On the Synthesis of System-Level Dynamic Equations for Mechatronic Systems

Antonio Diaz-Calderon, Christiaan J. J. Paredis[a], Pradeep K. Khosla
Institute for Complex Engineered Systems
Carnegie Mellon University
Pittsburgh, PA 15213
Email: adiaz@cs.cmu.edu, cjp@cs.cmu.edu, pkk@cs.cmu.edu
Phone: (412) 268-5214, FAX: (412) 268-5229

Abstract

This paper presents a novel methodology for deriving the dynamic equations of mechatronic systems from component models that are represented as linear graphs. This work is part of a larger research effort in composable simulation. *In this framework, CAD models of system components are augmented with simulation models describing the component's dynamic behavior in different energy domains. By composable simulation we mean then the ability to automatically generate system-level simulations through composition of individual component models. This paper focuses on the methodology to create the system-level dynamic equations from a high-level system description within CAD software. In this methodology, a mechatronic system is represented by a single system graph. This graph captures the interactions between all the components within and across energy domains — rigid-body mechanics, electrical, hydraulic, and signal domains. From the system graph, the system-level dynamic equations can be derived independently of the underlying energy domains. In the final step, we reduce and order the dynamic equations for efficient computation.*

Keywords: mechatronics, composable simulation, system-level modeling, linear graphs.

a. Corresponding author.

Contents

Figures

Tables

1.Introduction

The work presented in this paper is part of a larger effort to develop a framework for *composable simulation*. By composable simulation we mean the ability to generate system-level simulations automatically by simply organizing the system components in a CAD system. A system component can be either a physical component (electrical motor, gearbox, etc.) or an information technology component (embedded controller or other software component). Each of these system components has one or more simulation models associated with it describing its dynamics in multiple energy domains, across energy domains, and possibly at multiple levels of accuracy (with varying computational requirements). When these system components are combined into a complete system, our framework will automatically combine a selection of the associated component models into a system-level simulation. The user interaction occurs thus at the level of composition of *system components* rather than *simulation components* as in most traditional simulation environments (Matlab, Easy5, etc.). These traditional simulation environments do not consider the mapping from system components to simulation models. This mapping is not one-to-one. The system-level simulation model is not simply a concatenation of individual component models, but may require combining multiple system components into one simulation model (to avoid algebraic loops or index problems) or conversely may require multiple simulation components for a single physical component (describing its behavior in multiple energy domains for instance). Raising the level of user interaction to composition of system components rather than composition of simulation models will result in a significant reduction of effort in creating and modifying system-level simulations and will reduce the simulation and modeling expertise required of the user. Our framework for composable simulation will therefore enable the designers and control engineers to verify their physical designs and control software with much less effort and time than is required in current simulation environments.

In this paper, we will address an issue that typically occurs in the simulation of mechatronic systems, namely, combining software components and symbolic equation manipulation. Mechatronic systems span multiple energy domains (e.g. mechanical, electrical, hydraulic) and include information technology components (such as control algorithms or signal processing). Mathematical models of physical subsystems are in general represented by a set of symbolic differential-algebraic equations (DAEs) while information technology subsystems are usually represented as computer code. From a modeling perspective, information technology components can be considered as black boxes, i.e., port-based objects with multiple inputs and outputs. Within our framework for composable simulation, the information technology components will be included in the resultant set of DAEs (we call these the *system equations*). Some of the inputs to these components may come from the environment (e.g. a reference signal) while some others may come from symbolic equations describing the mechatronic system. Similarly, the outputs of these components may be used in symbolic equations or by other information technology components. To produce correct simulation code, we must consider the information technology components while evaluating the system equations. This is necessary to obtain the evaluation order of the combined system of equations and information technology components.

To address the composable simulation problems outlined above, we have developed a methodology based on a system graph representation. We have extended the system graph to include different energy domains and information technology components through a combination of linear graphs and block diagrams, resulting in a unified system graph representation. The system graph captures the topology of the energy flow in the system, and is used to generate the set of differential-algebraic equations that describe the system behavior. The block-diagram component of the system graph represents the signal flow in the information-technology components. To address the problem of composable simulation at the software integration level, we have defined a software architecture that supports the integration of software modules[5].

The relationship between physical systems and linear graphs was first recognized by Trent[19] and by Brannin[2]. Roe[14] and Koenig[8] apply the theory of linear graphs to the systems theory and provide important results that can be directly related to the two basic laws in circuit theory: Kirchhoff's voltage and current laws. Linear graph theory has been used in the analysis of rigid body dynamics[9, 10,11,12,18] and in the analysis of other engineering systems that include interaction between different energy domains[13].

Besides linear graphs, bond graphs[7, 15] have also been used for system modeling. Bond graphs are energy-based system descriptions in which energy elements are connected by energy conserving junction structures. Similar to our approach, bond graphs define a minimal set of generalized elements that can be used to model system behavior across energy domains. Connections between elements are made through power bonds which represent the power flow in the system. Although bond graphs (with appropriate extensions) can be used to represent mechatronic systems, we have chosen linear graphs for the following reasons. Linear graphs can be more easily adapted to model 3D rigid body mechanics. Furthermore, linear system graphs reflect the topology of the physical system directly, making it easier for non-specialists to create system descriptions.

Composition of simulation models can be accomplished by combining fundamental building blocks described in a high level object-oriented modeling language[3, 4, 6]. The object-oriented approach facilitates model reuse and simplifies maintenance. Using these modeling languages, software executables can be generated automatically from individual submodels and the interactions between them.

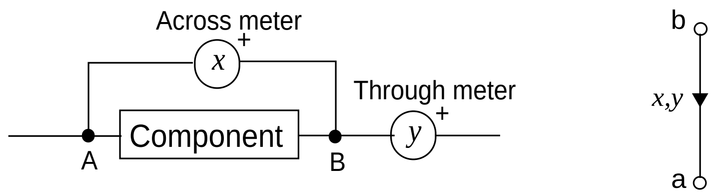
This paper is organized as follows. In Section 2 we describe the proposed modeling methodology to model mechatronic systems. In Section 3 we present algorithms to synthesize the system graph from the geometric description of the mechatronic system. We proceed in Section 4 to present algorithms to derive the governing system equations, and we conclude in Section 5.

2.Modeling of mechatronic systems

Linear graph theory is a branch of mathematics that studies the algebraic and topological properties of topological structures known as graphs. In this context, a physical system can be regarded as a collection of components and terminal points[19]. Between any two terminals, a pair of oriented measurements can be taken, namely *across* and *through* measurements, as shown in Figure 1. The variables associated with this pair of measurements are called *terminal variables*. The mathematical relations between the terminal variables define the component's physical characteristics and are called *terminal equations*.

The graph representation of the component is a directed edge that joins two the terminal points. This graph representation is called *terminal graph* of the component, and the *system graph* is the collection of terminal graphs connected at the appropriate nodes. In a mechatronic system, the system graph may be non-connected, due to the presence of processes in different energy domains.

Figure 1. Through and across measurements on a general two-terminal element and its terminal graph.

