Appendix B: Matlab Toolbox for Linear Graphs

By Haoxiang Lang, Eric McCormick, and Clarence W. de Silva

This Appendix presents a custom Matlab toolbox, called LGtheory, which provides a robust and automated method for generating Linear Graph (LG) models of multi-domain (multi-physics or mixed) 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 to evaluate larger, more complex multi-domain systems. The Toolbox can be downloaded from the following link:

https://github.com/GRASP-ONTechU/Linear_Graph

B.1 Inputting a Linear Graph into the LGtheroy Toobox

B.1.1 LGtheory Tooblbox 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 using 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 using 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 as given in Table B.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 the system elements are correctly accounted for (specifically in the case of spring elements in the mechanical domain), and ensure that the appropriate variable

types are applied to each element. The energy domains are specified using the indexing values given in Table B.1:

TableB.1: Index Values of Element Types and Energy Domains for Inputs of LGtheory Toolbox.

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 correspond to 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 given in Table B.2.

Table B.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

B.1.2 LGtheory Toolbox Outputs

Similar to the inputs, the output of the function, to the user (Model), is a structure array containing various information used to construct the state-space model through the LG approach, as well as, the vectors and the 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 the system elements and the connection to the system nodes, and zeros to represent the absence of connection between the elements and the nodes.

`Tree` – Normal Tree Matrix:

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

`Branches` – Branches Vector:

A vector that defines the column-wise indexes and the 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 that represents the co-tree of the LG model in the form of an incidence matrix.

`Links` – Links Vector:

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

`Across_Vars` – Across-Variables Vector:

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

`Through_Vars` – Through-Variables Vector:

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

`Prime` – Primary-Variables Vector:

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

`Secon` – Secondary-Variables Vector:

A vector that 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 that 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 resistance 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 the A-Type elements in the normal tree and the through-variables of the 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, as defined in Table B.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 directly affect the outputs of the system.

`E` – Input Derivative Matrix

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

B.1.3 Suggested Best Practices

While the LGtheory toolbox is robust in relation to how it handles inputs, there are several best practices that should be employed by the user to ensure that they are providing the inputs to the LG model properly.

The user should specify the ground node in the source and the target vector input fields as a "1". This is required as MATLAB index arrays and matrices start from 1 instead of 0, making the work done by the toolbox much simpler. Each node added to the system will be incremented by 1. If a value for a node is skipped, or the ground node is not specified as "1", it will likely generate an error message or an incorrect result from the toolbox.

For systems that contain transfer elements (transformers or gyrators), the first (input) and the second (output) ports of these two-port elements should be placed in successive order within the toolbox input fields without any other transfer port elements in between. This is because the toolbox couples and evaluates the successive transfer 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 transfer element successively (immediately next to each other) in the LG inputs in order to ensure that they will be coupled and evaluated together and correctly. While the successive ordering of the transfer port elements is important, the order of the nodes to which they are attached are not required to be successive (i.e., port 1 can be connected to node 5 and ground, while port 2 is connected to node 8 and the ground).

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

When specifying the domain indices 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 the 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 (the primary domains are electrical, mechanical translational and translational, hydraulic/fluid, and thermal) they 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 sufficiently simplified to ensure that there are no redundant state-variables (in addition to the number governed by the system order). 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.

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

$$C_e = \frac{1}{\sum_i \frac{1}{C_i}} \tag{B.1}$$

$$L_e = \frac{1}{\sum_i \frac{1}{L_i}} \tag{B.2}$$

B.1.4 Example of Inputting a Linear Graph Model into the LGtheory Toolbox

A DC motor with an inertial load is used as an example now, to demonstrate the processes associated with the LGtheory toolbox for generating the state-space representation of an LG model.

