Automated Generation and Simulation of Component-Based Generalized Bond Graphs

Siyuan Dai, Xenofon Koutsoukos
Institute for Software-Integrated Systems
Department of Electrical Engineering and Computer Science
Vanderbilt University
siyuan.dai@vanderbilt.edu, xenofon.koutsoukos@vanderbilt.edu

Keywords: bond graph, generalized bond graph, components, simulink, modelica

Abstract

The concept of causality presents a problem for component-based modeling of bond graphs, the uncertainty of the causality of each interaction bond. As a result we use generalized bond graphs instead, which is mathematically represented by its Dirac structure. Through the use of generalized bond graphs we can represent each component as a causal computational model (hybrid-input-output representation) or as an acausal computational model (kernel representation). Our contribution consists of developing a method which exploits the Dirac structure of a generalized bond graph component, and generates both a causal and acausal computational model for that component. A case study, with subsequent simulation results, is used to showcase both methods and help compare and contrast both computational models.

1. INTRODUCTION

Bond graph is a useful tool for modeling physical systems, especially when multiple domains are involved. One of its major distinctions from other tools such as signal-flow graphs is that every arc in a bond graph represents an exchange of power, rather than an exchange of signal. A bond graph can generate equations and block diagrams using the idea of causality [5].

Our paper was motivated by an idea of hierarchy: a (global) bond graph model broken down into interacting components, where each component is also bond graph and each interaction is a bond connecting two components. A causality assignment on the bond graph requires strokes to be assigned to every bond in the model, which includes the interaction bonds between components.

The causality of interaction bonds can be easily determined when looking at the global model, but if we only look at a particular component there is no way of determining the causality of an interaction bond; like a resistor, its causality can go either way and still be valid. Without determining the causality of interaction bonds, the bond graph model cannot be simulated using a causal simulation tool.

In order to circumvent the causality problem, we modify our motivation by defining components as generalized bond graphs. Each component contains a Dirac structure, a set of power ports, and a set of interaction ports; power ports are attached to traditional bond graph elements such as resistive elements, storage elements, and source elements while interaction ports connect the components to each other. In this paper, components are defined in terms of their physical world counterparts (motors, loads, etc).

Using generalized bond graphs, we can represent a component as either a causal computational model or an acausal computational model. The causal computational model requires all equations to have an inherent cause and effect relationship between variables, whereas the acausal computational model does not have this stringent requirement. In this paper we will compare and contrast the processes in which we derive causal or acausal computational models from generalized bond graphs.

Our contributions include an approach that uses generalized bond graph techniques to model of systems in a compositional way. A system can be fully described by its components and their interactions with each other. We exploit the Dirac structure of each component; component Dirac structures can generate component block diagram subsystems, which can connect together to form the full system. Due to the compositional nature of the Dirac structure, components are hierarchical, that is composition of components can also be characterized by a Dirac structure. In this paper we present the derivations of causal and acausal computational models from the generalized bond graph of components. Our method uses two Dirac structure forms, the kernel representation and the hybrid-input-output representation [10].

In causal modeling, the causality of an interaction port must be determined. For acausal modeling, the causality of an interaction port can be ignored. Therefore, we can see that causal modeling of a component will require a knowledge of its neighboring components, while for acausal modeling, each component can be modeled separately.

Given a generalized bond graph model, we can break its Dirac structure down into interacting components. Every

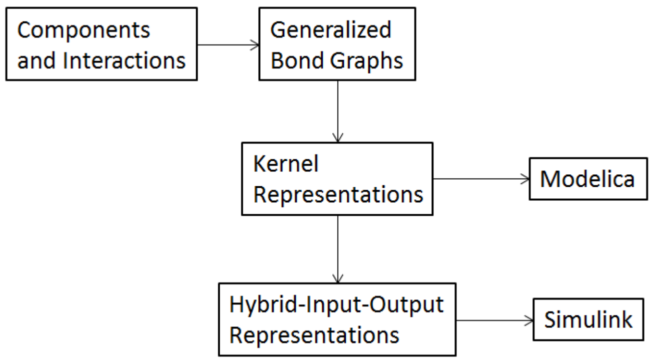
Components and Interactions
Generalized Bond Graphs
Kernel Representations
Modelica
Hybrid-Input-Output Representations
Simulink

Figure 1. Flow chart of the simulation method in this paper

component will have a Dirac structure, which will directly lead to a kernel representation; the kernel representation is an acausal computational model for the component. Using the kernel representation of each component we are able to also derive their respective hybrid-input-output representations, which is a causal computational model. A simple flow chart of the entire process is illustrated in Figure 1.

This paper is organized as follows. In Section 2., short background information on bond graph and generalized bond graph modeling is provided. Section 3. describes component-based acausal simulation method of systems. Section 4. extends the acausal method to a causal component-based simulation method. A case study which showcases our methods is presented in Section 5.. Finally, analysis and discussion of the algorithm is given in Section 6..

2. BACKGROUND

Bond graphs were introduced in 1961 by Dr. Henry Paynter as a means to model physical systems, using the notion of power flow, rather than signal flow. One of the major concepts in bond graph theory is causality, which determines the signal direction of a particular bond. Causality is described in bond graphs using causal strokes, where the placement of the causal stroke determines the direction of effort [4]. A bond graph with no causal conflicts can be transformed into state space equations and block diagrams for simulation [7]. We will use the elastic hoist device featured in Broenink's paper [7] as an example to illustrate concepts. Figure 2 shows the schematic of the elastic hoist device, and Figure 3 contains its corresponding bond graph model.

