capacitor and inductor elements, respectively, the BG approach represents mass elements as inductors and spring elements as capacitors instead. Furthermore, the BG approach makes use of the generalized momentum (p) and displacement (q) variables as the state-variables when producing the state-space model; by contrast, the LG approach utilizes the across- and through-variables as the state variables. This is another key advantage of LG modeling, as the across- and through variables of the system are more intuitive to understand physically compared to the abstract concepts of the generalized momentum and displacements (i.e. current vs. charge and voltage vs. flux). This not only makes the LG produced state-space model easier to understand, but also makes it easier to manually produce the $\boldsymbol{C}$ and $\boldsymbol{D}$ matrices of the output equation of the state-space model without requiring to account for the abstract state-variable terms.

2.4.3 Recent Research and Software Availability

The final notable difference observed is the differing amounts of available research and published literature related to each of the two methods. While the literature reviews provided in Sections 2.2 and 2.3 are not exhaustive for either approach, it does demonstrate the clear preference researchers have for the BG modeling method. This is further highlighted by the fact that since this research began in May of 2018, a number of publications relating to the application of BG modeling in a variety of fields have been produced [51, 52, 53]; whereas, papers on LG theory (besides the work presented in this thesis) could not be found upon review. The likely cause of this discrepancy in publications between LGs and BGs is believed to be due to the lack of commercial or openly available software tools for the automatic evaluation of LG models. On the other hand, there are a number of software tools based on the BG modeling approach, such as 20-sim, SYMBOLS, and the BG add-on block library for Simulink called BG V.2.1 which have been utilized in many of the reviewed publications. This lack of a software tool means that researchers wishing to make use of the LG approach are required to either solve their systems manually, or as seen in [21], convert their models into other forms for automatic evaluation. This makes the LG approach unappealing to researchers wishing to perform complex simulations on large multi-domain systems because often, manual formulation of these systems can be time-consuming or impractical.

2.5 Summary

The literature presented in this chapter demonstrates that there is a need for further research and advancement in the field of LG theory when compared to the similar BG modeling technique. While the presented literature does highlight some interesting applications of the LG approach to a variety of unique system types and energy-domains, the frequency of publications related to this approach for the analysis of complex mechatronic systems is significantly less than that of the BG approach.

While the seeming lack of interest in this field of study by researchers is a drawback of the LG approach, a comparison of the two methods presented in this chapter shows that the use of LG theory for the modeling of dynamic systems has some significant benefits over the BG modeling approach. These benefits include the close resemblance which LG models have to their respective systems, and the intuitive nature in which these models can be constructed when compared to the BG approach. Similarly, the network-like representation of the LG method facilitates the analogous application of familiar node and loop equations commonly used in circuit analysis to systems outside of the electrical energy-domain. Finally, the variables used as a result of the across and through analogy of the LG approach results in an easily understood state-space model consisting of common state-variable types, as opposed to the generalized displacements and momentums used in BG modeling.

Despite the key benefits demonstrated for LGs over BGs, the question remains as to why the BG approach is the more popular modeling technique. One possible reason, identified by the analysis of the presented literature, is that many different software packages are available for automating the direct analysis of BG models; whereas, there are currently no available software packages for the direct analysis of LG models. While software packages have existed in the past, their current lack of availability is a clear gap in LG theory research. With the development of such a software tool, researchers will be more eager to employ the LG approach for advanced applications such as automated design evolution of engineering systems, as well as analysis of larger, more complex mechatronic systems.

Due to the demonstrated advantages of the LG approach over BG modeling, and the clear gaps in available software tools available for this method, the research presented in this thesis will be focused on the development of an automatic LG-based software tool written in the MATLAB programming language. In Chapter 3, the design, development, and validation of this toolbox will be presented, along with various examples of using this tool to derive the state-space model of dynamic systems.

Table 2-2: Literature Review Summary

Literature Review	Descriptions and Gaps in Research
2.2 Review of Linear Graph Theory Literature	• Lack of software tools for direct research into LG approach • General lack of research into applications of LG method • Integration of LG method with GP and GA to automated system design evolution (currently requires conversion of LG to Simscape)
2.3 Review of Bond Graph Theory Literature	• Very popular method of modeling for a variety of energy-domains and mechatronic systems • Many software packages available for evaluation of BG models
2.4 Comparison of Linear Graph and Bond Graph Modeling Approaches	• Publications on LG method are infrequent compared to BG method • Key benefits of LG method over BG method highlighted • Popularity of BG related to wide availability of related software tools

Chapter 3. Development of a Linear Graph-Based MATLAB Toolbox

3.1 Introduction

This chapter presents the development of a custom software toolbox, called LGtheory, which provides a robust and automated method for evaluating LG models of multi-domain engineering systems within the MATLAB programming environment. The need for such a toolbox is required because, while the LG approach is 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. Additionally, there are currently no software packages available that directly evaluate LG models; thus, the requirement for such a software package to fill this gap in research.

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 [9]; DynaFlexPro, developed at the University of Waterloo in the mid 2000s [13, 12]; and LG2ss, developed at the University of British Columbia in the 2010s [54]. Unfortunately, the lgraph software was only available on MIT workstation computers at the time, 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 some of the underlying technology is based on the LG approach, the program does not directly evaluate systems represented 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 chapter is to provide a tool for the automated evaluation of LG models, particularly facilitating education and research. The functionality and capabilities of the LGtheory toolbox are demonstrated in this chapter through the step-by-step evaluation of an electromechanical system consisting of a DC motor with an inertial load. The outputs of the state-space model produced by the LGtheory toolbox are simulated and compared against an equivalent model constructed in Simulink Simscape. The comparison of these results demonstrates that LGtheory is a fully capable program that produces accurate state-space models of multi-domain dynamic systems.

3.2 Inputting a Linear Graph Model into the LGtheory Toolbox

3.2.1 LGtheory Toolbox Inputs

In order to properly use the LGtheory MATLAB toolbox, users must understand the requirements for converting LG models into the acceptable inputs of the toolbox. The main function of the toolbox, which converts the input LG model into the corresponding state-space representation, is:

```
[Model] = LGtheory(LG);
```

Where the input to the function (`LG`) is a structure array containing the following fields:

`S` – Source Vector:

> A vector which defines the starting (reference) nodes of each system element (represented as directed branches) in a column-wise manner.

`T` – Target Vector:

> A vector which defines the end nodes (points of action) of each system element (represented as directed branches) in a column-wise manner.

`Type` – Type Vector:

A vector which defines the element type of each branch of the LG model in a column-wise manner. The element types are specified using indexing values based on Table 3-1:

`Domain` – Domain Vector:

A vector which defines the energy domain of each system element in a column-wise manner. These values ensure that the parameters of system elements are correctly accounted for (specifically in the case of spring elements in the mechanical domains), and ensures that the appropriate variable types are applied to each element. The energy domains are specified using indexing values based on Table 3-1:

Table 3-1: Index Values of Element Types and Energy Domains for LGtheory Toolbox Inputs

Index	Element Type	Index	Energy Domain
1	Across-Variable Source	0	Generalized
2	A-Type Element	1	Electrical
3	Transformer	2	Mechanical Translational
4	Gyrator	3	Mechanical Rotational
5	D-Type Element	4	Hydraulic/Fluid
6	T-Type Element	5	Thermal
7	Through-Variable Source		

`Var_Names` – Variable Names Vector:

A vector which defines the variable names of each system element in a column-wise manner. The variable name inputs must be symbolic variables or functions defined using either the `sym` or `syms` commands in MATLAB. These variable names will be the system parameters used to symbolically represent the matrices of the state-space model.

`y` – Output Vector:

A vector which defines the desired output variables of the state-space model. The output variable names must be symbolic variables defined using either the `sym` or `syms` commands in MATLAB. The symbolic variables must be created based on the desired variable(s) (across or through) and the energy domain(s) of the system element(s) using the formats described in Table 3-2.

Table 3-2: Format for Defining State-Space Outputs to LGtheory for each Energy Domain and Variable Type

Energy Domain	Across-Variable	Through-Variable
Generalized	`f_<name>(t)`	`v_<name>(t)`
Electrical	`V_<name>(t)`	`i_<name>(t)`
Mechanical Translational	`v_<name>(t)`	`F_<name>(t)`
Mechanical Rotational	`Omega_<name>(t)`	`Tau_<name>(t)`
Hydraulic/Fluid	`P_<name>(t)`	`Qf_<name>(t)`
Thermal	`T_<name>(t)`	`Q_<name>(t)`

*Where <name> is replaced with the symbolic variable name for the element of interest

3.2.2 LGtheory Toolbox Outputs

Similar to the inputs, the output of the function to the user (`Model`) is a structure array containing the various information used to construct the state-space model through the LG approach, as well as, the vectors and matrices of the state-space model itself. The following fields are contained within the structure array:

`In` – Incidence Matrix:

A sparse matrix representation of the topology of an LG model using “1” and “-1” to represent the directionality of system elements and connection to system nodes, and zeros to represent the lack of connections between elements and nodes.

`Tree` – Normal Tree Matrix:

A matrix which represents the normal tree of the LG model in an incidence matrix form.

`Branches` – Branches Vector:

A vector which defines the column-wise indexes and element types of the normal tree branches with non-zero values, and the column-wise indexes of the co-tree links with zeros.

`CoTree` – Co-Tree Matrix:

A matrix which represents the co-tree of the LG model in an incidence matrix form.

`Links` – Links Vector:

A vector which defines the column-wise indexes and element types of the co-tree links with non-zero values, and the column-wise indexes of the normal tree branches with zeros.

`Across_Vars` – Across-Variables Vector:

A vector which defines the across-variables of all system elements in a column-wise manner using the naming convention defined in Table 3-2.

`Through_Vars` – Through-Variables Vector:

A vector which defines the through-variables of all system elements in a column-wise manner using the naming convention defined in Table 3-2.

`Prime` – Primary-Variables Vector:

A vector which defines the primary-variables of the system as the across-variables of normal tree branches and the through-variables of co-tree links. The primary variables are important to identify because they are the variables which determine the energy stored by each independent energy storage element within the system.

`Secon` – Secondary-Variables Vector:

A vector which defines the secondary-variables of the system as the through-variables of normal tree branches and the across-variables of co-tree links. These variables have no impact on the energy storage of the independent system elements.

`Params` – State-Space Parameters Vector:

A vector which defines the parameters of each system element in a column-wise manner based on the variable name defined by the user. While similar to the Var_Names vector, this vector accounts for the requirement to inverse the spring constants of the mechanical domains, and the resistive parameters in some cases.

`x` – State Vector:

A column vector containing the state-variables of the state-space model, defined as the across-variables of A-Type elements in the normal tree and the through-variables of T-Type elements in the co-tree.

`u` – Input Vector

A column vector containing the input-variables of the state-space model, as defined by the source elements of the LG model.

`elem_eqns` – Elemental/Constitutive Equations Vector:

A column vector containing the constitutive (elemental) equations of each passive element, defined as per Table 1-2.

`cont_eqns` – Continuity Equations Vector:

A column vector containing the continuity (node) equations for each passive branch of the normal tree, isolated for the secondary variable of that element.

`comp_eqns` – Compatibility Equations Vector:

A column vector containing the compatibility (loop) equations for each passive link of the normal tree, isolated for the secondary variable of that element.

`A` – State Matrix

An $n \times n$ matrix (where n is the order of the system) which defines how the current state-variables affect the rate of change of the future system states.

`B` – Input Matrix

An $n \times r$ matrix (where r is the number of system inputs) which defines how the current system input-variables affect the rate of change of the future system states.

`C` – Output Matrix

An $m \times n$ matrix (where m is the number of outputs) which defines how the current state-variables affect the outputs of the system.

`D` – Feedforward Matrix

An $m \times r$ matrix which defines how the current system input-variables affect the outputs of the system.

`E` – Input Derivative Matrix

An $n \times r$ matrix which defines how the derivative of input-variables affect the rate of change of the future system states. This matrix only occurs for some cases where the LG model contains dependent energy storage elements.

`F` – Output Derivative Matrix

An $m \times r$ matrix which defines how the derivative of input-variables affect the system outputs. This matrix only occurs for some cases where the LG model contains dependent energy storage elements.

3.2.3 Best Practices

While the LGtheory toolbox is robust with how it handles inputs, there are some best practices which should be employed by the user to ensure that they are inputting their LG model correctly.

Users are required to specify the ground node in the source and target vector input fields as a "1". This is required as MATLAB indexes arrays and matrices from one instead of zero, making much of the work done by the toolbox much simpler. Each node added to the system will be incremented in value by 1. If a value for a node is skipped, or the ground node is not specified as “1”, it will likely result in an error message or an incorrect result from the toolbox.

For systems that contain transducer elements (transformers or gyrators), the first and second ports of these two-port elements should be placed in successive order within the toolbox input fields without any other transducer port elements in between. This is because the toolbox couples and evaluates the successive transducer ports in the order in which they are listed in the inputs. Because of this, it is considered best practice to list the two ports of the same transducer element immediately next to each other in the LG inputs in order to ensure that they will be coupled and evaluated together correctly. While the successive ordering of transducer port elements is important, the order of the nodes for which they are attached are not required to be successive (i.e. port one can be connected to node 5 and ground, while port two is connected to node eight and ground).

Similarly, the variable names of the two ports of a transducer element should be differentiated from each other in the Var_Names input field. While this is not required for transducers spanning multiple energy domains because the variable types on either side will be different from one another, it is a requirement for single-domain transducers. Typically, this is done using an alphabetical differentiation of each port (i.e. $TF1a$ and $TF1b$), as transducer pairs are typically differentiated from one another numerically in systems containing multiple two-port elements.

When specifying the domain indexes of the LG model, it is important that the user defines these index values correctly for each element contained in the system as the across- and through-variables associated with each element will be determined based on these values.

Likewise, the parameters of specific system element types will be changed based on the requirements of each domain. If the user wishes to leave the domains in their generalized terms in order to evaluate systems outside of the primary domains supported by the toolbox (electrical, mechanical translational and translational, hydraulic/fluid, and thermal) it must keep in mind that the toolbox will be unable to automatically adjust these parameters, and that this must be done manually when the parameter values are being substituted into the state-space model.

Before the user converts their model into inputs for the toolbox, it must be ensured that the LG model is simplified sufficiently to ensure that there is no excess of state-variables. This is done by combining A-Type elements that are directly in series and T-Type elements that are directly in parallel. Similarly, the user must ensure that their model is complete with no loose elements (no nodes with only a single element attached). In either of these cases, the LGtheory toolbox will return an error message to the user.

The equations for converting direct series A-Type elements and direct parallel T-Type elements are:

$$C_e = \frac{1}{\sum_i \frac{1}{C_i}} \tag{30}$$

$$L_e = \frac{1}{\sum_i \frac{1}{L_i}} \tag{31}$$

3.2.4 Example of Inputting a Linear Graph Model to the LGtheory Toolbox

For this Chapter, the following system of a DC motor with an inertial load will be used as an example for demonstrating the processes of the LGtheory toolbox for deriving the state-space representation of LG models.

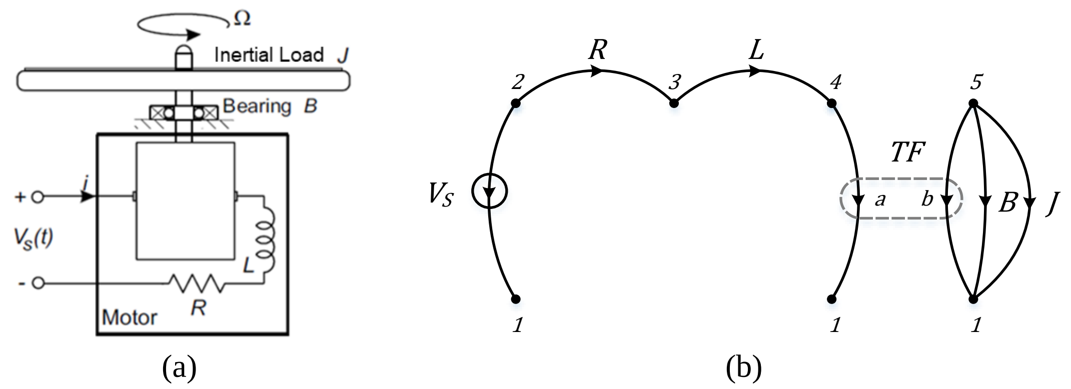


Figure 3-1(a) Schematic Diagram [5] and (b) LG model of DC Motor with Inertial Load System [5]

Based on the LG model of the system presented in Figure 3-1, the following inputs must be defined for the LGtheory MATLAB toolbox:

```
LG.S = [2 2 3 4 5 5 5];
LG.T = [1 3 4 1 1 1 1];
LG.Type = [1 5 6 3 3 5 2];
LG.Domain = [1 1 1 1 3 3 3];
syms s R L TFa TFb B J
LG.Var_Names = [s R L TFa TFb B J];
syms i_TFa(t) Tau_TFb(t) Omega_J(t)
LG.y = [i_TFa(t) Tau_TFb(t) Omega_J(t)];
[Model] = LGtheory(LG);
```

For this example, the outputs of the state-space model were specified as being the current of the electrical port of the DC motor (`i_TFa(t)`), the torque output of the motor on the mechanical side (`Tau_TFb(t)`), and the rotational velocity of the inertial load (`Omega_J(t)`).

3.3 LGtheory Toolbox Functions

This section describes the various processes and algorithms used by the sub-functions of the toolbox to convert the LG model representation inputted to the LGtheory function into the corresponding state-space model of the system.

3.3.1 Check Model Inputs

```
CheckModel(LG);
```

Before the LG model can be evaluated, the inputs must be checked to ensure that they meet the requirements of the LGtheory toolbox.

First, the toolbox will check to ensure that the model is complete (closed) and that no loose elements (open loops) exist within the model inputs. A loose element can be detected by examining the source and target vectors for any node values that only occur once between both vectors. If such a node is found, it is then known that the node has only a single element attached to it, thus meaning that the model contains an open loop.

Next, the toolbox will check for an excess of state-variables. This is done by checking the inputs for A-Type elements that are in direct series, or T-Type elements that are in direct parallel. If either case is found, an error message is displayed to the user informing them of the excess state variables.

If neither of these cases is detected, evaluation of the model inputs shall continue.

3.3.2 Conversion to Incidence Matrix Representation

```
[Model] = IncidenceMatrix(LG);
```

In order to mathematically represent the topology and directionality of 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 (Source node), and a “1” in the row corresponding to the node that the element is entering (Target node).

The toolbox constructs the incidence matrix of the LG model by initializing a zero matrix with as many rows as there are nodes (determined by the highest number in the source and target vectors) and as many columns are there are elements (determined by the length of the source and target vectors). The program then indexes through the columns of both vectors and the incidence matrix and assigns a “-1” or “1” in the incidence matrix row indexes corresponding to the source and target vectors, respectively.