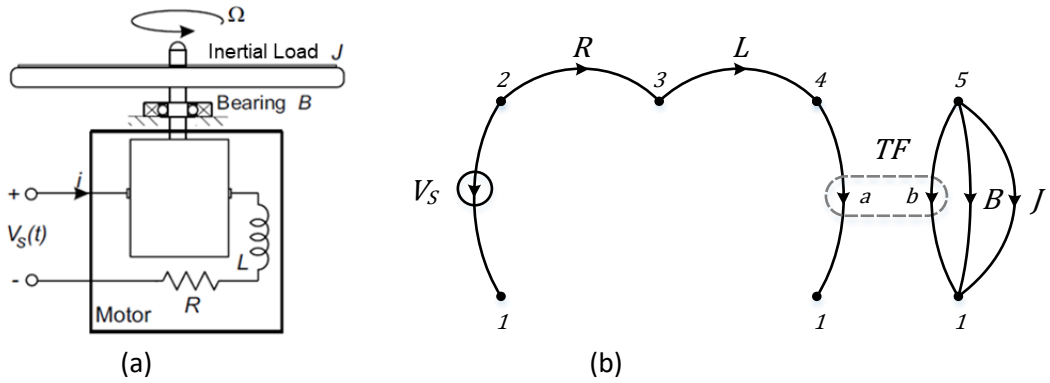

FIGURE B.1

(a) Schematic diagram and; (b) LG model of DC motor with inertial load.

Based on the LG model of the system presented in Figure B.1, the following inputs should 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 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 (angular velocity) of the inertial load (Omega_J(t)).

B.2 LGtheory Toolbox Functions

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

B.2.1 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 the target vectors for any node values that only occur once between the two vectors. If such a node is found, then it is known that the node has only a single element attached to it, meaning that the model contains an open loop.

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

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

B.2.2 Conversion to Incidence Matrix Representation

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

In order to mathematically represent the topology and directionality of an LG model 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 there are as nodes (determined by the highest number in the source and target vectors) and as many columns there are as elements (determined by the length of the source and the 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 indices corresponding to the source and the target vectors, respectively.

From the LG model presented in Figure B.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:

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

B.2.3 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 (see Appendix A). The normal tree is important in the LG approach as it allows for the classification of the primary and the secondary variables, and also 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 has the same size as the incidence matrix of the LG model but with a depth of 2^T, where T is the number of transfer 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 transfer elements, where each page (or layer) of the three-dimensional matrix represents a different permutation of the normal tree.

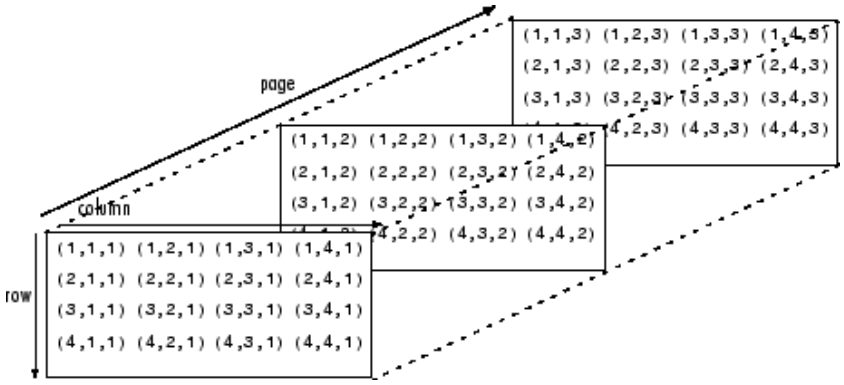

FIGURE B.2

An example of indexing values of a multidimensional array in MATLAB.

Next, the program adds all A-type (across-type) 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 matrix.

Transfer elements are then added to the normal tree incidence matrices with only one transfer 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 the transfer element branches are created throughout the 2^T layers.

The algorithm then cycles through the passive elements in the order: A-Type elements, D-Type elements, and T-Type elements. For each column-wise element that 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 T-type (through-type) source elements in every layer of the normal tree matrix. Since it is required that no T-type source elements are included in the normal tree, the layer index of any permutation of the normal tree matrix that requires the addition of a T-type source in order to be completed shall be flagged in a logical vector as an invalid normal tree configuration.

Once this process is completed for all elements of the LG model, each normal tree configuration is revaluated for any potential loops that may have been created as the result of the required inclusion of A-type source elements and transfer port elements. Similarly, each configuration is evaluated to determine whether all nodes are contained within the normal tree matrix. Any configuration that 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 that contain the most A-Type elements. If there are multiple layers that contain the most A-Type elements, the algorithm will then select the layer that contains the least T-Type elements, as this combination of most A-Type and least T-Type elements ensures that the program identifies the most independent energy storage elements in the system.

