

# Automatic Generation of System-Level Dynamic Equations for Mechatronic Systems

**Antonio Diaz-Calderon<sup>a</sup>, Christiaan J. J. Paredis, 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. The complete modeling process is illustrated with an example of a missile seeker.*

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

---

a. Corresponding author.

# 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, gear-box, 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/Simulink, 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 pro-

cessing). 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<sup>7</sup>.

The relationship between physical systems and linear graphs was first recognized by Trent<sup>31</sup> and by Brannin<sup>2</sup>. Roe<sup>26</sup> and Koenig<sup>13</sup> 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<sup>14,17,18,19,30</sup> and in the analysis of other engineering systems that include interaction between different energy domains<sup>21</sup>.

Besides linear graphs, bond graphs<sup>12,27</sup> 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<sup>4,5,11</sup>. The object-oriented approach facilitates model reuse and simplifies maintenance. Using these modeling languages, software executables can be generated automatically from individual sub-models and the interactions between them.

In the commercial world, there are a number of simulation packages specifically designed for particular application areas. In the area of CAE there exist several packages for rigid body dynamics such as ADAMS<sup>20,23,24</sup>, DADS<sup>15,22</sup>, and Mesa Verde<sup>32</sup>. The main characteristic of these systems is that the equations of motion are generated numerically from the geometric description of the mechanism. Some (i.e., ADAMS and DADS) can be integrated with CAD packages such that the geometry of the mechanism is directly derived from the CAD model.

A second group of commercial simulation packages provides support for general systems modeling and simulation. We can identify three main approaches: 1) block-diagrams, 2) object-oriented modeling and 3) bond graphs. Systems belonging to the first category include EASY5<sup>3</sup> and Matlab/Simulink<sup>16</sup>. Both systems are based on an interactive environment for modeling where the user defines the system as a network of interconnected blocks. EASY5 however takes the modeling approach a step further in which the system is mod-

eled by defining the interactions between components instead of simulation blocks as with the block-diagram approach.

In the second category we can include Dymola<sup>9</sup>. Dymola is an object-oriented language and a program for modeling large systems. Reuse of modeling knowledge is supported by use of libraries containing model classes and through inheritance. Dymola also supports the new object-oriented modeling language Modelica<sup>10</sup>.

Finally, 20-sim<sup>6</sup> models systems based on the bond graph approach. Dymola and 20-sim represent the model dynamics in non-causal form which provides greater modeling flexibility by not forcing the modeler to use predefined input/output relationships when defining model dynamics.

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<sup>31</sup>. 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 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.

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<sup>26</sup>, 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  $\mathbf{T}$  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.

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	$p(t)$
Mechanic	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  $x_1, x_2, \dots, x_e$  and  $\mathbf{y}$  a vector of through variables  $y_1, y_2, \dots, y_e$  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  $\mathbf{M}_1, \mathbf{M}_2, \dots, \mathbf{M}_p$  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.

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

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

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  $\mathbf{A}$ :

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

## 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 sub-graphs; a *spanning tree*  $\mathbf{T}$  (or *spanning 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  $\mathbf{T}$  and some chords, the cut is unique. It is clear that for a tree  $\mathbf{T}$  with  $v - 1$  branches, there will be as many unique cuts. The sum of the vertex equations for all nodes within the cut-set<sup>b</sup> contains only through variables corresponding to the cut-set elements. The set of all cut-set equations can be written in the form:

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

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:

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

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<sup>13</sup> (i.e.,  $\mathbf{A}\mathbf{B}^T = \mathbf{0}$  and  $\mathbf{B}\mathbf{A}^T = \mathbf{0}$ ), we can obtain an expression for  $\mathbf{B}_T$ :

$$\mathbf{B}_T = -\mathbf{A}_C^T \quad (5)$$

---

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

## 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).

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.

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.

## 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. The approach to building a system in the system editor is called *schematic-diagram*. In this approach the modeling is done at the

component level and the interaction between components is defined by connections between *terminals*. 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. Terminal connections and edge associations arise from the interconnection of elements in non-mechanical modeling layers. On the other hand, mechanical interactions such as rigid connections, prismatic joints or revolute joints arise from the interconnection of two rigid bodies.

