



3 1293 00898 2104

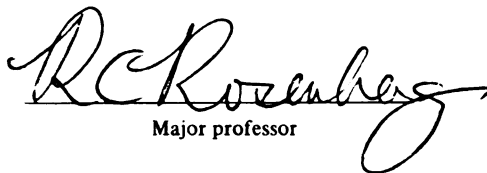
This is to certify that the  
thesis entitled  
User-Guided Reduction of Certain Storage Fields  
in Multiport Systems

presented by

John Michael McCalla

has been accepted towards fulfillment  
of the requirements for

M. S. degree in M. E.

  
Major professor

Date May 16, 1991

**LIBRARY**  
**Michigan State**  
**University**

**PLACE IN RETURN BOX** to remove this checkout from your record.  
**TO AVOID FINES** return on or before date due.

DATE DUE	DATE DUE	DATE DUE
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

**MSU is An Affirmative Action/Equal Opportunity Institution**

c:\circ\deldue.pn3-p.1

**USER-GUIDED REDUCTION OF CERTAIN STORAGE FIELDS  
IN MULTI-PORT SYSTEMS**

**By**

**John Michael McCalla**

**A THESIS**

**Submitted to  
Michigan State University  
in partial fulfillment of the requirements  
for the degree of**

**MASTER OF SCIENCE**

**Department of Mechanical Engineering**

**1991**

656-440-

## **ABSTRACT**

### **USER-GUIDED REDUCTION OF CERTAIN STORAGE FIELDS IN MULTI-PORT SYSTEMS**

**By**

**John Michael McCalla**

A methodology is developed which replaces linear, dependent storage elements in a bond graph with an equivalent element. This substitution, which utilizes a causal path variable trace, restores integral causality to the bond graph. The procedure reduces the number of energy variables, thereby simplifying the system model, and facilitates the formulation of explicit state equations. This algorithm is implemented in the ENPORT software and applied to several examples. Simulation results demonstrate a new option for the engineer in designing certain types of systems.

**for Mom and Dad**

## **ACKNOWLEDGEMENTS**

**Thanks to Dr. Rosenberg, whose guidance kept me inspired.**

## **TABLE OF CONTENTS**

<b>LIST OF FIGURES</b>	<b>vi</b>
<b>LIST OF SYMBOLS</b>	<b>vii</b>
<b>Chapter 1. Introduction</b>	<b>1</b>
<b>Chapter 2. Identification of Derivative Causality</b>	<b>3</b>
2.1 Identifying Derivative Causality	3
2.2 Solution Options for Derivative Causality	6
<b>Chapter 3. The Storage Field Reduction Technique</b>	<b>9</b>
3.1 Identifying a Storage Field	9
3.2 The Reduction Technique	11
3.3 Example Problems	16
<b>Chapter 4. Algorithm Implementation</b>	<b>19</b>
4.1 Design of Implementation	19
4.2 Examples of Use	21
4.3 Description of Software	26
<b>Chapter 5. Conclusions</b>	<b>28</b>
5.1 Summary	28
5.2 Recommendations for Further Development	29
<b>APPENDIX A Dynamic Augmentation of Roller Coaster Model</b>	<b>31</b>
<b>APPENDIX B Dynamic Augmentation of Robot Arm Model</b>	<b>33</b>
<b>APPENDIX C The FORTRAN Algorithm</b>	<b>36</b>
<b>LIST OF REFERENCES</b>	<b>70</b>



## LIST OF FIGURES

Figure 1.	Derivative and Integral Forms of C and I elements	3
Figure 2.	Bond Graph of Roller Coaster Model	4
Figure 3.	C-Subsystem Required for Dynamic Augmentation	6
Figure 4.	Augmented Bond Graph Model	7
Figure 5.	Inertial Storage Field	11
Figure 6.	Bond Graph Model with Derivative Causality	16
Figure 7.	Reduced Bond Graph Model	17
Figure 8.	Bond Graph Model with Derivative Causality	18
Figure 9.	Flow Chart of Methodology	20
Figure 10.	Lever Arm System	21
Figure 11.	Bond Graph Model of Lever Arm System	22
Figure 12.	Reduced Form of Bond Graph Model	24
Figure 13.	Bond Graph of Field with GY Element	24
Figure 14.	Reduced Form of Bond Graph	25
Figure 15.	Flow Chart of Subroutines	27
Figure A-1.	Roller Coaster Model	31
Figure B-1.	Robot Arm in Vertical Plane	34

## LIST OF FIGURES, CONTINUED

<b>Figure B-2. Bond Graph Model of Robot Arm</b>	<b>35</b>
<b>Figure B-3. Augmented Bond Graph Model of Robot Arm</b>	<b>36</b>

## LIST OF SYMBOLS

<b>C:</b>	<b>compliance</b>
<b>d/dt:</b>	<b>time derivative</b>
<b>e:</b>	<b>effort (force)</b>
<b>f:</b>	<b>flow (velocity)</b>
<b>ge:</b>	<b>effort gain</b>
<b>gf:</b>	<b>flow gain</b>
<b>I:</b>	<b>inertia coefficient</b>
<b>m:</b>	<b>mass</b>
<b>p:</b>	<b>momentum</b>
<b>q:</b>	<b>displacement</b>
<b>t:</b>	<b>time</b>

## Chapter 1. Introduction

One purpose of constructing a bond graph is to obtain a set of differential equations that describe the motion of a physical system. Through the use of power bonds and elements that represent various pieces of a physical system, an engineer can construct a bond graph model that closely represents an actual system. State equations for the system can be generated by using a straightforward procedure that follows the details of the bond graph.