The normal tree resulting from this process for the example system is shown in Figure B.3, where the solid lines represent branches and the dashed lines represent links.

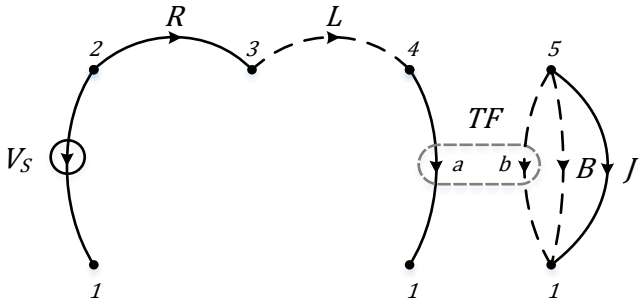

FIGURE B.3

Normal Tree of the DC Motor with Inertial Load LG model.

The resulting incidence matrix representation of the normal tree is:

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

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

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

B.2.4 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 B.2. For the present example, these two arrays would be as follows:

$$Across\_Vars = [V_s \ V_R \ V_L \ V_{TF_a} \ \omega_{TF_b} \ \omega_B \ \omega_J] \tag{B.6}$$

$$Through\_Vars = [i_s \ i_R \ i_L \ i_{TF_a} \ \mathrm{T}_{TF_b} \ \mathrm{T}_B \ \mathrm{T}_J] \tag{B.7}$$

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

$$Prime = [V_s \ V_R \ i_L \ V_{TF_a} \ \mathrm{T}_{TF_b} \ \mathrm{T}_B \ \omega_J] \tag{B.8}$$

$$Secon = [i_s \ i_R \ V_L \ i_{TF_a} \ \omega_{TF_b} \ \omega_B \ \mathrm{T}_J] \tag{B.9}$$

Finally, the program 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 input-variables as a vector containing the source variable type of each source element:

$$\boldsymbol{x} = \left[\omega_J \ i_L\right]^T \tag{B.10}$$

$$\boldsymbol{u} = [V_s]^T \tag{B.11}$$

B.2.5 Constitutive Equations

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

The constitutive equations of the system are created for all passive elements. 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:

$$\begin{aligned} \frac{d\omega_J}{dt} &= \frac{1}{J}\,\mathrm{T}_J \\ \frac{di_L}{dt} &= \frac{1}{L}\,V_L \\ V_R &= R \cdot i_R \\ \mathrm{T}_B &= B \cdot \omega_B \\ V_{TF_a} &= TF \cdot \omega_{TF_b} \\ \mathrm{T}_{TF_b} &= -TF \cdot i_{TF_a} \end{aligned} \tag{B.12}$$

B.2.6 Network Equations

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

The continuity (node) equations of an LG model are formed using the contouring method. This method involves "virtual cutting" around a node or a set of nodes (as shown in Figure B.4 for the contour of the *K* element) in such a way that only a single branch is intersected by the contour. This contour can then be treated in a similar manner as a junction in the Kirchhoff's Current Law, where the sum of all the through-variables entering and exiting the contour is equal to zero. A continuity equation is constructed for each passive branch of the normal tree, where each equation is rearranged to isolate the secondary variable of the passive branch.

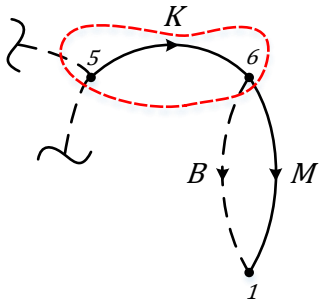

FIGURE B.4

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 the 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 that is not contained in the normal tree, where each equation is rearranged to isolate the passive link's secondary variable.

Figure B.5 illustrates the contours of each passive normal tree branch (shown by red broken lines), and the re-inclusion of the passive co-tree links (shown by light blue lines) to the LG model.