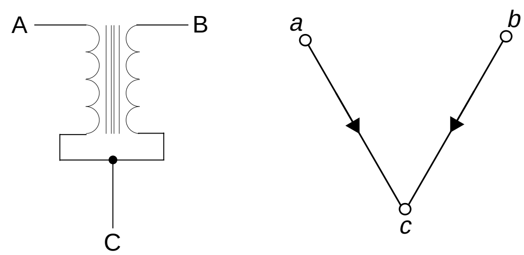


Based on the type of relationship between the terminal variables, one can distinguish three classes of elements: passive elements (that can be further divided into dissipative and non-dissipative elements), generators, and transducers. A dissipative element is one which cannot supply energy to the system while a non-dissipative element, does not dissipate energy but can store it for later recovery. These elements can be divided in two categories: *delay* elements which store energy by means of their through variables, and *accumulator* elements which store energy by means of their across variables. The second class of components contains the generators or drivers. A driver forces an across or through quantity to follow a prescribed function of time. The third class of elements, the transducers (also referred to as couplers), transmits energy from one part of the system to another. An ideal transducer is a transducer that can neither store nor dissipate energy, i.e., there is no energy loss in the component.

Interactions between different energy domains, cannot be described with a two-terminal element. It is necessary to introduce elements that have more than two terminals — *n-terminal* elements. Within this category we find the transducer elements defined previously. The system graph associated with an n-terminal element will be derived from measurements taken between pairs of terminals. However as is shown by Roe[14], we only need $n - 1$ across measurements to completely determine the across variables between any pair of terminals. This number corresponds to the number of branches in a tree selected in the graph:

the terminal graph of an n-terminal element is the tree **T** of $n-1$ edges connecting the n vertices corresponding to the n terminals of the system component. To illustrate this case consider the electric transformer (a 3-terminal system component) shown in Figure 2. Two across measurements will completely determine the device giving a terminal graph with two edges.

Figure 2. n-terminal component.



As is the case with two-terminal elements, the edges in a terminal graph of an n-terminal element will be associated with measurements taken between terminal pairs in the physical system.

In summary, there exists an isomorphism between a linear graph and a physical system provided that one can define pairs of across and through variables. For a system composed of m subsystems, the *system graph* is the union of all terminal graphs for all the components of the system.

Table 1 shows all the various variables associated with the different energy domains considered.

Table 1. Through and across variables for various energy domains

Type of system	**Through Variable**		**Across variable**	
	Name	**Symbol**	**Name**	**Symbol**
General		$x(t)$		$y(t)$
Electrical	Current	$i(t)$	Voltage	$v(t)$
Hydraulic	Fluid flow	$g(t)$	Pressure	$p(t)$
Mechanical	Force, Torque	$f(t)$, $\tau(t)$	Displacement	$\mathbf{r}(t)$, $\theta(t)$

All derivatives of across or through variables are across or through variables as well. For example, velocity and acceleration are also across variables.

Let $\mathbf{x}$ be a vector of across variables $x_1, x_2, \ldots, x_e$ and $\mathbf{y}$ a vector of through variables $y_1, y_2, \ldots y_e$ associated with a system graph of e edges. The terminal variables are chosen such that the power of a component is characterized by their product.

2.1 Representation of topology

The topology of the system graph with v vertices and e edges can be represented by an incidence matrix $\bar{\mathbf{A}}$. The incidence matrix is a $v \times e$ matrix in which each element can have the value +1, -1, or 0 if the edge e is negatively, positively, or not incident onto node v, respectively. In general, for a graph with p connected components, the incidence matrix is a direct sum: a matrix $\mathbf{M}$ is said to be a direct sum of $\mathbf{M}_1, \mathbf{M}_2, \ldots \mathbf{M}_p$ if for any $\mathbf{M}_k$ in $\mathbf{M}$ no nonzero element of $\mathbf{M}_k$ lies in a row or column of $\mathbf{M}$ associated with any of the other submatrices. The $\mathbf{M}_k$ matrices can be regarded as the incidence matrices of each of the p connected components.

As an example consider the mechatronic system shown in Figure 3 for which the system graph is shown in Figure 4. The mechatronic system consists of two energy domains (mechanical and electrical) and a number of components within each energy domain.

Figure 3. Positioning system.

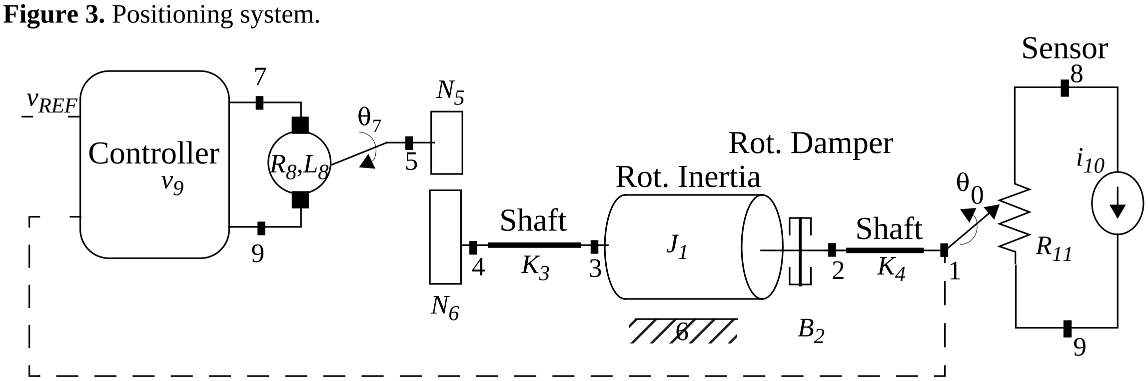
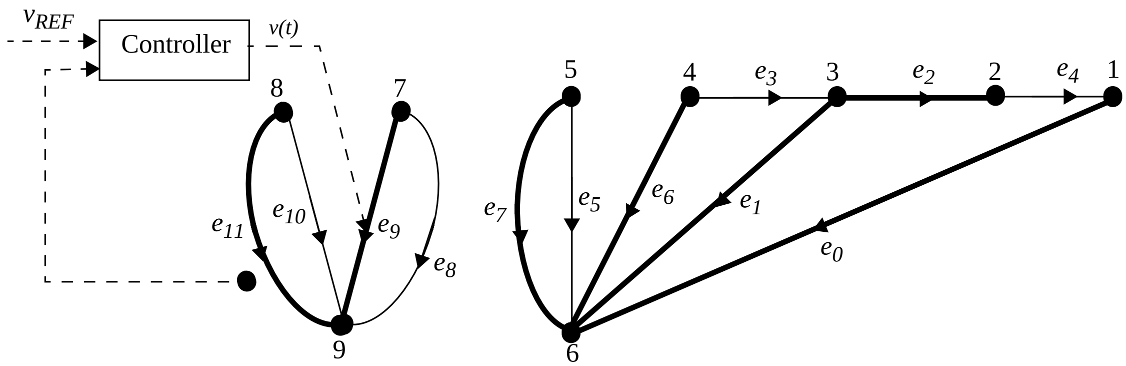


Figure 4. System graph for the positioning system.