From the LG model presented in Figure 3-1, and the source and target vectors specified in the example input to MATLAB, the following incidence matrix is produced for the DC motor with the inertial load system:

$$In = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccccc} V_s & R & L & TF_a & TF_b & B & J \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & -1 \end{bmatrix} \end{array} \tag{32}$$

3.3.3 Building the Normal Tree

```
    [Model] = BuildNormalTree(LG,Model);
```

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 the classification of the primary and secondary variables, as well as, for providing a systematic process of identifying independent and dependent energy storage elements.

The process for constructing the normal tree starts by creating an empty three-dimensional incidence matrix for the tree (and one for the co-tree) which is the same size as the incidence matrix of the LG model but with a depth of 2^T, where T is the number of transducer elements contained within the LG model. This depth is required for evaluating all possible normal tree configurations that can result due to the inclusion of transducers elements, where each page (or layer) of the three-dimensional matrix represents a different permutation of the normal tree.

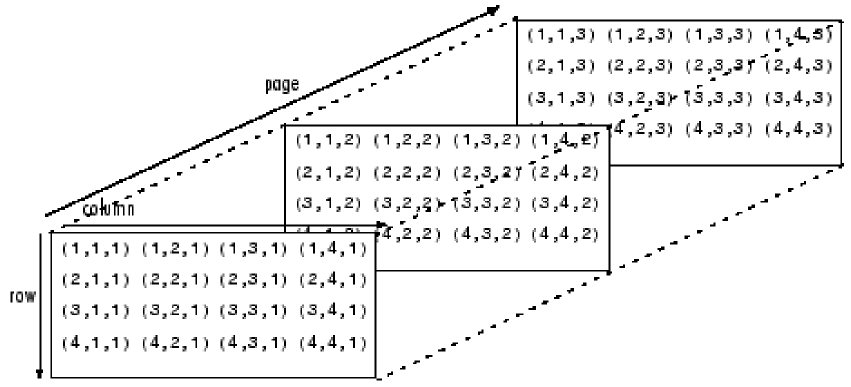


Figure 3-2: Example of Indexing Values of a Multidimensional Array in MATLAB [55]

Next, the program adds all across-variable source elements to each of the normal tree configurations. This is done for each element by adding the corresponding column of the LG model incidence matrix to the same column of the normal tree matrices.

Transducer elements are then added to the normal tree incidence matrices with only one transformer port, and either both or neither gyrator ports being included in the normal tree. This is done in a looping process which ensures that all possible combinations of transducer branches are created throughout the 2^T layers.

The algorithm then cycles through the passive elements in the order of A-Type, D-Type, and T-Type elements, respectively. For each column-wise element which is added to a layer of the normal tree matrix, a depth-first-search loop detecting algorithm is used to determine if the inclusion of the most recent element resulted in the creation of a loop in the normal tree. If no loop is detected, the element remains in the normal tree. If a loop is detected, the last element added to the normal tree is removed from that layer and is instead added to the corresponding layer of the co-tree matrix.

Finally, this process is attempted again for the through-variable source elements in every layer of the normal tree matrix. Since it is required that no through-variable source elements are included in the normal tree, the layer index of any permutation of the normal tree matrix which requires the addition of a through-variable source in order to be completed shall be flagged in a logical vector as an invalid normal tree configuration.

Once this process is complete for all elements of the LG model, each normal tree configuration is revaluated for any potential loops which may have been created as the result of the required inclusion of across-variable source and transducer port elements. Similarly, each configuration is evaluated to determine whether all nodes are contained within the normal tree matrix. Any configuration which is determined to violate either of these requirements is assigned a logical value for its layer index, denoting it as being invalid.

In order to make the final decision on which configuration will be selected as the normal tree, the algorithm finds the valid permutations which contain the most A-Type elements. In the case that, there are multiple layers that contain the most A-Type elements, the algorithm will then select the layer which contains the least T-Type elements as this combination of most A-Type and least T-Type elements ensure that the program identifies the most independent energy storage elements in the system.

The normal tree resulting from this process for the example system can be seen in Figure 3-3, where the solid lines represent branches and the dashed lines represent links.

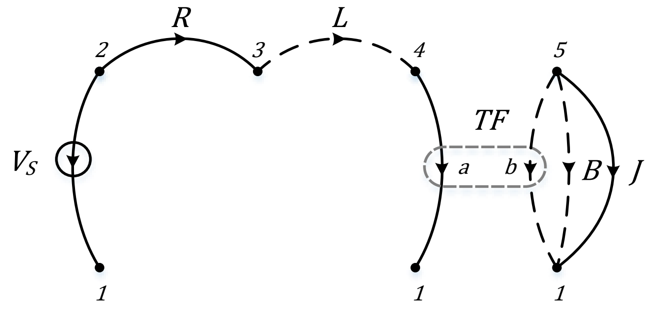


Figure 3-3: Normal Tree of the DC Motor with Inertial Load LG model

The resulting incidence matrix representation of the normal tree is:

$$Tree = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccccc} V_s & R & L & TF_a & TF_b & B & J \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \end{array} \quad (33)$$

The corresponding incidence matrix representation of the co-tree is:

$$CoTree = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccccc} V_s & R & L & TF_a & TF_b & B & J \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & 0 \end{bmatrix} \end{array} \quad (34)$$

3.3.4 Variable Classification

```
[Model] = ClassifyVariables(LG,Model);
```

Once the normal tree of the LG model has been identified, it can be utilized to assist in the process of variable classification. First, the toolbox creates two arrays that contain the across- and through-variables of all the system elements in symbolic form based on the formats specified in Table 3-2. For this example, these two arrays would be as follows:

$$Across\_Vars = [V_s \ V_R \ V_L \ V_{TF_a} \ \omega_{TF_b} \ \omega_B \ \omega_J] \tag{35}$$

$$Through\_Vars = [i_s \ i_R \ i_L \ i_{TF_a} \ \mathrm{T}_{TF_b} \ \mathrm{T}_B \ \mathrm{T}_J] \tag{36}$$

Similarly, the two vectors for the primary- and secondary-variables, where the primary-variables are classified as the across-variables of the branches and the through-variables of the links, and the secondary variables are classified as the through-variables of the branches and the across-variables of the links:

$$Prime = [V_s \ V_R \ i_L \ V_{TF_a} \ \mathrm{T}_{TF_b} \ \mathrm{T}_B \ \omega_J] \tag{37}$$

$$Secon = [i_s \ i_R \ V_L \ i_{TF_a} \ \omega_{TF_b} \ \omega_B \ \mathrm{T}_J] \tag{38}$$

Finally, the program then identifies the state-variables as a vector containing the across-variables of the A-type branches, and the through-variables of the T-type links, and the output-variables as a vector containing the source variable type of each source element:

$$\boldsymbol{x} = [\omega_J \ i_L]^T \tag{39}$$

$$\boldsymbol{u} = [V_s]^T \tag{40}$$

3.3.5 Constitutive Equations

```
[Model] = ElementalEquations(LG,Model);
```

The constitutive equations of the system are created for all passive elements. Referring to Table 1-2, the constitutive equations of each passive element can be formed. Once formed, the program rearranges each equation to isolate for the primary-variable (or its derivative) associated with that element.

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

$$\begin{gathered} \frac{d\omega_J}{dt} = \frac{1}{J}\,\mathrm{T}_J \\ \frac{di_L}{dt} = \frac{1}{L}\,V_L \\ V_R = R \cdot i_R \\ \mathrm{T}_B = B \cdot \omega_B \\ V_{TF_a} = TF \cdot \omega_{TF_b} \\ \mathrm{T}_{TF_b} = -TF \cdot i_{TF_a} \end{gathered} \tag{41}$$

3.3.6 Network Equations

```
[Model] = NetworkEquations(Model);
```

The continuity (node) equations of an LG model are formed using the contouring method. This method involves “cutting” around a node or set of nodes (as shown in Figure 3-4 for the contour of the M element) 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, where each equation is rearranged to isolate for the secondary variable of the passive branch.

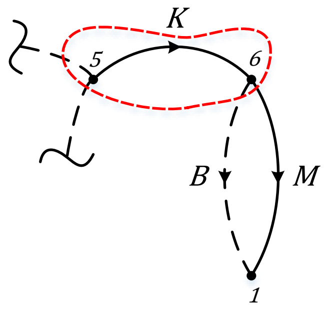


Figure 3-4: Example of Contour Enveloping Multiple Nodes

Similarly, the compatibility (loop) 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 formed from that element's inclusion. 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.

Figure 3-5 illustrates the contours of each passive normal tree branch (shown in red), and the re-inclusion of the passive co-tree links (shown in blue) to the LG model.

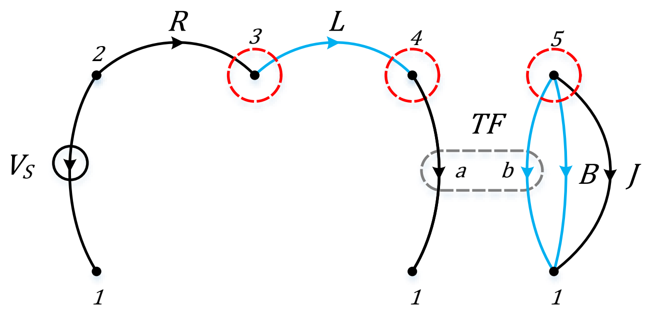


Figure 3-5: LG Model with Contour and Temporary Link Inclusion Overlay

While being able to identify and evaluate the loops and contours of this model is simple to accomplish manually for a human, having a computer program recognize these patterns can be much more difficult. So, in order to accomplish this, the concept of the fundamental cut set will be employed [14, 56].

This concept involves partitioning the LG model's incidence matrix into two sub-matrices of the normal tree and co-tree. The incidence matrices found from building the normal tree can be used, but the columns representing the elements that are not contained in either tree must be removed. Likewise, row one, representing the ground node of the model, will be removed from the matrices as well. This results in the following sub-matrices:

$$\boldsymbol{A_{tr}} = \begin{array}{c c} & \begin{array}{cccc} V_s & R & TF_a & J \end{array} \\ \begin{array}{c} 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} -1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \end{array} \tag{42}$$

$$\boldsymbol{A_{co}} = \begin{array}{c c} & \begin{array}{ccc} L & TF_b & B \end{array} \\ \begin{array}{c} 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & -1 & -1 \end{bmatrix} \end{array} \tag{43}$$

Next, the fundamental cut set matrix is found as:

$$\boldsymbol{F_c} = [\boldsymbol{H} \quad \boldsymbol{I}] \tag{44}$$

Where,

$$\boldsymbol{H} = \boldsymbol{A_{tr}^{-1} A_{co}} \tag{45}$$

And I is an identity matrix of corresponding size to H. For the example system, the fundamental cut set matrix will be:

$$\boldsymbol{F_c} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{46}$$

Next, the across- and through-variables of the LG model must be separated based on whether their corresponding element exists in the normal tree or co-tree. The following is the resulting vectors for the example system:

$$\boldsymbol{v_{tr}} = \begin{bmatrix} V_s \\ V_R \\ V_{TF_a} \\ \omega_J \end{bmatrix}, \quad \boldsymbol{v_{co}} = \begin{bmatrix} V_L \\ \omega_{TF_b} \\ \omega_B \end{bmatrix} \tag{47}$$

$$\boldsymbol{f_{tr}} = \begin{bmatrix} i_s \\ i_R \\ i_{TF_a} \\ \mathrm{T}_J \end{bmatrix}, \quad \boldsymbol{f_{co}} = \begin{bmatrix} i_L \\ \mathrm{T}_{TF_b} \\ \mathrm{T}_B \end{bmatrix} \tag{48}$$

The continuity equations for each passive branch (and, in this case, the across-variable source element) can subsequently be found as:

$$\boldsymbol{f_{tr}} = -\boldsymbol{H}\boldsymbol{f_{co}} \tag{49}$$

Which, for the example system would result in:

$$\begin{bmatrix} i_s \\ i_R \\ i_{TF_a} \\ \mathrm{T}_J \end{bmatrix} = -\begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} i_L \\ \mathrm{T}_{TF_b} \\ \mathrm{T}_B \end{bmatrix} \tag{50}$$

$$\begin{gathered} i_s = -i_L \\ i_R = i_L \\ i_{TF_a} = i_L \\ \mathrm{T}_J = -\mathrm{T}_{TF_b} - \mathrm{T}_B \end{gathered} \tag{51}$$

The compatibility equations for each passive link can subsequently be found as:

$$\boldsymbol{v_{co}} = \boldsymbol{H^T v_{tr}} \tag{52}$$

Which, for the example system would result in:

$$\begin{bmatrix} V_L \\ \omega_{TF_b} \\ \omega_B \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_s \\ V_R \\ V_{TF_a} \\ \omega_J \end{bmatrix} \tag{53}$$

$$\begin{gathered} V_L = V_s - V_R - V_{TF_a} \\ \omega_{TF_b} = \omega_J \\ \omega_B = \omega_J \end{gathered} \tag{54}$$

3.3.7 Creating the State-Space Matrices

```
    [Model] = StateSpaceMatrices(LG,Model);
```

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.

$$\frac{d\omega_J}{dt} = \frac{-\mathrm{T}_B - \mathrm{T}_{TF_b}}{J} \tag{55}$$

$$\frac{di_L}{dt} = \frac{-V_R - V_{TF_a} + V_s}{L} \tag{56}$$

$$V_R = R \cdot i_L \tag{57}$$

$$\mathrm{T}_B = B \cdot \omega_J \tag{58}$$

$$V_{TF_a} = TF_a \cdot \omega_J \tag{59}$$

$$\mathrm{T}_{TF_b} = -TF_a \cdot i_L \tag{60}$$

The newly created equations are then 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. For the example system, column vector $\boldsymbol{x}$ will consist of equations (55) and (56), column vector $\boldsymbol{d}$ will be empty as there are no dependent storage elements (as determined by the normal tree), and column vector $\boldsymbol{p}$ will consist of equations (57)-(60). These vectors can thus be written as the following matrix equations:

$$\dot{\boldsymbol{x}} = \boldsymbol{Px} + \boldsymbol{Qp} + \boldsymbol{Rd} + \boldsymbol{Su} \tag{61}$$

$$\boldsymbol{d} = \boldsymbol{M}\dot{\boldsymbol{x}} + \boldsymbol{N}\dot{\boldsymbol{u}} \tag{62}$$

$$\boldsymbol{p} = \boldsymbol{Hx} + \boldsymbol{Jp} + \boldsymbol{Kd} + \boldsymbol{Lu} \tag{63}$$

Therefore,

$$\dot{\boldsymbol{x}} = [0]_{2\times 2}\boldsymbol{x} + \begin{bmatrix} 0 & -\frac{1}{J} & 0 & -\frac{1}{J} \\ -\frac{1}{L} & 0 & -\frac{1}{L} & 0 \end{bmatrix}\boldsymbol{p} + [0]\boldsymbol{d} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix}\boldsymbol{u} \tag{64}$$

$$\boldsymbol{d} = [0]\dot{\boldsymbol{x}} + [0]\dot{\boldsymbol{u}} \tag{65}$$

$$\boldsymbol{p} = \begin{bmatrix} 0 & R \\ B & 0 \\ TF_a & 0 \\ 0 & -TF_a \end{bmatrix}\boldsymbol{x} + [0]_{4\times 4}\boldsymbol{p} + [0]\boldsymbol{d} + [0]_{4\times 1}\boldsymbol{u} \tag{66}$$

The general solution to the state-space equation is formed by isolating $\boldsymbol{d}$ in (62) and $\boldsymbol{p}$ in (63), and substituting the results into (61). Once simplified, this process results in the following general formulation of the state-space model:

$$\dot{\boldsymbol{x}} = \boldsymbol{Ax} + \boldsymbol{Bu} + \boldsymbol{E}\dot{\boldsymbol{u}} \tag{67}$$

Where,

$$\boldsymbol{A} = [\boldsymbol{I} - (\boldsymbol{QK'} + \boldsymbol{R})\boldsymbol{M}]^{-\boldsymbol{1}}(\boldsymbol{P} + \boldsymbol{QH'}) \tag{68}$$

$$\boldsymbol{B} = [\boldsymbol{I} - (\boldsymbol{QK'} + \boldsymbol{R})\boldsymbol{M}]^{-\mathbf{1}}(\boldsymbol{S} + \boldsymbol{QL'}) \tag{69}$$

$$\boldsymbol{E} = [\boldsymbol{I} - (\boldsymbol{QK'} + \boldsymbol{R})\boldsymbol{M}]^{-\mathbf{1}}(\boldsymbol{R} + \boldsymbol{QK'})\boldsymbol{N} \tag{70}$$

And,

$$\boldsymbol{K'} = [\boldsymbol{I} - \boldsymbol{J}]^{-\mathbf{1}}\boldsymbol{K} \tag{71}$$

$$\boldsymbol{H'} = [\boldsymbol{I} - \boldsymbol{J}]^{-\mathbf{1}}\boldsymbol{H} \tag{72}$$

$$\boldsymbol{L'} = [\boldsymbol{I} - \boldsymbol{J}]^{-\mathbf{1}}\boldsymbol{L} \tag{73}$$

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. $\boldsymbol{d} = 0$), 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:

$$\boldsymbol{A} = \boldsymbol{P} + \boldsymbol{QH'} \tag{74}$$

$$\boldsymbol{B} = \boldsymbol{S} + \boldsymbol{QL'} \tag{75}$$

2. If the system contains dependent energy storage elements (i.e. $\boldsymbol{d} \neq 0$) but contains no input derivatives (i.e. $\dot{\boldsymbol{u}} = 0$), 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 still calculated using (68) and (69), respectively.

As previously stated, the LGtheory toolbox program determines from the normal tree that the example system contains no dependent energy storage elements ($\boldsymbol{d} = 0$), meaning that this system falls into scenario 1, as described above. The MATLAB program subsequently extracts the necessary matrices and performs calculations for the state-space matrices using (74) and (75), to obtain:

$$\boldsymbol{A} = [0]_{2\times2} + \begin{bmatrix} 0 & -\frac{1}{J} & 0 & -\frac{1}{J} \\ -\frac{1}{L} & 0 & -\frac{1}{L} & 0 \end{bmatrix} [\boldsymbol{I}_{4\times4} - [0]_{4\times4}]^{-1} \begin{bmatrix} 0 & R \\ B & 0 \\ TF_a & 0 \\ 0 & -TF_a \end{bmatrix} \tag{76}$$

$$\boldsymbol{A} = \begin{bmatrix} -\frac{B}{J} & \frac{TF}{J} \\ -\frac{TF}{L} & -\frac{R}{L} \end{bmatrix} \tag{77}$$

And,

$$\boldsymbol{B} = \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} + \begin{bmatrix} 0 & -\frac{1}{J} & 0 & -\frac{1}{J} \\ -\frac{1}{L} & 0 & -\frac{1}{L} & 0 \end{bmatrix} [\boldsymbol{I}_{4\times4} - [0]_{4\times4}]^{-1} [0]_{4\times1} \tag{78}$$

$$\boldsymbol{B} = \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} \tag{79}$$

The output equations are then 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 LGtheory by examining the continuity and compatibility equations, as well as, the substituted constitutive equations from this section, and selecting equations that can be isolated for 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 $\boldsymbol{D}$, and potentially $\boldsymbol{F}$, matrices are thus extracted from these equations.