The *Sequential Causal Assignment Procedure* (SCAP) is an algorithm that assigns causality to every bond in a bond graph model [11]; an application to hybrid bond graphs allows for the derivation of state equations [13]. Methods like SCAP do not allow for the analysis of a system by components; just like any other bond graph algorithms, it requires knowledge of the global model. Therefore, our goal is to generate computational models for each component of a system,

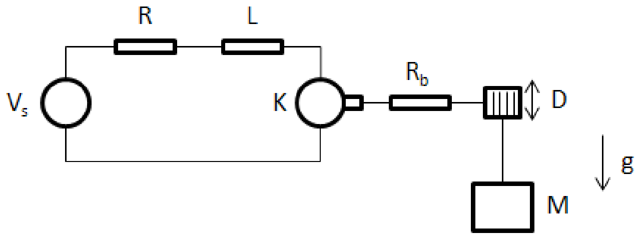

Figure 2. Schematic of the elastic hoist device [7]

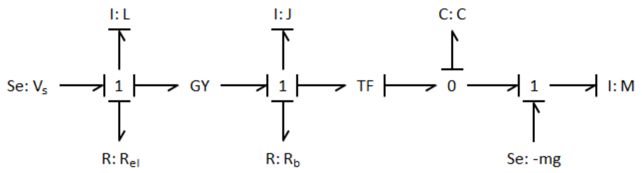

Figure 3. Bond graph model for the hoist device

which we can then put together in order for simulations.

Generalized bond graph modeling is different from bond graph modeling in two ways, absence of causality and use of symplectic gyrators [10]. A symplectic gyrator is a unit gyrator that inverts the roles of the effort and flow [6]. In generalized bond graphs, symplectic gyrators are attached to I-storage elements, which allows C and I storage elements to be treated in a unified way. The generalized bond graph model of the elastic hoist device is shown in Figure 4, where we partition the model into four components (power supply, motor, cable drum, and load).

Central to every generalized bond graph model is a Dirac structure, which is a representation of its power conserving interconnection structure [10]. This power conserving interconnection structure contains all non-single-port elements, such as transformers, gyrators, 1 junctions, and 0 junctions. The Dirac structure is commonly represented by a kernel representation [10] as seen in equation (1):

$$\mathcal{D}(x) = \{(e,f) \in T_z Z \times T_z * Z : E(x)e + F(x)f = 0\}, \quad (1)$$

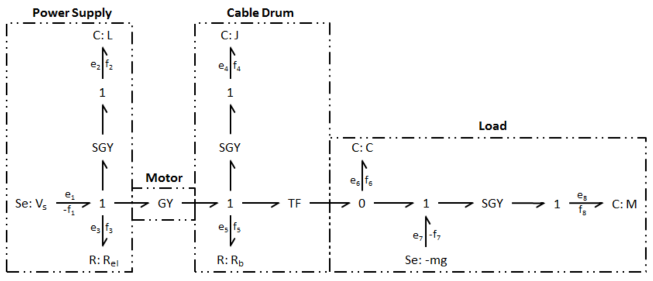

Figure 4. Generalized bond graph model for the elastic hoist device with components

where the dimensions of $F(x)$ and $E(x)$ are both $n \times n$, where n is the total number of non-power conserving elements. The $F(x)$ and $E(x)$ matrix must satisfy the following two conditions:

1. $\operatorname{rank}\left[F(x) \quad | \quad E(x)\right] = n$
2. $E(x)F^{\mathsf{T}}(x) + F(x)E^{\mathsf{T}}(x) = 0$

Looking at the power supply in Figure 4, an automated method will provide the following kernel representation shown in equation (2), with e_α and f_α as the effort and flow of the interaction port, respectively. The Dirac structure of a model can be derived from its generalized bond graph in an automated fashion, which we will describe in detail in Section 3..

$$\begin{bmatrix} -1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_\alpha \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_\alpha \end{bmatrix} = 0, \quad (2)$$

Using the kernel representation of a Dirac structure, we can also derive an input-output form. There are two square matrices in a kernel representation, and one of them must be full rank in order for an input-output form to be derived. In this paper we define it so that we want to manipulate the kernel representation so to make the $F(x)$ matrix full rank. Using equation (2) we can see that its $F(x)$ matrix is full rank, which allows for:

$$\begin{aligned} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_\alpha \end{bmatrix} &= - \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} -1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_\alpha \end{bmatrix} \\ &= \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & -1 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_\alpha \end{bmatrix}. \end{aligned} \quad (3)$$

Equation (3) is called the hybrid-input-output representation [10] of the power supply. This representation implies a causal relationship between its inputs and outputs, which contrasts with the acausal kernel representation in equation (2).

The Dirac structure is central to component-based analysis, because the composition of any number of Dirac structures is again a Dirac structure [14]. Composition of Dirac structures is defined as the power-conserving interconnections between the Dirac structures [8]. In this paper we describe each component of a system by its Dirac structure; their interactions become the composition of their Dirac structures.

3. ACAUSAL MODELING AND SIMULATION

In this section, we present a component-based method for transforming generalized bond graph components to acausal compuational models. We begin by deriving the kernel representation of each Dirac structure component from its generalized bond graph model. Using the kernel representations, we can perform acausal simulation.

The most common way of forming the kernel representation of a Dirac structure is by writing down junction equations and combining/eliminating the equations. This method is not ideal for systems with thousands of junction equations. Another way uses input-output generalized junction structures (IOGJS) to calculate the Dirac structure [10]; this method necessitates the conversion of a regular generalized bond graph model into an IOGJS model, which will introduce additional power-conserving elements and source elements.