The data flow diagram illustrating the flow of data through the system is shown in Figure 9. After the design is defined in the system editor, the topology and geometry of the system is derived from the high-level description given in the system editor. The analysis occurs in two parts: the analysis of the mechanical domain, and the analysis of the non-mechanical energy domains. In the mechanical domain, the analysis starts by extracting the kinematic and geometric information from the CAD model. This information is used in the generation of the mechanical system graph. The output of this process is passed to Dynaflex to generate the equations of motion of the mechanism. The analysis of the non-mechanical energy domains starts by constructing the system graph from the topological information provided by the system editor. The system graph is then used to write the terminal equations in causal form from which later a state space form is derived. This set of equations is combined with the equations derived from Dynaflex and with the equations derived from the signal domain. This new set of equations is then transformed to a set of differential algebraic equations which are sorted into a computationally feasible evaluation order<sup>8</sup>. The output of this process is a system in Block Lower Triangular (BLT) form which can be simulated by the simulation kernel. Currently, the BLT form is translated into and simulated in the ASCEND<sup>25</sup> system. Finally, the output of the ASCEND simulation is sent to the visualization process.

### 3.1 Synthesis of the system graph for non-mechanical energy domains

Terminal connections represent the interaction between components within the non-mechanical energy domains. Interactions are non-causal which means that the terminals involved in the connection do not have a predefined direction. A terminal connection between two terminals indicates that both terminals are mapped to a single node in the system graph.

As defined in Section 2, the mathematical model of a component consists of two parts: the terminal equations (behavior) and the terminal graph (topology). The generation of the system graph is a two step process. First, the terminal graphs of the individual components are instantiated to create a disconnected graph with  $n_c$  components, where  $n_c$  is the number of terminal graphs in the system. Second, the information provided by the terminal connections is used to reduce the graph to a non-connected graph with  $n_E < n_c$  components, where  $n_E$  is the number of energy domains involved in the design.

We define a topological operator  $merge(u, v)$  as follows: let  $\mathbf{G}$  be the system graph defined by  $\mathbf{G} = (V, E)$  where  $V$  is the set of vertices and  $E$  is the set of edges represented as ordered sets  $\{u, v\}$ . Then,

$$merge(u, v) \sim \begin{cases} \{u, v\} \in V \Rightarrow V \leftarrow V - \{v\}, \\ \forall (e \in E), v \in e \Rightarrow e \leftarrow (e - \{v\}) \cup \{u\} \end{cases} \quad (6)$$

The generation of the system graph can be thought of as the reduction of a non-connected graph with  $n_c$  components to a non connected graph with  $n_E$  components by a successive application of the merge operator to the terminal graphs of the components.

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:

$$v(t) = K_m \dot{\theta}(t) + R + L \frac{d}{dt} i(t) \quad (7)$$

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.

### **3.2 Synthesis of the system graph for 3D Mechanics.**

The difference between the system graph for non-mechanical energy domains and the system graph for the mechanical domain lies in the dimensionality of the terminal variables. Variables in the mechanical domain are elements of  $\mathfrak{R}^3$  whereas variables in the system graph are elements of  $\mathfrak{R}$  (i.e., scalars.) Once the system graph for the mechanical domain is generated, 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<sup>30</sup>. DynaFlex is a research system developed at the University of Waterloo and is based on 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). DynaFlex is based on the same principles as those used in the derivation of the system equations for non-mechanical energy domains and can therefore be seamlessly integrated with our approach.

Our approach is general enough to accept different mechanism analysis tools; however the only restriction is that it must provide dynamic equations in symbolic form. The reason being that these equations are to be combined later with the remaining system equations derived from the other non-mechanical energy domains (see Figure 9.)

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. Our work on kinematic and geometric analysis<sup>28</sup> allows us to automatically determine the instantaneous kinematic relationships between components in the mechanism. This geometric analysis 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. As a result, modifications to the geometry

of a component in a CAD system are automatically updated in the corresponding simulation model.