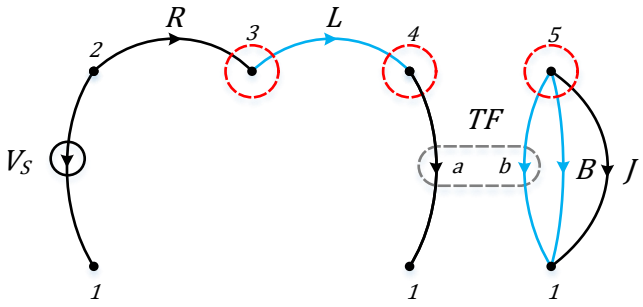

FIGURE B.5

LG Model with Contour and Temporary Link Inclusion Overlay.

While the manual process of identify and evaluating the loops and contours of this model is simple, the computer-automation of the process, using a program, for recognizing these patterns can be much more difficult. So, in order to accomplish this using a computer program, the concept of the fundamental cut set is employed.

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

$$\boldsymbol{A_{tr}} = \begin{matrix} \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} \begin{matrix} \begin{matrix} V_s & R & TF_a & J \end{matrix} \\ \begin{bmatrix} -1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \end{matrix} \tag{B.13}$$

$$\boldsymbol{A_{co}} = \begin{matrix} \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} \begin{matrix} \begin{matrix} L & TF_b & B \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & -1 & -1 \end{bmatrix} \end{matrix} \tag{B.14}$$

Next, the fundamental cut set matrix is found as:

$$\boldsymbol{F_c} = [\boldsymbol{H} \quad \boldsymbol{I}] \tag{B.15}$$

where,

$$\boldsymbol{H} = \boldsymbol{A}_{\boldsymbol{tr}}^{-\mathbf{1}}\boldsymbol{A}_{\boldsymbol{co}} \tag{B.16}$$

Here, $\boldsymbol{I}$ is an identity matrix of corresponding size to $\boldsymbol{H}$. For the present example, the fundamental cut set matrix will be:

$$\boldsymbol{F}_{\boldsymbol{c}} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.17}$$

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

$$\boldsymbol{v}_{\boldsymbol{tr}} = \begin{bmatrix} V_S \\ V_R \\ V_{TF_a} \\ \omega_J \end{bmatrix}, \quad \boldsymbol{v}_{\boldsymbol{co}} = \begin{bmatrix} V_L \\ \omega_{TF_b} \\ \omega_B \end{bmatrix} \tag{B.18}$$

$$\boldsymbol{f}_{\boldsymbol{tr}} = \begin{bmatrix} i_S \\ i_R \\ i_{TF_a} \\ \mathrm{T}_J \end{bmatrix}, \quad \boldsymbol{f}_{\boldsymbol{co}} = \begin{bmatrix} i_L \\ \mathrm{T}_{TF_b} \\ \mathrm{T}_B \end{bmatrix} \tag{B.19}$$

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

$$\boldsymbol{f}_{\boldsymbol{tr}} = -\boldsymbol{H}\boldsymbol{f}_{\boldsymbol{co}} \tag{B.20}$$

For the present example, this would result in:

$$\begin{bmatrix} i_S \\ i_R \\ i_{TF_a} \\ \mathrm{T}_J \end{bmatrix} = -\begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} i_L \\ \mathrm{T}_{TF_b} \\ \mathrm{T}_B \end{bmatrix} \tag{B.21}$$

$$\begin{gathered} i_S = -i_L \\ i_R = i_L \\ i_{TF_a} = i_L \\ \mathrm{T}_J = -\mathrm{T}_{TF_b} - \mathrm{T}_B \end{gathered} \tag{B.22}$$

Subsequently, the compatibility equation for each passive link can be found as:

$$\boldsymbol{v_{co} = H^T v_{tr}} \tag{B.23}$$

For the present example, this would result in:

$$\begin{bmatrix} V_L \\ \omega_{TF_b} \\ \omega_B \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_s \\ V_R \\ V_{TF_a} \\ \omega_J \end{bmatrix} \tag{B.24}$$

$$\begin{aligned} V_L &= V_s - V_R - V_{TF_a} \\ \omega_{TF_b} &= \omega_J \\ \omega_B &= \omega_J \end{aligned} \tag{B.25}$$