In this subsection, we formulate a method of deriving the Dirac structure of a component by examining the structural properties of its power conserving elements. Our method sequentially combines all power-conserving elements of a component to generate its kernel representation equations. The main idea in this method is recognizing that there is a pattern to the Dirac structure of every power conserving element. A transformer or a gyrator has only one possible kernel representation due to its SISO (one bond in, one bond out) nature.

$$\begin{bmatrix} -1 & r \\ 0 & 0 \end{bmatrix} \begin{bmatrix} e_i \\ e_o \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -1 & \frac{1}{r} \end{bmatrix} \begin{bmatrix} f_i \\ f_o \end{bmatrix} = 0, \quad (4)$$

$$\begin{bmatrix} -1 & 0 \\ 0 & \frac{1}{k} \end{bmatrix} \begin{bmatrix} e_i \\ e_o \end{bmatrix} + \begin{bmatrix} 0 & k \\ -1 & 0 \end{bmatrix} \begin{bmatrix} f_i \\ f_o \end{bmatrix} = 0. \quad (5)$$

Equations (4) and (5) describe the kernel representations of a transformer and a gyrator, repectively, where r is the transformer turning ratioa and k is the gyrator ratio. Notice how the first columns of the E and F matrices are negative, this is due to the fact that the first columns corresponds to the input ports. In this paper we have chosen the convention that input columns are negative.

A zero or one junction, on the other hand, can have an infinite number of kernel representations due to its MIMO nature (multiple bonds in, multiple bonds out). However, there is a pattern to the matrices; for a zero and one junction with n bonds, their kernel representations are shown in equations (6) and (7). Note that in these two equations we only show the structural formats of their kernel representations, columns will be assigned as negative or positive depending on direction of bonds:

$$\begin{bmatrix} 1 & 1 & 0 & \cdots & \cdots & 0 \\ 1 & 0 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \ddots & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \ddots & \vdots \\ 1 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ \vdots \\ e_{n-1} \\ e_n \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_{n-1} \\ f_n \end{bmatrix} = 0, \tag{6}$$

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ \vdots \\ e_{n-1} \\ e_n \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & \cdots & \cdots & 0 \\ 1 & 1 & 0 & \cdots & \cdots & 0 \\ 1 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & 0 & \ddots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & 1 & 0 \\ 1 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_{n-1} \\ f_n \end{bmatrix} = 0. \tag{7}$$

Our algorithm takes a component's generalized bond graph model and constructs its kernel representation. First it constructs the kernel representation of every power-conserving element inside the component using equations (4) – (7). Then it combines all individual kernel representations together using equation substitution methods to form the full kernel representation of the component. Note that the algorithm only takes into account power conserving elements, because they form the Dirac structure equations. A summary of the algorithm is listed in Algorithm 1.

Algorithm 1 GBG2DS

```
-for all Components
-   for all Power-conserving elements
-       Find E and F based on pattern;

-   while Interconnections left
-       Substitute Dirac structure equations to form combined
Dirac structure;
```

Modelica is a multi-domain modeling language for component-based modeling of systems developed by the Modelica Association [2]. One of Modelica's major distinctions from other modeling languages is that equations in Modelica describes an equality rather than an assignment; therefore, there is no inherent causality in the Modelica language [2]. Its acausal nature makes Modelica an appealing language to use for simulating the kernel representation of a Dirac structure.

Having the kernel representation of components, we can simulate them acausally using s Modelica tool/software. We need to write out equations of the kernel representation of each component and the constituent equations of bond graph elements (such as source, storage, and resistive) on the power ports of each component. One of the advantages of using acausal methods is independent analysis; the kernel representation of each component can be derived without knowledge of its neighboring components or the global model. The acausal method can be automated into Modelica code, which a Modelica software can simulate. A summary of the algorithm is listed in Algorithm 2.

Algorithm 2 DS2Modelica

```
-for all Components
-   Set parameters of power ports, gyrators, and transformers;
-   Define power port variables;
-   Define interaction port variables;
-   Write out Dirac structure equations;

-   for all Power ports
-       Write out constitutive bond graph element equations;
```

4. CAUSAL MODELING AND SIMULATION

In addition to acausal computational models from the kernel representation of components, we can also derive their respective hybrid-input-output representations, which are causal computational models. In causal analysis, there can be no causal conflict, and the causality of every bond must be determined. These extra steps make deriving causal computational models more complicated than deriving acausal computation models. At the heart of the causal computation model is the hybrid-input-output representation of the Dirac structure. In this section, we propose a method based on elementary column operations of $E(x)$ and $F(x)$ in order obtain the hybrid-input-output representation.

We illustrate the elementary column operations with a simple example. Consider the following kernel representation of a system with only two power ports:

$$\begin{bmatrix} E_1 & E_2 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + \begin{bmatrix} F_1 & F_2 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = 0. \tag{8}$$

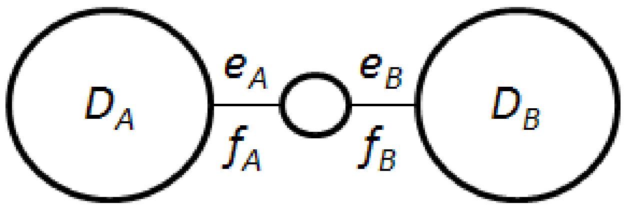

Figure 5. Simple diagram of two interacting Dirac structures [8]

Mathematically, it is equivalent to the following representation:

$$\begin{bmatrix} E_1 & F_2 \end{bmatrix} \begin{bmatrix} e_1 \\ f_2 \end{bmatrix} + \begin{bmatrix} F_1 & E_2 \end{bmatrix} \begin{bmatrix} f_1 \\ e_2 \end{bmatrix} = 0. \tag{9}$$

The difference lies in the fact that the positions of E_2 and F_2 are swapped. This elementary column operation interchanges a column in $E(x)$ with a corresponding column in $F(x)$. This technique is important in the derivation of the hybrid-input-output representation of a component.