For the example system, the variables of interest were specified as the current supplied to the motor, the torque output by the motor, and the rotational velocity of the inertial load. For these output variables, the following $\boldsymbol{C}$ and $\boldsymbol{D}$ matrices are produced:

$$\boldsymbol{C} = \begin{bmatrix} 0 & 1 \\ 0 & -TF \\ 1 & 0 \end{bmatrix} \qquad \boldsymbol{D} = 0 \tag{80}$$

3.3.8 Standard State-Space Form Conversion

```
[Model] = StandardForm(Model);
```

In the case of some LG models, the state-space matrices produced by the toolbox will result in a non-standard form of the state-space model:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{Ax} + \boldsymbol{Bu} + \boldsymbol{E}\dot{\boldsymbol{u}} \\ \boldsymbol{y} &= \boldsymbol{Cx} + \boldsymbol{Du} + \boldsymbol{F}\dot{\boldsymbol{u}} \end{aligned} \tag{81}$$

These additional matrices, $\boldsymbol{E}$ and $\boldsymbol{F}$, represent the effects which the derivative of the input variables have on the rate of change of the state-variables and on the output-variables, respectively.

While the main LGtheory function of the toolbox will return all state-matrices in a non-standard form (assuming the input LG model results in the additional matrices), the `StandardForm` function is included in the LGtheory toolbox library in order to automate the conversion of non-standard state-space models to standard form.

This function works by transforming the state-variables of the system, as well as the output ($\boldsymbol{B}$) and feedforward matrices ($\boldsymbol{D}$), to a modified set which accounts for the derivative(s) of the input variable(s) and eliminates the $\boldsymbol{E}$ matrix:

$$\boldsymbol{x}' = \boldsymbol{x} - \boldsymbol{Eu} \tag{82}$$

$$\boldsymbol{B}' = \boldsymbol{AE} + \boldsymbol{B} \tag{83}$$

$$\boldsymbol{D}' = \boldsymbol{CE} + \boldsymbol{D} \tag{84}$$

The state-space model can thus be expressed in these terms:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x}' + \boldsymbol{B}'\boldsymbol{u} \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x}' + \boldsymbol{D}'\boldsymbol{u} + \boldsymbol{F}\dot{\boldsymbol{u}} \end{aligned} \tag{85}$$

3.4 Results and Discussion

In order to validate the results for the example system produced by the LGtheory toolbox, the state-space matrices output by the program was simulated in MATLAB using commands from the Control System Toolbox. The same electromechanical system 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 [6] and a step input of 12V for the voltage source to the motor. Figure 3-6 shows the Simscape representation of the evaluated system.

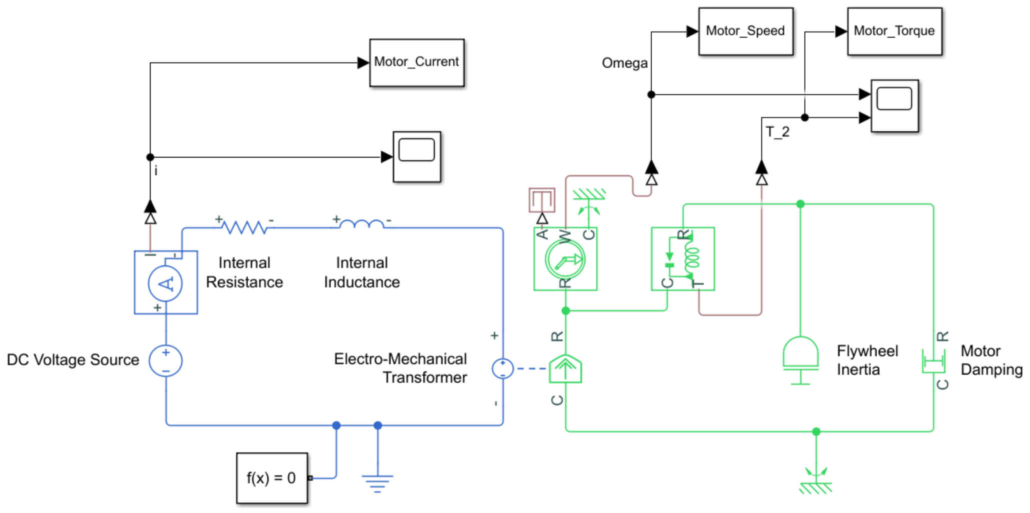


Figure 3-6: Simulink Simscape Model of a DC Motor with an Inertial Load

The results of the simulations for both the LGtheory produced state-space model and the Simulink Simscape model are presented in Figure 3-7 for the dynamic response of the specified system outputs. From these graphs, a strong agreement between the two results can be observed. 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. In addition, this comparison also demonstrated the computational efficiency of the LGtheory toolbox over the Simulink Simscape simulation, with the LGtheory toolbox being able to consistently compute and display its results in less time.

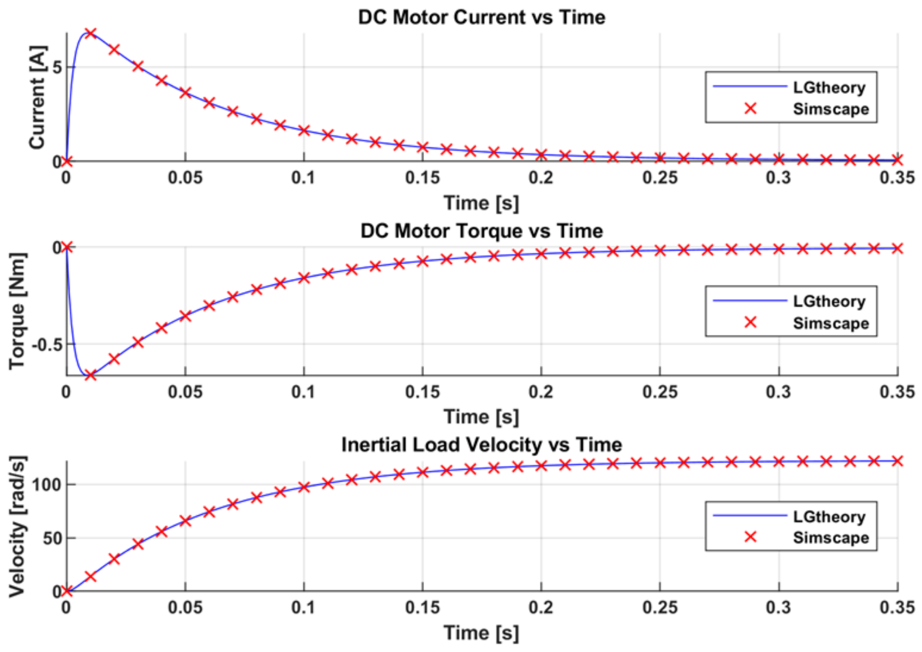


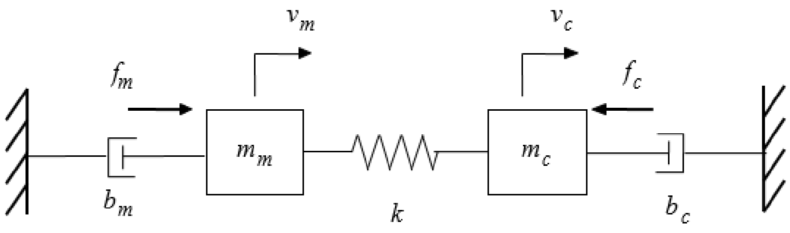
Figure 3-7: LGtheory MATLAB Simulation vs. Simulink Simscape Simulation Results

Note that for the results of both methods, the torque of the DC motor transducer element is determined to be negative. This is due to the equation for a transformer element, shown in Table 1-2, which relates the through-variable of the first port (current) to the through-variable of the second (torque) by a relation of $-1/TF$. This negative is required to signify that power is flowing out of the transducer element and into the inertial load element, which would be shown as positive if graphed.

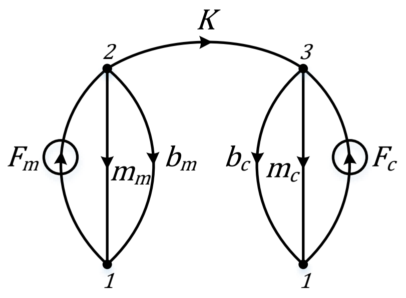
3.5 Additional Examples using LGtheory Toolbox

3.5.1 Example 1: Mechanical Translational System

The following system consists of two mass elements attached to each other with a spring and each attached to ground through damper elements.



(a)



(b)

Figure 3-8: (a) System model and (b) LG model of a Mechanical Translational System [5]

The following are the inputs to the LGtheory toolbox for this system:

```
LG.S =      [1 2 2 2 3 3 1]; %Source vector
LG.T =      [2 1 1 3 1 1 3]; %Target vector
LG.Type =   [7 2 5 6 2 5 7]; %Type vector
LG.Domain = [2 2 2 2 2 2 2]; %Domain vector
syms m m_m b_m K m_c b_c c
LG.Var_Names = [m m_m b_m K m_c b_c c];
syms v_m_m(t) v_m_c(t)
LG.y = [v_m_m(t) v_m_c(t)];
[Model] = LGtheory(LG);
```

The following equations represent the state-space model produced by the toolbox with outputs specified as the velocity of the mass elements:

$$\begin{bmatrix} \dot{v}_{m_m} \\ \dot{v}_{m_c} \\ \dot{F}_K \end{bmatrix} = \begin{bmatrix} -\frac{b_m}{m_m} & 0 & -\frac{1}{m_m} \\ 0 & -\frac{b_c}{m_c} & \frac{1}{m_c} \\ K & -K & 0 \end{bmatrix} \begin{bmatrix} v_{m_m} \\ v_{m_c} \\ F_K \end{bmatrix} + \begin{bmatrix} \frac{1}{m_m} & 0 \\ 0 & \frac{1}{m_c} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} F_m \\ F_c \end{bmatrix} \tag{86}$$

$$\begin{bmatrix} v_{m_m} \\ v_{m_c} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} v_{m_m} \\ v_{m_c} \\ F_K \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} F_m \\ F_c \end{bmatrix} \tag{87}$$

3.5.2 Example 2: Hydro-mechanical System

The following system consists of a rotational torque source, representing the power from an electric motor, powering a positive-displacement pump and piston which actuates a mass element attached to a spring and to ground in the mechanical translational domain.

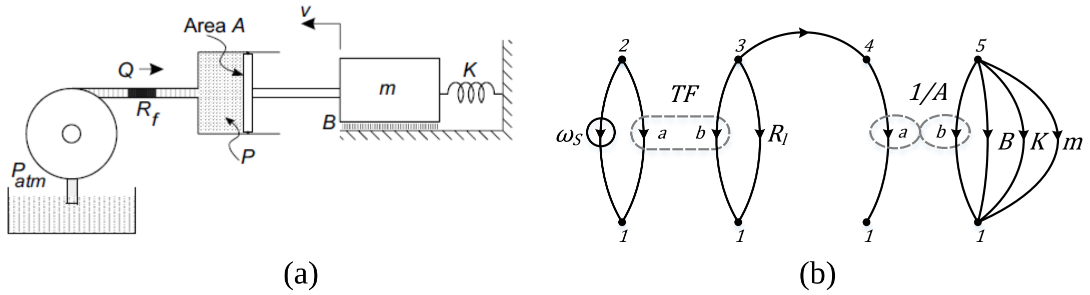


Figure 3-9: (a) System model and (b) LG model of a Hydro-mechanical System [5]

The following are the inputs to the LGtheory toolbox for this system:

```
LG.S =        [2 2 3 3 3 4 5 5 5 5];
LG.T =        [1 1 1 1 4 1 1 1 1 1];
LG.Type =     [1 3 3 5 5 4 4 5 6 2];
LG.Domain = [3 3 4 4 4 4 2 2 2 2];
syms s TF R_l R_f A B K m
LG.Var_Names = [s TF TF R_l R_f 1/A 1/A B K m];
syms P_R_f(t) v_m(t)
LG.y = [P_R_f(t) v_m(t)];
[Model] = LGtheory(LG);
```

The following equations represent the state-space model produced by the toolbox with outputs specified as the pressure of the fluid between the pump and the piston, and the velocity of the mass element:

$$\begin{bmatrix} \dot{v_m} \\ \dot{F_k} \end{bmatrix} = \begin{bmatrix} \frac{A^2(R_f + R_l - R_f R_l TF)}{m(R_l TF - 1)} - \frac{B}{m} & -\frac{1}{m} \\ K & 0 \end{bmatrix} \begin{bmatrix} v_m \\ F_k \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} [\omega_s] \tag{88}$$

$$\begin{bmatrix} P_{R_f} \\ v_m \end{bmatrix} = \begin{bmatrix} -AR_f & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v_m \\ F_k \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} [\omega_s] \tag{89}$$

3.6 Summary

The 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 evaluating LG models, most of these programs are no longer available for direct research and education related to the LG approach. The LGtheory MATLAB toolbox fills this gap by providing a complete and robust method for evaluating LG models in the MATLAB programming 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 an equivalent Simscape model. The results of this comparison demonstrated strong agreement between the LGtheory and Simscape modeling and simulation methods.

With the successful development of an automated tool for evaluating LG models, this approach can now be employed to facilitate more advanced applications, such as the automated evolutionary design of engineering systems, and the modeling of larger more complex multi-domain dynamic systems. These applications shall be demonstrated in Chapters 4 and 5 of this thesis, respectively.

Chapter 4. Automated Multi-Domain Engineering Design through Linear Graphs and Genetic Programming

4.1 Introduction

From voice and facial recognition software to self-driving vehicles and beyond, the applications of artificial intelligence (AI) and machine learning techniques to various aspects of our lives is an ever-growing field of research. One such up and coming focus of this research field is the use of machine learning in the design of mechatronic engineering systems. Similar to AI and machine learning, the field of mechatronic engineering, involving the design of multi-energy-domain systems, is one that has become ever more prevalent. Systems that were once purely single-domain in nature (mechanical, electrical, hydraulic, thermal, etc.) are now being improved immensely through integration of various energy-domain subsystems. The benefits of such integration include more robust functionality, improved efficiency and reliability, an enhanced ability for system optimization, and more accurate control.

The GP approach, which is implemented in the present work through the use of the GPLAB MATLAB toolbox [57], is a machine learning and evolutionary computing strategy which genetically evolves computer programs in the form of tree structures as a method of problem-solving. Although the idea of evolving computer programs had been theorized for many years before, the paradigms that govern the GP process were formally introduced by John Koza in 1990 [58]. Since its introduction, numerous applications of GP have been explored in various fields such as engineering, computer science, biomedical, and chemistry [59]. Applications of GP in engineering design include pre-existing examples of filter circuit synthesis, as demonstrated in [60, 61, 62], but the use of GP along with LG theory for the evolutionary design of multi-domain mechatronic systems is still rather scarce [63], with the only publication not evolving LG models directly, but rather, models converted from LG form into Simulink Simscape. Although the present work is only concerned with the evolution of filter circuits in the electrical-domain, by demonstrating the successful implementation of the LGtheory MATLAB toolbox for the purposes of the automated design of electronic circuits, this work may then be extended in a straightforward manner to systems encompassing multiple energy domains.

This chapter proposes a methodology of integrating LG modeling with the GP method for automating the evolution of multi-domain engineering systems using the in-house developed LGtheory toolbox and the GPLAB toolbox in MATLAB. The purpose of this work is to demonstrate the potential of combining the two fields, machine learning and mechatronics design, in order to achieve computer-automated design of multi-domain engineering systems. With the integration of the GP and LG approaches, an evolutionary design method is proposed with detailed explanations of the terminology and design steps. Examples of designing various electronic filter circuits are presented to demonstrate the application of the proposed method.

4.2 Genetic Programming

GP is a machine learning and evolutionary computing technique that generates computer programs consisting of various functions and terminals in an evolutionary manner based on the concept of natural evolution. In natural evolution, members of a population adapt to changing environments over numerous generations. During this process, members who inherit or develop beneficial characteristics become more likely to survive, reproduce, and pass on these characteristics to future generations. Alternatively, members who inherit or develop unbeneficial characteristics are more likely to die off before reproduction. This process of natural selection means that, over time, the weaker members of the population are unable to pass on their non-advantageous characteristics, thus strengthening the total population.

GP represents members of the population in the form of tree structures, consisting primarily of two main components: functions, which are sub-programs that conduct a specific procedure or routine based on input values and return or perform some output action; and terminals, which can either be functions which have no inputs, or operands which only provide input values to other functions in the tree. The topological structure of these trees is represented by nodes which denote the aforementioned functions and terminals, and lines which represent the hierarchical connections between nodes. In the tree structure, all internal nodes must be functions as they have to accept input actions from sub-nodes and provide an output action their parent node, whereas all outermost nodes of the tree must be terminals as they can only provide an output action to an internal node function.

All functions and terminals used in GP must satisfy the property of closure, which requires that all functions take into consideration type consistency and evaluation safety [64]. This is important because as the GP process adapts and manipulates trees stochastically through crossover and mutation operations, it is possible that any combination of functions and terminals may occur; thus, consistency in function input/output types, or a method of predictably converting types, is required to ensure that any possible tree structure can be evaluated properly. Similarly, evaluation safety is required as some functions may fail during runtime under certain conditions (i.e., a divide function may fail if the operand of the denominator is 0). It is common practice to ensure that these functions are protected by checking for conditions that would lead to failure and, if so, perform some predetermined output action instead (e.g., if dividing by 0, always return 1). Alternatively, it is possible to allow these failures to occur while applying a large fitness penalty to these solutions in order to potentially eliminate them from the population in subsequent generations.

4.2.1 Fitness Function

Similar to how the environment tests an individual's ability to survive in natural evolution, the fitness function in GP evaluates a solution's ability to produce the desired result. This method of evaluation can be conducted in any number of ways that the user sees fit, and can optimize for either maximizing or minimizing the resulting fitness values. One of the most common ways to determine fitness is to calculate the absolute error between the produced result and the desired outcome. For this particular type of fitness measurement, it is desired to minimize the fitness of each member, as this fitness value relates directly to the total error for that individual; thus, by prioritizing the minimization of fitness for this type of evolution, preceding generations will strive to achieve lower error. Ultimately, after every member of each generation has been produced and evaluated, the individual with the most desirable fitness value, be it lowest or highest, is chosen as the solution to the GP problem.