The incidence matrix for the system graph for the mechatronic system is

$$\bar{\mathbf{A}} = \left[\begin{array}{cccccccc|cccc} 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 \end{array}\right] = \left[\begin{array}{c|c} \bar{\mathbf{A}}_1 & 0 \\ \hline 0 & \bar{\mathbf{A}}_2 \end{array}\right] \tag{1}$$

This matrix is a direct sum of the incidence matrices for the mechanical and the electrical energy domains. It can be shown that for each connected component k, only $v_k - 1$ rows of the submatrix $\bar{\mathbf{A}}_k$ are linearly independent. When one node is identified as the datum node within each energy domain and the corresponding row is deleted from $\bar{\mathbf{A}}_k$, the resultant matrix is called the *reduced* incidence matrix $\mathbf{A}$:

$$\mathbf{A} = \left[\begin{array}{cccccccc|cccc} 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right] \tag{2}$$

2.2 Constraint equations

The terminal equations are insufficient to describe the mechatronic system completely. An additional e equations are required to define a well posed problem: *2e* equations in *2e* unknowns. These additional equations are derived from the connectivity of the components given by the topology of the system graph. We now regard the system graph as two subgraphs; a *spanning tree* **T** (or *spannig forest* if the graph is non-connected) and a *cotree* (*coforest*.) Without loss of generality, assume that the system graph is connected $(p = 1)$. We can identify $v - 1$ pairs of terminal variables (x_T, y_T) with the *branches* of the spanning tree and $e - v + 1$ terminal variables (x_C, y_C) with the *chords* of the cotree. If the system graph is divided in two non-connected subgraphs by a cut including exactly one branch of **T** and some chords, the cut is unique. It is clear that for a tree **T** with $v - 1$ branches, there will be as many unique cuts. The sum of the vertex equations for all nodes within the cut-set[a] contains only through variables corresponding to the cut-set elements. The set of all cut-set equations can be written in the form:

$$\begin{bmatrix} \mathbf{U}_T & \mathbf{A}_C \end{bmatrix} \begin{bmatrix} \mathbf{y}_T \\ \mathbf{y}_C \end{bmatrix} = \mathbf{0} \tag{3}$$

where $\mathbf{U}_T$ is a unit matrix of dimension $(v - 1)$ and the cut-set matrix $\begin{bmatrix} \mathbf{U}_T & \mathbf{A}_C \end{bmatrix}$ can be derived by applying row operations on the reduced incidence matrix $\mathbf{A}$.

Each chord in the system graph is uniquely associated with a loop in the system. For a given loop, its orientation will be determined by the orientation of the defining chord. A new matrix, namely the circuit matrix $\mathbf{B}$ that captures the connectivity relations between circuits and edges can be defined. The circuit equations associated with each circuit can be written in the form:

$$\begin{bmatrix} \mathbf{B}_T & \mathbf{U}_C \end{bmatrix} \begin{bmatrix} \mathbf{x}_T \\ \mathbf{x}_C \end{bmatrix} = \mathbf{0} \tag{4}$$

a. A cut-set is a set of edges that divide the graph into exactly two components.

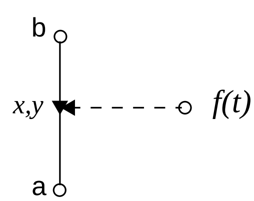
Where $\mathbf{U}_C$ is a square unit matrix with dimensions equal to the number of chords in the system graph. From the *principle of orthogonality*, which states that the vector space spanned by the rows of the incidence matrix $\mathbf{A}$ and the vector space spanned by the rows of the circuit matrix $\mathbf{B}$ are orthogonal complements[8] (i.e., $\mathbf{A}\mathbf{B}^{\mathsf{T}} = \mathbf{0}$ and $\mathbf{B}\mathbf{A}^{\mathsf{T}} = \mathbf{0}$), we can obtain an expression for $\mathbf{B}_T$:

$$\mathbf{B}_T = -\mathbf{A}_C^{\mathsf{T}} \tag{5}$$

2.3 Low-power component modeling

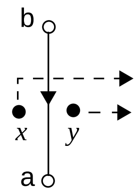
In order to include information technology components as well as other types of low-power devices in the system graph, it is necessary to extend the system graph representation for the inclusion of signals. A *signal* represents the value of some system variable as a function of time. To introduce signals in the system graph we define the concept of *variable elements*. A variable element is an element that can have one or more input signals that modify its response. The simplest variable element is the signal-controlled across or through driver. In this case, either the across or through variable associated with the terminal graph will be completely defined by the signal: $x(t) = f(s(t))$ or $y(t) = h(s(t))$ where x and y represent across and through variables, respectively. Similarly, a variable passive element is also signal-controlled, but here, the input signal is modulating one of the element parameters (Figure 5). Output signals are obtained from the system graph as “measurements” of system variables (Figure 6).

Figure 5. Terminal graph of variable elements.



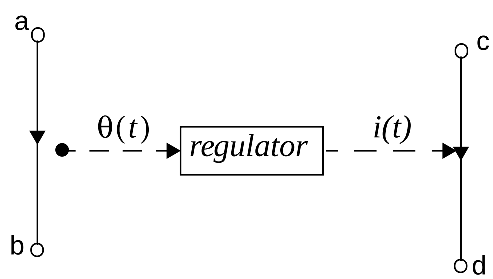
In the context of mechatronics, it is important to have a system representation that is capable of handling signal elements. Mechatronic systems include information technology components for which there is no energy flow and that therefore cannot be represented by a terminal graph. As an example consider an embedded controller. The control algorithms are provided as algorithmic components that must interact with the rest of the system but do not generate or transfer power.

Figure 6. Reading values from a terminal graph.



To illustrate this consider a portion of a positioning system composed of an angular position sensor, a regulator, and a current source (Figure 7). The regulator obtains the signal input from the position sensor to provide an output signal used to modulate the current source.

Figure 7. A positioning system The system graph shows the interaction between the signal block and the terminal graph.



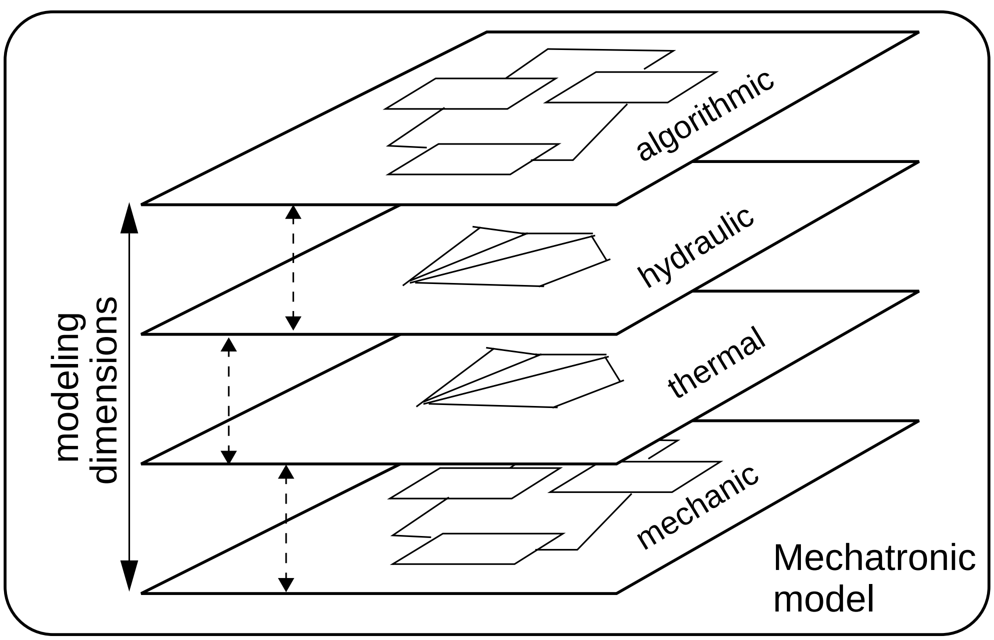
3.Synthesis of the System Graph for Mechatronic Systems