Similar to the basic modeling elements we defined in Section 2, Dynaflex provides a set of modeling elements for mechanical systems, including<sup>30</sup>: body elements, arm elements (position vectors), motion and force drivers, spring-damper-actuator elements, and joint elements. This classification of elements is used to assign weights in the normal tree selection algorithm presented in Section 4. Once the mechanical system graph has been defined—as indicated later in the section—penalty costs are assigned to the edges of the system graph based on the type of element they model. This weighted graph is used to find a tree that will define the causality of the terminal equations associated with the edges in the mechanical system graph. This tree is then found by applying the minimum cost spanning tree to the weighted graph.

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.  $\xi$  contains the bodies to be combined into the current composite, and *SYSTEM* is the set of CG nodes of the system graph.

**A1.** Set  $CLOSED \leftarrow \emptyset$ ,  $LUMPS \leftarrow \emptyset$

**A2.** While  $SYSTEM \neq \emptyset$  do

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

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

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

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

**A5.** While  $OPEN \neq \emptyset$  do

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

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

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

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

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

**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  $\alpha$  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 10.

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)
REVOLUTE	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 11. Secondly, Algorithm A identifies the composites listed in Table 3.

**Table 3.** Composite bodies found by Algorithm A

<b>Composite</b>		<b>Constituents</b>	
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

<b>Type of Joint</b>	<b>Reference body</b>	<b>Secondary body</b>
REVOLUTE	BODY_2	BODY_3
REVOLUTE*	BODY_2	BODY_3
REVOLUTE	BODY_3	BODY_1
REVOLUTE*	BODY_3	BODY_1

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 12.

In addition to the inertial parameters and the kinematic properties, dynamic elements such as external forces, and forces acting between any two bodies are included in Dynaflex representation. For this example, only two force elements are introduced: e8 and e12, 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. The edges (e1, e3, e5) represent the weights of BODY\_2, BODY\_3, and BODY\_1, respectively. If the characteristics of the geometric components are changed, the dynamic model is automatically updated to reflect those changes.

## 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:

$$\dot{x} = f(x, t) \tag{8}$$

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,  $s$ :

$$p^{(n)} = f(s) \tag{9}$$

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

$$s = g(p) \tag{10}$$

Then, by substituting the constraint equations (10) into the terminal equations (9), we get a minimal set of dynamic equations of the form:

$$p^{(n)} = f(g(p)) \quad (11)$$

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:

$$\begin{aligned} \mathbf{y}_T &= -\mathbf{A}_C \mathbf{y}_C \\ \mathbf{x}_C &= -\mathbf{B}_T \mathbf{x}_T \end{aligned} \quad (12)$$

From equations (10) and (12), 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 (12) are identified as *secondary variables* of the system graph.

Based on the selection of primary and secondary variables, we can obtain dynamic equations of the form (11) 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  $\mathbf{G}$  is found by defining a real function  $w: e \rightarrow \mathfrak{R}^+$  on the edges of  $\mathbf{G}$  that computes the weight of the edges as follows:

Let  $\kappa_\alpha$  and  $\kappa_\tau$  represent the highest derivative order of all accumulator elements and all delay elements respectively, and  $O: e \rightarrow \mathfrak{R}^+$  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  $\mathbf{G}$  as follows: let all across drivers and generalized across drivers belong to the class  $c^\Delta$ , accumulator elements to the classes

$$c_i^\alpha = \{e_\alpha | O(e_\alpha) = \kappa_\alpha - i\} \quad i = 0, \dots, \kappa_\alpha - 1, \quad (13)$$

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

$$c_i^\tau = \{e_\tau | O(e_\tau) = \kappa_\tau - i\} \quad i = 0, \dots, \kappa_\tau - 1. \quad (14)$$

Finally, all through drivers and generalized through drivers will belong to class  $c^\Phi$ . The weight functions  $w$  defined on the edges of  $\mathbf{G}$  are chosen for each class such that

$$w^\Delta < w_0^\alpha < w_1^\alpha < \dots < w_{\kappa_\alpha - 1}^\alpha < w^\delta < w_0^\tau < w_1^\tau \dots < w_{\kappa_\tau - 1}^\tau < w^\Phi \quad (15)$$

where  $w^\Delta$  is the weight function associated to class  $c^\Delta$ ,  $w_i^\alpha$  is the weight function associated to class  $c^\alpha$ ,  $w_i^\delta$  is the weight function associated to class  $c^\delta$ ,  $w_i^\tau$  is the weight function associated with class  $c^\tau$ , and  $w_i^\Phi$  is the weight function associated to class  $c^\Phi$ .

