1999 SCS European Simulation Symposium, Erlangen, Germany, October 26-28 1999.

A COMPOSABLE SIMULATION ENVIRONMENT FOR MECHATRONIC SYSTEMS

Antonio Diaz-Calderon, Christiaan J. J. Paredis[1], Pradeep K. Khosla
Department of Electrical and Computer Engineering
Institute for Complex Engineered Systems
Carnegie Mellon University
Pittsburgh, PA 15213, USA
E-mail: cjp@cmu.edu

KEYWORDS

Composable simulation, design verification, linear graphs, CAD, mechatronic systems.

ABSTRACT

We present a software environment for composable simulation of mechatronic systems. By composable simulation we mean the ability to automatically generate simulations from individual component models through manipulation of the corresponding physical components in a CAD system. This form of virtual prototyping will reduce the design cycle significantly by providing immediate feedback to the designer with minimal intervention of simulation and modeling specialists.

INTRODUCTION

The work presented in this paper is the result of an ongoing 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/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.

To address the composable simulation problem outlined above, we developed a methodology based on a system graph representation. The system graph is a linear graph which captures the topology of the energy flow in the system. We have extended the system graph to include signal components by combining the linear graph with block diagrams, resulting in a unified system graph representation. The system graph is used to generate the set of differential-algebraic equations that describe the system behavior including the information technology components.

RELATED WORK

The relationship between physical systems and linear graphs was first recognized by Trent (Trent 1995) and by Brannin (Brannin 1966). Linear graph theory has been used in different engineering domains, including, systems theory (Roe 1966; Koenig 1967), analysis of rigid body dynamics (McPhee et al. 1996), and on the analysis of multi-energy domain engineering systems (Muegge 1996). The system graph approach that we are developing builds on linear graph theory.

Another graph representation is bond graphs (Karnopp et al. 1990). 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 because they can be more easily adapted to model 3D rigid body mechanics, and reflect the topology of the physical system directly.

1. Corresponding author.

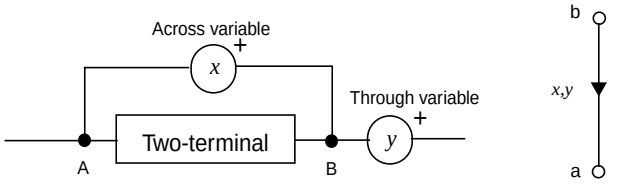

Figure 1: Through and across measurements on a general two-terminal element and its terminal graph.

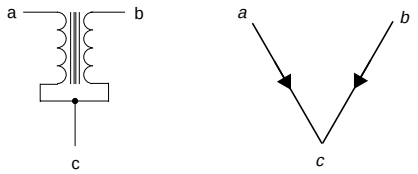

Figure 2: n-terminal component.

Composition of simulation models can also be accomplished by combining fundamental building blocks described in a high level object-oriented modeling language (Cellier 1993; Elmqvist and Bruck 1995). 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.

A different approach to composability of simulation models is presented in (Diaz-Calderon et al. 1998). In this work a software architecture that supports the integration of simulation modules is defined: composition is achieved by connecting software components through a well defined interface. The arrangement of components defines the system to be executed by the simulation engine.

To compose simulation models directly from 3D mechanical systems, in (Diaz-Calderon et al. 1999a) we present a methodology that derives the system graph of a 3D mechanical system directly from the geometry. This system graph is then used to derive the dynamic equations for the 3D mechanism (Diaz-Calderon et al. 1999b).

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 (Trent 1955). 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.

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 1966), 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.

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 the terminal variables associated with different energy domains. The derivatives of across or through variables are across or through variables as well. For example, velocity and acceleration are across variables while the derivatives of force and torque are also through variables.

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	$f(t)$, $\tau(t)$	Displace-ment	$\mathbf{r}(t)$, $\theta(t)$

The algebraic properties of a linear graph determine two sets of *constraint equations*: the $e - v + p$ fundamental circuit equations and the $v - p$ cut-set equations of a system graph **G** with e edges, v vertices and p connected components (1), where **A** and **B** are the incidence and circuit matrices respectively.

$$\mathbf{x}_C(t) = -\mathbf{B}_T\mathbf{x}_T(t) \quad \text{and} \quad \mathbf{y}_T(t) = -\mathbf{A}_C\mathbf{y}_C(t) \qquad (1)$$

Together these equations form a system of e linearly independent equations in $2e$ unknowns. To find a unique solution to this system, we add the e independent equations that are derived from the relationships between the across and through variables for the components in the system graph. In general, for an n-terminal component, there will be n terminal equations.