4.2.2 Selection

Once the fitness of the population has been evaluated, the selection process occurs. Again, this process can be conducted in a number of ways depending on the specific application, but one of the most common methods used is the roulette wheel approach. This method replicates the process of spinning a roulette wheel where each member of the population occupies a portion of the wheel with a size corresponding to their fitness with respect to the total fitness of the population. This means that members with more desirable fitness values will be more likely to be selected as parents for reproduction than members with less desirable fitness values. Another common method for selection is the tournament approach. This method involves forming small subgroups of random individuals and pitting them against each other. The individual with the most desirable fitness of the subgroup is then selected as a parent for reproduction.

4.2.3 Genetic Operations

In GP, two primary genetic operations (and many variants of these operations) can be utilized during the reproduction process in order to introduce new individuals and diverse characteristics into the population. These operations are crossover and mutation:

- Crossover occurs between two individuals that have been selected as parents for reproduction. A random node is selected from the trees of each parent and the associated branches are swapped with one another. This process results in two new children who differ slightly from their parents in order to further explore the solution space.
- Mutation occurs for a single individual who has been selected to be a parent. A random node is selected in this individual and the associated tree branch is replaced with a random branch that is created using the available functions and terminals. This process results in one new child who is similar to its parent but with some newly introduced characteristics. This operation is beneficial for introducing more solution diversity into the population

4.2.4 GPLAB

GPLAB is a GP-based MATLAB toolbox, developed by Sara Silva at the University of Coimbra, Portugal, which was released for public use in July of 2003 [65]. This toolbox has been under consistent development since its initial release, with the most recent version at the time of writing, version 4.04, being released in June of 2018. While GPLAB's default configuration is for more basic tasks such as symbolic regression, it is also possible to customize various aspects of the toolbox, such as functions and terminal sets, fitness functions, selection procedures, genetic operations, etc., in order to facilitate the needs of more advanced research.

Since GPLAB is written in MATLAB, along with other benefits such as its robustness and flexibility for user modification, it can be easily adapted to be compatible with any other MATLAB-based scripts or libraries such as the LGtheory MATLAB toolbox. Due to these reasons, GPLAB was selected for implementation with LGtheory for the present work.

4.3 Design Evolution of Electronic Filter Circuits

Electronic filters are signal processing electric circuits consisting of discrete electrical components with the purpose of attenuating unwanted frequencies ranges while maintaining the desired ranges. There are many types of electronic filters with different functions, applications, and benefits. The purpose of the present section is to explore the application of combined LG and GP approach in the design of linear passive low-pass, high-pass, and band-pass filter circuits, and thereby validate the proposed method. Passive filters are filter circuits that contain passive electrical components, such as capacitors, inductors, and resistors. They do not contain active components, such as operational amplifiers, which require external power sources to operate.

4.3.1 Embryo Model

The design evolution process begins with an initial embryo model in order to provide a basis on which the evolutionary process can commence. For all three filters that will be designed in the following sections, the embryo model will consist of a voltage source, V_S, in series with a resistive element, R_S, representing the sources internal resistance, and another resistive element, R_L, representing the load resistance, which will be added to the evolved models between their last (n^{th}) node and the ground node, 1. The values pertaining to the source voltage and the resistances of the embryo model are specified by the user during setup in order to accommodate the specific needs of the system. Figure 4-1 illustrates the embryo system model; the evolved filter circuit is constructed within this embryo between the nodes highlighted with the red squares, 3 and *n*. In order to evaluate the evolved models, the voltage of the load resistance element is selected as the output of the state-space model and the frequency response of this element is observed via a Bode plot for each iteration of the evolutionary design process.

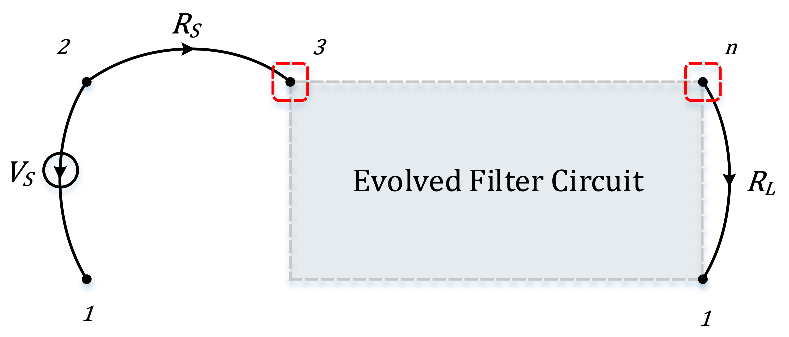


Figure 4-1: Embryo LG Model for Evolving Filter Circuits

4.3.2 Functions

The following sections describe the functions utilized by the GP toolbox to facilitate the construction of the LG models for the filter circuits.

4.3.2.1 Series

The series function takes two input actions to the function and constructs the resulting sub-models in a series manner. In the simplest case, if both input actions are the addition of single elements to the model, the series function will place both of them in series with one another in the system model. For more complex cases, this function will produce the sub-models that result from the associated input branches in such a way that the resulting system components are constructed in series with one another as well.

4.3.2.2 Split

The split function takes two input actions and constructs the resulting sub-models off of the same base node independently of one another. After a split occurs, each resulting sub-model will be constructed separately from one another and grounded independently, resulting in each sub-model being parallel with one another between the node where the split occurs and the ground node. This splitting function is important for the design of electronic filters as it results in a ladder-like topology that is required for constructing higher-order filter circuits.

4.3.3 Terminals

The Add_A, Add_D, and Add_T terminals each add their respective passive LG elements to the embryo system model. These terminals perform the addition of their respective elements in an identical manner, with the exception of the element type index value which must correspond with the type of element that is being added to the system. When an element is added, a new node is also created, upon which the model can be built further. If this newly created node is not to be further built upon during the GP process, it will be converted into a ground node. Additionally, these terminals randomly select the parameter values associated with the newly added element within a range of values which are determined based on the source/load resistance values and the cutoff frequencies specified by the user. This parameter selection process is done in order to ensure that the GP fitness will converge toward an optimal value quickly by reducing the solution space which the algorithm must explore.

4.3.4 Fitness Function

Each model generated by the GP algorithm is subsequently sent to the LGtheory toolbox in order to extract the state-space model of the system. If this toolbox determines that it is impossible to extract the state-space models of the evolved systems due to any issues with the model construction by the GP (e.g., excess or lack of state variables, incomplete models, etc.) an error message is returned to the fitness function by the toolbox. Upon detection of such error messages, the fitness function will end for that particular system and a large penalty value shall be applied to the fitness of that individual.

If a state-space model can be extracted, the frequency response of the system is evaluated using the `bode` function in MATLAB. From the data produced by the `bode` function, the magnitude of the output voltage at each point along the plot is evaluated against the desired voltage output for that specific frequency. Ideally, this means that for frequencies that are desired to be passed, the magnitude of the output voltage will be equal to the input voltage, and for unwanted frequencies, the magnitude of the output voltage will be equal to zero. The absolute error values found between the desired and the actual voltage values within the entire frequency range are summed and equated to the fitness of the evolved system. In band pass filter evolution, the absolute error values associated with the pass band frequencies are squared to increase their impact on the fitness value. This is due to the pass band generally being narrow compared to the total frequency range; squaring the error of the pass band helps to equalize the algorithm's prioritization of the pass band and attenuated frequencies, thus tending to generate a more optimal system.

The GP algorithm is run for the amount of generations and population size specified by the user, and the evolved model that results in the lowest fitness value is selected and presented to the user as the final design.

4.3.5 Interpreting Tree Structures

Figure 4-2 shows the tree structure evolved for the low pass filter model. The GP algorithm executes functions in such a manner that the first sub-tree from the function node is evaluated in its entirety before the second sub-tree is considered. This is represented in the figure by the arrows lining the perimeter of the tree in a counter-clockwise direction, which defines the order in which the terminal nodes are executed, and thus, the order in which elements are added to the system.

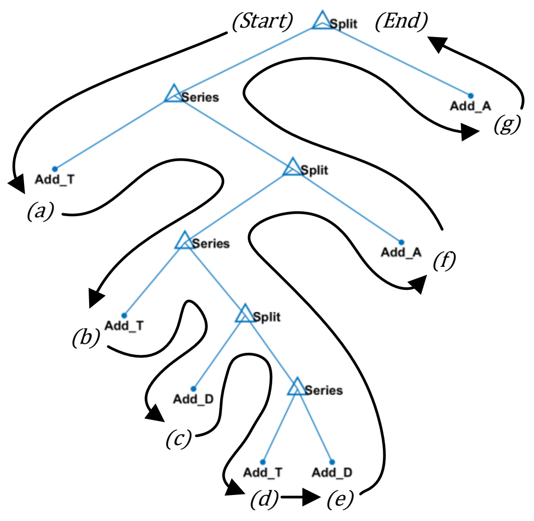


Figure 4-2: Tree Structure of the Evolved Low Pass Filter with the Directionality of Node Execution

This tree structure is interpreted as follows: at (Start), a split is created at node 3 of the embryo model. At (a), an inductor (T-Type) element is added to the model at node 3 and a new node, 4, is created. Another split is created at node 4, and at (b), a second inductor element and node, 5, are added to the model. At this newly created node, another split occurs with the addition of a resistive (D-Type) element at (c), with the node created by the addition of this element being connected to the ground. A series connection of the third inductor and the second resistor elements are added to node 5 at (d) and (e), creating new nodes 6 and 7. Following the arrow from (e) to (f) results in the grounding of node 7, and the addition of a capacitor (A-Type) element between the previously split node 4 and a new node, which is subsequently grounded. Finally, following the arrow from (f) through (g) to (End) results in the addition of a second capacitor element between node 3 and ground.

After the tree has been interpreted by the GP algorithm and the LG model constructed, the load resistance element is added to the system between the highest value node, in this case 6, and ground. This results in the final low pass filter LG model, as shown in Figure 4-4.

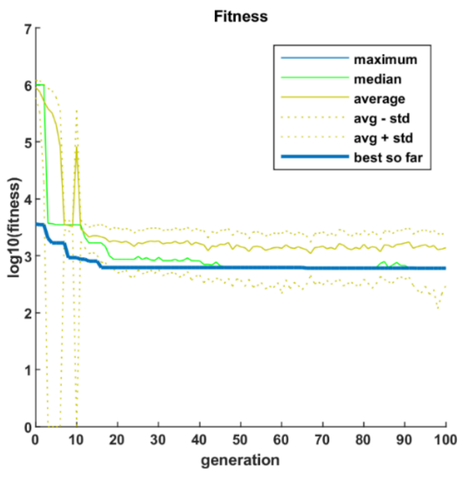
4.4 Results and Discussion

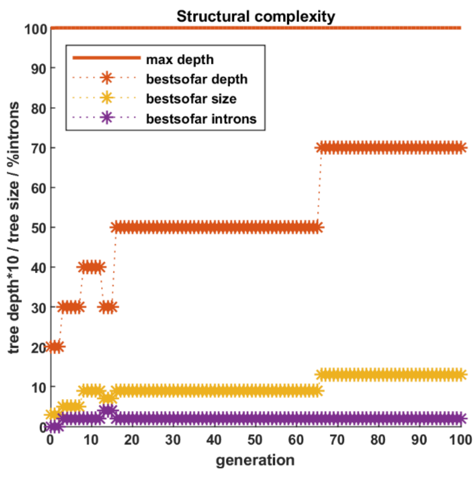
4.4.1 Low Pass Filter Evolution

The low pass filter evolution program was run for a system containing a 10-volt source with an internal resistance of 750 Ohms, and a load resistance of 50 Ohms. The filter was designed for a cutoff frequency of 50 kHz over 100 generations with a population size of 50. Figure 4-2 of the previous section shows the tree structure that was constructed by the GP algorithm for these input values.

The two graphs shown in Figure 4-3 demonstrate the evolution of the fitness and structural complexity, respectively, of the evolved tree structures over the course of 100 generations.

Figure 4-3 (a) shows the change in the fitness of the "best so far" tree structure, as well as, the median and average fitness values, and the average standard deviations of all trees for each generation of the evolution process. Similarly, Figure 4-3 (b) shows the evolving structural complexity of the "best so far" solution in the form of depth, size, and percentage of introns of the associated tree structure.

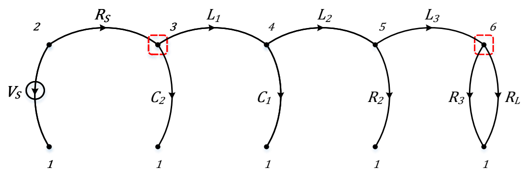


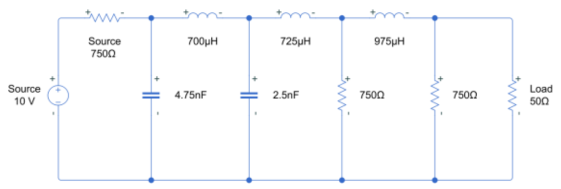


(a) (b)

Figure 4-3: (a) Fitness and (b) Structural Complexity for the Evolution of the Low Pass Filter

The resulting LG model and the equivalent circuit diagrams are shown in Figure 4-4.





(a) (b)

Figure 4-4: (a) LG Model and (b) Circuit Diagram of the Evolved Low Pass Filter

The frequency response of the system is shown in Figure 4-5.

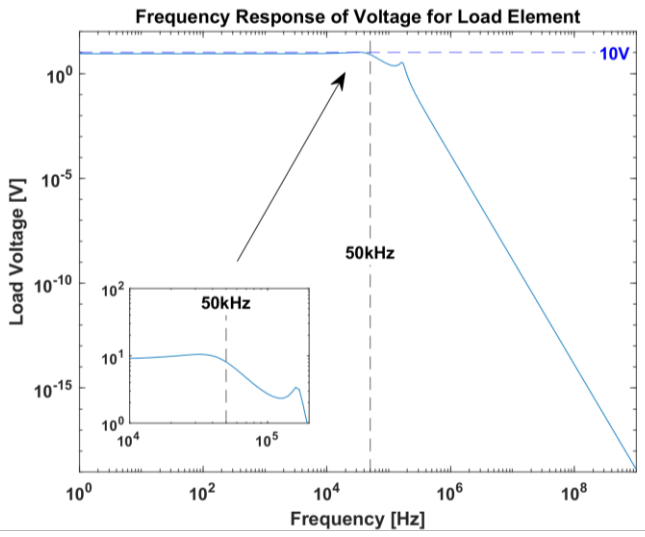


Figure 4-5: Frequency Response Function of the Evolved Low Pass Filter

4.4.2 High Pass Filter Evolution

The high pass filter evolution program was run for a system containing a 10-volt source with an internal resistance of 750 Ohms, and a load resistance of 50 Ohms. The filter was designed for a cutoff frequency of 300 kHz over 100 generations, with a population size of 50. Figure 4-6 shows the tree structure that was constructed by the GP algorithm for these input values.

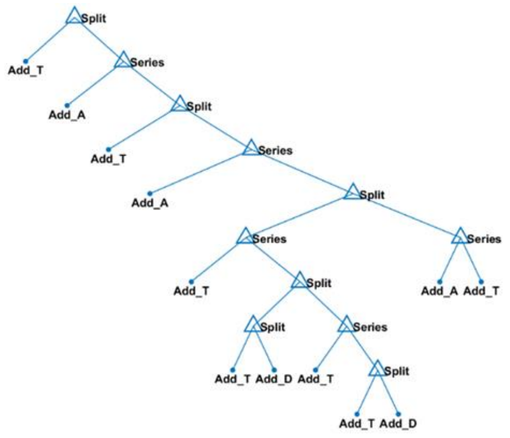
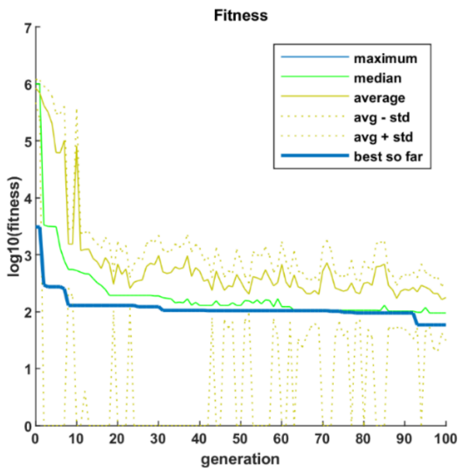
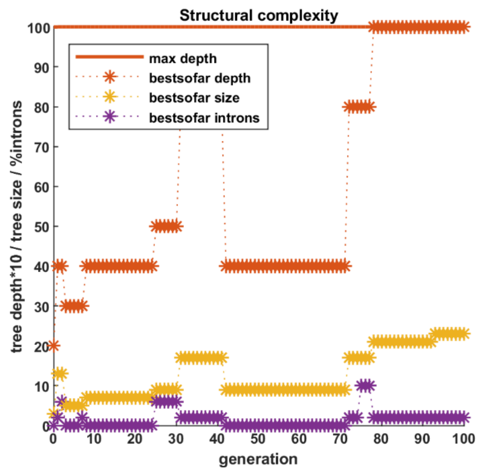


Figure 4-6: Tree Structure of the Evolved High Pass Filter

The two graphs shown in Figure 4-7 demonstrate the evolution of the fitness and structural complexity of the evolved tree structures over the course of 100 generations.



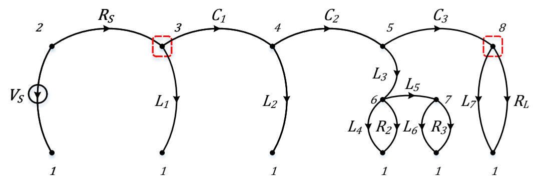
(a)



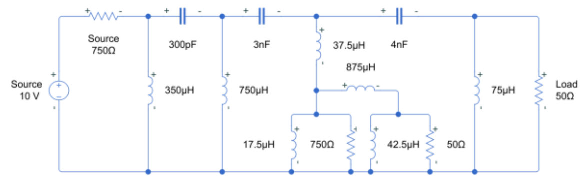
(b)

Figure 4-7: (a) Fitness and (b) Structural Complexity for the Evolution of the High Pass Filter

The resulting LG model and the equivalent circuit diagrams are shown in Figure 4-8.



(a)



(b)

Figure 4-8: (a) LG Model and (b) Circuit Diagram of the Evolved High Pass Filter

The frequency response function of the system is shown in Figure 4-9.