From bond graph theory we know that source elements have fixed causality; sources of effort have effort-out-flow-in causality, while sources of flow have flow-out-effort-in causality. As a result, the Dirac structure must solve the flow of a source of effort given its effort (backwards for sources of flow). In order to do so we need to apply the elementary column operation to all source of flow power ports.

Although storage elements have preferred causality, we still want to fix their causality to integral causality, due to the issues that arises with implementing differentiators. As a result of this constraint, we do not need to apply the elementary column operation to storage power ports. The difficult part of this process lies in determining column swap for resistive and interaction power ports. Resistive power ports can be either effort-out-flow-in or effort-in-flow-out; there is no *a priori* way of determining how each resistor will behave. The main focus of this algorithm is on the behavior of interaction power ports.

Consider the two Dirac structures in Figure 5, D_A and D_B, connected through an interaction. From a structural perspective, if the interaction power port of D_A is effort-out-flow-in, the interaction power port of D_B has to be flow-out-effort-in, and vice versa. Therefore, we can declare that for two Dirac structures that share the same interaction, the causality of their interaction ports is opposite.

Similar to resistive power ports, there are two possible causality configurations for interaction power ports. In order to determine the causality of an interaction power port on a component, we must examine the component along with all neighboring components. In this section we propose an interaction port propagation assignment (IPPA) algorithm which iterates through each component sequentially and determines the causality of every interaction power port.

Our IPPA algorithm takes advantage of the indifferent causality nature of resistors. Because of its indifferent causality, a resistor can never be the source of a causal conflict, which allows for the analysis of each component with its resistive columns intentionally left out. With resistive power ports ignored for the time being, the only power ports with questionable causality are interaction power ports.

Similar to assigning causal strokes to a bond graph model, we select a component with the least number of interaction ports and most number of constraint ports (source and storage) with which to begin propgation of interaction port assignments. We analyze the ranks of the $F(x)$ matrix of the starting component and perform combinations of elementary column operations until it has full rank; additionally, do the same thing for the neighboring components. The process gets repeated until every interaction port is assigned. A summary of the algorithm is listed in Algorithm 3:

Algorithm 3 IPPA

```
-for all Components
-   Swap all source of flow columns;
-   Remove all resistive columns;

-   while There are still unassigned interactions
-      Find component with greatest interaction to constraint ratio;
-      Propagate interaction assignment;

-   Put back all resistive columns;
```

Now that all interaction ports are assigned, we must determine the causality of every resistive power port. In order to do so we analyze the rank of $F(x)$ of individual components; we then apply the elementary column operation to the appropriate resistive power ports. Note that for components with multiple resistors, there can be more than one unique solution of column swapped resistive ports, but any of the solutions will work in simulation. A summary is listed in Algorithm 4:

Algorithm 4 Kernel2Hybrid

```
-for all Components
-   Determine which resistive columns to swap;
-   Swap said resistive columns to obtain full ranked $F$;
-   $J \leftarrow -F^{-1}E$;
```

In Simulink, every block has a clearly defined input-output relationship. With each component's hybrid-input-output representation, we can derive their corresponding block diagrams and simulate them inside Simulink. Our software uses Simulink to create the block diagrams, where each component is modeled with the subsystem block. Inside each component subsystem, there is another subsystem (the Dirac structure of the component) along with its attached ports; the Dirac structure subsystem consists of a series of sums and gains, depicting the dynamics of the Dirac structure equations. Power ports are modeled as follows:

1. Source of effort: effort modeled as a step input (i.e. a constant voltage source); flow sunk to a scope
2. Source of flow: flow modeled as a step input (i.e. a constant current source); effort sunk to a scope
3. Storage port: flow goes into an integrator and a gain (inverse of the storage value) to obtain effort
4. Resistive port: effort and flow have a linear relationship, which can be represented by a gain, whose value depends on the orientation of the resistive element
5. Interaction port: effort and flow modeled as Inport and Outport

Algorithm 5 creates block diagrams from hybrid-input-output representations.

Algorithm 5 DS2BD

```
-for all Components
-   for all Power ports
-       if Non-interaction port
-           Model according to rules;
-       if Interaction port
-           Model as Simulink inports/outports;

-Connect all components;
```

5. CASE STUDY: ELASTIC HOIST DEVICE

The generalized bond graph model of the elastic hoist device is built in GME [1], which is a modeling tool developed by the Institute for Software-Integrated Systems at Vanderbilt University. Inside GME we defined a generalized bond graph meta model, which allows for a simple generalized bond graph model to be created. In addition to typical bond graph elements, we also introduce an interaction element, which models the interaction between components.

The meta model has an interpreter attached to it, written in C++, that generates structural data from the generalized bond graph model. Structural data then gets passed through Algorithm 1, written in MATLAB [12], is used to calculate the E and F matrices of every component. We chose MATLAB as an essential component in the software package due to its abilities in handling matrices.