Components in a mechatronic system are represented by one or more edges in the system graph connecting well defined interface points. The subset of edges of the system graph that represent a system component is called *terminal graph*.The model of a system component includes both the terminal equations and the associated terminal graph. The terminal graph provides the topological structure of the system component while the terminal equations provide the mathematical model of the basic operation of the system component.

In order to include software components as well as other types of low-power devices in the system graph modeling approach, it is necessary to extend our view of the modeling elements presented so far to include the use of signals. A *signal* represents the flow of some system variable value at a very low power level. In (Diaz-Calderon et al. 1999c), we present an extension to the linear graph representation to accept the modeling of low power components.

FRAMEWORK FOR COMPOSABLE SIMULATION

The framework for composable simulation is based on the system graph representation of 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 based on *schematic-diagrams.* In this approach, the modeling is performed at the component level and the interaction between components is defined by connections between *terminals*. 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) terminal connections, 2) edge associations, and 3) mechanical interactions. 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.

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 predefined direction. A terminal connection between two terminals indicates that both terminals are mapped to a single node in the system graph.

The process of generating 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 nonconnected graph with $n_E < n_c$ components, where n_E is the number of energy domains involved in the design (Diaz-Calderon et al. 1999b).

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.

Mechanical interactions are handled differently due to the difference in dimensionality of the terminal variables involved. The generation of the system graph involves a direct translation of the kinematic information into the linear graph representation (Diaz-Calderon et al. 1999a). 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, 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 (Diaz-Calderon et al. 1999b).

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:

$$\dot{x} = f(x, t) \tag{2}$$

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:

$$p^{(n)} = f(s) \tag{3}$$

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

$$s = g(p) \tag{4}$$

Then, by substituting the constraint equations (4) into the terminal equations (3), we get a minimal set of dynamic equations of the form:

$$p^{(n)} = f(g(p)) \tag{5}$$

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 equation (1) 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:

$$\begin{aligned} \mathbf{y}_T &= -\mathbf{A}_C \mathbf{y}_C \\ \mathbf{x}_C &= -\mathbf{B}_T \mathbf{x}_T \end{aligned} \tag{6}$$

From equations (4) and (6), 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 (6) are identified as s*econdary variables* of the system graph.

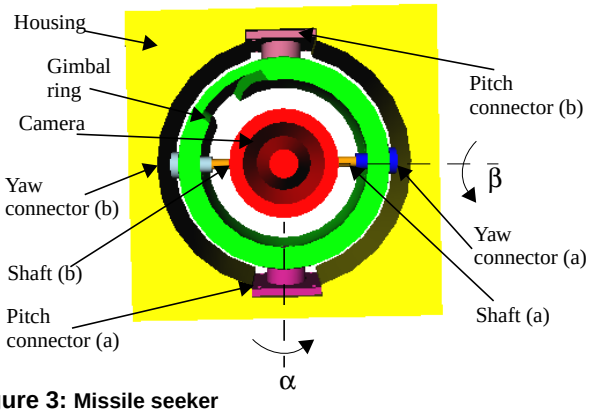

Figure 3: Missile seeker

Based on the selection of primary and secondary variables, we can obtain dynamic equations of the form (5) 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. An algorithm to automatically derive the normal tree of a system graph is presented in (Diaz-Calderon et al. 1999b).

Once the state space form of the dynamic equations is found, the system of equations is augmented with the equations derived from the signal domain which may include references to software components. This new system of equations completely describes the mechatronic system. The equations are then sorted into Block Triangular Form (BLT) to obtain a computational order of evaluation of the equations and software components (Diaz-Calderon et al. 1999c).

EXAMPLE

To illustrate the concepts presented in this paper we have selected the design of a missile seeker shown in Figure 3.

The topological information of this design is specified in the *System Editor* as shown in Figure 4 where nodes represent components and edges between components represent physical connections established between components. The components in the system are port-based objects (Diaz-Calderon et al. 1998) that have a well defined interface that prescribes their interaction with the environment. In addition to the interface, each component includes its mathematical model from which the terminal graphs are instantiated.