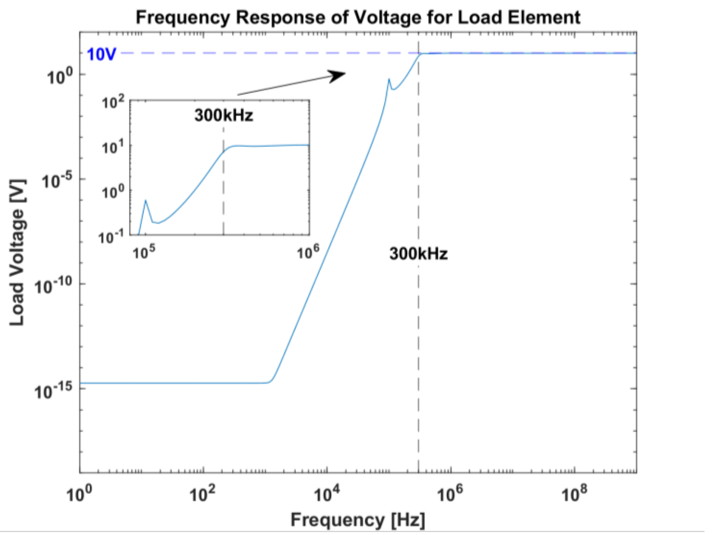


Figure 4-9: Frequency Response Function of the Evolved High Pass Filter

4.4.3 Band Pass Filter Evolution

The band pass filter evolution program was run for a system containing a 10-volt source with an internal resistance of 750 Ohms, and a load resistance of 50 Ohms. The filter was designed for a lower cutoff frequency of 20 kHz and an upper cutoff frequency of 250 kHz, over 100 generations with a population size of 50. Figure 4-10 shows the tree structure, which was constructed by the GP algorithm for these input values.

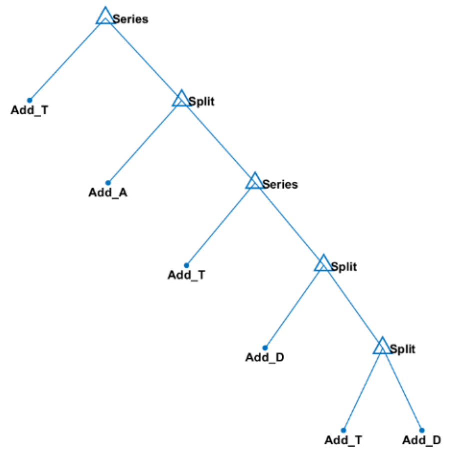
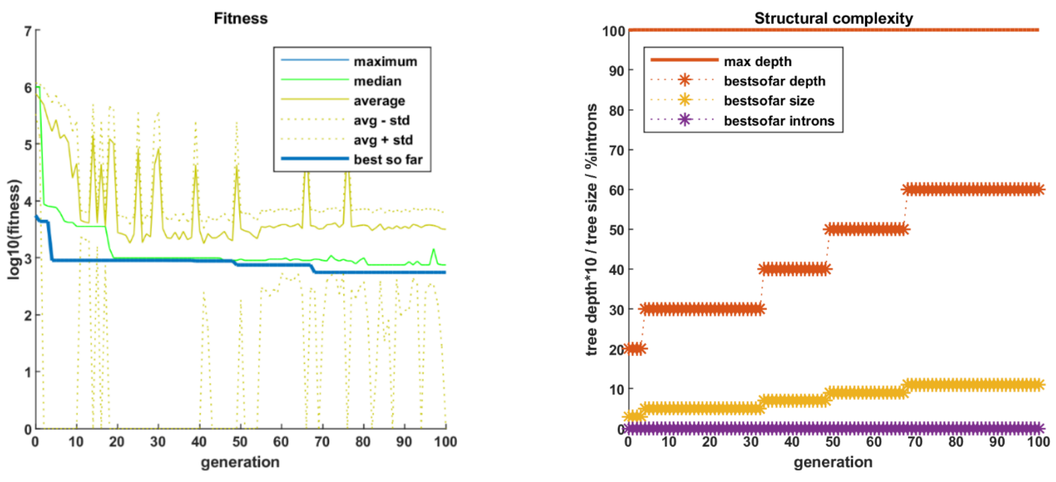


Figure 4-10: Tree Structure of the Evolved Band Pass Filter

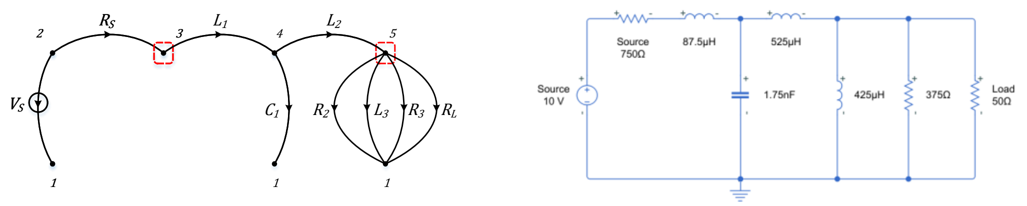
The two graphs shown in Figure 4-11 demonstrate the evolution of the fitness and structural complexity of the evolved tree structures over the course of 100 generations.



(a) (b)

Figure 4-11: (a) Fitness and (b) Structural Complexity for the Evolution of the Band Pass Filter

The resulting LG model and equivalent circuit diagrams are shown in Figure 4-12.



(a) (b)

Figure 4-12: (a) LG Model and (b) Circuit Diagram of the Evolved Band Pass Filter

The frequency response function of the system is shown in Figure 4-13.

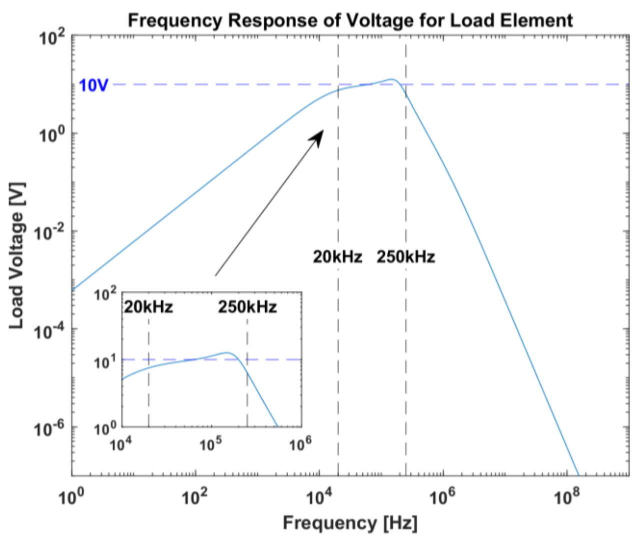


Figure 4-13: Frequency Response Function of the Evolved Band Pass Filter

3.1 Summary

This Chapter presented the application of the combined LG modeling and GP design approaches for the evolutionary design of passive electronic filter circuits. While the use of GP for the design of electronic filters is not new, this application of design evolution was utilized in order to demonstrate the capabilities of LG modeling and the LGtheory MATLAB toolbox for automated design by evaluating the evolved systems via dynamic simulation.

In this Chapter, three types of passive filter circuits (low pass, high pass, and band pass filters) were designed using the combination of LG modeling and GP. The results of the frequency response of these systems demonstrated that the combined LG-GP program was able to successfully evolve high-order filter circuits that were effective at attenuating the voltages of undesirable frequencies while maintaining the voltages of desired frequencies.

While the present application of designing filter circuits is only concerned with the electrical energy domain, due to the unified and integrated nature of LG modeling, a characteristic which other single-domain evaluation tools lack, it is possible to extend the present work to other domains, and even combinations of different domains, for the design of complex multi-domain mechatronic system. The present results further validate the capabilities of the LGtheory MATLAB toolbox, which was developed as a robust and reliable software package for the design and simulation of complex dynamic systems.

Additional validation of the LGtheory toolbox's capabilities shall be provided in the coming chapter through the dynamic modeling of a complex multi-domain system in the form of a mobile robot.

Chapter 5. Dynamic Modeling and Simulation of a 4-Wheel Skid-Steer Mobile Robot using the Linear Graph Approach

5.1 Introduction

In comparison to LG theory, the BG theory is more widely utilized by researchers for applications related to mechatronics and robotics. Examples of this approach applied in these fields can be seen in [36, 39], as well as, applications related specifically to the modeling of mobile robotic system dynamics in [66] and [67]. By comparison, recent applications of the LG approach to robotics is relatively limited to [19] and [20]. Despite this discrepancy in publications between the two approaches, the LG method provides various benefits over the BG method, along with the versatility and robustness which LG modeling provides for evaluating such multi-domain systems.

To demonstrate the versatility and robustness of the LG approach, the work presented in this chapter proposes an LG model representation of the dynamics of a 4-wheel skid-steer mobile robot, the Clearpath Husky. Due to this model's complexity, the LGtheory MATLAB toolbox, developed in our lab to address the lack of available LG-based software tools [68], will be utilized to automate the process of extracting the dynamic system's state-space model. This model will then be further enhanced with a GA-based parameter estimation procedure, similar to [20], in order to calibrate the unknown parameter values of the model. Validation of the proposed model will then be performed via comparisons of the trajectory response of the simulated LG model against data collected from a Robot Operating System (ROS) based Gazebo simulation and from experimental data of real-world driving scenarios using the Clearpath Husky mobile robotic system.

5.2 Linear Graph Modeling of the Mobile Robot Subsystems

The LG model of the Clearpath Husky robot consists of various subsystems that encompass multiple domains and functions of the robotic system. The complete model includes the electrical subsystem, consisting of the DC motors powered by a battery voltage source, the drivetrain systems, consisting of primary and secondary axles and wheels for both the independent left and right side drivetrain systems, and the rotational and translational dynamics subsystems, representing the linear and rotational movement of the entire mobile robot, respectively.

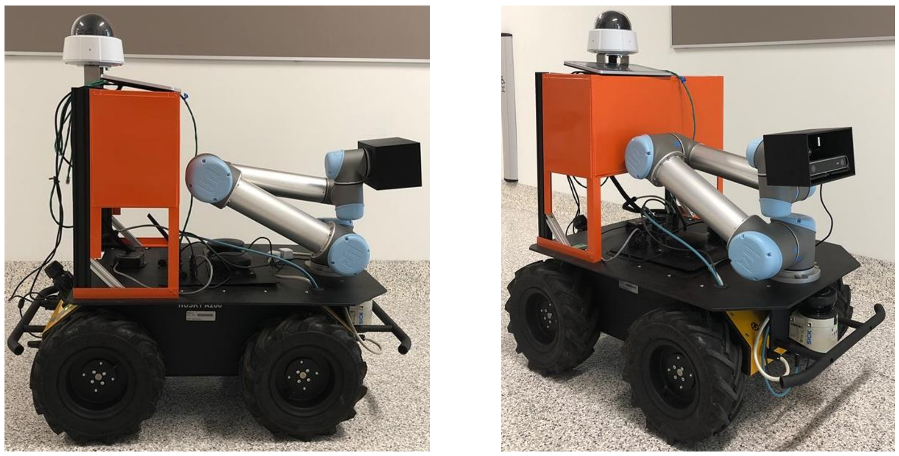
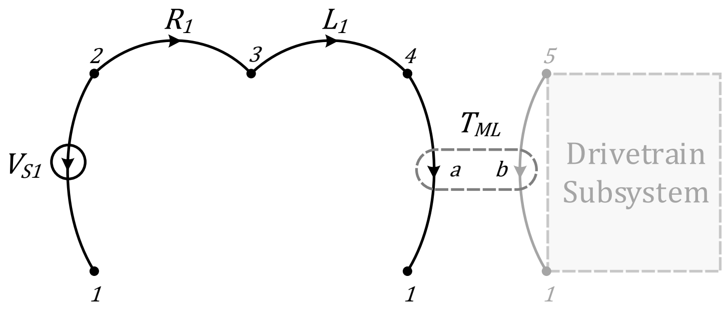


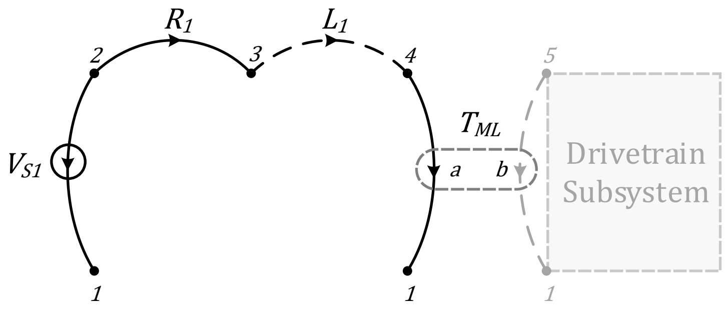
Figure 5-1: Clearpath Husky Mobile Robotic System

5.2.1 Electrical Subsystem

The Husky's electrical subsystem consists of two 24V brushed DC motors, each with a 78.71:1 gearbox reduction attachment, which transmit the output torque of the motors directly to the primary axles of the two independent drivetrains. These DC motors are modeled as series LG circuits consisting of a controlled DC voltage source (V_{S1}), a D-Type resistive element (R_1), a T-Type inductive element (L_1) and a two-port transformer element representing the combined output of the motor torque constant and the gearbox reduction (T_{ML}).



(a)



(b)

Figure 5-2: (a) LG Model and (b) Normal Tree of the Husky Robot's Electrical Subsystem

The equations produced by the LGtheory toolbox for the electrical subsystem based on the constitutive equations and normal tree of the LG model are as follows:

The continuity equations for each passive branch are:

$$i_{R_1} = i_{L_1} \tag{90}$$

$$i_{T_{ML}} = i_{L_1} \tag{91}$$

The compatibility equation for each loop formed by the temporary inclusion of each passive link is:

$$V_{L_1} = V_{s1} - V_{T_{ML}} - V_{R_1} \tag{92}$$

5.2.2 Drivetrain and Wheels Subsystem

The Husky's drivetrain subsystem consists of two independent drivetrains each powered by its own DC motor. Each of these independent drivetrains consists of a primary axle, in which the DC motor is directly attached, and a secondary axle, which receives power from the primary axle via a 1:1 belt drive system. Each of these independent drivetrains power both of the wheels on their respective side of the vehicle. Figure 5-3 shows a diagram of the Husky's system configuration with drivetrain components and sensors used for data collection.

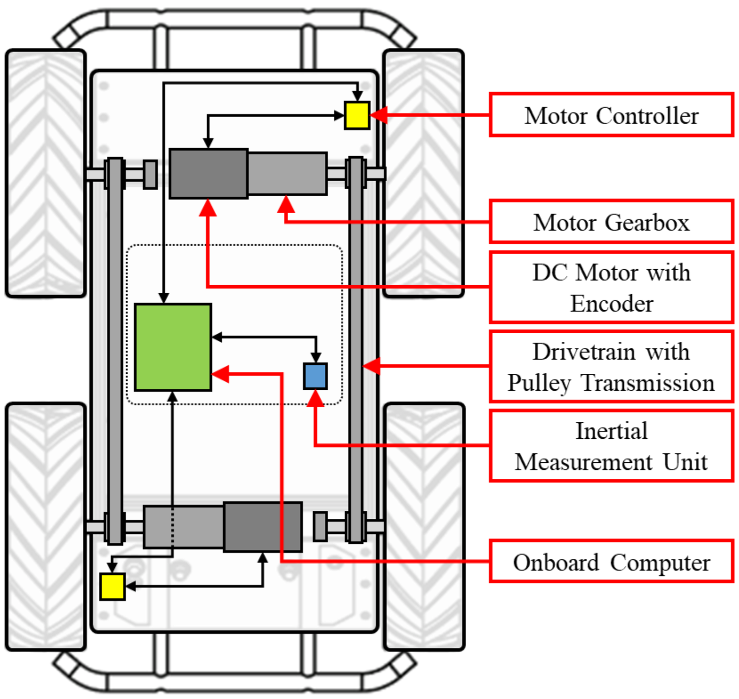


Figure 5-3: Systematic Diagram of the Clearpath Husky Robot

The work in this chapter demonstrates two methods for modeling the drivetrain subsystem: the first method, referred to as the expanded model, accounts for the dynamics of all the wheels on the vehicle separately and considers the compliance and slippage of the belt drive system; and the second method, referred to as the simplified model, assumes the belt drive system to be rigid and models the inertias of the two axles of each drivetrain as a single element.

For the expanded model (considering only the drivetrain on the left side of the vehicle, as the other half of the drivetrain will be modeled in the same manner), the second branch of the two-port element (coming from the corresponding motor of the electrical subsystem) splits at its upper node into four paths.

The first two of these paths, which are set in parallel with the second port of the motor transformer, consist of elements that represent the parameters of the primary axle of the drivetrain. The first path contains an A-Type element (J_{RL}) and the second path consists of a D-Type element (B_{RL}) in series with two transformer elements (TF_1 and TF_2). The first path represents the combined inertial load of the wheel and shaft of the primary axle; whereas, the second path collectively represents all of the energy being lost or transferred out of the drivetrain system. The D-Type element represents the energy dissipation from the system due to friction in the shaft bearings, as well as, the friction of the wheels slipping on the driving surface, and the transformer elements represent the transmission of torque from the drivetrain subsystem to the translational and rotational dynamics subsystems of the Husky vehicle.

The third and fourth paths consist of a D-type (B_{BeltL}) and a T-type (K_{BeltL}) element in parallel, representing the compliance and slip of the belt in the pulley transmission system. These paths then split at their shared second node into two more paths representing the inertial load (J_{FL}) and energy dissipation/transmission (B_{FL}, TF_3, and TF_4) of the secondary axle, similar to the primary axle.

Figure 5-4 demonstrates the aforementioned benefits of LG modeling over other modeling approaches. The LG model representation provides a closer resemblance to the physical system than any other form of dynamic system modeling, as can been seen by how the elements representing the dynamics of each axle overlay their respective wheels, and how the parallel D-type and T-type elements closely resemble the belt connecting the two shafts to one another.