This methodology produces explicit state equations,  $\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{t})$ , when the bond graph model contains only integral causality. Implicit,  $\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{t})$ , state equations arise when the model contains derivative causality. Causality provides this information through its inherent representation of the constitutive relationships for each energy variable. These energy variables are identified through energy storing elements inertia and capacitance. Thus, even if the model contains only one energy variable with derivative causality, the entire system of state equations will be implicit and the solution of the equations will require a robust, implicit integrator. For a discussion of causality in bond graphs see, e.g., Rosenberg and Karnopp. [1]

A methodology has been developed in which a storage field reduction takes place. The technique entails following a causal path that leads to the source of the derivative causality for the dependent

energy variable. This methodology reduces the number of energy variables by combining the effects of the dependent variable with an independent variable. Thus the technique restores integral causality to the bond graph and allows an explicit integrator to solve the system of equations. This reduction decreases the complexity of the system state equations by providing only those energy variables that are state variables.

This project implemented the storage field reduction technique as an algorithm in the dynamic software simulation package ENPORT, presented by ROSENCODE Associates in [2]. The user of this package now has the option to allow the program to make an attempt at solving any derivative causality problem. The algorithm, which will be presented in Chapter 3, determines if a particular model can be corrected using this storage field reduction technique. Once deemed eligible, the bond graph is adjusted in a user-interactive manner. The experienced system modeler will find this option beneficial due to the luxury of rapid equivalent coefficient computation by the algorithm.

The details of the algorithm from initial development stages to the final process, its implementation in ENPORT, and several examples are presented in Chapters 2, 3, and 4. Finally, Chapter 5 summarizes the methodology and gives a set of recommendations for further development.

## Chapter 2. Identification of Derivative Causality

### 2.1 Identifying Derivative Causality

Derivative causality presents dependent energy storing variables to the engineer attempting to write a system of state equations. The system equations cannot be reduced to explicit state-space form, in general, when derivative causality is present in the model. Identifying derivative causality is thereby the initial step in forming explicit state equations. This identification can take place after causality has been placed on the bond graph model according to the Sequential Causality Assignment Procedure (SCAP). [1] The number of energy storing variables can be found at this point by noting the number of energy storing elements in the model, assuming they are all 1-port. The dependent energy variables are evidenced by the causal stroke on the bond connecting the C or I element to the model. Figure 1 shows the derivative and integral forms for both the inertial and the capacitance elements.

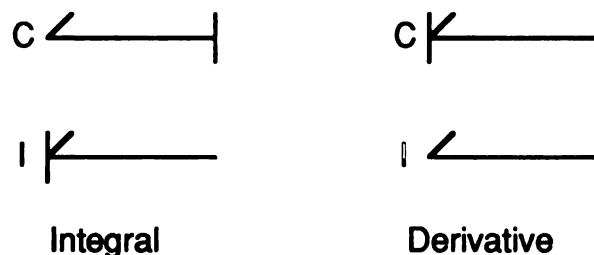


Figure 1. Derivative and Integral Forms of C and I elements

An example of a system model with derivative causality is shown in Figure 2, which is a bond graph model of a roller coaster originally

studied by Emery. [3] The physical system model is shown in Figure A-1, Appendix A.

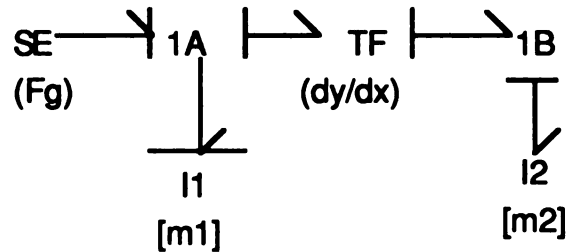


Figure 2. Bond Graph of Roller Coaster Model

Note the derivative causality resulting on the I2 element. The formulation of state equations using this bond graph model would produce a system of nonlinear, implicit equations. The general form of the state equations arising from both integral and derivative causality will be detailed in the remaining portions of this section, where we restrict our attention to linear elements as noted.

The constitutive law for a linear capacitance element in a bond graph always relates effort (e.g., force) to displacement (e.g., deflection). Displacement is the integral of flow (e.g., velocity). The constitutive equation for a linear capacitance element with integral causality is as follows:

$$e = \frac{q}{c} = \frac{\int^t f dt}{c} \quad (1)$$

studied by Emery. [3] The physical system model is shown in Figure A-1, Appendix A.

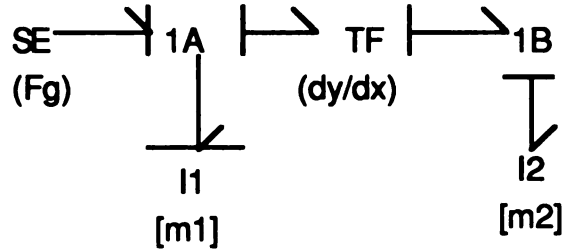


Figure 2. Bond Graph of Roller Coaster Model

Note the derivative causality resulting on the I2 element. The formulation of state equations using this bond graph model would produce a system of nonlinear, implicit equations. The general form of the state equations arising from both integral and derivative causality will be detailed in the remaining portions of this section, where we restrict our attention to linear elements as noted.

The constitutive law for a linear capacitance element in a bond graph always relates effort (e.g., force) to displacement (e.g., deflection). Displacement is the integral of flow (e.g., velocity). The constitutive equation for a linear capacitance element with integral causality is as follows:

$$e = \frac{q}{C} = \frac{\int f dt}{C} \quad (1)$$



However, the constitutive equation for a C element with derivative causality is quite different, namely:

$$f = \frac{d}{dt}q = \frac{d}{dt}Ce = C\frac{de}{dt} \quad (2)$$

Note that these two sets of equations are simply two ways of expressing the same constitutive law. Each causality indicates how to use the relation and whether integration or differentiation is implied.