The system graph for a mechatronic system is constructed with the help of a *system editor* that is tightly integrated with a CAD system. As is shown in Figure 8, the system editor is based on the concept of *modeling layers* each of which represents a different energy domain of the system. The modeling layer for the mechanical energy domain is implemented in a CAD system. When a component is brought into the system editor, its constituting models are included in their respective modeling layers. It is then the task of the user to identify the interactions between components. Interactions are classified as: 1) mechanical interactions, 2) terminal connections, and 3) edge associations. Mechanical interactions such as rigid connections, prismatic joints or revolute joints arise from the interconnection of two rigid bodies. The method for deriving the system-graph representation of the mechanical system is presented in Section 3.1. Terminal connections on the other hand, represent the interaction of components within the other energy domains. A terminal connection between two terminals indicates that both terminals are mapped to a single node in the system graph. Edge associations arise from the energy exchange between different energy domains. They occur when system variables in the terminal equations of a component are associated with other edges in the terminal graph. For example consider the terminal equation of the electrical edge of a DC motor:

$$v(t) = K_m\dot{\theta}(t) + R + L\frac{d}{dt}i(t) \tag{6}$$

Variable $\dot{\theta}(t)$ is a system variable that is associated with an edge (in the mechanical system graph) that is not part of the electrical domain. These types of variables are called *exogenous* since they are assumed to be known within that portion of the model. The definition of exogenous variables within a terminal equation establishes an association between edges in the system graph.

Figure 8. Modeling layers of a mechatronic system.



3.1 Synthesis of the system graph for 3D Mechanics.

The dynamic equations of the 3D mechanics of the system are derived using a sub-module (Dynaflex) which is specifically designed for the analysis of three dimensional constrained mechanical systems[18]. Dynaflex is a graph-theoretic approach in which the connectivity of the bodies in the mechanism and the forces acting on them are represented by a linear graph (mechanical system graph). The difference between the system graph for non-mechanical energy domains and the graph for the mechanical domain lies in the different dimension of the spaces defining the terminal variables. Variables in the mechanical domain are elements of $\Re^3$ whereas variables in the system graph are elements of $\Re$ (i.e., scalars.)

The system graph of the mechanical system captures the topology of the mechanism. However, to have a complete model, geometric and inertial information must be added to the topology. This information is derived from the Intelligent Assembly Modeling and Simulation (IAMS) tool kit[16]. Based on the geometry of the mechanism the IAMS tool kit will automatically determine the instantaneous kinematic relationships between components in the mechanism. IAMS further provides information about the origin of the inertial frame, center of mass of each body, location of articulation points in each body, type of joint, and points of application of internal forces.

Similar to the basic modeling elements we defined in Section 2, Dynaflex provides a set of modeling elements for mechanical systems, including[18]: body elements, arm elements (position vectors), motion and force drivers, spring-damper-actuator elements, and joint elements.

The process for obtaining the graph representation suitable for Dynaflex consists of three steps. First an *extended* system graph is generated. This step maps the geometry of the mechanism directly into a linear graph representing its topology. The second step identifies composite bodies consisting of rigidly connected subcomponents. In a final step, composite bodies are replaced by single bodies reducing the system graph to a minimal graph with the same topological properties.

The generation of the system graph involves a direct translation of the kinematic information into the linear graph representation. In general, the result of the first stage is an extended system graph that includes all kinematic information including fixed joints and redundant joints. However, to avoid structural singularities and indexing problems, and to improve the efficiency of the symbolic computations in Dynaflex, we simplify this initial system graph by lumping all rigidly connected bodies into a single composite body.

Composite bodies are identified by performing a depth-first traversal on the extended system graph. The algorithm explores all paths created by rigid connections and collects all bodies along the path into a single composite body. The algorithm can be stated as follows:

Algorithm A. *(Composite body identification)*. The algorithm takes as an input the system graph and generates as output the set of composite bodies. The algorithm uses the following sets to keep track of all nodes in the graph: the set CLOSED, contains all nodes already visited. The set LUMPS contains all the composite bodies in the system. OPEN is a set containing all the nodes to-be-visited. ξ contains the bodies to be combined into the current composite, and SYSTEM is the set of CG nodes of the system graph.

A1. Set $CLOSED \leftarrow \varnothing$, $LUMPS \leftarrow \varnothing$

A2. While $SYSTEM \neq \varnothing$ do

A3. Set $cg \leftarrow first(SYSTEM)$, $\xi \leftarrow \{cg\}$,

A4. $OPEN \leftarrow sucessors(cg) \setminus CLOSED$,

$SYSTEM \leftarrow SYSTEM \setminus \{cg\}$

$CLOSED \leftarrow CLOSED \cup \{cg\}$

A5. While $OPEN \neq \varnothing$ do

A6. $cg \leftarrow first(OPEN)$,

A7. $\xi \leftarrow \xi \cup \{cg\}$,

$OPEN \leftarrow (OPEN \setminus \{cg\}) \cup (sucessors(cg) \setminus CLOSED)$

$SYSTEM \leftarrow SYSTEM \setminus \{cg\}$

$CLOSED \leftarrow CLOSED \cup \{cg\}$

A8. Continue A5.

A9. Set $LUMPS \leftarrow LUMPS \cup \{\xi\}$

A10. Continue A2.

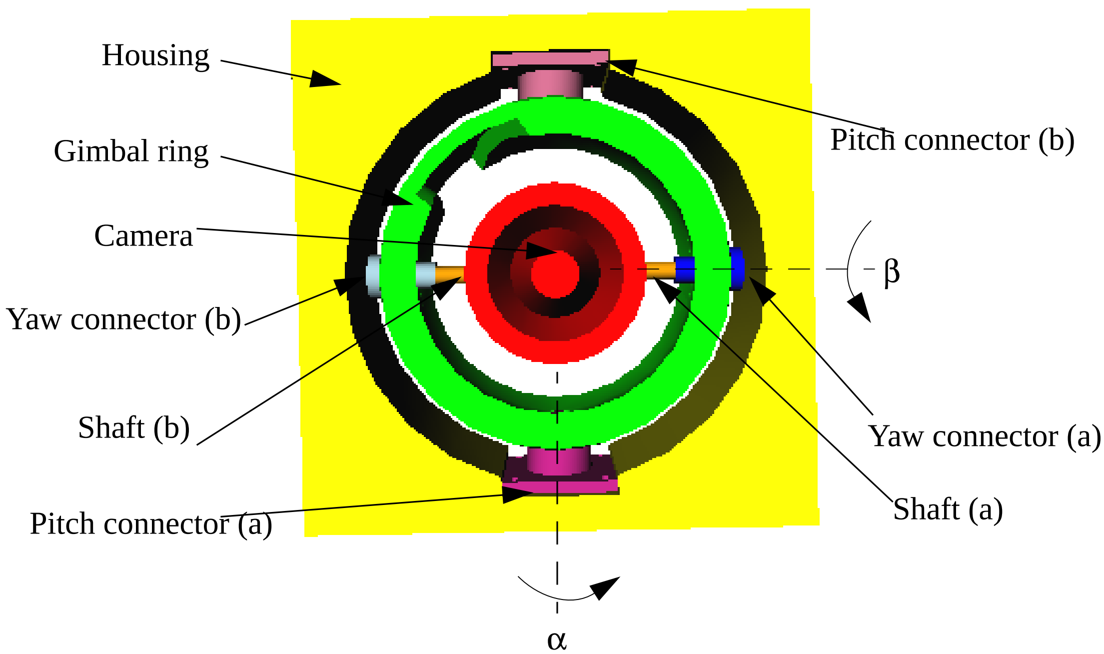
A11. The algorithm terminates. We have checked all bodies in the system and have defined the composite bodies that must be created.