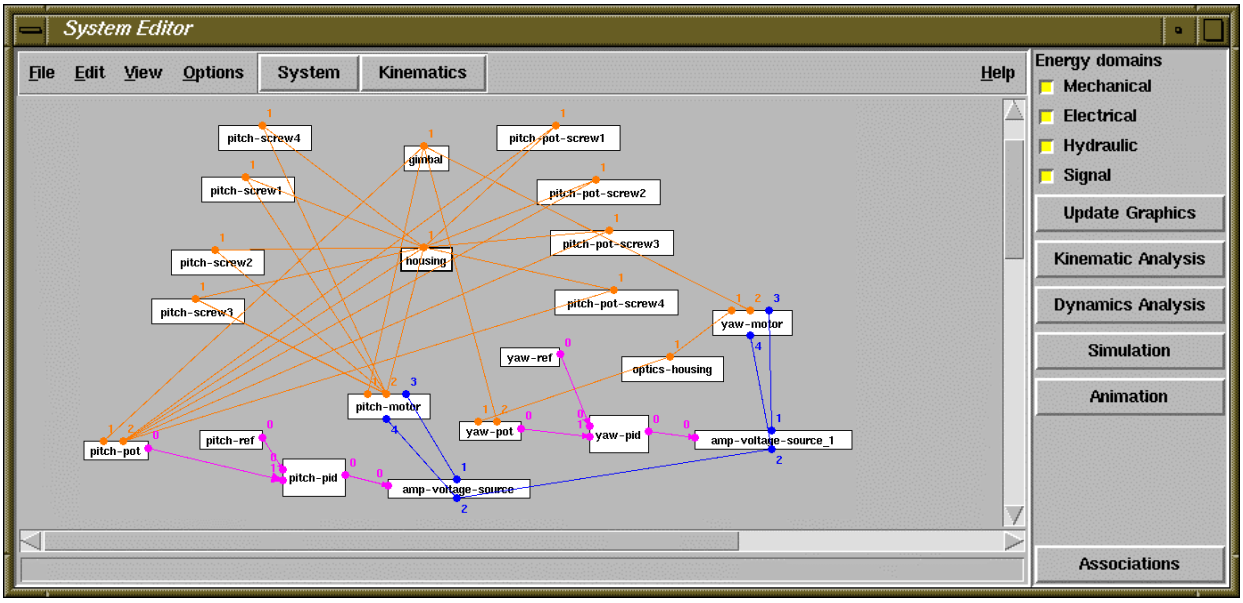

Figure 4: System editor: missile seeker input graph

Synthesis of the System Graph.

Terminal connections derived from the system description are used to reduce the electrical system graph to a connected graph with $n_E < n_c = 1$.

The generation of the mechanical system graph involves a direct translation of the kinematic information into the linear graph representation. The result of this stage is a mechanical system graph that includes all kinematic information including fixed joints and redundant joints. However, to avoid structural singularities and indexing problems, we simplify this initial mechanical 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 starting from the node representing the center of mass of a body (Diaz-Calderon et al. 1999b). Once composite bodies have been identified, the mechanical system graph is topologically modified such that composite bodies are combined into single bodies and redundant joints are removed. In this context, redundant joints are joints that duplicate already existing kinematic constraints; for instance, co-linear revolute joints.

Redundant joints need to be removed from the representation to prevent us from interpreting the result as an overconstrained system. Possibly overconstrained systems can be recognized in the system graph as kinematic loops. Our work on geometric and kinematic analysis (Sinha et al. 1998) allows us to determine whether a kinematic loop contains a redundant joint or whether it results in an overconstrained system.

Combining bodies joined by fixed joints requires the system to extract and combine inertial parameters. This step involves access to the CAD models to automatically derive these properties. This implies that changes in the geometry will be automatically propagated to changes in the simulation model.

Geometric analysis on the seeker shows it contains 9 bodies (not counting the screws for the sake of clarity): housing, gimbal ring, camera, pitch connector (2) yaw connector (2), shaft (2). The kinematic analysis of the system reveals the kinematic constraints among the bodies (Table 2). Based on these constraints, the bodies that can be combined to form composites are identified.

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

The application of the process outlined above yields a mechanical system graph with only three bodies.

Finally, collecting the system graphs for the electrical and mechanical subsystems into a single graph yields a non-connected system graph (Figure 5).

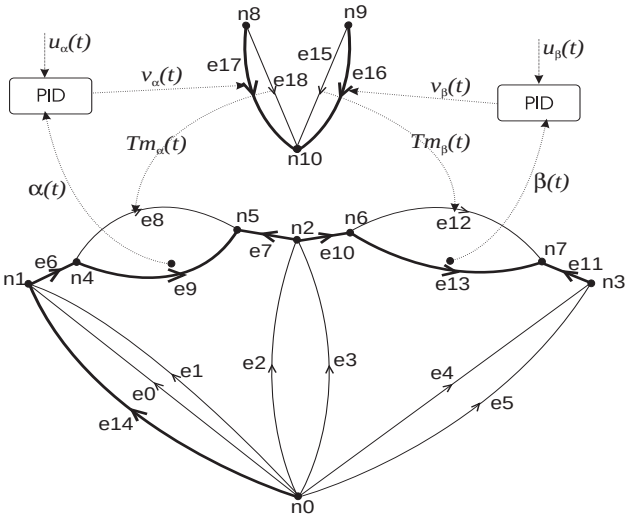

Figure 5: System graph of the seeker design

Synthesis of Dynamic Equations.