The same ideas apply to the linear inertial element. The inertial element relates flow to momentum, and momentum is the time integral of effort. The constitutive equation for a linear inertial element with integral causality is as follows:

$$f = \frac{p}{I} = \frac{\int e dt}{I} \quad (3)$$

The constitutive equation for an inertial element with derivative causality is shown below:

$$e = \frac{d}{dt}p = \frac{d}{dt}If = I\frac{df}{dt} \quad (4)$$

These two sets of expressions represent the same constitutive law in different forms.

## 2.2 Solution Options for Derivative Causality

When derivative causality arises in a model to be simulated, three solution options exist:

1. use robust, implicit integrator
2. use Dynamic Augmentation
3. use Storage Field Reduction

The first option entails finding a robust, implicit integrator which would provide a means for numerically solving the implicit system of differential equations. This option is an issue involving numerical integration methodology and software implementation. It will not be discussed in further detail in this work.

Dynamic augmentation of a bond graph model involves the introduction of additional energy variables to facilitate dynamic independence of all the energy variables. For some examples in mechanical systems see [5], [6], and [7]. This procedure often requires the insertion of capacitance elements into the bond graph model. These insertions take the form shown in Figure 3.

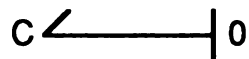


Figure 3. C-Subsystem Required for Dynamic Augmentation

The insertion of these additional elements, and the state variables they represent, takes place at strategic points in the model. The capacitance elements represent stiff springs in mechanical systems and are inserted at connection points in the model. Thus, the

coefficients for these elements are large,  $k = 1/C$ , in order to represent actual local material stiffness. The ideal location for insertion of the element remains a model specific issue.

The dynamic augmentation methodology provides explicit state equations without eliminating any of the energy variables which may be of interest. However, the computational problems associated with dynamic augmentation are significant. A variable step integrator capable of dealing with stiff systems is required for numerical solution. The inserted variables represent the error associated with this modelling technique and provide insight into error control during simulation.

Dynamic augmentation can be applied to the roller coaster bond graph model shown in Figure 2. The augmented model is shown in Figure 4.

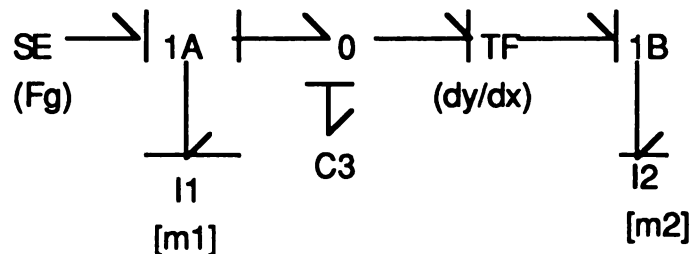


Figure 4. Augmented Bond Graph Model

Note the elimination of derivative causality with the insertion of the stiff spring, C3. Further discussion of this model may be found in Appendix A. An additional example of the dynamic augmentation

procedure can be found in Appendix B, which details the development of a robot arm in the vertical plane.

The storage field reduction of a bond graph is the third option for dealing with derivative causality. The application of causality, the crucial step in the construction of a bond graph, leaves a causal path. This path is used to locate the source of derivative causality and to eliminate it from the bond graph. This methodology provides explicit state equations by reducing the effects of a dependent energy variable into the parameter of an independent energy variable. Nonlinear and linear storage fields are both conceivably eligible for this reduction. The remaining chapters of this thesis will develop an algorithm for linear storage fields whose derivative causality is a direct result of the application of integral causality on a one port inertial or capacitance element during SCAP.

## **Chapter 3. The Storage Field Reduction Technique**

### **3.1 Identifying a Storage Field**

The causal path provides a means for the elimination of derivative causality in a bond graph model. During the second phase of SCAP integral causality may be assigned to either a capacitance or inertial element. This random application may result in the assignment of derivative causality to another I or C element during the extend portion of this phase. This particular problem can be corrected by following the path created during the causal extension process.

The identification of a storage field is the initial step in the reduction technique. Once derivative causality has been identified on a linear, 1-port inertial or capacitance element, the causal path trace can begin. Consider a derivative inertial element and its potential causal path. The inertial causal trace follows a flow path. If the flow path enters a zero junction with more than two ports, the exiting flow path is not unique and the trace cannot continue. The dependent energy storing variable is a function of more than one energy and/or source variable if this split path situation occurs.

The elements through which the causal path can travel include any number of one junctions, zero junctions, and linear, constant coefficient transformers. The inertial causal path trace leads to either an independent energy storing variable or to a source variable. If the path leads to a linear, independent, 1-port inertial element,

the storage field has been identified. However, if the causal path leads to a source variable the technique terminates because the dependent energy variable is not solely a function of an independent energy variable and cannot be reduced by our algorithm. Once the field has been identified, the reduction can commence.

Similarly, if a capacitance derivative element exists in the model the causal trace follows an effort path. If the effort causal path enters a one junction with more than two ports this trace terminates due to the non-unique exiting path. The dependent energy variable is a function of more than one energy and/or source variable and cannot be reduced using this technique when this split path occurs. The elements eligible to be on the causal path are the same as for the inertial element's trace. If the causal path leads to a linear, independent, 1-port capacitance element, the storage field has been identified and reduction can commence.

The reduction of an inertial element into a capacitance element, or vice versa, can occur only if the causal path between these two elements contains an odd number of linear, constant coefficient gyrators. The unique relationships of the gyrator make this combination possible by reversing the causal path trace variable ( $e$  or  $f$ ) through the element and allowing for the combination of mass and stiffness parameters.