In other words, the weight function  $w$  ranks the edges of  $\mathbf{G}$  based on their respective classes. Any weight function that satisfies the ranking in equation (15) is admissible.

The next step in our approach is to find a *minimum cost spanning tree*<sup>c</sup> that minimizes the total cost (weight) of the weights assigned to the branches of the tree. Aho et. al.<sup>1</sup> shows that it is always possible to find such a tree based on the following property: let  $V$  be the set of vertices of  $\mathbf{G}$  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.<sup>1</sup>

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

---

c. A spanning tree for  $\mathbf{G}$  is a tree that connects all vertices in  $\mathbf{G}$ .

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 topological information of this design is specified in the *System Editor* as indicated in Figure 13; the nodes represent components, and edges represent physical connections established between components. The components in the system are port-based objects<sup>7</sup> that have a well defined interface that prescribes its interaction with the environment. In addition to the interface each component includes its mathematical model from which the terminal graphs are instantiated in the system graph with  $n_c = 4$ . Terminal connections derived from the system description are used to perform successive applications of the merge operation to reduce the electrical system graph to a connected graph with  $n_E < n_c = 4$ . For instance the electrical side of the `pitch-motor` and the `amp-voltage-source` are connected through connections<sup>d</sup> (3, 1) and (4,2) which result in two merge operations as indicated in Figure 14. A similar process occurs with the other two electric components leaving an electric system graph with two components. However, the terminal connection between the `amp-voltage-sources` defines the common ground which merges the ground nodes of the two components reducing the electrical system graph to a connected graph with a single component.

This process completely defines the system graph for the system. The system graph is a non-connected graph with two connected components: one connected component for the electrical domain and one connected component for the mechanical domain (Figure 15.)

To write a system of differential equations in state space form, we proceed to find the normal forest (i.e., a normal tree in each connected component of the system graph). This is accomplished by assigning penalty weights to the edges of the system graph and then computing the minimum cost spanning tree on each connected component. For the electri-

---

d. Terminal connections are represented as non-directed edges to reflect the non-causal relation between the terminals.

cal system graph the weights are assigned based on the form of the terminal equation associated with the edge which are shown in Table 5:

**Table 5.** Terminal equations for the electrical components in the seeker.

Domain	Component	Edge	Terminal equation
Electrical	Pitch motor	18	$v_{18}(t) = k_{m_\alpha} \dot{\theta}_9(t) + R_{18} i_{18}(t) + L_{18} \frac{d}{dt} i_{18}(t)$
	Yaw motor	15	$v_{15}(t) = k_{m_\beta} \dot{\theta}_{13}(t) + R_{15} i_{15}(t) + L_{15} \frac{d}{dt} i_{15}(t)$
	Controlled voltage driver	17	$V_{17}(t) = E_\alpha(t)$
	Controlled voltage driver	16	$V_{16}(t) = E_\beta(t)$

Following the classification rules defined before, the elements in the electrical system graph fall in the following classes:

**Table 6.** Classification of electrical components in the electrical system graph

Edge	Class	Weight
18	first order delay	$w_{18}$
15	first order delay	$w_{15}$
17	signal-controlled across driver	$w_{17}$
16	signal controlled across driver	$w_{16}$

where the weights  $w_i$  are selected such that

$$(w_{16} = w_{17}) < (w_{15} = w_{18}) \quad (16)$$

A similar approach is applied to the mechanical system graph resulting in the normal forest shown in Figure 15 and indicated by bold lines. The next step in our derivation is to write the system of equations from the system graph. We use Dynaflex to derive the equations of motion of the mechanism while our system takes care of the non-mechanical energy domain. In this process, the equations are first written in causal form derived from the normal forest as indicated in Equation (17). The last equation in (17) represents the equations of motion generated by Dynaflex where  $\mathbf{M}$  is the inertia matrix of the system,  $\mathbf{V}$  is the

vector of centrifugal and coriolis terms,  $\mathbf{G}$  is the vector of gravity terms, and  $\mathbf{F}$  is the vector of external forces such as friction forces or other non-rigid body effects.