B.2.7 Creating the State-Space Matrices

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

On 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 (redundant, auxiliary) variables.

$$\frac{d\omega_J}{dt} = \frac{-\mathrm{T}_B - \mathrm{T}_{TF_b}}{J} \tag{B.26}$$

$$\frac{di_L}{dt} = \frac{-V_R - V_{TF_a} + V_s}{L} \tag{B.27}$$

$$V_R = R \cdot i_L \tag{B.28}$$

$$\mathrm{T}_B = B \cdot \omega_J \tag{B.29}$$

$$V_{TF_a} = TF_a \cdot \omega_J \tag{B.30}$$

$$\mathrm{T}_{TF_b} = -TF_a \cdot i_L \tag{B.31}$$

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 present example, the column vector $\boldsymbol{x}$ will consist of equations (3.26) and (3.27), the column vector $\boldsymbol{d}$ will be empty as there are no dependent storage elements (as determined by the normal tree), and the column vector $\boldsymbol{p}$ will consist of equations (B.27)-(B.31). These vectors can be written as the following matrix equations:

$$\boldsymbol{\dot{x} = Px + Qp + Rd + Su} \tag{B.32}$$

$$\boldsymbol{d} = \boldsymbol{M}\dot{\boldsymbol{x}} + \boldsymbol{N}\dot{\boldsymbol{u}} \tag{B.33}$$

$$\boldsymbol{p} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{J}\boldsymbol{p} + \boldsymbol{K}\boldsymbol{d} + \boldsymbol{L}\boldsymbol{u} \tag{B.34}$$

Therefore,

$$\dot{\boldsymbol{x}} = [0]_{2\times 2}\boldsymbol{x} + \begin{bmatrix} 0 & -\frac{1}{J} & 0 & -\frac{1}{J} \\ -\frac{1}{L} & 0 & -\frac{1}{L} & 0 \end{bmatrix}\boldsymbol{p} + [0]\boldsymbol{d} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix}\boldsymbol{u} \tag{B.35}$$

$$\boldsymbol{d} = [0]\dot{\boldsymbol{x}} + [0]\dot{\boldsymbol{u}} \tag{B.36}$$

$$\boldsymbol{p} = \begin{bmatrix} 0 & R \\ B & 0 \\ TF_a & 0 \\ 0 & -TF_a \end{bmatrix}\boldsymbol{x} + [0]_{4\times 4}\boldsymbol{p} + [0]\boldsymbol{d} + [0]_{4\times 1}\boldsymbol{u} \tag{B.37}$$

The general solution to the state-space equation is formed by isolating $\boldsymbol{d}$ in (3.33) and $\boldsymbol{p}$ in (3.34), and substituting the results into (3.32). Once simplified, this process results in the following general formulation of the state-space model:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{E}\dot{\boldsymbol{u}} \tag{B.38}$$

where,

$$\boldsymbol{A} = [\boldsymbol{I} - (\boldsymbol{Q}\boldsymbol{K}' + \boldsymbol{R})\boldsymbol{M}]^{-1}(\boldsymbol{P} + \boldsymbol{Q}\boldsymbol{H}') \tag{B.39}$$

$$\boldsymbol{B} = [\boldsymbol{I} - (\boldsymbol{Q}\boldsymbol{K}' + \boldsymbol{R})\boldsymbol{M}]^{-1}(\boldsymbol{S} + \boldsymbol{Q}\boldsymbol{L}') \tag{B.40}$$

$$\boldsymbol{E} = [\boldsymbol{I} - (\boldsymbol{Q}\boldsymbol{K}' + \boldsymbol{R})\boldsymbol{M}]^{-1}(\boldsymbol{R} + \boldsymbol{Q}\boldsymbol{K}')\boldsymbol{N} \tag{B.41}$$

And,

$$\boldsymbol{K}' = [\boldsymbol{I} - \boldsymbol{J}]^{-1}\boldsymbol{K} \tag{B.42}$$

$$\boldsymbol{H}' = [\boldsymbol{I} - \boldsymbol{J}]^{-1}\boldsymbol{H} \tag{B.43}$$

$$\boldsymbol{L}' = [\boldsymbol{I} - \boldsymbol{J}]^{-1}\boldsymbol{L} \tag{B.44}$$

Depending on the system that is being considered, 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:

$$\boldsymbol{A} = \boldsymbol{P} + \boldsymbol{Q}\boldsymbol{H}' \tag{3.45}$$

$$\boldsymbol{B} = \boldsymbol{S} + \boldsymbol{Q}\boldsymbol{L}' \tag{3.46}$$

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 (3.39) and (3.40), respectively.