### 3.2 The Reduction Technique

The storage field reduction derives from consideration of the system equations. The coefficients of the elements along the causal path will represent these equations in a reduced form. The development of the relations will utilize inertia elements for the energy variables as an illustration. Equivalent relationships for the capacitance elements will be discussed after the initial development. Consider the details shown in Figure 5.

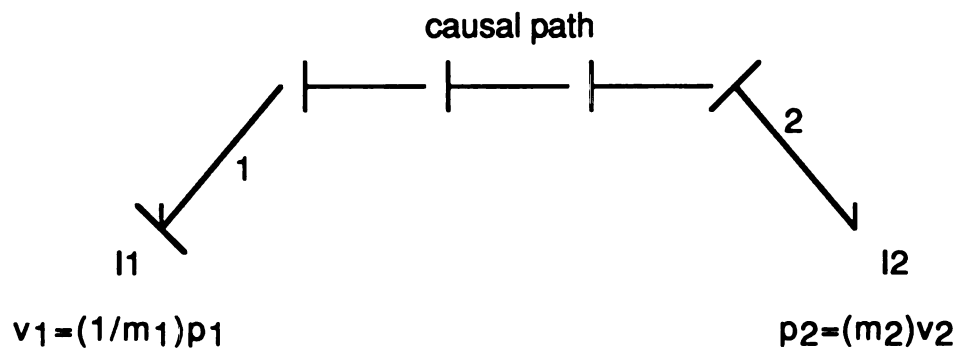


Figure 5. Inertial Storage Field

Following the causal path entails traversing through the different types of elements mentioned in section 3.1. The coefficients of these elements are recorded for this procedure in order for the reduction to account for all the system relationships. These coefficients are represented on either an effort gain,  $ge$ , or a flow gain,  $gf$ . Thus, by noting these coefficients we can reduce the dependent energy variable's effects into the independent energy variable. For the example in Figure 5, the total expression for  $e_1$  yields equation (5) on the following page.

$$e_1 = e_{12} + e_{1r} \quad (5)$$

The effort,  $e_{12}$ , is the contribution due to I2 and  $e_{1r}$  is the contribution of the rest of the system. We can represent  $e_{12}$  as shown in equation (6).

$$e_{12} = (g_e) \dot{p}_2 \quad (6)$$

The effort gain,  $g_e$ , represents the coefficients obtained from the effort path from I1 to I2. The integration of equation (6) yields equation (6a).

$$p_{12} = (g_e)p_2 \quad (6a)$$

Substituting the constitutive relationship for I2, equation (7) into equation (6a) we find (8).

$$p_2 = (m_2)f_2 \quad (7)$$

$$p_{12} = (g_e m_2)f_2 \quad (8)$$

Equation (9) represents the principle upon which the identification of unique causal path is based. Notice that the dependent energy variable,  $f_2$ , is a function of only one independent energy variable,  $f_1$ . The causal flow path does not branch at any point in the trace. The flow gain,  $g_f$ , represents the coefficients obtained on the flow path from I2 to I1.

$$f_2 = (g_f)f_1 \quad (9)$$

Substitution of equation (9) into equation (8) yields (10).



$$p_{12} = (g_e m_2 g_f) f_2 \quad (10)$$

Equations (11-13) detail the integration and substitution steps necessary for the computation of the equivalent parameter. This computation requires a choice of element for reduction, and for the sake of example I2 will be reduced. The integration of equation (5), disregarding the effort associated with the rest of the model because it plays no role in the reduction process, yields (11).

$$p_1 = \int (e_{1r} + e_{12}) dt = \int e_{1r} dt + p_{12} \quad (11)$$

Substitution of equation (10) into equation (11) yields (12).

$$p_1 = \int e_{1r} dt + (g_e m_2 g_f) f_1 \quad (12)$$

Substituting the constitutive relationship for I1 into equation (12) yields (13).

$$m_1 \dot{f}_1 - (g_e m_2 g_f) \dot{f}_1 = \int e_{1r} dt \quad (13)$$

Rearranging equation (13) yields the equivalent relationship (14).

$$(m_1 - g_e m_2 g_f) \dot{f}_1 = \int e_{1r} dt \quad (14)$$

Since the effort and flow gains can differ by a sign at most, the recording of one of these values is sufficient for storage field reduction. If the signs of the gains are opposite, the reduction equations will introduce a negative sign to ensure that the resulting equivalent coefficient is always positive, as in (14). Thus, the equivalent inertial parameter is represented in (15).

$$m_{1eq} \dot{f}_1 = \int e_{1r} dt \quad (15)$$

The relationships for the causal path between capacitance elements are similar. In this case, the effort relationship comparable to (9) is the defining equation. If the dependent capacitance variable is a function of only one other independent capacitance energy variable, the reduction can take place. This reduction will yield an equivalent stiffness parameter which can be inserted into the selected capacitance energy variable.

The relationships for the causal path between an inertial element and a capacitance element are similar. The odd number of gyrator elements required for this combination of energy variables ensures that the equivalent coefficient will have the correct form. Since either energy variable may remain in the model, the element reduction choice will determine the form of the equivalent parameter, inertial or capacitance.

The effort and flow gains warrant a brief explanation. When the causal path travels through a power conserving zero or one junction, the power directions must be examined. Through a one junction, the flow gain will have an identity relationship, while the effort gain may have a sign change. Similarly, through a zero junction the flow gain may have a sign change, while the effort gain will have an identity relationship. Thus, the source of possible negative signs in the equivalence calculation has been identified. Similarly, if a linear, constant coefficient transformer is entered on the path, the

coefficient recorded is the transformer modulus. When the path enters a linear, constant coefficient gyrator the recorded coefficient is the gyrator modulus.