The next step in our derivation is to write the system of equations from the system graph. 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 (Aho et al. 1987) on each connected component (Diaz-Calderon et al. 1999b). For the electrical system graph the weights are assigned based on the form of the terminal equation associated with the edges which are shown in Table 3.

Table 3: Terminal equations for the electrical components in the seeker

Component	Edge	Terminal equation
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	$V_{16}(t) = E_\beta(t)$

The form of the equation associated with an edge specifies its class which can be delay, accumulator, driver or transducer.

Once the normal forest is found (shown by bold lines in Figure 5), we proceed with the derivation of the dynamic equations. We use Dynaflex (Shi 1998) 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 (7). The last equation in (7) represents the equations of motion generated by Dynaflex where **M** is the inertia matrix of the system, **V** is the vector of centrifugal and coriolis terms, **G** is the vector of gravity terms, and **F** is the vector of external forces such as friction forces or other non-rigid body effects.

$$\begin{aligned} \frac{d}{dt}i_{18}(t) &= f_\alpha(\dot{\theta}_9, i_{18}, v_{18}) \\ \frac{d}{dt}i_{15}(t) &= f_\beta(\dot{\theta}_{13}, i_{15}, v_{15}) \\ \tau &= \mathbf{M}(\Theta)\ddot{\Theta} + \mathbf{G}(\Theta) + \mathbf{V}(\Theta, \dot{\Theta}) + \mathbf{F}(\Theta, \dot{\Theta}) \end{aligned} \tag{7}$$

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 sorted into BLT form and integrated in time to obtain the response of the system. In Figure 5, the input reference signal to the system is represented by *u(t)* while the sensor is represented by the edge labeled $\beta(t)$. The mechanical system receives input torques that are generated by the transducer represented by *Tm(t)*.

SUMMARY

The composable simulation approach to modeling and simulation of mechatronic systems such as the one presented in this paper offer new and promising possibilities in the design arena. Despite the advances in modeling and simulation included in many simulation environments, the task of creating a simulation model for a mechatronic system still requires significant expertise. As a solution to this problem, we have presented a framework for composable simulation in which CAD component models are automatically combined to create system-level simulation models and in which the inertial and kinematic properties of the design are automatically derived from the CAD model.

ACKNOWLEDGMENTS

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.

REFERENCES

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

Branin, F. H. 1966. "The algebraic-topological basis for network analogies and the vector calculus." *Symposium on Generalized Networks* (Polytechnic Institute of Brooklyn).

Cellier, F. E. 1993. "Automated formula manipulation supports object-oriented continuous-system modeling." *IEEE Control Systems*, Vol. 2, No. 13.

Diaz-Calderon A., C. J. J. Paredis, and P. K. Khosla. 1998. "A modular composable software architecture for the simulation of mechatronic systems." *ASME Design Engineering Technical Conference, 18th Computers in Engineering Conference (*Atlanta, GA, September).

Diaz-Calderon A., C. J. J. Paredis, and P. K. Khosla. 1999a. "On the synthesis of the system graph for 3D mechanics." *American Control Conference* (San Diego, CA, June).

Diaz-Calderon A., C. J. J. Paredis, and P. K. Khosla. 1999b. "Automatic generation of system-level dynamic equations for mechatronic systems." Tech. Report 04-14-9. Institute for Complex Engineered Systems, Carnegie Mellon University, Pittsburgh PA.

Diaz-Calderon A., C. J. J. Paredis, and P. K. Khosla. 1999c. "Combining information technology components and symbolic equation manipulation in modeling and simulation of mechatronic systems." *IEEE International Symposium on Computer Aided Control System Design* (Island of Hawaii, HI, August).

Elmqvist, H., and D. Bruck. 1995. "Constructs for object-oriented modeling of hybrid systems." *Eurosim Simulation Congress* (Vienna, Austria, September).

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

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

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

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

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

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

Sinha, R., C. J. J. Paredis; S. K. Gupta; and P. K. Khosla. 1998. "Capturing Articulation in Assemblies from Component Geometry." *Proceedings of the ASME Design Engineering Technical Conference (*Atlanta, GA, September).

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