5.1. SIMULATION OF ACAUSAL MODELS

The kernel representation of the power supply is presented in Section 2.. For the motor, we obtain the following kernel representation:

$$\begin{bmatrix} -1 & 0 \\ 0 & \frac{1}{K} \end{bmatrix} \begin{bmatrix} e_\alpha \\ e_\beta \end{bmatrix} + \begin{bmatrix} 0 & K \\ 1 & 0 \end{bmatrix} \begin{bmatrix} f_\alpha \\ f_\beta \end{bmatrix} = 0, \qquad (10)$$

for the cable drum, we obtain the following kernel representation:

$$\begin{bmatrix} -1 & 0 & 1 & \frac{D}{2} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} e_\beta \\ e_4 \\ e_5 \\ e_\gamma \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & \frac{2}{D} \end{bmatrix} \begin{bmatrix} f_\beta \\ f_4 \\ f_5 \\ f_\gamma \end{bmatrix} = 0, \quad (11)$$

for the load, we obtain the following kernel representation:

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ -1 & 0 & -1 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} e_\gamma \\ e_6 \\ e_7 \\ e_8 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} f_\gamma \\ f_6 \\ f_7 \\ f_8 \end{bmatrix} = 0, \qquad (12)$$

The system contains eight power ports, and their constituent differential equations are as follows:

1. Power port 1 (voltage source): $e_1 = V_s$
2. Power port 2 (inductor): $\frac{de_2}{dt} = \frac{f_2}{L}$
3. Power port 3 (resistor): $e_3 = Rf_3$
4. Power port 4 (moment of inertia): $\frac{de_4}{dt} = \frac{f_4}{J}$
5. Power port 5 (bearing friction): $e_5 = R_b f_5$
6. Power port 6 (elasticity): $\frac{de_6}{dt} = \frac{f_6}{C}$
7. Power port 7 (gravitational force): $e_7 = mg$
8. Power port 8 (mass of load): $\frac{de_8}{dt} = \frac{f_8}{M}$

With the kernel form for each component and the constituent differential equations of the eight power ports established, we can simulate the dynamics of the system using the Modelica language. We implemented our Modelica code in OpenModelica [3], which is an open-source Modelica modeling and simulation environment supported by the Open Source Modelica Consortium (OSMC).

For simulation we chose $V_s = 1$, $L = 2$, $R = 5$, $J = 4$, $R_b = 3$, $C = 0.001$, $mg = 100$, $M = 10$, transformer turns ratio of 2, and gyrator turns ratio of 2.5. We ran the simulation with a time of two seconds. In our results we show the four states of the system: inductor state ($x_1 = Le_2$), moment of inertia state ($x_2 = Je_4$), elasticity state ($x_3 = Ce_6$), and mass of load state ($x_4 = Me_8$). Our simulation plots (Figure 6) the four states of the system (as shown below), running 2 seconds.

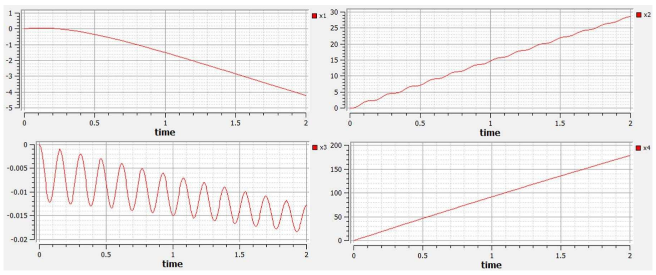
Figure 6. Simulation of the 4 states using Modelica

5.2. SIMULATION OF CAUSAL MODELS

The causal method requires a causality assignment to each interaction power port. First, all resistive columns of the kernel representation of each component are hidden, which yields the following four expressions.

Power Supply

$$\begin{bmatrix}-1&0&1\\0&1&0\\0&0&0\\0&0&0\end{bmatrix}\begin{bmatrix}e_1\\e_2\\e_\alpha\end{bmatrix}+\begin{bmatrix}0&1&0\\1&0&0\\1&0&0\\1&0&1\end{bmatrix}\begin{bmatrix}f_1\\f_2\\f_\alpha\end{bmatrix}$$

Motor

$$\begin{bmatrix}-1&0\\0&\frac{1}{K}\end{bmatrix}\begin{bmatrix}e_\alpha\\e_\beta\end{bmatrix}+\begin{bmatrix}0&K\\1&0\end{bmatrix}\begin{bmatrix}f_\alpha\\f_\beta\end{bmatrix}$$

Cable Drum

$$\begin{bmatrix}-1&0&\frac{D}{2}\\0&1&0\\0&0&0\\0&0&0\end{bmatrix}\begin{bmatrix}e_\beta\\e_4\\e_\gamma\end{bmatrix}+\begin{bmatrix}0&1&0\\1&0&0\\1&0&0\\1&0&\frac{2}{D}\end{bmatrix}\begin{bmatrix}f_\beta\\f_4\\f_\gamma\end{bmatrix}$$

Load

$$\begin{bmatrix}0&0&0&1\\-1&0&-1&0\\-1&1&0&0\\0&0&0&0\end{bmatrix}\begin{bmatrix}e_\gamma\\e_6\\e_7\\e_8\end{bmatrix}+\begin{bmatrix}0&0&1&0\\0&0&0&1\\0&0&0&0\\1&1&-1&0\end{bmatrix}\begin{bmatrix}f_\gamma\\f_6\\f_7\\f_8\end{bmatrix}$$

We can see that the power supply contains 2 constraints, the motor contains no constraints, the cable drum contains 1 constraint, and the load constains 3 constraints, thus going by the method derived in Section 4., we will begin analysis with the load. The $F(x)$ matrix of the load has a rank of 3 but a dimension of 4, which means that the elementary column operation needs to be applied to one column; the only option is the interaction port (E_γ, F_γ), which results in the following expression:

$$\begin{bmatrix}0&0&0&1\\0&0&-1&0\\0&1&0&0\\1&0&0&0\end{bmatrix}\begin{bmatrix}f_\gamma\\e_6\\e_7\\e_8\end{bmatrix}+\begin{bmatrix}0&0&1&0\\-1&0&0&1\\-1&0&0&0\\0&1&-1&0\end{bmatrix}\begin{bmatrix}e_\gamma\\f_6\\f_7\\f_8\end{bmatrix} \tag{13}$$