The storage field reduction algorithm, shown in the following sequence, presents the ideas detailed in a systematic manner.

1. Allow causality to be assigned by SCAP. General non-linear models are permissible at this point.
- 2 Identify the existence of derivative causality. Allow simulation software to formulate implicit state equations for models with derivative causality.
3. Trace the specific structure of each storage field, with the result being complete (causal path is unique) or incomplete (causal path is not unique).
4. For complete, linear storage fields:  
calculate the causal path gains.
5. Select the location for the equivalent node.
6. Express the constitutive equation in terms of a stiffness or mass parameter.
7. Form the modified model with a single storage element and proper equivalence parameter.
8. Eliminate unnecessary elements from the model.

The elimination of unnecessary elements in the reduced bond graph is a critical final step in the process. This elimination technique uses the causal path to determine if there are elements in the bond graph that serve no purpose. These elements include: a one junction, zero junction, transformer or gyrator with only one bond. If found, the process eliminates these elements, thus reducing unnecessary steps in the equation formulation procedure. Once this elimination is complete the process can repeat in order to reduce other sources of derivative causality. If all storage fields containing derivative causality can be reduced, then explicit state equations can be formed for the entire system.

### 3.3 Example Problems

This technique can be applied to a wide range of physical system models, including non-linear models which contain linear storage fields. Perhaps the simplest case is that of two masses travelling at the same velocity. Consider the portion of a bond graph model shown in Figure 6, which contains an element (I2) with derivative causality.

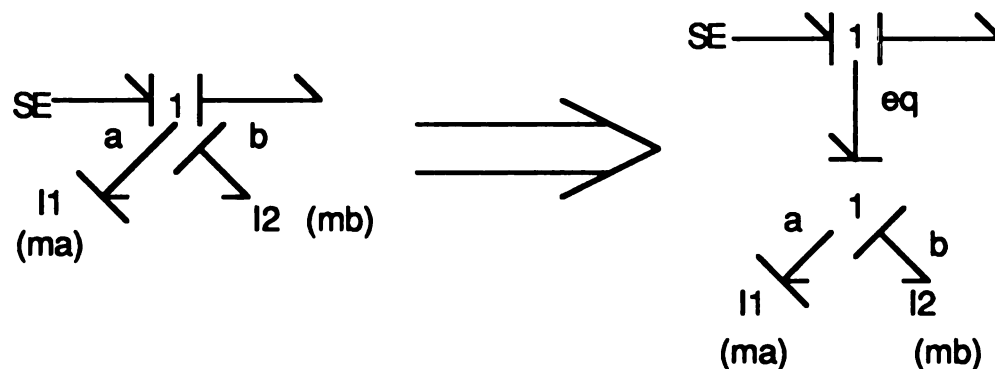


Figure 6. Bond Graph Model with Derivative Causality

The flow path from I1 to I2 yields (13).

$$g_f = 1.0 \quad (13)$$

The effort path between I1 and I2 yields (14).

$$g_e = -1.0 \quad (14)$$

If we select I2 for reduction, the equivalent parameter is calculated as shown in equation (15).

$$m_{eq} = m_a - (1.0)(m_b)(-1.0) \quad (15)$$

Thus, the equivalent parameter is:  $m_a + m_b$ . The model can now be represented as shown in Figure 7.

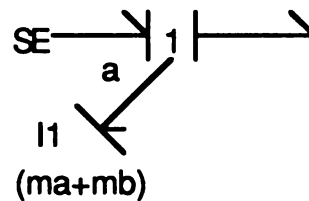


Figure 7. Reduced Bond Graph Model.

This simple example provides an illustration of the storage field reduction technique. Consider the portion of a bond graph shown in Figure 8. This example shows a derivative C element whose initial effort path leads directly into a one junction that has two exiting effort paths.

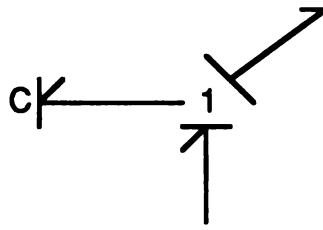


Figure 8. Bond Graph Model with Derivative Causality.

The trace would end at this point due to the ambiguity of the exiting effort path. The solution to this type of problem would entail a matrix reduction operation and is discussed in the recommendations section in Chapter 5.

## **Chapter 4. Algorithm Implementation**

### **4.1 Design of Implementation**

This methodology has been implemented into the ENPORT software simulation package. The form of the algorithm used to develop the FORTRAN code is compact and takes advantage of the processes discussed previously. These processes reduce the required storage space and increase the speed of calculation of the equivalent coefficient. The algorithm checks on the linearity of each element in the trace by examining the form of each coefficient it encounters. The computer implemented algorithm incorporates the identification process into the actual reduction by noting the coefficients along the initial examination of the path. The presentation of this process is shown in Figure 9 on the following page.