Algorithm A uses the predicate `successors`, which given a node in the system graph, it returns the adjacent nodes if the path to the successors is through a rigid connection.

The last stage in the synthesis of the system graph is to perform the reduction process that will combine the identified bodies into single composite bodies and remove redundant joints. Redundant joints are detected by loops where the bodies in the loop appear more than once. If two joints are found to be redundant, that can be interpreted in two ways. First, the joint axes may be colinear. This would result in an overconstrained mechanism for which one of the two joints can be discarded for analysis purposes. The second interpretation is when the joint axes are not colinear making some angle α between them. This configuration results in an overconstrained body for which we cannot discard any of the joints for analysis purposes. In this event, the algorithm reports the problem to the user stating that the system is fully constrained.

As an example of how these steps are followed consider the design of a missile seeker shown in Figure 9.

Figure 9. Missile seeker.



This design contains 9 bodies: housing, gimbal ring, camera, pitch connector (2) yaw connector (2), shaft (2). A kinematic description of the system reveals that there are a number of bodies that may be combined to form composites (Table 2).

Table 2. Kinematic description for the seeker system

Type of Joint	Reference body	Secondary body
FIXED	housing	pitch connector (a)
FIXED	housing	pitch connector (b)
REVOLUTE*	pitch connector (a)	gimbal ring
REVOLUTE	pitch connector (b)	gimbal ring
FIXED	gimbal ring	yaw connector (a)
REVOLUTE*	yaw connector (a)	shaft (a)
FIXED	gimbal ring	yaw connector (b)
REVOUTE	yaw connector (b)	shaft (b)
FIXED	shaft (a)	camera
FIXED	shaft (b)	camera

From the kinematic description shown in Table 2, the first stage of our derivation generates an extended system graph shown in Figure 10. Secondly, Algorithm A identifies the composites listed in Table 3.

Table 3. Composite bodies found by Algorithm A

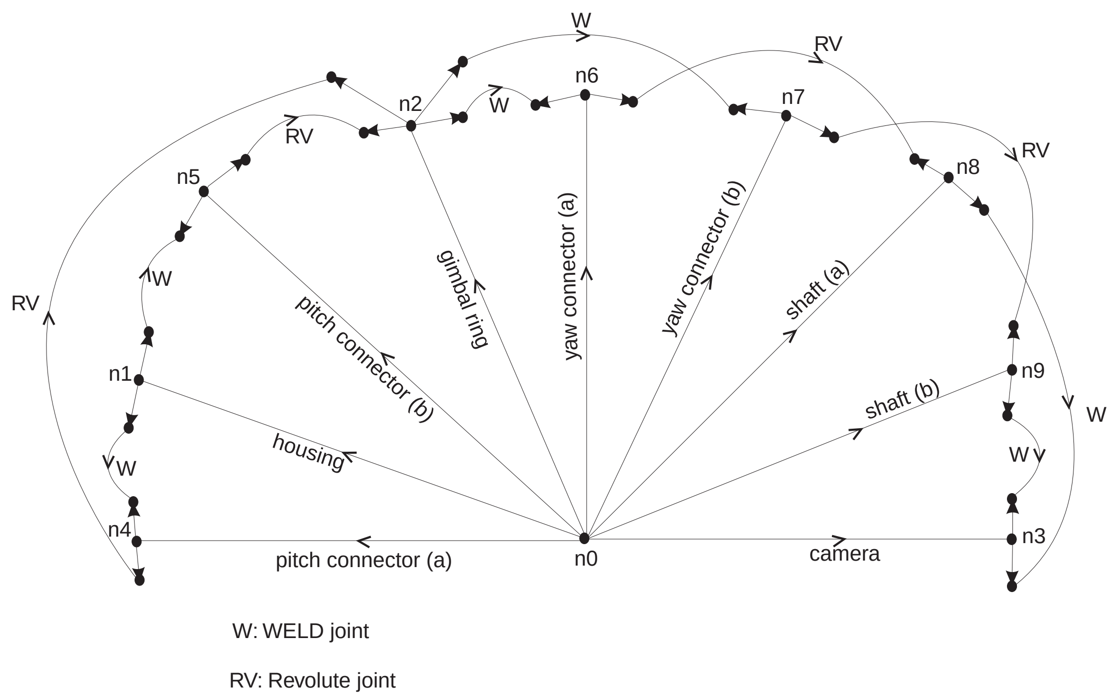
BODY_1	shaft (b)	camera	shaft (a)
BODY_2	housing	pitch connector (b)	pitch connector (a)
BODY_3	gimbal ring	yaw connector (b)	yaw connector (a)

Finally, the reduction stage yields the following kinematic relations:

Table 4. Kinematic description for the composite bodies in the seeker

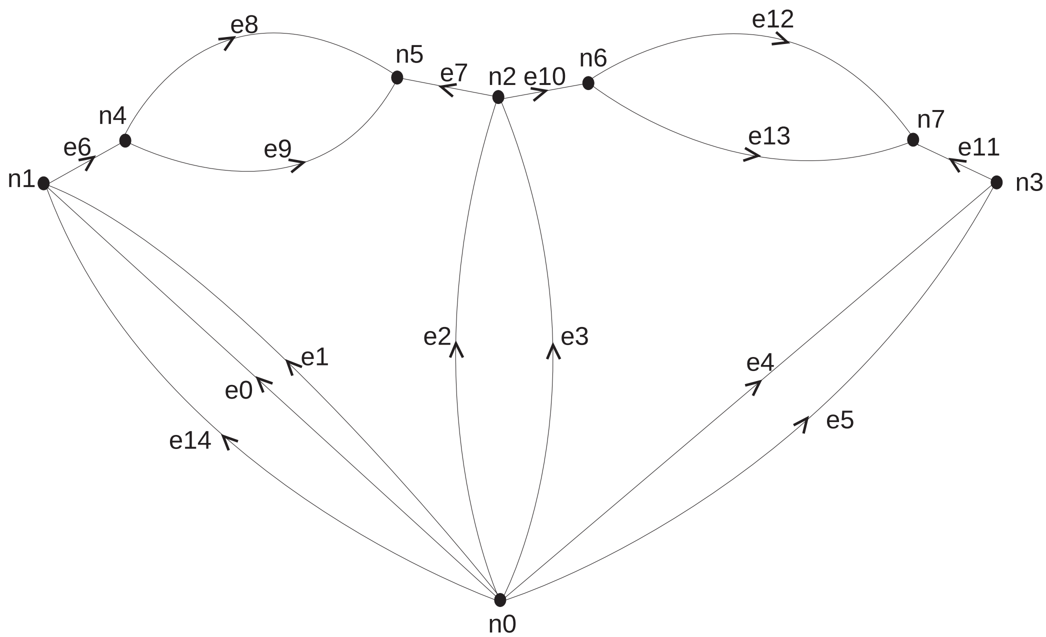
Type of Joint	Reference body	Secondary body
REVOLUTE	BODY_2	BODY_3
REVOLUTE*	BODY_2	BODY_3
REVOLUTE	BODY_3	BODY_1
REVOLUTE*	BODY_3	BODY_1

Figure 10. Extended system graph. Only joint and body elements are shown for clarity.