We can then propagate to the cable drum, which is attached via interaction port (E_γ, F_γ). The interaction port (E_γ, F_γ) in the cable drum then becomes a fixed constraint, which allows us to look at interaction port (E_β, F_β). We see that the $F(x)$ matrix of the cable drum has full rank, so interaction port (E_β, F_β) is fine the way it is:

$$\begin{bmatrix}-1&0&\frac{D}{2}\\0&1&0\\0&0&0\\0&0&0\end{bmatrix}\begin{bmatrix}e_\beta\\e_4\\e_\gamma\end{bmatrix}+\begin{bmatrix}0&1&0\\1&0&0\\1&0&0\\1&0&\frac{2}{D}\end{bmatrix}\begin{bmatrix}f_\beta\\f_4\\f_\gamma\end{bmatrix} \tag{14}$$

The interaction port (E_β, F_β) in the motor, on the other hand, needs the elementary column operation, which leads to the $F(x)$ matrix of the motor to no longer be full rank. We then apply the elementray column operation to its interaction port (E_α, F_α):

$$\begin{bmatrix}0&K\\1&0\end{bmatrix}\begin{bmatrix}f_\alpha\\f_\beta\end{bmatrix}+\begin{bmatrix}-1&0\\0&\frac{1}{K}\end{bmatrix}\begin{bmatrix}e_\alpha\\e_\beta\end{bmatrix} \tag{15}$$

Finally propagation reaches the final Dirac structure, power supply, which requires no elementary column operation:

$$\begin{bmatrix}-1&0&1\\0&1&0\\0&0&0\\0&0&0\end{bmatrix}\begin{bmatrix}e_1\\e_2\\e_\alpha\end{bmatrix}+\begin{bmatrix}0&1&0\\1&0&0\\1&0&0\\1&0&1\end{bmatrix}\begin{bmatrix}f_1\\f_2\\f_\alpha\end{bmatrix} \tag{16}$$

With the causality of all interaction ports determined, we can unhide all resistive columns, transforming equations 2 through 12 into:

$$\begin{bmatrix}-1&0&1&1\\0&1&0&0\\0&0&0&0\\0&0&0&0\end{bmatrix}\begin{bmatrix}e_1\\e_2\\e_3\\e_\alpha\end{bmatrix}+\begin{bmatrix}0&1&0&0\\1&0&0&0\\1&0&1&0\\1&0&0&1\end{bmatrix}\begin{bmatrix}f_1\\f_2\\f_3\\f_\alpha\end{bmatrix}=0, \tag{17}$$

$$\begin{bmatrix}0&K\\1&0\end{bmatrix}\begin{bmatrix}f_\alpha\\f_\beta\end{bmatrix}+\begin{bmatrix}-1&0\\0&\frac{1}{K}\end{bmatrix}\begin{bmatrix}e_\alpha\\e_\beta\end{bmatrix}=0, \tag{18}$$

$$\begin{bmatrix}-1&0&1&\frac{D}{2}\\0&1&0&0\\0&0&0&0\\0&0&0&0\end{bmatrix}\begin{bmatrix}e_\beta\\e_4\\e_5\\e_\gamma\end{bmatrix}+\begin{bmatrix}0&1&0&0\\1&0&0&0\\1&0&1&0\\1&0&0&\frac{2}{D}\end{bmatrix}\begin{bmatrix}f_\beta\\f_4\\f_5\\f_\gamma\end{bmatrix}=0, \tag{19}$$

$$\begin{bmatrix}0&0&0&1\\0&0&-1&0\\0&1&0&0\\1&0&0&0\end{bmatrix}\begin{bmatrix}f_\gamma\\e_6\\e_7\\e_8\end{bmatrix}+\begin{bmatrix}0&0&1&0\\-1&0&0&1\\-1&0&0&0\\0&1&-1&0\end{bmatrix}\begin{bmatrix}e_\gamma\\f_6\\f_7\\f_8\end{bmatrix}=0, \tag{20}$$

We can see that all four $F(x)$ matrices are full rank, which means elementary column operations are unnecessary for any

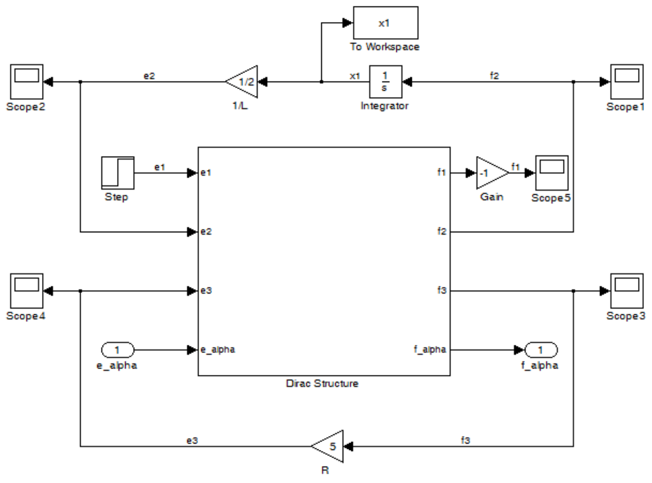

Figure 7. Block diagram for the power supply

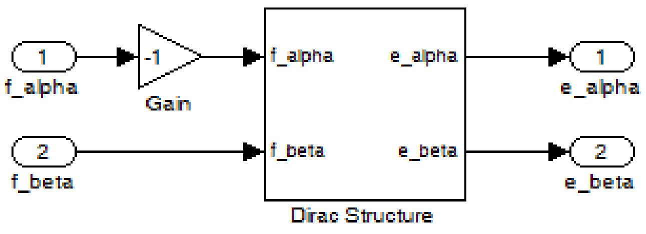

Figure 8. Block diagram for the motor

of the resistive power ports. Therefore, we can formulate the hybrid-input-output representations of each component as follows (power supply's hybrid input output representation was already formulated in equation (3)):