The causal path trace begins at the derivative element and continues through to the independent energy storing element. The computer implemented algorithm requires that the linear coefficients associated with each element passed during the trace be noted. Several options are employed by the algorithm inserted in the software. The process allows the user to select the element that is to be reduced. Default is the derivative element originally found by the algorithm. Additionally, the user can provide the form of the equation for the modified element. Default is the original form of the equation for the unaltered element. The software provides a listing of the elements in the causal path strictly for user information. If a model has multiple elements with derivative

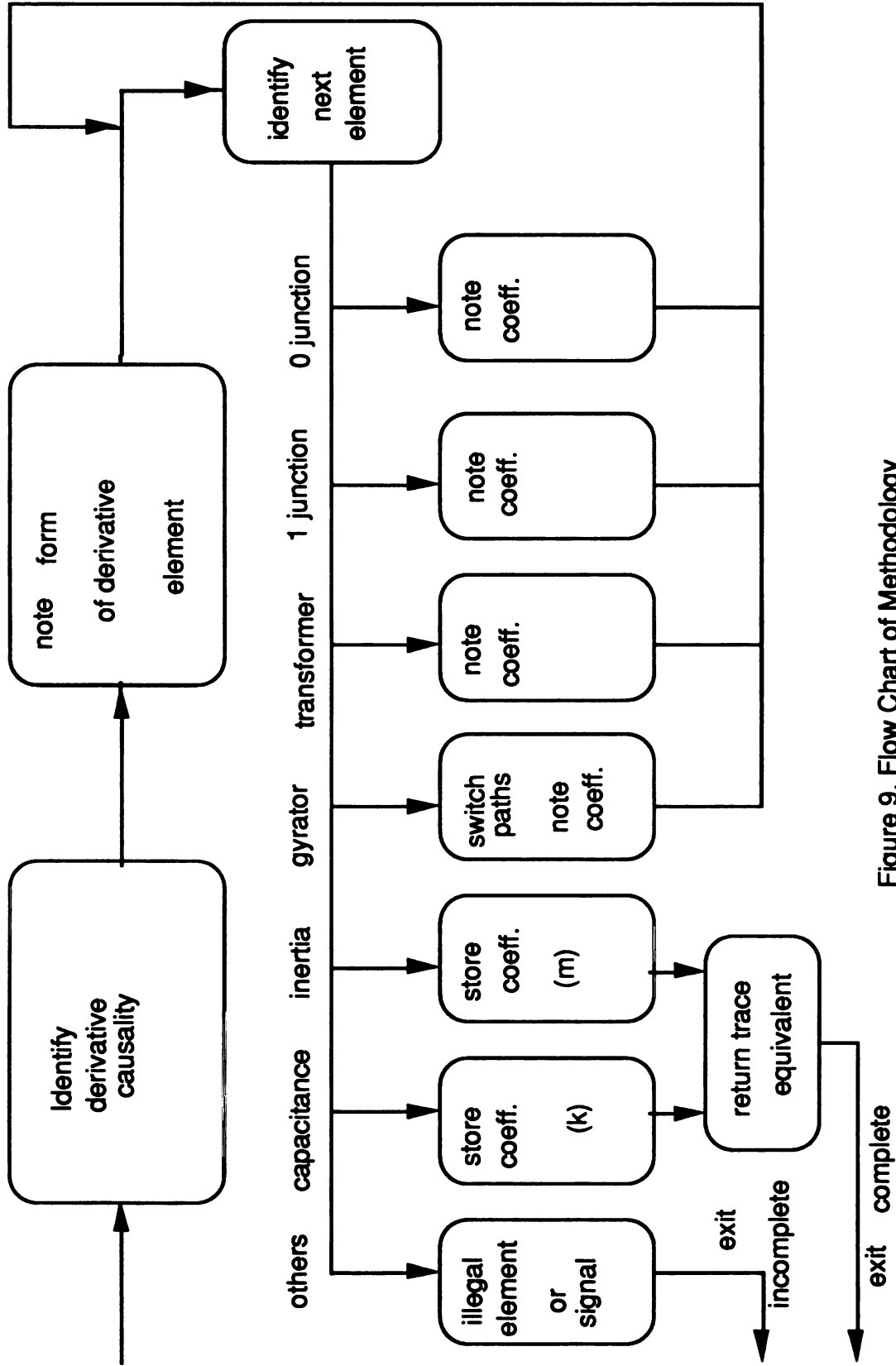


Figure 9. Flow Chart of Methodology



causality, the process is repeated until all eligible forms of derivative causality have been reduced.

Upon encountering an ineligible storage field with derivative causality the software determines under what class of problem this particular storage field falls and provides the user with a message to that effect. This identification tool was installed in the hopes that future software development will utilize the identification process already encoded in this portion of the program. The topic of further research will be discussed in Chapter 5, section 5.2.

#### 4.2 Examples of Use

The following physical system model can be described in bond graph form. As the illustration shows, the system consists of two point masses attached at either ends of a massless bar which is pivoted in the middle.

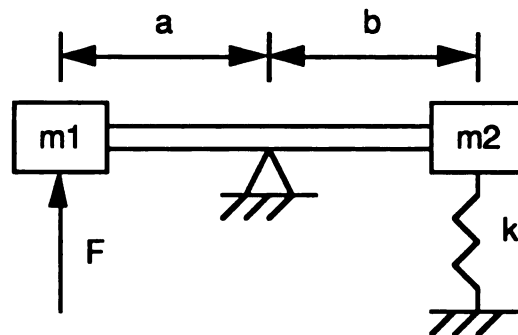


Figure 10. Lever Arm System

The bond graph of this system is shown in Figure 11. Derivative causality results on the I2 element as a direct result of the

assignment of integral causality on the I1 element. This model is eligible for the storage field reduction technique.