Notice that there are two revolute joints per pair of composite bodies. Kinematic analysis reveals that the rotation axes of each pair of joints coincide. For the Dynaflex analysis, one of the two points is removed to avoid concluding overconstrained kinematics; only the joints marked with an asterisk are considered. At the end of the reduction process, we obtain the reduced system graph shown in Figure 11.

Figure 11. Reduced system graph.



To conclude the Dynaflex description, dynamic elements are introduced consisting of generalized forces provided by motors, external forces applied to the bodies, and forces acting between two bodies. For this example, only two force elements are introduced: e9 and e13, which are the result of the motors built into the corresponding joints. Furthermore, we introduce gravity forces acting on the bodies at their center of mass (e1, e3, e5) representing the weight of BODY_2, BODY_3, and BODY_1, respectively.

4.Synthesis of system equations

Once a mechatronic system has been described as a system graph, the dynamic equations can be derived from the graph without the need to consider the underlying physics in each of the energy domains. As mentioned in Section 2, the system equations can be derived by simultaneously considering the *e* terminal equations and the *e* independent topological constraints (cut-set and circuit equations). The remaining questions that we will address in this section are: which topological constraints need to be considered, and which of the two system variables (across or through) should be the independent variable in each of the *e* terminal equations? Both of these questions are answered in the normal tree selection algorithm presented in this section.

The terminal equations plus any independent set of *e* constraint equations unambiguously define the dynamics of the system. However, before these equations can be numerically solved they must be expressed in state space form in which the derivatives of a state *x* are expressed as explicit functions of the states and time:

$$\dot{x} = f(x, t) \tag{7}$$

Expressing the equations of the system in this form implies using the smallest possible number of equations (equal to the order of the system) and expressing the high order derivatives as a function of low order derivatives of state variables, in each equation

This can be accomplished in the following way. Let us divide the system variables into two groups: primary variables and secondary variables — one of each for every edge. Assume now that in the terminal equation of an edge, the highest order derivative of the primary variable *p* is expressed as a function of the secondary variable, *s*:

$$p^{(n)} = f(s) \tag{8}$$

On the other hand, assume that in the constraint equations the secondary variables are expressed as a function of the primary variables:

$$s = g(p) \tag{9}$$

Then, by substituting the constraint equations (9) into the terminal equations (8), we get a minimal set of dynamic equations of the form:

$$p^{(n)} = f(g(p)) \tag{10}$$

which is exactly the desired state-space representation.

The final step in the derivation of our approach is the selection of the primary and secondary variables. According to equations (3) and (4) the dependent variables in the constraint equations are the through variables in the branches of the tree and the across variables in the chords of the cotree:

$$\begin{aligned} \mathbf{y}_T &= -\mathbf{A}_C \mathbf{y}_C \\ \mathbf{x}_C &= -\mathbf{B}_T \mathbf{x}_T \end{aligned} \tag{11}$$

From equations (9) and (11), we can identify primary variables with the set of $v - p$ across variables associated with the branches of a forest and the set of $e - v + p$ through variables associated with the chords of a coforest. Similarly, the dependent variables in equation (11) are identified as s*econdary variables* of the system graph.

Based on the selection of primary and secondary variables, we can obtain dynamic equations of the form (10) by selecting a tree on the system graph such that the following two conditions are satisfied: 1) the highest order derivatives of as many primary variables as possible appear in the terminal equations as functions of secondary variables and low order derivatives of primary variables, and 2) the terminal equations contain as few derivatives of secondary variables as possible. The tree that satisfies these two conditions is called a *normal tree* of the system graph.

The normal tree of a system graph **G** is found by defining a real function $w: e \to \Re^+$ on the edges of **G** that computes the weight of the edges as follows:

Let κ_α and κ_τ represent the highest derivative order of all accumulator elements and all delay elements respectively, and $O{:}e \to \Re^+$ be a real function defined on the edges that computes the highest derivative order of the element associated with edge *e*. Next, classify the edges of **G** as follows: let all across drivers and generalized across drivers belong to the class c^Δ, accumulator elements to the classes

$$c_i^\alpha = \{e_\alpha | O(e_\alpha) = \kappa_\alpha - i\} \qquad i = 0, \ldots, \kappa_\alpha - 1 \ , \tag{12}$$

dissipator elements to class $c^\delta = \{e_\delta | O(e_\delta) = 0\}$, and delay elements to the classes

$$c_i^\tau = \{e_\tau | O(e_\tau) = \kappa_\tau - i\} \qquad i = 0, \ldots, \kappa_\tau - 1 \ . \tag{13}$$

Finally, all through drivers and generalized through drivers will belong to class c^Φ. The weight functions *w* defined on the edges of **G** are chosen for each class such that

$$w^\Delta < w_0^\alpha < w_1^\alpha < \ldots < w_{\kappa_\alpha - 1}^\alpha < w^\delta < w_0^\tau < w_1^\tau \ldots < w_{\kappa_\tau - 1}^\tau < w^\Phi \tag{14}$$

where w^{Δ} is the weight function associated to class c^{Δ}, w_i^{α} is the weight function associated to class c^{α}, w_i^{δ} is the weight function associated to class c^{δ}, w_i^{τ} is the weight function associated with class c^{τ}, and w_i^{Φ} is the weight function associated to class c^{Φ}.

In other words, the weight function *w* ranks the edges of **G** based on their respective classes. Any weight function that satisfies the ranking in equation (14) is admissible.

The next step in our approach is to find a *minimum cost spanning tree*[b] that minimizes the total cost (weight) of the weights assigned to the branches of the tree. Aho et. al.[1] shows that it is always possible to find such a tree based on the following property: let *V* be the set of vertices of **G** and *U* be a proper subset of *V*. If $e_{min} = (u, v)$ is an edge of lowest cost such that $u \in U$ and $v \in V - U$, then there is a minimum cost spanning tree that includes e_{min}. The proof of this property is outside the scope of this article but can be found in Aho et. al.[1]