$$\begin{bmatrix} e_\alpha \\ e_\beta \end{bmatrix} = \begin{bmatrix} 0 & K \\ -K & 0 \end{bmatrix} \begin{bmatrix} f_\alpha \\ f_\beta \end{bmatrix}. \tag{21}$$

$$\begin{bmatrix} f_\beta \\ f_4 \\ f_5 \\ f_\gamma \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & -1 & -\frac{D}{2} \\ 0 & 1 & 0 & 0 \\ 0 & \frac{D}{2} & 0 & 0 \end{bmatrix} \begin{bmatrix} e_\beta \\ e_4 \\ e_5 \\ e_\gamma \end{bmatrix}. \tag{22}$$

$$\begin{bmatrix} e_\gamma \\ f_6 \\ f_7 \\ f_8 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} f_\gamma \\ e_6 \\ e_7 \\ e_8 \end{bmatrix}. \tag{23}$$

The consituent differential equations of each power port are implemented as described at the end of Section 4.. Each of these hybrid-input output forms, along with constitutive differential equations of power ports, can then be converted into block diagrams, Figures 7 through 10. The combined form for the whole system is shown in Figure 11.

Simulation results of the states of the system are shown in Figure 12, using the same simulation parameters as the Modelica simulations.

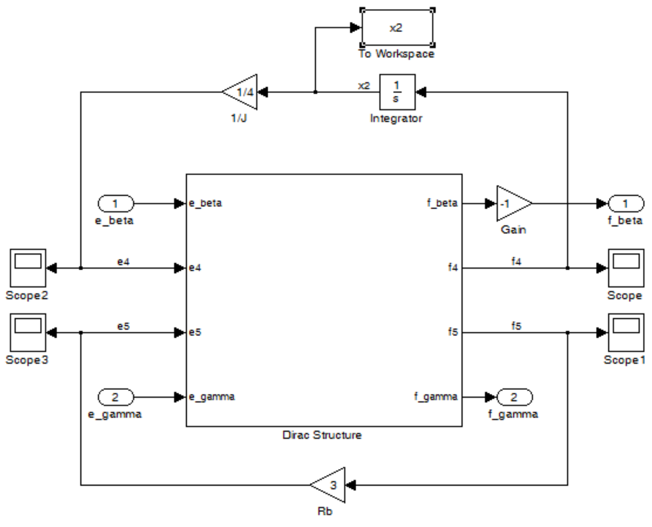

Figure 9. Block diagram for the cable drum

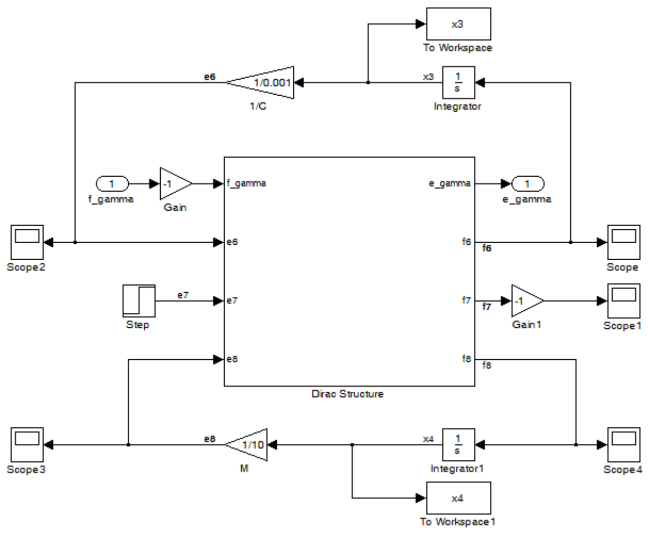

Figure 10. Block diagram for the hoist device load

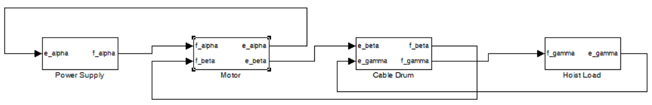

Figure 11. Block diagram for the elastic hoist device

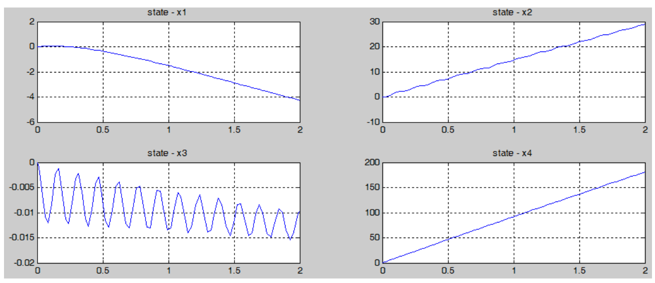

Figure 12. Simulation of the 4 states using Simulink

6. DISCUSSION AND FUTURE WORK

The most important point of this paper is the distinction between acausal and causal component-based simulation of systems. In Section 3. we have seen that acausal simulation is generated from the kernel representations of each component, which are derived independently of each other. Causal simulation of a component, on the other hand, requires knowledge of its interacting neighbor components in order to determine the causality of each interaction port. This difference makes causal computational models of components more difficult to generate as compared to acausal computational models.

However, the causal method does have one significant advantage over the acausal method: identification of causality conflicts. Any conflicts in causality will result in an under-ranked $F(x)$ matrix, which will result in the inability to derive the hybrid-input-output representation. The acausal method, on the other hand, has no way of identifying causal conflicts; kernel representations of every component will not give any indication of causal conflicts.

Another major difference between the causal and acausal method occurs during simulation time. The acausal method (Modelica) uses a Differential Algebraic System Solver (DASSL) engine which symbolically manipulates the equations in order to solve for all the variables. The causal method uses an Ordinary Differential Equation (ODE) solver, which uses numerical integration techniques to solve for all variables. DAE's are generalizations of ODE's, and includes fundamental mathematical properties that makes them computationally more expensive to solve compared to ODE's.