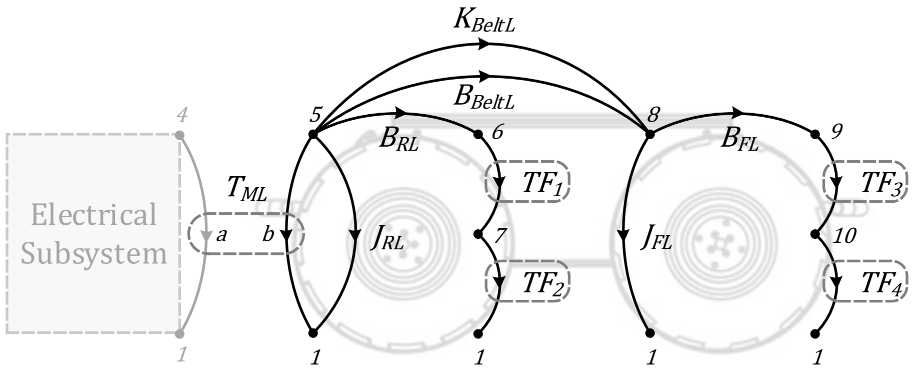
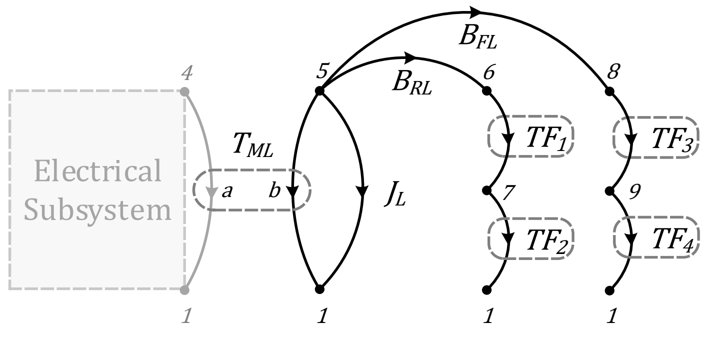


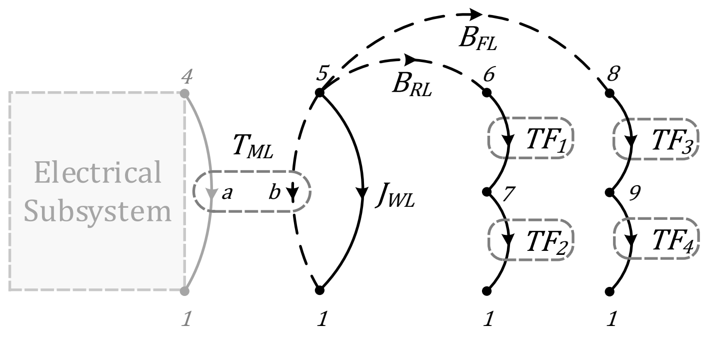
Figure 5-4: LG Model of the Left Side Drivetrain Subsystem Overlaid on the Profile of the Husky Robot

For the simplified model, shown in Figure 5-5, it is assumed that the belt is rigid and slip free, thus eliminating the compliance between the primary and secondary axles of the drivetrain. With this change, the model can be further simplified by combining the inertia of the primary and secondary axels into a single element (J_L).

Although the simplified model sacrifices some resemblance to the physical system, this model was found to still accurately represent the dynamics of the mobile robot while greatly reducing the complexity and size of the final state-space model.



(a)



(b)

Figure 5-5: (a) Simplified LG Model and (b) Normal Tree of Left Side Husky Drivetrain Subsystem

The equations produced by the LGtheory toolbox for the simplified drivetrain subsystem based on the constitutive equations and normal tree of the LG model are as follows:

The continuity equations for each passive branch are:

$$\tau_{J_{LW}} = -\tau_{B_{FL}} - \tau_{B_{RL}} - \tau_{T_{ML}} \qquad (93)$$

$$\tau_{TF_1} = \tau_{B_{RL}} \qquad (94)$$

$$\tau_{TF_2} = \tau_{B_{RL}} \qquad (95)$$

$$\tau_{TF_3} = \tau_{B_{FL}} \qquad (96)$$

$$\tau_{TF_4} = \tau_{B_{FL}} \qquad (97)$$

The compatibility equations for each loop formed by the temporary inclusion of each passive link are:

$$\omega_{T_{ML}} = \omega_{J_{LW}} \tag{98}$$

$$\omega_{B_{RL}} = \omega_{J_L} - \omega_{TF_1} - \omega_{TF_2} \tag{99}$$

$$\omega_{B_{FL}} = \omega_{J_L} - \omega_{TF_3} - \omega_{TF_4} \tag{100}$$

5.2.2.1 Drivetrain Transformer Equations

The four transformer elements of the half drivetrain model shown in Figure 5-5 represents the conversion of the wheel torques into the tractive forces which propel the vehicle linearly (TF_{odd}), and the conversion of that tractive forces into the moments which rotates the vehicle (TF_{even}). The equations which define these power conversions are determined as follows:

For TF_{odd}, the rotation of the vehicle's wheels due to the power provided by the DC motors creates friction between the tread of the tire and the driving surface (F_{F_i}). This traction between the two surfaces creates a counterforce on the vehicle (F_{W_i}), which propels the vehicle forward.

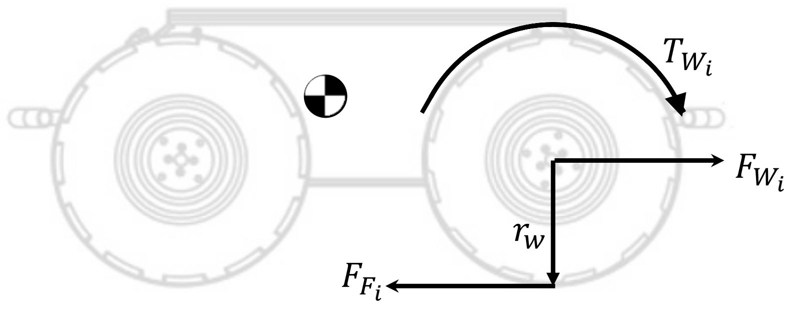


Figure 5-6:Diagram of Wheel Forces Produced by the Torque of the Wheels

Referring to the constitutive equations for a transformer in Table 1-2, f_1 is the generalized through-variable of the first port, representing the wheel torque, and f_2 is the generalized through-variable of the second port, representing the tractive force of the wheels. In order to convert the wheel torque to a force, f_1 must be divided by the radius of the wheel (r_W), therefore:

$$TF_{odd} = \frac{1}{r_W} \tag{101}$$

Similarly, for TF_{even}, the tractive force which propels the vehicle, created by the torque of the DC motors, also produces the moment which rotates the vehicle.

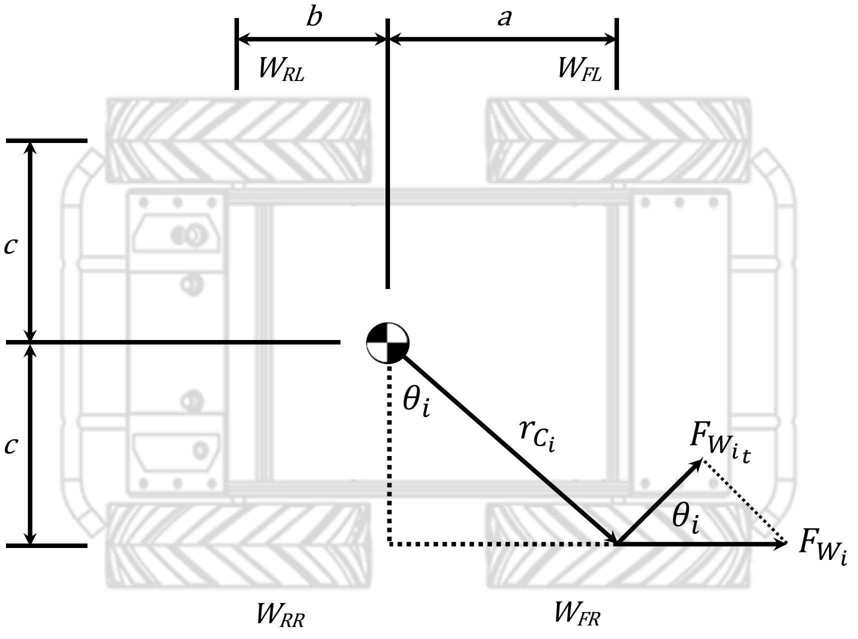


Figure 5-7: Diagram of Rotational Moments of the Husky Vehicle Produced by the Torque of the Wheels

Referring to the same constitutive equations for a transformer as before, f_1 is the generalized through-variable of the first port, representing the wheel torque, and f_2 is the generalized through-variable of the second port, representing the rotational moment the wheels induce on the vehicle. In this transformer, two conversions occur; firstly, the conversion of the wheel torque to tractive force, followed by the conversion of the tractive force to a rotational moment. Again, to create tractive force, the torque of the wheel must be divided by the radius of the wheel. To convert this tractive force to rotational moment, the distances from the center of mass of the vehicle to the contact point of each wheel and the driving surface (r_{c_i}) must be found. The rotational moment on the vehicle produced by each wheel can then be determined by multiplying these distances by the tangential component of each wheels' tractive force. For the following transformer equation, the "+" is for wheels on the right side of the vehicle, as a forward rotation of those wheels will result in a force which produces a counter-clockwise (positive) moment on the vehicle, and the "-" is for wheels on the left side of the vehicle, as a forward rotation of those wheels will result in a force which produces a clockwise (negative) moment on the vehicle:

$$TF_{even} = \pm\, cos\left(\theta_{W_i}\right) \cdot r_{C_i} \cdot \frac{1}{r_W} \tag{102}$$

Where TF_2 corresponds to the rear left wheel (W_{RL}), TF_4 the front left (W_{FL}), TF_6 the front right (W_{FR}), and TF_8 the rear right (W_{RR}), as shown in Figure 5-7. Therefore, for TF_2 and TF_8 the following conditions apply:

$$\theta_{W_{RL}} = \theta_{W_{RR}} = tan^{-1}\left(\frac{b}{c}\right) \tag{103}$$

$$r_{C_{RL}} = r_{C_{RR}} = \sqrt{(b)^2 + (c)^2} \tag{104}$$

And for TF_4 and TF_6 the following conditions apply:

$$\theta_{W_{FL}} = \theta_{W_{FR}} = tan^{-1}\left(\frac{a}{c}\right) \tag{105}$$

$$r_{C_{FL}} = r_{C_{FR}} = \sqrt{(a)^2 + (c)^2} \tag{106}$$

5.2.3 Mobile Robot Translational Dynamics Subsystem

The translational dynamics of the Husky robot are determined by the summation of the traction forces produced by the wheels of the vehicle. This is represented in LG form by placing the second ports of the odd-numbered transformer elements (representing the conversion of wheel torque to traction force) in parallel with an A-Type element (M_{Husky}) representing the mass of the Husky vehicle. This configuration means that the forces produced by each tire will be summed in order to induce an acceleration on the vehicle when the wheels produce a non-zero resultant force, and allow the vehicle to remain stationary when the tire forces are balanced (i.e. stopped, zero turning).

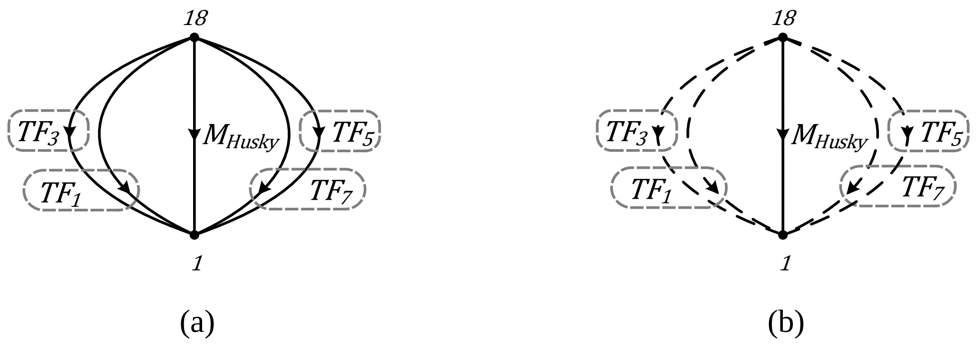


Figure 5-8: (a) LG Model and (b) Normal Tree of the Husky Robot Translational Dynamics Subsystem

The equations produced by the LGtheory toolbox for the mobile robot translational dynamics subsystem based on the constitutive equations and normal tree of the LG model are as follows:

The continuity equation for the passive branch is:

$$F_{M_{Husky}} = -F_{TF_1} - F_{TF_3} - F_{TF_5} - F_{TF_7} \tag{107}$$

The compatibility equations for each loop formed by the temporary inclusion of each passive link are:

$$v_{TF_1} = v_{M_{Husky}} \tag{108}$$

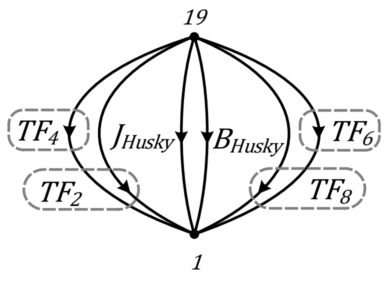
$$v_{TF_3} = v_{M_{Husky}} \tag{109}$$

$$v_{TF_5} = v_{M_{Husky}} \tag{110}$$

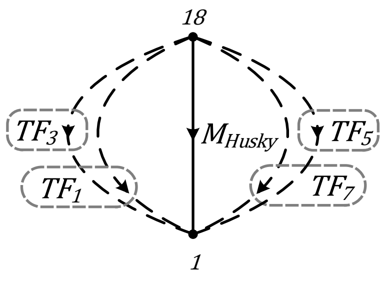
$$v_{TF_7} = v_{M_{Husky}} \tag{111}$$

5.2.4 Mobile Robot Rotational Dynamics Subsystem

The rotational dynamics of the Husky robot are determined by the summation of the rotational moments produced by the wheels of the vehicle. This is represented in LG form by placing the second ports of the even numbered transformer elements (representing the conversion of wheel torque to rotational moment) in parallel with a D-Type (B_{Husky}) and an A-Type element (J_{Husky}) representing the resistance to rotational movement and the inertia of the Husky vehicle, respectively. This configuration means that the torque produced by each tire will be summed in order to induce a rotation in the vehicle when the wheels produce a non-zero resultant moment, and allow the vehicle to remain stationary when the rotational moments are balanced (i.e. not turning).



(a)



(b)

Figure 5-9: (a) LG Model and (b) Normal Tree of the Husky Robot Rotational Dynamics Subsystem

The equations produced by the LGtheory toolbox for the mobile robot rotational dynamics subsystem based on the constitutive equations and normal tree of the LG model are as follows:

The continuity equation for the passive branch is:

$$\tau_{J_{Husky}} = -\tau_{TF_2} - \tau_{TF_4} - \tau_{TF_6} - \tau_{TF_8} - \tau_{B_{Husky}} \tag{112}$$

The compatibility equations for each loop formed by the temporary inclusion of each passive link are:

$$\omega_{TF_2} = \omega_{J_{Husky}} \tag{113}$$

$$\omega_{TF_4} = \omega_{J_{Husky}} \tag{114}$$

$$\omega_{TF_6} = \omega_{J_{Husky}} \tag{115}$$

$$\omega_{TF_8} = \omega_{J_{Husky}} \tag{116}$$

$$\omega_{B_{Husky}} = \omega_{J_{Husky}} \tag{117}$$

5.3 Mobile Robot Linear Graph Model and Parameter Estimation

5.3.1 Complete Mobile Robot Linear Graph Model

The complete Husky Robot LG model is produced by combining two sets of the electrical and drivetrain subsystem models (each representing one half of the Husky powertrain) with the models of the translational and rotational dynamic subsystems. The simplified drivetrain model was chosen as it demonstrated accurate results while also reducing the complexity of the state-space model and run-time of the GA-based parameter estimation process. This results in the LG model presented in Figure 5-10, which encompasses the dynamics of the entire Husky system.

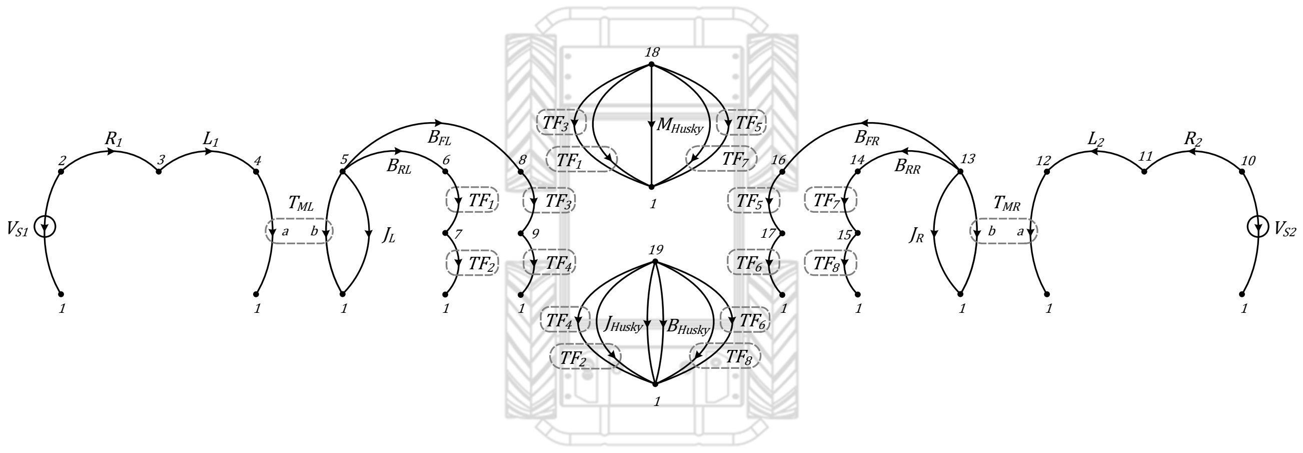


Figure 5-10:Complete LG Model of the Clearpath Husky Robot with Simplified Drivetrain Subsystems

By generating and evaluating equations (90)-(100),(107)-(117) for the various subsystem models, as well as, the corresponding equations for the right side powertrain, and the constitutive equations for each element, the LGtheory toolbox automated the process of creating the following state-space model of the Husky system (note that for these equations, the “Husky” subscript has been replaced with “H”):

$$\frac{\boldsymbol{dx}}{\boldsymbol{dt}} = \boldsymbol{A}\begin{bmatrix}\omega_{J_{LW}}\\ \omega_{J_{RW}}\\ v_{M_H}\\ \omega_{J_H}\\ i_{L_1}\\ i_{L_2}\end{bmatrix} + \boldsymbol{B}\begin{bmatrix}V_{s1}\\ V_{s2}\end{bmatrix} \tag{118}$$

$$\begin{bmatrix}\omega_{\mathrm{J_{LW}}}\\ \omega_{\mathrm{J_{RW}}}\\ \mathrm{v_{M_H}}\\ \omega_{\mathrm{J_H}}\end{bmatrix} = \boldsymbol{C}\begin{bmatrix}\omega_{J_{LW}}\\ \omega_{J_{RW}}\\ v_{M_H}\\ \omega_{J_H}\\ i_{L_1}\\ i_{L_2}\end{bmatrix} + \boldsymbol{D}\begin{bmatrix}V_{s1}\\ V_{s2}\end{bmatrix} \tag{119}$$

Where, the state-space matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, and $\boldsymbol{D}$ are:

$$\boldsymbol{A} = \begin{bmatrix} \frac{-B_{FL} - B_{RL}}{J_L} & 0 & \frac{B_{FL}TF_3 + B_{RL}TF_1}{J_L} & \frac{B_{FL}TF_4 + B_{RL}TF_2}{J_L} & \frac{T_{ML}}{J_L} & 0 \\ 0 & \frac{-B_{FR} - B_{RR}}{J_R} & \frac{B_{FR}TF_5 + B_{RR}TF_7}{J_R} & \frac{B_{FR}TF_6 + B_{RR}TF_8}{J_R} & 0 & \frac{T_{MR}}{J_R} \\ \frac{B_{FL}TF_3 + B_{RL}TF_1}{M_H} & \frac{B_{FR}TF_5 + B_{RR}TF_7}{M_H} & \frac{-B_{RL}TF_1^2 - B_{FL}TF_3^2 - B_{FR}TF_5^2 - B_{RR}TF_7^2}{M_H} & \frac{-B_{FL}TF_3TF_4 - B_{FR}TF_5TF_6 - B_{RL}TF_1TF_2 - B_{RR}TF_7TF_8}{M_H} & 0 & 0 \\ \frac{B_{FL}TF_4 + B_{RL}TF_2}{J_H} & \frac{B_{FR}TF_6 + B_{RR}TF_8}{J_H} & \frac{-B_{FL}TF_3TF_4 - B_{FR}TF_5TF_6 - B_{RL}TF_1TF_2 - B_{RR}TF_7TF_8}{J_H} & \frac{-B_{RL}TF_2^2 - B_{FL}TF_4^2 - B_{FR}TF_6^2 - B_{RR}TF_8^2 - B_H}{J_H} & 0 & 0 \\ -\frac{T_{ML}}{L_1} & 0 & 0 & 0 & -\frac{R_1}{L_1} & 0 \\ 0 & -\frac{T_{MR}}{L_2} & 0 & 0 & 0 & -\frac{R_2}{L_2} \end{bmatrix} \tag{120}$$

$$\boldsymbol{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1/L_1 & 0 \\ 0 & 1/L_2 \end{bmatrix} \tag{121}$$

$$\boldsymbol{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{122}$$

$$\boldsymbol{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{123}$$

The outputs of the state-space model where chosen to be the state-variables representing the rotational velocities of the left and right wheels, as well as, the linear and rotational velocities of the mobile robot itself. Choosing these outputs allows for a comparison of these states to the corresponding Gazebo simulated and experimental state values of the robot for the purposes of model validation and parameter estimation.