As previously stated, for the present example, the LGtheory toolbox program determines from the normal tree that it 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 (B.45) and (B.46), to obtain:

$$\boldsymbol{A} = [0]_{2\times2} + \begin{bmatrix} 0 & -\frac{1}{J} & 0 & -\frac{1}{J} \\ -\frac{1}{L} & 0 & -\frac{1}{L} & 0 \end{bmatrix} [\boldsymbol{I}_{4\times4} - [0]_{4\times4}]^{-1} \begin{bmatrix} 0 & R \\ B & 0 \\ TF_a & 0 \\ 0 & -TF_a \end{bmatrix} \tag{B.47}$$

$$\boldsymbol{A} = \begin{bmatrix} -\frac{B}{J} & \frac{TF}{J} \\ -\frac{TF}{L} & -\frac{R}{L} \end{bmatrix} \tag{B.48}$$

and,

$$\boldsymbol{B} = \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} + \begin{bmatrix} 0 & -\frac{1}{J} & 0 & -\frac{1}{J} \\ -\frac{1}{L} & 0 & -\frac{1}{L} & 0 \end{bmatrix} [\boldsymbol{I}_{4\times4} - [0]_{4\times4}]^{-1} [0]_{4\times1} \tag{B.49}$$

$$\boldsymbol{B} = \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} \tag{B.50}$$

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 and also the substituted constitutive equations from this section, and selecting the 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 extracted from these equations.

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

$$\boldsymbol{C} = \begin{bmatrix} 0 & 1 \\ 0 & -TF \\ 1 & 0 \end{bmatrix} \qquad \boldsymbol{D} = 0 \tag{B.51}$$

B.2.8 Standard State-Space Form Conversion

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

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

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{E}\dot{\boldsymbol{u}} \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} + \boldsymbol{F}\dot{\boldsymbol{u}} \end{aligned} \tag{B.52}$$

The additional matrices, $\boldsymbol{E}$ and $\boldsymbol{F}$, represent the effects that the derivatives of the input variables have on the rate of change of the state-variables and on the output-variables.

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 a non-standard state-space model into the standard form.

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

$$\boldsymbol{x}' = \boldsymbol{x} - \boldsymbol{E}\boldsymbol{u} \tag{B.53}$$

$$\boldsymbol{B}' = \boldsymbol{A}\boldsymbol{E} + \boldsymbol{B} \tag{B.54}$$

$$\boldsymbol{D}' = \boldsymbol{C}\boldsymbol{E} + \boldsymbol{D} \tag{B.55}$$

The state-space model can then be expressed as follows:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x}' + \boldsymbol{B}'\boldsymbol{u} \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x}' + \boldsymbol{D}'\boldsymbol{u} + \boldsymbol{F}\dot{\boldsymbol{u}} \end{aligned} \tag{B.56}$$