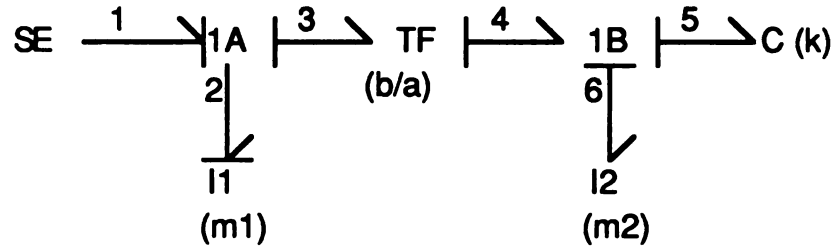


Figure 11. Bond Graph Model of Lever Arm System

The flow path from I2 to I1 yields equation (16).

$$f_6 = (b/a)f_2 \quad (16)$$

The effort path from I1 to I2 yields equation (17)

$$e_2 = -(b/a)e_6 \quad (17)$$

The equivalent parameter is shown in equation (18).

$$m_{eq} = m_1 + (b/a)^2 m_2 \quad (18)$$

The text on the following page shows the exact format used by the ENPORT software package when reducing a storage field in a bond graph model.

\*\*\* The number of dependent C and I ports is: 1.

-----  
 L: List the current equations  
 M: Modify the equations  
 S: Show function types available  
 D: list nodes with Default equations  
 T: list/modify named\_parameters  
 C: Causality display or assignment  
 A: Alleviate derivative causality  
 P: Proceed to the solver menu  
 R: Return to the main menu  
 H: Help  
 -----

Enter option (P):A

The path contains the following elements:

I2 1B TF 1A I1

Which element would you like to reduce?

I2 or I1

Enter selection (I2 )::

Hit <return> to continue...

Select equation type for I1

(GAIN ):

Trace is complete

As the text shows, the algorithm alleviates the derivative causality in a user interactive manner. The reduced bond graph model is shown in Figure 12. Note the absence of derivative causality in the model; I2 has been reduced.

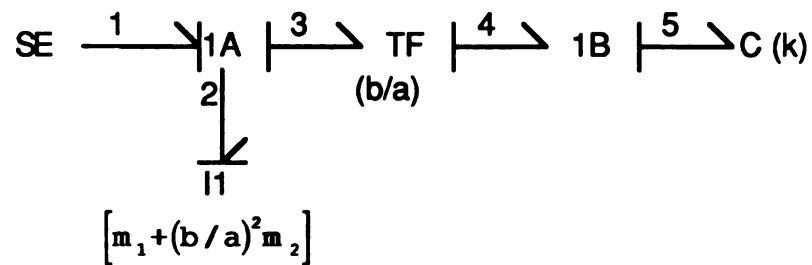


Figure 12. Reduced Form of Bond Graph Model

Equation development can now proceed to yield a system of explicit state equations because no derivative causality exists in the model.

An example bond graph is shown in Figure 13. Derivative causality results on the C1 element as a direct result of the assignment of integral causality on the I1 element.

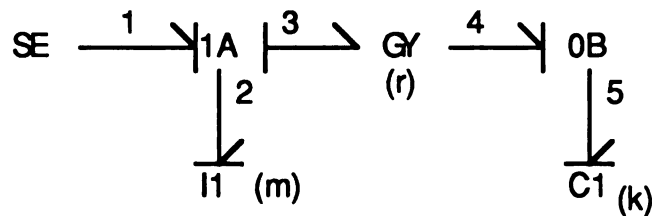


Figure 13. Bond Graph of Field with GY Element

The algorithm eliminates the derivative causality in a user interactive manner. The reduced bond graph model is shown in Figure 14. Note the absence of derivative causality in the model. The linear spring, C1, has been reduced.

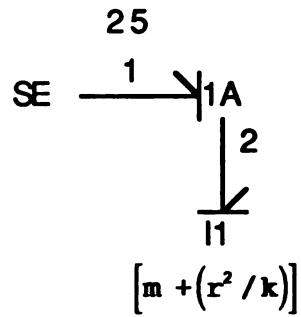


Figure 14. Reduced Form of Bond Graph

The unnecessary gyrator and 0 junction have been eliminated from the bond graph model as they now serve no purpose and equation development will now yield a system of explicit state equations. These examples provide insight into the basic principles involved in the reduction of linear storage fields with derivative causality.

### 4.3 Description of Software

The FORTRAN algorithm included in Appendix C contains six subroutines: TBACK, INIT, TRACE, REDUCE, ELIM, and REASSIGN. The controlling subroutine, TBACK, calls INIT, TRACE, and REDUCE. INIT and TRACE make no subroutine calls, while REDUCE calls ELIM and REASSIGN. When the controlling subroutine TBACK is called, it identifies derivative elements and calls INIT. INIT simply initializes the variables for each causal path trace and returns to TBACK. The controlling routine, TBACK, then calls TRACE. This subroutine, which follows the causal path trace, records the corresponding coefficients and returns to TBACK.

TBACK calls REDUCE for the reduction portion of the algorithm. REDUCE asks the user for the element to be reduced and the form of the equivalent element equation. The REDUCE subroutine then calls ELIM. The elimination routine, ELIM, removes all unnecessary elements from the bond graph and returns to REDUCE. The storage field reduction routine, REDUCE, then calls REASSIGN to align equation numbers after the removal of elements from the bond graph. This reorganization is necessary for the proper storage of equations and their coefficients. REASSIGN returns to REDUCE when this process is complete. REDUCE then returns to TBACK and the method is repeated for other linear storage fields.

Figure 15, shown on the following page, presents a flow chart of the subroutines.