As mentioned in the background on LG theory provided in Chapter 1, a system containing eight transducers, such as the complete Husky LG model shown in Figure 5-10, has a possible 256 total normal tree configurations. While, with practice it can become much easier to manually identify the correct normal tree of a larger multi-domain system, the convenience of an automated tool such as LGtheory makes this process much easier, and ensures that no mistakes are made due to the possibility of human error associated with considering so many configurations. Similarly, with a total of 68 equations produced for the mobile robotic system, which must be reduced to the system order of six, this system would be extremely time consuming and impractical to be evaluated manually.

5.3.2 Parameter Estimation using Genetic Algorithms

GAs are a form of multi-point, population-based methods for simultaneously exploring multiple solutions to an optimization problem. Based on the same concepts as natural evolution, GAs reproduce and evolve members of a population, referred to as solutions, over many generations in order to obtain an optimized solution to a problem. Throughout this evolutionary process, new solutions inherit the beneficial characteristics from the successful solutions of past generations while also introducing new characteristics that may provide advantages over other solutions. Those solutions which are successful are more likely to reproduce, and thus, pass on their beneficial characteristics, while those that are less successful face the possibility of being purged from the population; this process tends to lead to a stronger population of solutions over many generations. GAs are stochastic in nature; therefore, they do not necessarily guarantee to find the most optimal solution to a problem; rather, this method is useful for more complex optimization problems that are difficult or unreasonable to solve through mathematical means [69].

For this work, the GA capabilities of MATLAB's Global Optimization Toolbox were utilized for the estimation of the unknown system parameters of the mobile robot's state-space model. The objective function of the parameter estimation GA is based on the sum of the absolute tracking errors in the x and y directions between the LG-based mobile robot simulation and the data obtained from Gazebo and from physical experiments with the Husky robot.

$$obj = \sum |x_{data}(t) - x_{LG}(t)| + \sum |y_{data}(t) - y_{LG}(t)| \tag{124}$$

While most of the system parameters of the Husky robot can be determined from manufacturer documentation or from component datasheets, some system parameters, which are based on specific environmental conditions, must be calibrated for. These parameters, which will be calibrated using a GA, are the unknown damping constant values shown in Table 5-1 representing the losses of the drivetrain systems due to slip and friction (B_{LW}, B_{RW}), and the resistance to rotational movement between the Husky vehicle and the driving surface (B_{Husky}).

Additionally, the GA will calibrate the coefficient value (c) of a simple multivariable function which is used to estimate the motor voltage signals from the recorded command velocity signals sent to the robot. Since real-time measurements of the motor voltages cannot be obtained at a fast enough rate from ROS for the physical experiments, the following functions have to be utilized in order to estimate the voltage inputs to the state-space model:

$$V_{s1} = 24 \cdot (c \cdot v_t + 0.541 \cdot c \cdot v_r) \quad (125)$$

$$V_{s2} = 24 \cdot (c \cdot v_t - 0.541 \cdot c \cdot v_r) \quad (126)$$

Where v_t and v_r are the translational velocity and rotational velocity command values, respectively. It was found through experimentation with the command signals to the motor that a rotational velocity command which was numerically equivalent to a translational velocity command would result in 54.1% rotational speed of the husky wheels; hence, the inclusion of the 0.541 values in equations (125) and (126).

Table 5-1 shows the known and unknown parameter values for the mobile robot state-space model:

Table 5-1: State-Space Parameters of Mobile Robot LG Model Including Unknown Values

Description	Parameter	Value	Units
Voltage Inputs	V_{s1} , V_{s2}	±24	V
Internal Motor Resistance	R_1 , R_2	0.46	Ω
Internal Motor Inductance	L_1 , L_2	0.22	mH
Motor Torque Constant	k_t	0.044488	$N \cdot m/A$
Gearbox Ratio	GR	78.71 : 1	Gear Ratio
Motor Transformer Ratio	T_{ML} , T_{MR}	$k_t \times GR$	$N \cdot m/A$
Drivetrain Inertia	J_{LW} , J_{RW}	0.08	$kg \cdot m^2$
Drivetrain Damping	B_{RL} , B_{FL} , B_{FR} , B_{RR}	Unknown	$rad/(N \cdot m \cdot s)$
Power Conversion	TF_{odd}	Equation (101)	
Transformer Ratios	TF_{even}	Equation (102)	
Husky Mass	M_{Husky}	48.39	kg
Husky Rotational Damping	B_{Husky}	Unknown	$rad/(N \cdot m \cdot s)$
Husky Inertia	J_{Husky}	3.0556	$kg \cdot m^2$

5.4 Results and Discussion

5.4.1 ROS Gazebo Simulation Environment

Gazebo is a physics-based 3D environment for simulating the rigid-body dynamics of robotic systems. Along with dynamic simulations, Gazebo can be used for simulating sensor readings, evaluating and training AI control systems, and much more. Integration of Gazebo with ROS allows for the data transfer and communication capabilities which ROS facilitates. These capabilities allow for recording of the command signals and sensor data of the robot using the following ROS topics: /tf, the transformation frames of the robot used for linear speed and trajectory; /imu_um7/data, the IMU data readings for rotational speed of the husky; /joint_states, the encoder readings of each wheel; and /husky_velocity_controller/cmd_vel, the command signals sent to the robot from the gamepad controller used to drive the robot. ROS then allows this recorded data to be filtered for the command data and replayed as input to a Gazebo simulation in order to attempt replicating the results.

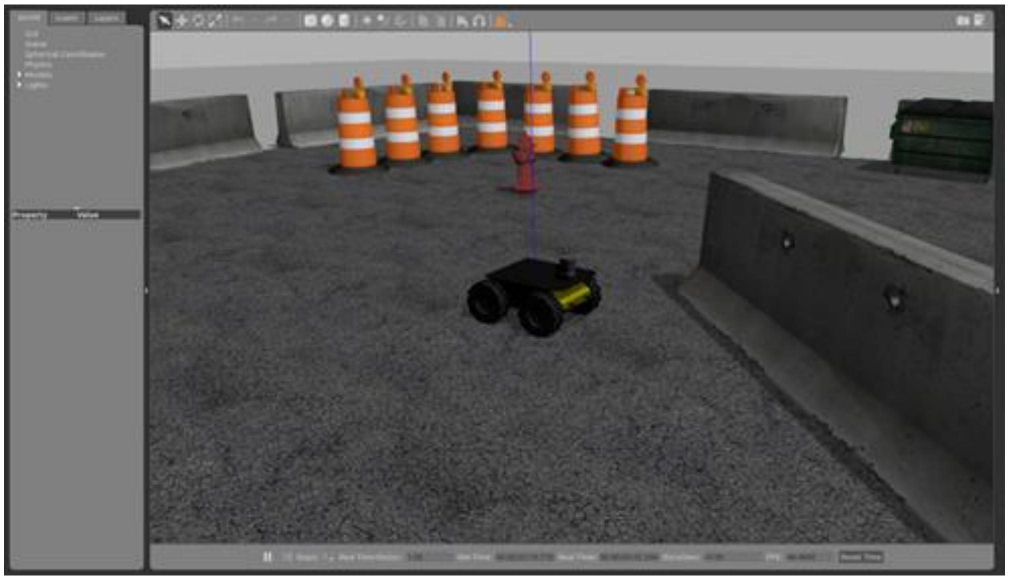


Figure 5-11: Example of the Gazebo User Interface and Simulation Environment

5.4.2 Genetic Algorithm Estimation and Trajectory Results Comparisons

Initial calibration of the Husky LG model was conducted using data collected from the physical robot while completing a circular maneuver. This maneuver involved an initial straight trajectory before entering a circular trajectory. Once one full revolution was complete, the Husky exited the circle again in a straight path.

Table 5-2 shows the variables being adapted by the GA parameter estimation procedure, the upper and lower bounds of their searches, and the results obtained which gave the optimal simulation results:

Table 5-2: LG Model Parameters Adapted by the GA with Resulting Values

Variable	$B_{FL}, B_{RL}, B_{RR}, B_{FR}$	B_{Husky}	c
Upper Bounds	1	1	0.75
Lower Bounds	100	100	1.00
Results	1.3016	12.8650	0.8961

Figure 5-12 shows the evolution of the parameter estimation GA used to calibrate the model, which was run with a population of 100 solutions, for a maximum of 100 generations, and a crossover fraction of 0.5.

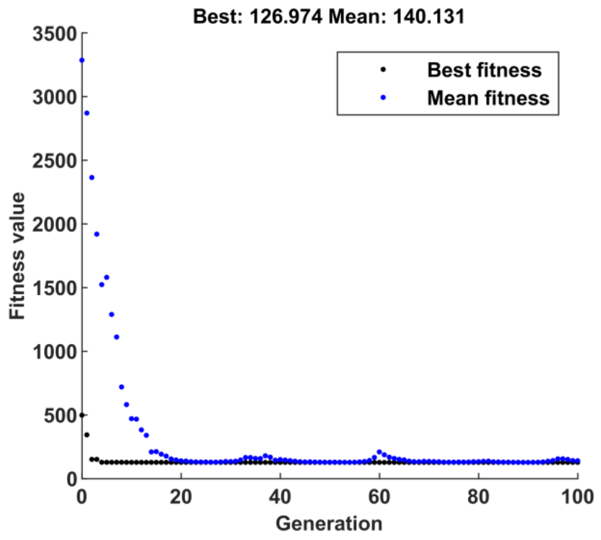


Figure 5-12: GA Evolution of the Parameter Estimation Procedure

Figure 5-13 demonstrates the trajectory response and tracking error of the calibrated LG model against the trajectory of the physical robot, as well as, the Gazebo simulated robot for the circular calibration maneuver.

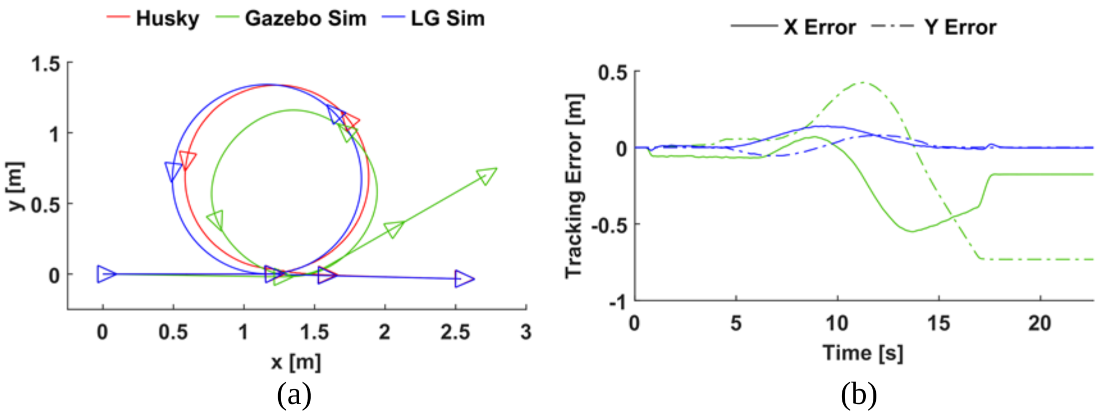


Figure 5-13: (a) Trajectory and (b) Tracking Error Results of LG and Gazebo Simulations Against Experimental Data for a Circle Maneuver

As can be observed from this figure, the trajectory of the LG model closely resembles the trajectory of the real robot; whereas, the trajectory of the Gazebo simulation differs significantly despite the same system inputs being used for both simulation approaches. The bounds of the maximum tracking error of the LG based simulation for this maneuver are:

$$|\bar{X}| \leq 0.1397[m] \qquad |\bar{Y}| \leq 0.0819[m]$$

Figure 5-14 demonstrates the trajectory response and tracking error of the previously calibrated LG model and Gazebo simulations against the physical robot for an S-bend maneuver.

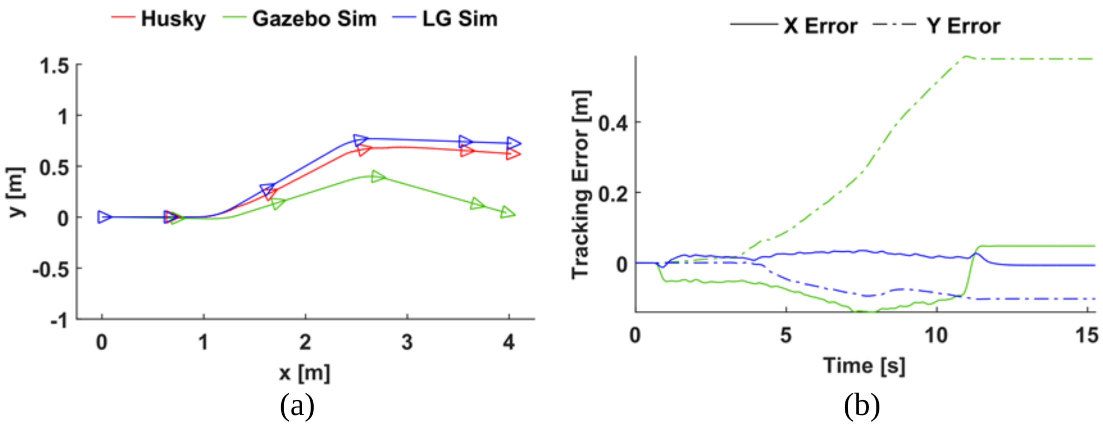
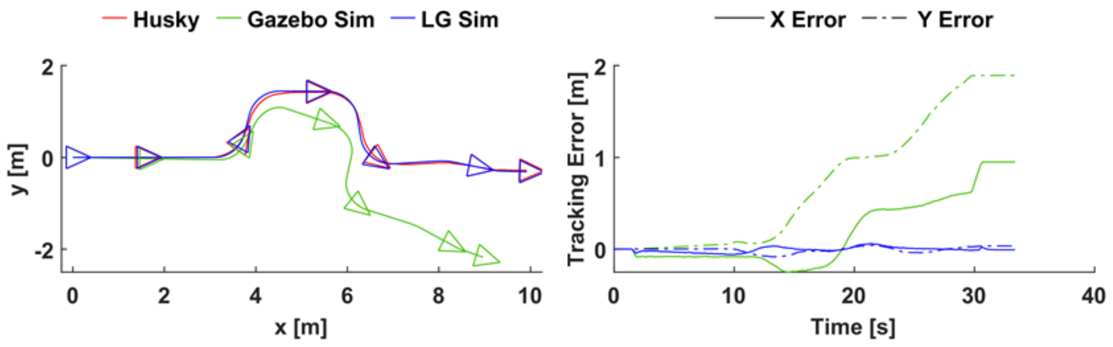


Figure 5-14: (a) Trajectory and (b) Tracking Error Results of LG and Gazebo Simulations Against Experimental Data for an S-Bend Maneuver

As can be observed from this figure, the trajectory of the LG model closely resembles the trajectory of the real robot, while, again, the trajectory of the Gazebo simulation differs more significantly. The bounds of the maximum tracking error of the LG based simulation for this maneuver are:

$$|\bar{X}| \leq 0.0352[m] \qquad |\bar{Y}| \leq 0.1027[m]$$

Figure 5-15 demonstrates the trajectory response and tracking error of the previously calibrated LG model and Gazebo simulations against the physical robot for an obstacle avoidance maneuver.



(a) (b)

Figure 5-15: (a) Trajectory and (b) Tracking Error Results of LG and Gazebo Simulations Against Experimental Data for an Obstacle Avoidance Maneuver

Again, the trajectory of the LG model closely resembles the trajectory of the real robot; whereas, the trajectory of the Gazebo simulation differs significantly after the initial turn. The bounds of the maximum tracking error of the LG based simulation for this maneuver are:

$$|\bar{X}| \leq 0.0613[m] \qquad\qquad |\bar{Y}| \leq 0.0854[m]$$

Additional validation of the LG model can be observed by graphing the output variables of the LG model against the corresponding data measurements from the sensors of the mobile robot for the obstacle avoidance maneuver, as shown in Figure 5-16. From this figure, a strong conformance can be observed between the response of the simulated output states and the Husky sensor readings despite the fact that the LG model was not calibrated for this specific maneuver.

In each of the three demonstrated maneuvers, the calibrated LG model provides a significantly more accurate trajectory response than the Gazebo Simulation. The primary cause of error for the Gazebo simulation is the excessive skidding which can be observed during the rotational movements of the Husky during the simulation. For the circling maneuver, this excessive skidding resulted in a smaller, oval-shaped trajectory, compared to the larger, circular trajectory of the LG model. Similarly, for the S-bend and obstacle avoidance maneuvers, the excessive skidding present in the Gazebo simulation has resulted in skewing of the Gazebo robot's otherwise accurate trajectories. While some noticeable error is present in the LG model trajectory, it is much less than that of the Gazebo simulations. The cause of this minimal error is likely due to the complexity associated with modeling the dynamics of the interactions between wheels of a skid steer vehicle and the driving surface, something which could be further addressed and improved in future work.