B.3 Additional Examples using LGtheory Toolbox

B.3.1 Example 1: Translational Mechanical System

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

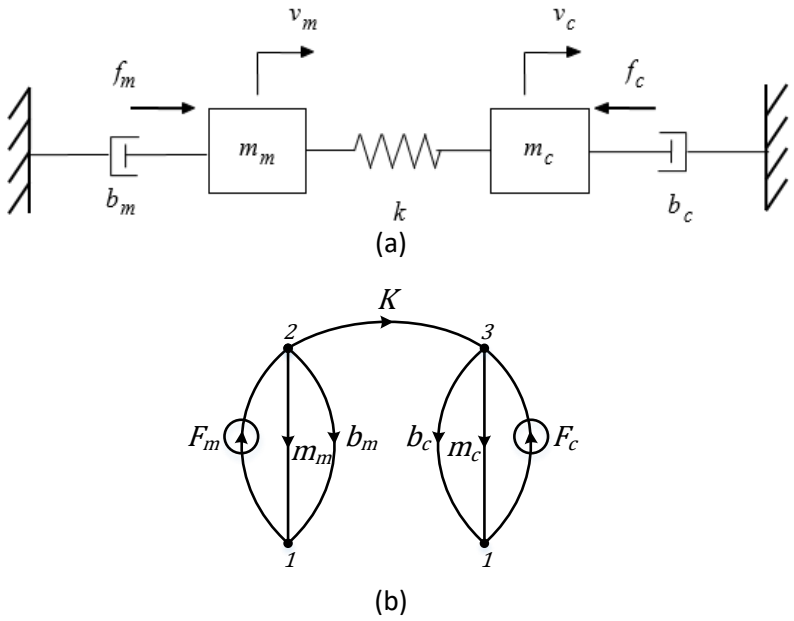

FIGURE B.6

(a) System model and (b) LG model of a Translational Mechanical System.

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 velocities of the mass elements:

$$\begin{bmatrix} \dot{v}_{m_m} \\ \dot{v}_{m_c} \\ \dot{F}_K \end{bmatrix} = \begin{bmatrix} -\frac{b_m}{m_m} & 0 & -\frac{1}{m_m} \\ 0 & -\frac{b_c}{m_c} & \frac{1}{m_c} \\ K & -K & 0 \end{bmatrix} \begin{bmatrix} v_{m_m} \\ v_{m_c} \\ F_K \end{bmatrix} + \begin{bmatrix} \frac{1}{m_m} & 0 \\ 0 & \frac{1}{m_c} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} F_m \\ F_c \end{bmatrix} \tag{B.57}$$

$$\begin{bmatrix} v_{m_m} \\ v_{m_c} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} v_{m_m} \\ v_{m_c} \\ F_K \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} F_m \\ F_c \end{bmatrix} \tag{B.58}$$

B.3.2 Example 2: Hydro-mechanical System

The following multi-physics (multi-domain or mixed) system consists of a rotational torque source, representing the power from an electric motor, powering a positive-displacement pump and a piston, which actuates a mass element attached to ground through a spring. This is translational (rectilinear) system.

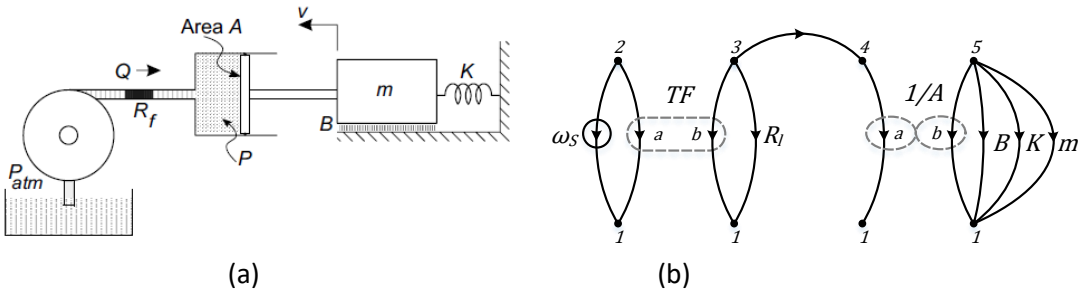

FIGURE B.7

(a) System Schematic mmodel and (b) LG model of a Hydro-mechanical System.

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 that is produced by the toolbox, with the outputs specified as the pressure of the fluid between the pump and the piston, and the velocity of the mass element:

$$\begin{bmatrix} \dot{v_m} \\ \dot{F_k} \end{bmatrix} = \begin{bmatrix} \frac{A^2(R_f + R_l - R_f R_l TF)}{m(R_l TF - 1)} - \frac{B}{m} & -\frac{1}{m} \\ K & 0 \end{bmatrix} \begin{bmatrix} v_m \\ F_k \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} [\omega_s] \tag{B.59}$$

$$\begin{bmatrix} P_{R_f} \\ v_m \end{bmatrix} = \begin{bmatrix} -AR_f & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v_m \\ F_k \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} [\omega_s] \tag{B.60}$$