In simulations such as this, algebraic loops can become a concern. In the method presented in this paper, we simulate systems and components using the hybrid-input-output representation of the Dirac structure; one of the key properties of that representation is that the $J(x)$ matrix is skew symmetric [10], which means that the entire diagonal line is zeros. We also constrained the modeling to only allow integral causality for storage elements. For these reasons our simulation technique does not generate any algebraic loops.

Our method does pose a few limitations that must be addressed. In this paper, we constrained the sources elements to be constant sources modeled by step inputs; this can be easily extended to include all kinds of non-constant sources. This limitation can be addressed in the metamodeling step where we add additional attributes to source elements and set those attributes accordingly. The purpose of our paper was to show methods in modeling interacting components, so we did not consider modifying the metamodel yet.

The methods introduced in this paper applies to linear systems only; future work can extend these methods to nonlinear systems. We assumed that all resistive elements are linear, which leads to its indifferent causality; we understand that in a physical system sometime there can be nonlinear resistive elements for which an inversion of its constitutive equations is difficult or even impossible. A possible way to address this problem is to define various types of resistive elements (linear vs. nonlinear) and setting causality constraints, or lack of causality constraints, appropriately.

Other future work can include comparing the total computational cost of the causal method and acausal method. The derivation of computational models for the causal method will take longer because of the extra steps, but its ODE solver will be less computationally expensive than the DAE solver of the acausal method. But due to the size of the system in this paper, there is really no obvious indication of which method is faster.

More importantly, we can analyze certain properties of a system in a component-based way. For example, controllability of a system is determined by the controllability indicators of its bond graph model [9]; however, the controllability indicator comes from the global model. Future work can be done to determine the controllabilty, and other system properties such as stability and verification, of a system by examining its components and how they interact. Additionally, we can create a library in Modelica with components that use Dirac structures.

ACKNOWLEDGEMENT

This work is supported in part by the National Science Foundation (CNS-1035655, CCF-0820088).

REFERENCES

[1] Generic modeling environment. `http://repo.isis.vanderbilt.edu/downloads?tool=GME`.

[2] Modelica. `http://www.Modelica.org/`.

[3] Openmodelica. `http://www.openmodelica.org/`.

[4] W. Borutzky. Bond graph modelling and simulation of multidisciplinary systems an introduction. *Simulation Modelling Practice and Theory*, 17(1):3 – 21, 2009.

[5] P. C. Breedveld. Modeling and simulation of dynamic systems using bond graphs. In *Control Systems, Robotics and Automation - Modeling and System Identification I*, volume 18-4-1 of *Encyclopedia of Life Support Systems*, pages 128–173. EOLSS Publishers Co. Ltd., Oxford, UK, 2008.

[6] P.C. Breedveld. Multibond graph elements in physical systems theory. *Journal of the Franklin Institute*, 319(1-2):1–36, 1985.

[7] Jan F. Broenink. Introduction to physical systems modeling with bond graphs. In *in the SiE whitebook on Simulation Methodologies*, pages 18–27, 1999.

[8] J. Cervera, A. J. van der Schaft, and A. Baños. Interconnection of port-hamiltonian systems and composition of dirac structures. *Automatica*, 43:214–217, February 2007.

[9] Ballance D.J. Gawthrop P.J. and Dauphin-Tanguy G. Controllability indicators from bond graphs.

[10] G. Golo, P.C. Breedveld, B.M. Maschke, and A.J. Schaft van der. Geometric formulation of generalized bond graph models - part i: Generalized junction structures, 2000. Imported from MEMORANDA.

[11] Margolis D.L. Rosenberg R.C. Karnopp, D.C. *System Dynamics: Modeling and Simulation of Mechatronic Systems*. John Wiley & Sons, 2000.

[12] MATLAB. *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts, 2010.

[13] Benjamin Podgursky, Gautam Biswas, and Xenofon Koutsoukos. Efficient tracking of behavior in complex hybrid systems via hybrid bond graphs. In *Annual Conference of the Prognostics and Health Management Society 2010*, Portland, OR, US, 10/2010 2010.

[14] S. Stramigioli V. Duindam, A. Macchelli and H. Bruyninckx. *Modeling and Control of Complex Physical Systems: The Port-Hamiltonian Approach*. Springer, 2009.

BIOGRAPHY

Siyuan Dai is a graduate student in the Department of Electrical Engineering and Computer Science at Vanderbilt University studying under Dr. Xenofon Koutsoukos. He is also a graduate research assistant in the Institute for Software-Integrated Systems (ISIS). He received his B.S. in Electrical Engineering from the University of Notre Dame in May 2010. His research work is in the area of cyber-physical systems emphasizing component-based modeling and analysis.

Xenofon Koutsoukos is an Associate Professor in the Department of Electrical Engineering and Computer Science at Vanderbilt University. He is also a Senior Research Scientist in the Institute for Software-Integrated Systems (ISIS). Before joining Vanderbilt, Dr. Koutsoukos was a Member of Research Staff in the Xerox Palo Alto Research Center (PARC) (2000-2002), working in the Embedded Collaborative Computing Area. He received his PhD in Electrical Engineering from the University of Notre Dame in 2000. His research work is in the area of cyber-physical systems with emphasis on formal methods, distributed algorithms, diagnosis and fault tolerance, and adaptive resource management. He has published numerous journal and conference papers and he is coinventor of four US patents. He was the recipient of the NSF Career Award in 2004 and the Excellence in Teaching Award in 2009 from the Vanderbilt University School of Engineering.