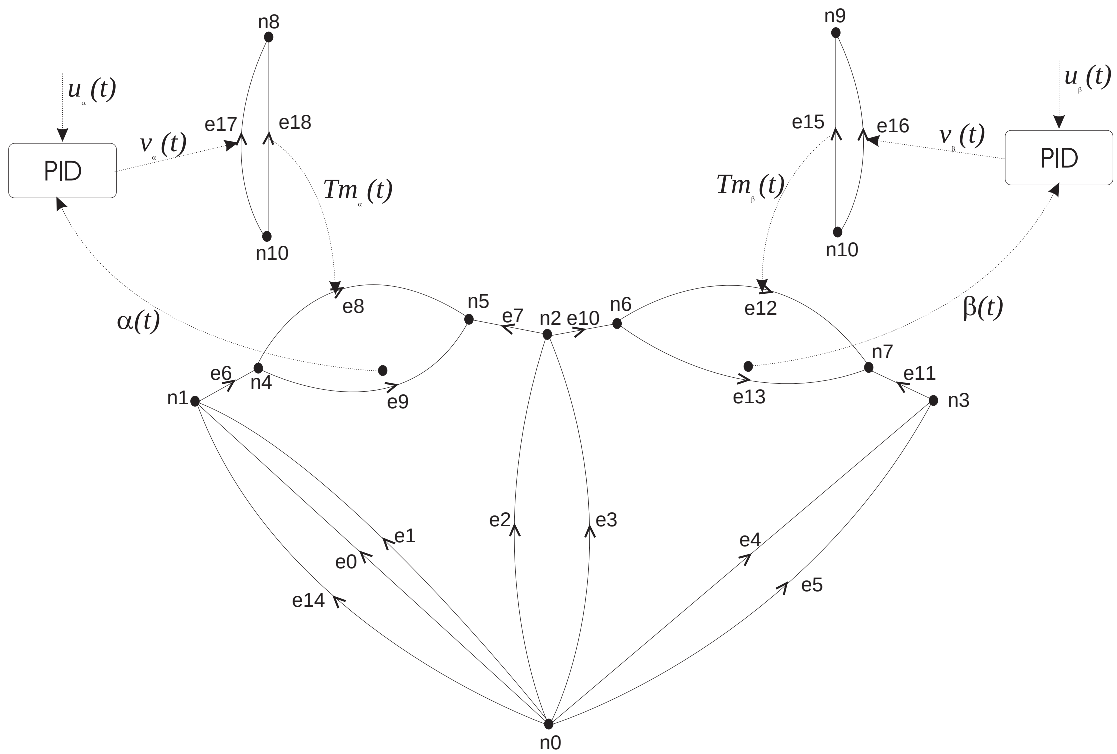
Once a normal tree has been selected, we can write *e* cut-set and circuit equations, and *e* terminal equations. The constraint equations together with the terminal equations constitute the set of equations for the complete solution of the mechatronic system excluding the mechanical energy domain. To completely specify the mechatronic system, these equations are combined with the dynamic equations for the 3D mechanism derived by Dynaflex. This new set of equations will be in general a set of differential algebraic equations (DAE) which constitute the set of equations for the complete system.

We now revisit the missile seeker example introduced in the previous section. The system graph is generated as prescribed in Section 3 and the result is shown in Figure 12. We can now proceed to select the normal forest (which is indicated by bold lines in Figure 12) and once the forest is selected the system equations are derived. As shown in the appendix, there are two second order differential equations that correspond to the mechanical system, two first order differential equations corresponding to the electric domain of the motors and four algebraic equations that correspond to the coupling equations and the controllers used in the design. These equations can be transformed to a set of six first order differential equa-

b. A spanning tree for **G** is a tree that connects all vertices in **G**.

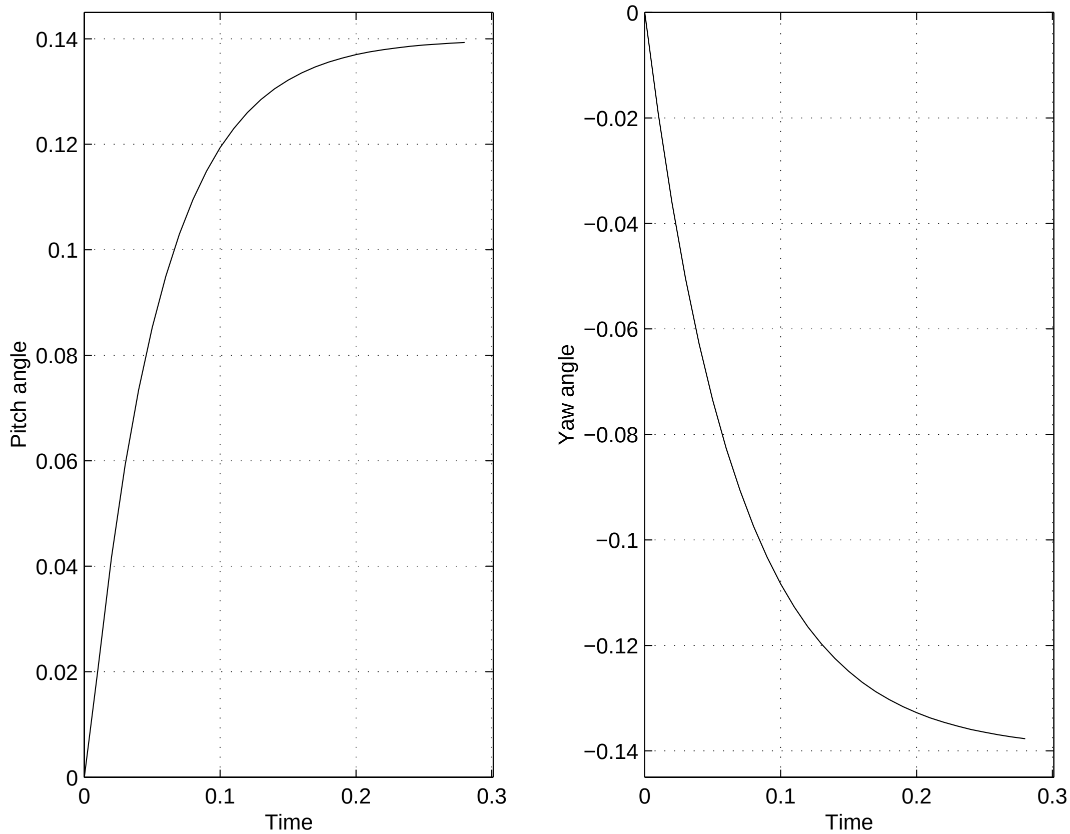
tions and the algebraic equations can be substituted in the resultant set. This equations are then integrated over time to show the behavior of the system.

Figure 12. System graph for the missile seeker.



As an example, Figure 13 shows the angular position of the gimbal ($\alpha(t)$) and the camera assembly ($\beta(t)$) when step functions of 0.14 rad and -0.14 rad are applied to the respective controllers.

Figure 13. System response as a function of time.



5.Conclusions

In this paper, we have presented a framework for composable simulation in which component models are automatically combined to create system-level simulation models. Our approach is based on linear graph theory. Component models are represented as terminal graphs with the corresponding terminal equations. These subgraphs are combined into a system graph which is a domain independent representation of the system dynamics. To derive the dynamic equations from the system graph, a minimum cost spanning tree algorithm is used, resulting ina set of equations in state-space form. We illustrated the framework with an example of a missile seeker.

6.Acknowledgments

Our thanks are due to Dr. John J. McPhee and Dr. Pengfei Shi from the Motion Research Group in the Department of Systems Design Engineering at the University of Waterloo for their help in the use of Dynaflex.

This research was funded in part by DARPA under contract ONR # N00014-96-1-0854, by the National Institute of Standards and Technology, by the Pennsylvania Infrastructure Technology Alliance, by the National Council of Science and Technology of Mexico (CONACyT), and by the Institute for Complex Engineered Systems at Carnegie Mellon University.

7.References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Data structures and algorithms*. Reading, Massachusetts: Addison-Wesley, 1987.

[2] F. H. Branin, “The algebraic-topological basis for network analogies and the vector calculus,” presented at Symposium on Generalized Networks, Polytechnic Institute of Brooklyn, 1966.

[3] F. E. Cellier, Continuous system modeling. Springer-Verlag, 1991.

[4] F. E. Cellier, Automated formula manipulation supports object-oriented continuous-system modeling. IEEE Control Systems, Vol. 2, No. 13, April 1993. pp. 28-38.

[5] A. Diaz-Calderon, C. J. J. Paredis, and P. K. Khosla, “A modular composable software architecture for the simulation of mechatronic systems,” presented at ASME Design Engineering Technical Conference, 18th Computers in Engineering Conference, Atlanta, GA, 1998.