$$\begin{aligned}\frac{d}{dt}i_{18}(t) &= f_{\alpha}(\dot{\theta}_9, i_{18}, v_{18}) \\ \frac{d}{dt}i_{15}(t) &= f_{\beta}(\dot{\theta}_{13}, i_{15}, v_{15}) \\ \tau &= \mathbf{M}(\Theta)\ddot{\Theta} + \mathbf{G}(\Theta) + \mathbf{V}(\Theta, \dot{\Theta}) + \mathbf{F}(\Theta, \dot{\Theta})\end{aligned}\tag{17}$$

The dynamic equations are augmented using the coupling equations defined by the associations indicated in the system editor to complete the definition of the system of ODEs. This system is then transformed into six first order differential equations to which the equations derived from the signal domain are appended. This last step yields the dynamic equations that describe the behavior of the seeker including the controllers from the signal domain.

The system of ODEs is then integrated over time to obtain the behavior of the system (Figure 16.) As an example of the output and one of the visualization options, the system is presented with step functions as inputs. Figure 17 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.

## 5. Conclusions

The composable simulation approach offers new and promising possibilities in the area of intelligent CAD. By composing simulations of mechatronic systems the designer is able to explore a larger portion of the design space. This results in improved quality and reduced costs of the designed artifact. In our framework for composable simulation, the designer works at the level of system components. These components have multiple mathematical models associated with them. The mathematical models include CAD models (geometry), dynamic equations and topological information about the component. The framework for composable simulation automatically generates system-level simulations from the component models and their interactions as described by the user. Inertial and kinematic properties of the design are automatically derived from the CAD models, so that changes in geometry are automatically updated in the simulation models.

Our approach is based on linear graph theory. We extended the linear graph approach to include signal elements creating a hybrid representation for mechatronic systems. This representation combines linear graphs with block diagrams providing a multi-domain modeling framework. Component models of physical components are represented by terminal graphs with corresponding terminal equations. Terminal equations are expressed in non-causal form to reflect the fundamental nature of the physics of the modeled component. This approach provides greater modeling flexibility and promotes model reuse. Terminal graphs are combined into a system graph, which is a domain independent representation of the system dynamics. Furthermore, a minimum cost spanning tree algorithm is used to express the terminal equations and the constraint equations in causal form. A further reduction process results in dynamic 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. We would also like to thank the reviewers. Their insightful comments improved the quality of the article significantly.

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] The Boeing Company, "Easy5 Engineering analysis system," 1999.
- [4] F. E. Cellier, Continuous system modeling. Springer-Verlag, 1991.
- [5] F. E. Cellier, Automated formula manipulation supports object-oriented continuous-system modeling. IEEE Control Systems, Vol. 2, No. 13, April 1993. pp. 28-38.
- [6] Controllab Products B. V., "20-sim," AE Enschede, the Netherlands, 1999.
- [7] 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, September 1998.
- [8] A. Diaz-Calderon, C. J. J. Paredis, and P. K. Khosla, "Combining information technology components and symbolic equation manipulation in modeling and simulation of mechatronic systems," 1999 IEEE International Symposium on Computer Aided Control System Design, Island of Hawaii, Hawaii, August 1999.
- [9] Dynasim AB, "Dymola," Lund, Sweden, 1999.
- [10] H. Elmqvist, S. E. Mattsson, and M. Otter, "Modelica: The new object-oriented modeling language," presented at The 12th European Simulation Multiconference, Manchester, UK, 1998.
- [11] H. Elmqvist, and D. Bruck. "Constructs for object-oriented modeling of hybrid systems." In Eurosim Simulation Congress, Vienna, Austria, September 11-15, 1995.
- [12] D. C. Karnopp, Margolis, D. L., and R. C. Rosenberg. System dynamics: A unified approach. John Wiley & Sons, Inc. New York, NY, 1990.
- [13] H. E. Koenig, Tokad, Y., Kesavan, H. K., and H. G. Hedges, Analysis of discrete physical systems. New York: MacGraw-Hill, 1967.
- [14] 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.
- [15] LMS CADSI, "DADS," 1999.