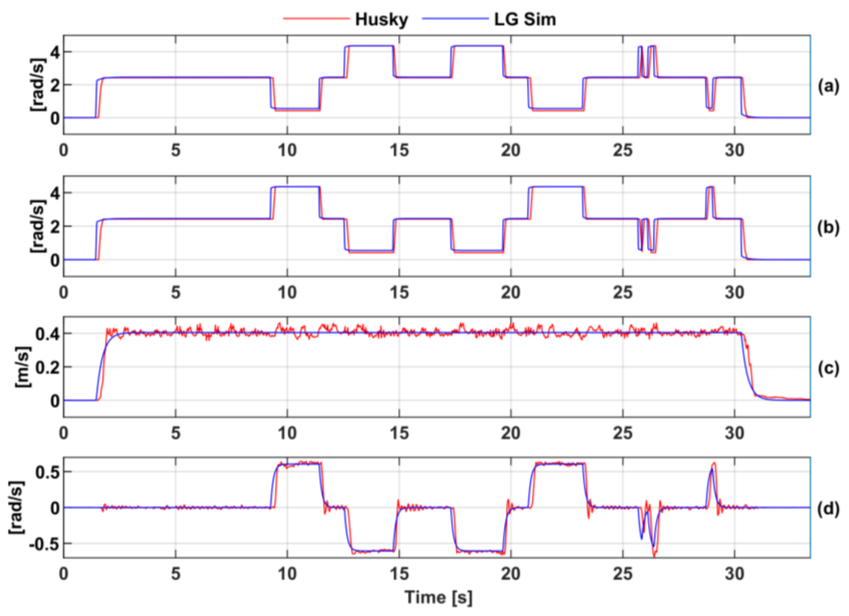


Figure 5-16: Response of the (a) Left and (b) Right Side Wheel Velocities, and the (c) Linear and (d) Rotational Velocities of the Husky LG model Against the Experimental Husky Data

5.5 Summary

In this chapter, an LG based method for modeling the dynamics of a mobile robotic system was presented. For this work, the in-house developed LGtheory MATLAB toolbox was utilized in order to automate the process of deriving the state-space model of such a complex system, and the Global Optimization Toolbox's GA capabilities were employed for calibration of the unknown system parameter values. The results of a comparison between a simulation of the LGtheory produced a state-space model for the mobile robot against a ROS/Gazebo-based simulation and experimental data collected while driving the Clearpath Husky mobile robot demonstrates the accuracy of the application of this modeling approach to a mobile robotic system. The successful application of this modeling approach to a mobile robotic system demonstrated in this work provides further validation of both the proposed LG model and the custom software used to construct it.

In the final chapter, the conclusions of this thesis will be presented along with an evaluation of how the objectives of this work have been met. In addition, recommendations for future work related to the research described in this thesis shall be provided.

Chapter 6. Conclusions and Future Work

6.1 Conclusions

While each energy domain tends to have its own independent methods of modeling, the graph-based methods of dynamic system modeling provide techniques for the evaluation of multi-domain systems in an integrated and unified manner. The literature of two prominent graph-based modeling approaches were evaluated in this thesis, and while the BG approach demonstrated more applications and research in the fields of mechatronics and robotics, the benefits and potential of the LG approach to these fields was also highlighted. The likely cause of such discrepancy in research between these two approaches was identified as the abundance of available BG-based software tools, and a complete lack of LG-based software to facilitate the evaluation of larger, and more complex systems using this method.

To address this, the LGtheory MATLAB toolbox was developed to provide a method of automating the derivation of state-space models using the LG approach. This toolbox provides users with a robust set of commands for the evaluation of LG models, as well as, the results for all steps of the LG evaluation process, including the constitutive, continuity, compatibility, and state-space equations of the system. For validation, the state-space output of the LGtheory toolbox for an electromechanical system was simulated and evaluated against an equivalent model produced in Simulink Simscape. The results of these simulations demonstrated strong agreement between these two approachs; thus, validating the accuracy of the LGtheory produced model.

Next, integration of the LGtheory toolbox with two forms of evolutionary computing (GP and GA) was performed in order to facilitate research into evolution-based design and optimization of LG models.

Through the integration of the LG and GP approaches, a methodology was proposed for the computer-automated design of mechatronic systems. Validation of this combined approach was demonstrated through the application of automatic electronic filter design. Results of this demonstration show that the proposed methodology was successful in designing low pass, high pass, and band pass filter circuits. These results demonstrate the potential of this methodology for applications related to the automatic design and modeling of more complex multi-domain systems.

An integrated LG and GA approach was then applied to the field of mobile robotics for the modeling, calibration, and simulation of a 4-wheeled skid steer robot. The benefits of the LGtheory toolbox are highlighted by this application as the complexity and size of this robotic system model would make manual evaluation impractical. The results of the trajectory response for a simulation of the proposed LG model was then compared to a ROS Gazebo simulation and experimental data collected from the robot for validation. This comparison demonstrated a much stronger agreement between the trajectories of the calibrated LG model and the physical mobile robotic system than that of the ROS Gazebo simulation; thus, validating the accuracy of the produced model.

In conclusion, the LGtheory toolbox addresses the gap identified in available LG-based software and provides a robust method for automatically evaluating LG models in the MATLAB programming environment. The goal of the development and applications of the toolbox presented in this thesis are to spark greater interest and facilitate further advancements in the field of LG research and education.

6.2 Future Work

Recommendations for future work related to the LGtheory toolbox and its applications are as follows:

6.2.1 LGtheory Toolbox

- Development of a user interface for direct graphical input of LG models
- Further development and maintenance to accommodate continued used of the toolbox
- Eventual open-source release of the toolbox to public

6.2.2 Automated Design Evolution using Linear Graphs

- Expansion of presented work to the automated design of the multi-domain system
- Exploration of automated design methodology with other machine learning techniques

6.2.3 Dynamic Modeling of Skid-Steer Mobile Robot using Linear Graphs

- Implementation of a more advanced tire and driving surface interaction model in order to address error still present in the currently proposed LG model

Bibliography

[1] L. Euler, "Solutio problematis ad geometriam situs pertinentis," *Commentarii academiae scientiarum Petropolitanae*, vol. 8, pp. 128-140, 1741.

[2] 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.

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

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

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

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

[7] M. Ganji, S. Behbahani and C. W. De Silva, "Integrated and object-oriented mechatronic modelling of piezoelectric transducers using linear graphs," *International Journal of Modelling and Simulation,* vol. 33, no. 1, pp. 15-24, 2013.

[8] D. Rowell and D. N. Wormley, System Dynamics: An Introduction, Upper Saddle River: Prentice Hall, 1997.

[9] 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.

[10] A. Adade Filho and J. Bosco Goncalves, "Automatic modelling of lumped parameters dynamic systems," *Controle & Automacao*, vol. 10, no. 1, pp. 1-12, 1999.

[11] J. McPhee, "On the use of linear graph theory in multibody system dynamics," *Nonlinear Dynamics*, vol. 9, no. 1-2, pp. 73-90, 1996.

[12] 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.

[13] 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.

[14] K. R. Uren and G. Van Schoor, "State space model extraction of thermohydraulic systems – Part I: A linear graph approach," *Energy*, vol. 61, pp. 368-380, 2013.

[15] K. R. Uren and G. Van Schoor, "State space model extraction of thermohydraulic systems – Part II: A linear graph approach applied to a Brayton cycle-based power conversion unit," *Energy*, vol. 61, pp. 381-396, 2013.

[16] 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.

[17] 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.

[18] C. W. De Silva, "Linear-graph modeling paradigms for mechatronic systems," *nternational Journal of Mechanical Engineering Education,* vol. 40, no. 4, pp. 305-328, 2012.

[19] A. S. Arifin, A. N. Poo, K. K. Tan and C. W. de Silva, "Evolution of a manipulator model using linear graphs and genetic algorithms," in *4th Asia International Symposium on Mechatronics*, Singapore, 2009.

[20] S. Gao and C. W. de Silva, "Formulation of a state-space model for a parallel manipulator using linear graphs," in *9th International Conference on Computer Science & Education*, Vancouver, 2014.

[21] L. B. Gamage, C. W. De Silva and R. Campos, "Design evolution of mechatronic systems through modeling, on-line monitoring, and evolutionary optimization," *Mechatronics,* vol. 22, no. 1, pp. 83-94, 2011.

[22] H. M. Paynter, Analysis and Design of Engineering Systems, Cambridge: The MIT Press, 1961.

[23] H. M. Paytner, "The Gestation and Birth of Bond Graphs," 2000. [Online]. Available: https://www.me.utexas.edu/~longoria/paynter/hmp/Bondgraphs.html. [Accessed August 2018].

[24] D. Karnopp and R. Rosenberg, Analysis and Simulation of Multiport Systems, Cambridge: The MIT Press, 1968.

[25] D. Karnopp and R. Rosenberg, "Application of Bond Graph Techniques to the Study of Vehicle Drive Line Dynamics," *Journal of Basic Engineering,* vol. 92, no. 2, pp. 355-362, 1970.

[26] R. Rosenberg and D. Karnopp, "Definition of the Bond Graph Language," *Journal of Dynamic Systems, Measurement, and Control,* vol. 92, no. 3, pp. 179-182, 1972.

[27] D. Karnopp, "Bond Graph Models for Fluid Dynamic Systems," *Journal of Dynamic Systems, Measurement, and Control,* vol. 94, no. 3, pp. 222-229, 1972.

[28] R. Rosenberg, "The Bond Graph as a Unified Data Base for Engineering System Design," *Journal of Engineering for Industry,* vol. 97, no. 4, pp. 1333-1337, 1975.

[29] D. M. Auslander, N. T. Tsai and F. Farazian, "Bond Graph Models for Torsional Energy Transmission," *Journal of Dynamic Systems, Measurement, and Control,* vol. 97, no. 1, pp. 53-59, 1975.

[30] D. M. Auslander, T. E. Lobdell and D. Chong, "A Large-Scale Model of the Human Cardiovascular System and its Application to Ballistocardiography," *Journal of Dynamic Systems, Measurement, and Control,* vol. 94, no. 3, pp. 230-238, 1972.

[31] J. R. Ort and H. R. Martens, "A Topological Procedure for Converting a Bond Graph to a Linear Graph," *Journal of Dynamic Systems, Measurement, and Control,* vol. 96, no. 3, pp. 307-314, 1974.

[32] A. Alabakhshizadeh, Y. Iskandarani, G. Hovland and O. Midtgård, "Analysis, modeling and simulation of mechatronic systems using the Bond Graph method," *Modeling, Identification and Control,* vol. 32, no. 1, pp. 35-45, 2011.

[33] M. M. Arezki, A. Smail and B. Djamel, "Modeling and simulation of mechatronic system to integrated design of supervision: using a bond graph approach," *Applied Mechanics and Materials,* vol. 86, pp. 467-470, 2011.

[34] F. Wu, H. He and Y.-M. Deng, "Applying bond graphs for design representation of mechatronic products," *Applied Mechanics and Materials,* Vols. 201-202, no. pt. 1, pp. 537-540, 2012.

[35] R. Chhabra and M. Reza Emami, "Holistic system modeling in mechatronics," *Mechatronics,* vol. 21, no. 1, pp. 166-175, 2011.

[36] M. Habibi and A. B. Novinzadeh, "Modeling and Designing an Intelligent Controller Using Bond Graph for a Satellite Controlled by Magnetic Actuators," *International Journal of Intelligent Mechatronics and Robotics*, vol. 2, no. 1, pp. 72-90, 2012.

[37] S. R. Sahoo and S. S. Chiddarwar, "Dynamic Modelling of Four Wheel Skid Mobile Robot by Unified Bond Graph Approach," in *2016 International Conference on Robotics: Current Trends and Future Challenges*, Thanjavur, 2016.

[38] S. R. Sahoo, S. S. Chiddarwar and V. Alakshendra, "Dynamic modelling and simulation of a three-Wheeled Omnidirectional Mobile Robot: Bond graph approach," in *Proceedings of the Advances in Robotics, AIR 2017*, New Delhi, 2017.

[39] J. Cui, Y. Ren, D. Yang and S. Zeng, "Reliability analysis of a novel design of pose deformation system for mobile robots through Bond Graph and Simulink," in *2016 IEEE Industry Applications Society Annual Meeting*, Portland, 2016.

[40] R. Loureiro, R. Merzouki and B. O. Bouamama, "Bond Graph Model Based on Structural Diagnosability and Recoverability Analysis: Application to Intelligent Autonomous Vehicles," *IEEE Transactions on Vehicular Technology,* vol. 61, no. 3, pp. 986-997, 2012.

[41] Y. Touati, R. Merzouki and B. O. Bouamama, "Robust diagnosis to measurement uncertainties using bond graph approach: Application to intelligent autonomous vehicle," *Mechatronics,* vol. 22, no. 8, pp. 1148-1160, 2012.

[42] L. Patnaik and L. Umanand, "inematics and dynamics of Jansen leg mechanism: A bond graph approach," *Simulation Modelling Practice and Theory,* vol. 60, pp. 160-169, 2016.

[43] A. I. Gal, L. Vladareanu and R. I. Munteanu, "Sliding motion control with bond graph modeling applied on a robot leg," *Revue Roumaine des Sciences Techniques, Serie Electrotechnique et Energetique,* vol. 60, no. 2, pp. 215-224, 2015.

[44] M. M. Gor, P. M. Pathak, A. K. Samantaray, J.-M. Yang and S. W. Kwak, "Control oriented model-based simulation and experimental studies on a compliant legged quadruped robot," *Robotics and Autonomous Systems,* vol. 72, pp. 217-234, 2015.

[45] M. M. Gor, P. M. Pathak, A. K. Samantaray, J.-M. Yang and S. W. Kwak, "Fault accommodation in compliant quadruped robot through a moving appendage mechanism," *Mechanism and Machine Theory,* vol. 121, pp. 228-244, 2018.

[46] M. M. Gor, P. M. Pathak, A. K. Samantaray, J.-M. Yang and S. W. Kwak, "Fault-tolerant control of a compliant legged quadruped robot for free swinging failure," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering,* vol. 232, no. 2, pp. 161-177, 2018.

[47] M. Roman, D. Popescu and D. Seliteanu, "An interactive teaching system for bond graph modeling and simulation in bioengineering," *Educational Technology & Society,* vol. 16, no. 4, pp. 17-31, 2013.

[48] L. Huang, G. Cheng, G. Zhu and D. Li, "Development of a bond graph based model library for turbocharged diesel engines," *Energy,* vol. 148, pp. 728-743, 2018.

[49] S. Behbahani and C. W. de Silva, "Mechatronic design evolution using bond graphs and hybrid genetic algorithm with genetic programming," *IEEE/ASME Transactions on Mechatronics,* vol. 18, no. 1, pp. 190-199, 2013.

[50] B. L. Samarakoon, L. B. Gamage and C. W. de Silva, "Design evolution of engineering systems using bond graphs and genetic programming," *Mechatronics,* vol. 33, pp. 71-83, 2016.

[51] A. Arshad, A. S. Kashif, M. Wasim, N. Mazhar and S. M. Tahir Zaidi, "Modelling inspiration and expiration mechanism of lungs using bond graph," in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, Vancouver, 2018.

[52] N. Villa-Villaseñor and R. Galindo-Orozco, "Bond graph modelling of a 4-parameter photovoltaic array," *Mathematical and Computer Modelling of Dynamical Systems,* vol. 24, no. 3, pp. 275-295, 2018.

[53] A. Mohammed and H. G. Lemu, "Wind turbine modelling using bond graph method," in *Advanced Manufacturing and Automation VIII. Lecture Notes in Electrical Engineering*, Singapore, 2019.

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

[55] The MathWorks, Inc., "Multidimensional Arrays," 2020. [Online]. Available: https://www.mathworks.com/help/matlab/math/multidimensional-arrays.html. [Accessed 3 March 2020].

[56] M. Scherrer and J. McPhee, "Dynamic modelling of electromechanical multibody systems," *Multibody System Dynamics,* vol. 9, no. 1, pp. 87-115, 2003.

[57] S. Silva and J. Almeida, "GPLAB - A Genetic Programming Toolbox for MATLAB," 2008.

[58] J. R. Koza, "Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems," Stanford University, Stanford, 1990.

[59] M. T. Ahvanooey, Q. Li, M. Wu and S. Wang, "A Survey of Genetic Programming and Its Applications," *KSII Transactions on Internet and Information Systems,* vol. 13, no. 4, pp. 1765-1794, 2019.

[60] J. R. Koza, F. H. Bennett III, D. Andre and M. A. Keane, "Automated Design of Both the Topology and Sizing of Analog Electrical Circuits using Genetic Programming," in *Artificial Intelligence in Design '96*, Stanford, 1996.

[61] V. Durev and E. Gadjeva, "Passive Circuit Synthesis using Genetic Algorithms in MATLAB," in *9th WSEAS International Conference on Evolutionary Computing (EC'08)*, Sofia, 2008.

[62] O. J. Ushie, M. F. Abbod and J. C. Ogbulezie, "The Use of Genetic Programming to Evolve Passive Filter Circuits," *International Journal of Engineering and Technology Innovation,* vol. 7, no. 4, pp. 255-268, 2017.

[63] L. B. Gamage, C. W. de Silva and R. Campos, "Design Evolution of Mechatronic Systems Through Modeling, On-Line Monitoring, and Evolutionary Optimization," *Mechatronics,* vol. 22, no. 1, pp. 83-94, 2012.

[64] R. Poli, W. B. Langdon and N. F. McPee, A Field Guide to Genetic Programming, Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk. (With contributions by J. R. Koza), 2008.

[65] S. Silva, "GPLAB - A Genetic Programming Toolbox for MATLAB," June 2018. [Online]. Available: http://gplab.sourceforge.net/.

[66] S. R. Sahoo and S. S. Chiddarwar, "Dynamic Modelling of Four Wheel Skid Mobile Robot by Unified Bond Graph Approach," in *2016 International Conference on Robotics: Current Trends and Future Challenges*, Thanjavur, 2016.

[67] S. R. Sahoo, S. S. Chiddarwar and V. Alakshendra, "Dynamic modelling and simulation of a three-Wheeled Omnidirectional Mobile Robot: Bond graph approach," in *Proceedings of the Advances in Robotics, AIR 2017*, New Delhi, 2017.

[68] E. McCormick, H. Lang and C. W. de Silva, "Research and Development of a Linear Graph-based MATLAB Toolbox," in *The 14th International Conference on Computer Science & Education (ICCSE 2019)*, Toronto, 2019, pp. 942-947.

[69] F. O. Karray and C. De Silva, Soft Computing and Intelligent Systems Design, Harlow: Addison Wesley, 2004.