[6] H. Elmqvist, and D. Bruck. “Constructs for object-oriented modeling of hybrid systems.” In Eurosim Simulation Congress, Vienna, Austria, September 11-15, 1995.

[7] D. C. Karnopp, Margolis, D. L., and R. C. Rosenberg. System dynamics: A unified approach. John Wiley & Sons, Inc. New York, NY, 1990.

[8] H. E. Koenig, Tokad, Y., Kesavan, H. K., and H. G. Hedges, Analysis of discrete physical systems. New York: MacGraw-Hill, 1967.

[9] T. W. Li, and G. C. Andrews, "Application of the vector-network method to constrained mechanical systems," *Mechanisms, Transmissions, and Automation in Design*, vol. 108, pp. 471-480, 1986.

[10] J. J. McPhee, "On the use of linear graph theory in multibody system dynamics," *Nonlinear Dynamics*, vol. 9, pp. 73-90, 1996.

[11] J. J. McPhee, M. G. Ishac, and G. C. Andrews, "Wittenburg's formulation of multibody dynamics equations from a graph-theoretic perspective," *Mechanism and Machine Theory*, vol. 31, pp. 202-213, 1996.

[12] J. J. McPhee, "Automatic generation of motion equations for planar mechanical systems using the new set of "branch coordinates"," *Mechanism and Machine Theory*, vol. 33, pp. 805-823, 1998.

[13] B. J. Muegge, Graph-theoretic modeling and simulation of planar mechatronic systems. MA. Sc. Thesis, Systems Design Engineering, University of Waterloo, Waterloo, 1996.

[14] P. H. O'n. Roe, *Networks and systems*. Reading, Massachusetts: Addison-Wesley, 1966.

[15] R. C. Rosenberg, and D. C. Karnopp. Introduction to physical system dynamics. McGraw-Hill, New York, NY, 1983.

[16] R. Sinha, C. J. J. Paredis; S. K. Gupta; and P. K. Khosla, "Capturing Articulation in Assemblies from Component Geometry," Proceedings of the ASME Design Engineering Technical Conference, September 1998, Atlanta, Georgia, USA

[17] S. Seshu and M. B. Reed, *Linear graphs and electrical networks*. Reading, Massachusetts: Addison-Wesley, 1961.

[18] P. Shi. "Flexible multibody dynamics: A new approach using virtual work and graph theory," Ph. D. Thesis, Systems Design Engineering, University of Waterloo, Waterloo, Canada, 1998.

[19] H. M. Trent, "Isomorphisms between oriented linear graphs and lumped physical systems," *The Journal of the Acoustical Society of America*, vol. 27, pp. 500-527, 1955.

8.Appendix

$$\ddot{\beta}(t) = (-0.18419821(10^{-6})\cos(\alpha(t))\dot{\beta}(t)^2\sin(\alpha(t)) \tag{15}$$
$$-0.42128079(10^{-6})\dot{\beta}(t)\dot{\alpha}(t)\cos(\alpha(t))$$
$$+0.31596059(10^{-6})\dot{\beta}(t)\dot{\alpha}(t)\sin(\alpha(t)) + 0.1579803(10^{-6})\dot{\alpha}(t)^2\sin(\alpha(t))$$
$$+0.1579803(10^{6})\dot{\beta}(t)^2\sin(\alpha(t)) - 0.63153674(10^{-6})\sin(\alpha(t))^2\dot{\beta}(t)^2$$
$$-0.00032463Tm_{\alpha}(t) + 0.00032463Tm_{\beta}(t) - 0.2106404(10^{-6})\dot{\beta}(t)^2$$
$$\cos(\alpha(t)) - 0.00048665\cos(\alpha(t))Tm_{\beta}(t) - 0.00064886\sin(\alpha(t))Tm_{\beta}(t)$$
$$+0.31576837(10^{-6})\dot{\beta}(t)^2 - 0.2106404(10^{-6})\dot{\alpha}(t)^2\cos(\alpha(t)))$$
$$/(-0.63153674(10^{-6})$$
$$\sin(\alpha(t))\cos(\alpha(t)) + 0.12389914(10^{-5}) - 0.18419821(10^{-6})\sin(\alpha(t))^2)$$

$$\ddot{\alpha}(t) = (0.36839643(10^{-6})\dot{\beta}(t)^2\sin(\alpha(t)) - 0.42128079(10^{-6})\dot{\beta}(t)\dot{\alpha}(t) \tag{16}$$
$$\cos(\alpha(t)) + 0.31596059(10^{-6})\dot{\beta}(t)\dot{\alpha}(t)\sin(\alpha(t)) + 0.63153673(10^{-6})\dot{\beta}(t)$$
$$\dot{\alpha}(t) + 0.18419821(10^{-6})\sin(\alpha(t))\dot{\alpha}(t)^2\cos(\alpha(t)) - 0.12630735(10^{-5})$$
$$\sin(\alpha(t))^2\dot{\beta}(t)\dot{\alpha}(t) + 0.1579803(10^{-6})\dot{\alpha}(t)^2\sin(\alpha(t))$$
$$+0.237036(10^{-5})\dot{\beta}(t)^2\sin(\alpha(t)) - 0.12630735(10^{-5})\sin(\alpha(t))^2\dot{\beta}(t)^2$$
$$-0.00032463Tm_{\alpha}(t) + 0.00487079Tm_{\beta}(t)$$
$$+0.00048665\cos(\alpha(t))Tm_{\alpha}(t) - 0.63153674(10^{-6})\sin(\alpha(t))^2\dot{\alpha}(t)^2$$
$$+0.31576837(10^{-6})\dot{\alpha}(t)^2 - 0.316048(10^{-5})\dot{\beta}(t)^2\cos(\alpha(t))$$
$$-0.0009733\cos(\alpha(t))Tm_{\beta}(t) - 0.00129773\sin(\alpha(t))Tm_{\beta}(t)$$
$$+0.36839643(10^{-6})\sin(\alpha(t))\dot{\beta}(t)\dot{\alpha}(t)\cos(\alpha(t)) + 0.63153674(10^{-6})\dot{\beta}(t)^2$$
$$-0.2106404(10^{-6})\dot{\alpha}(t)^2\cos(\alpha(t)) + 0.00064886\sin(\alpha(t))Tm_{\alpha}(t))$$
$$/(-0.63153674(10^{-6})$$
$$\sin(\alpha(t))\cos(\alpha(t)) + 0.12389914(10^{-5}) - 0.18419821(10^{-6})\sin(\alpha(t))^2)$$

$$\frac{d}{dt}i_{\alpha}(t) = -\frac{-E_{\alpha}(t) + 0.000813\dot{\alpha}(t) + 0.2i_{\alpha}(t)}{200(10^{-6})} \tag{17}$$

$$\frac{d}{dt}i_{\beta}(t) = -\frac{-E_{\beta}(t) + 0.00008\dot{\beta}(t) + 0.2i_{\beta}(t)}{200(10^{-6})} \tag{18}$$

$$Tm_{\alpha}(t) = 0.0813i_{\alpha}(t) \tag{19}$$

$$Tm_{\beta}(t) = 0.032i_{\beta}(t) \tag{20}$$

$$E_{\alpha}(t) = PID(u_{\alpha}(t), \alpha(t)) \tag{21}$$

$$E_{\beta}(t) = PID(u_{\beta}(t), \beta(t)) \tag{22}$$