- [16] The Mathworks Inc., "Matlab/Simulink," 1999.
- [17] J. J. McPhee, "On the use of linear graph theory in multibody system dynamics," *Nonlinear Dynamics*, vol. 9, pp. 73-90, 1996.
- [18] J. J. McPhee, M. G. Ishac, and G. C. Andrews, "Wittenburg's formulation of multi-body dynamics equations from a graph-theoretic perspective," *Mechanism and Machine Theory*, vol. 31, pp. 202-213, 1996.
- [19] 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.
- [20] Mechanical Dynamics Inc., "ADAMS," 1999.
- [21] B. J. Muegge, "Graph-theoretic modeling and simulation of planar mechatronic systems," MA. Sc. Thesis, Systems Design Engineering, University of Waterloo, Waterloo, 1996.
- [22] P. E. Nikravesh and I. S. Chung, "Application of euler parameters to the dynamic analysis of three-dimensional constrained mechanical systems," *Journal of Mechanical Design*, vol. 104, pp. 785-791, 1982.
- [23] N. Orlandea, M. A. Chace, and D. A. Calahan, "A sparsity-oriented approach to the dynamic analysis and design of mechanical systems - Part 1," *Journal of Engineering for Industry*, August, 1977.
- [24] N. Orlandea, M. A. Chace, and D. A. Calahan, "A sparsity-oriented approach to the dynamic analysis and design of mechanical systems - Part 2," *Journal of Engineering for Industry*, August, 1977.
- [25] P. C. Piela, T. G. Epperly, K. M. Westerberg, and A. W. Westerberg, "ASCEND: An object oriented computer environment for modeling and analysis. 1 - The modeling language," *Comput. Chem Engng*, vol. 15, no. 1, pp. 53-72, 1991.
- [26] P. H. O'n. Roe, *Networks and systems*. Reading, Massachusetts: Addison-Wesley, 1966.
- [27] R. C. Rosenberg, and D. C. Karnopp. Introduction to physical system dynamics. McGraw-Hill, New York, NY, 1983.

- [28] 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.
- [29] S. Seshu and M. B. Reed, *Linear graphs and electrical networks*. Reading, Massachusetts: Addison-Wesley, 1961.
- [30] 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.
- [31] 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.
- [32] J. Wittenburg and U. Wolz, “MESA VERDE: A symbolic program for nonlinear articulated-rigid-body dynamics,” presented at ASME Design Engineering Conference, Cincinnati, OH, 1985.

## 8. Appendix

$$\begin{aligned}
\ddot{\beta}(t) = & (-0.18419821(10^{-6})\cos(\alpha(t))\dot{\beta}(t)^2 \sin(\alpha(t)) \\
& - 0.42128079(10^{-6})\dot{\beta}(t)\dot{\alpha}(t)\cos(\alpha(t)) \\
& + 0.31596059(10^{-6})\dot{\beta}(t)\dot{\alpha}(t)\sin(\alpha(t)) + 0.1579803(10^{-6})\dot{\alpha}(t)^2 \sin(\alpha(t)) \\
& + 0.1579803(10^{-6})\dot{\beta}(t)^2 \sin(\alpha(t)) - 0.63153674(10^{-6})\sin(\alpha(t))^2 \dot{\beta}(t)^2 \\
& - 0.00032463Tm_{\alpha}(t) + 0.00032463Tm_{\beta}(t) - 0.2106404(10^{-6})\dot{\beta}(t)^2 \\
& \cos(\alpha(t)) - 0.00048665\cos(\alpha(t))Tm_{\beta}(t) - 0.00064886\sin(\alpha(t))Tm_{\beta}(t) \\
& + 0.31576837(10^{-6})\dot{\beta}(t)^2 - 0.2106404(10^{-6})\dot{\alpha}(t)^2 \cos(\alpha(t))) \\
& /(-0.63153674(10^{-6}) \\
& \sin(\alpha(t))\cos(\alpha(t)) + 0.12389914(10^{-5}) - 0.18419821(10^{-6})\sin(\alpha(t))^2)
\end{aligned} \tag{18}$$

$$\begin{aligned}
\ddot{\alpha}(t) = & (0.36839643(10^{-6})\dot{\beta}(t)^2 \sin(\alpha(t)) - 0.42128079(10^{-6})\dot{\beta}(t)\dot{\alpha}(t) \cos(\alpha(t)) + 0.31596059(10^{-6})\dot{\beta}(t)\dot{\alpha}(t) \sin(\alpha(t)) + 0.63153673(10^{-6})\dot{\beta}(t)\dot{\alpha}(t) + 0.18419821(10^{-6}) \sin(\alpha(t))\dot{\alpha}(t)^2 \cos(\alpha(t)) - 0.12630735(10^{-5}) \sin(\alpha(t))^2 \dot{\beta}(t)\dot{\alpha}(t) + 0.1579803(10^{-6})\dot{\alpha}(t)^2 \sin(\alpha(t)) \\
& + 0.237036(10^{-5})\dot{\beta}(t)^2 \sin(\alpha(t)) - 0.12630735(10^{-5}) \sin(\alpha(t))^2 \dot{\beta}(t)^2 - 0.00032463Tm_{\alpha}(t) + 0.00487079Tm_{\beta}(t) \\
& + 0.00048665 \cos(\alpha(t))Tm_{\alpha}(t) - 0.63153674(10^{-6}) \sin(\alpha(t))^2 \dot{\alpha}(t)^2 + 0.31576837(10^{-6})\dot{\alpha}(t)^2 - 0.316048(10^{-5})\dot{\beta}(t)^2 \cos(\alpha(t)) \\
& - 0.0009733 \cos(\alpha(t))Tm_{\beta}(t) - 0.00129773 \sin(\alpha(t))Tm_{\beta}(t) + 0.36839643(10^{-6}) \sin(\alpha(t))\dot{\beta}(t)\dot{\alpha}(t) \cos(\alpha(t)) + 0.63153674(10^{-6})\dot{\beta}(t)^2 \\
& - 0.2106404(10^{-6})\dot{\alpha}(t)^2 \cos(\alpha(t)) + 0.00064886 \sin(\alpha(t))Tm_{\alpha}(t) \\
& /(-0.63153674(10^{-6}) \sin(\alpha(t)) \cos(\alpha(t)) + 0.12389914(10^{-5}) - 0.18419821(10^{-6}) \sin(\alpha(t))^2)
\end{aligned} \tag{19}$$

$$\frac{d}{dt}i_{\alpha}(t) = -\frac{-E_{\alpha}(t) + 0.000813\dot{\alpha}(t) + 0.2i_{\alpha}(t)}{200(10^{-6})} \tag{20}$$

$$\frac{d}{dt}i_{\beta}(t) = -\frac{-E_{\beta}(t) + 0.00008\dot{\beta}(t) + 0.2i_{\beta}(t)}{200(10^{-6})} \tag{21}$$

$$Tm_{\alpha}(t) = 0.0813i_{\alpha}(t) \tag{22}$$

$$Tm_{\beta}(t) = 0.032i_{\beta}(t) \tag{23}$$

$$E_{\alpha}(t) = PID(u_{\alpha}(t), \alpha(t)) \tag{24}$$

$$E_{\beta}(t) = PID(u_{\beta}(t), \beta(t)) \tag{25}$$

# Captions

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

**Figure 2.** n-terminal component

**Figure 3.** Positioning system

**Figure 4.** System graph for the positioning system

**Figure 5.** Terminal graph of variable elements.

**Figure 6.** Reading values from a terminal graph

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

**Figure 8.** Modeling layers of a mechatronic system.

**Figure 9.** System data flow diagram.

**Figure 10.** Missile seeker

**Figure 11.** Extended system graph. Only joint and body elements are shown for clarity.

**Figure 12.** Reduced system graph.

**Figure 13.** System editor.

**Figure 14.** Topological operations to a connected electrical system graph.

**Figure 15.** System graph for the missile seeker.

**Figure 16.** Screen shoot of the complete CAD environment.

**Figure 17.** System response as a function of time.





