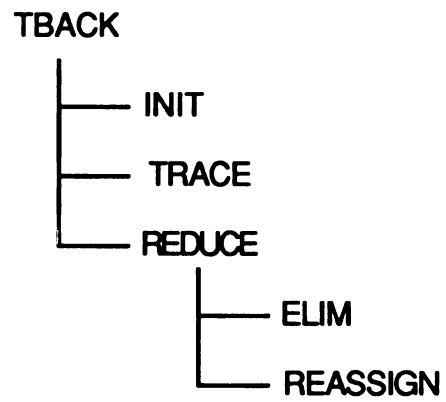
CALLING STRUCTURE

Figure 15. Flow Chart of Subroutines

## **Chapter 5. Conclusions**

### **5.1 Summary**

The process detailed in Chapters 3 and 4 alleviates derivative causality in certain linear storage fields by tracing a causal path from the dependent variable to the independent energy storing variable that forced the original application of the derivative causality. Through the use of a straightforward algorithm, the methodology records the coefficients of the elements in the path in order to calculate an equivalence parameter. This equivalence parameter is applied to the user selected element and the reduced element can then be removed from the bond graph. Thus, integral causality has been restored to the bond graph while keeping the effects of the derivative element encoded in the equations for the model. Equation formulation will thus yield a system of explicit state equations.

The methodology was derived with the intent of assisting engineers with their derivative causality problems. The causal path technique, although presented in a different manner, has been detailed previously in [1]. The form of the presentation of this technique and its implementation into ENPORT represented the main efforts in this project. The process enables the program to determine which bond graph modes are eligible for a direct explicit computational solution and which models need to be reduced before any explicit computation can take place.



The reduction takes place in a user interactive manner, with the user choosing the variable to be reduced along with the form of the equation for the equivalent energy variable. This interactive process serves not only as a means of alleviating derivative causality in a linear storage field, but also as an enhancement of the instructional powers of the ENPORT software package. This enhancement will serve as a rapid equivalence calculation technique. Thus, ENPORT users can rely on the software to calculate their linear equivalence parameters quickly and accurately.

## 5.2 Recommendations for Further Development

This project began with the intent of alleviating derivative causality in all its forms, including those that arise from non-linear elements. The range of the project was narrowed to its existing form when the details of dynamic augmentation were examined. Although dynamic augmentation yielded insight, it encompassed a range of problems much larger than originally anticipated. It is recommended that this procedure be examined in greater detail. The development of a set of standard rules for the insertion of capacitance elements would then be in order.

The extension of the storage field reduction technique to include the split path problems is recommended. This development should yield a matrix reduction technique which will enable the software to eliminate those dependent energy variables which are functions of more than one independent energy variable. It is hoped that this new process will take advantage of the enclosed method of identification

and the model reduction algorithm's ability to correct those forms of derivative causality which are already eligible for consideration.

Along the same line, the zero eigenvalue case may be eligible for a storage field reduction. This class of problems involves models with one stationary real mode. This stationary energy variable could be reduced into another energy variable and the equations for the model would be of lower order. This type of relationship arises for two capacitance elements on a one junction. As long as the initial displacements of the springs are equal, the effects of one element may be incorporated into the other without losing any dynamic information.

The nonlinear storage field may best be studied through a comparison of the dynamic augmentation method with a direct numerical solution based on implicit integration. It is not known which of these two solutions would yield the most favorable results generally. The final judgement should be made by the system modeler.

## **APPENDICES**

## APPENDIX A

### DYNAMIC AUGMENTATION OF ROLLER COASTER MODEL

The roller coaster system model was based on a simulation study begun by Scott Emery, an MSU graduate student in Mechanical Engineering. [3] This system introduces derivative causality to the bond graph when attempts are made to force the coaster to follow the track. Integral causality was restored to the system model through dynamic augmentation; the insertion of a 0 junction with a stiff spring between the vertical velocity 1 junction and the transformer that represented the  $dy/dx$  connection point. The modified system model was then examined using ENPORT and the results compare favorably with theory, i.e., the coaster followed the track and the velocity and acceleration profiles were as expected. The physical model follows in Figure A-1.

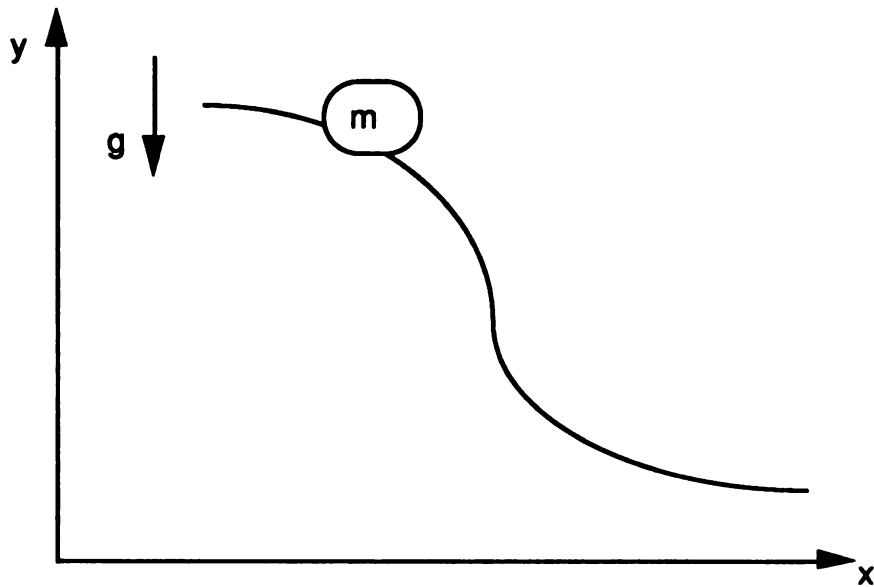


Figure A-1. Roller Coaster Model

The bond graph is shown in Figure 2 section 2.1. The derivative causality is reduced by the insertion of a stiff spring element at the connection point between the point mass and the path. The augmented bond graph is shown in Figure 4, section 2.2.

## APPENDIX B

### DYNAMIC AUGMENTATION OF ROBOT ARM MODEL

The robot arm system was based on the work presented by Tarokh in [4]. Figure B-1 shows the robot arm free to move in the vertical plane.

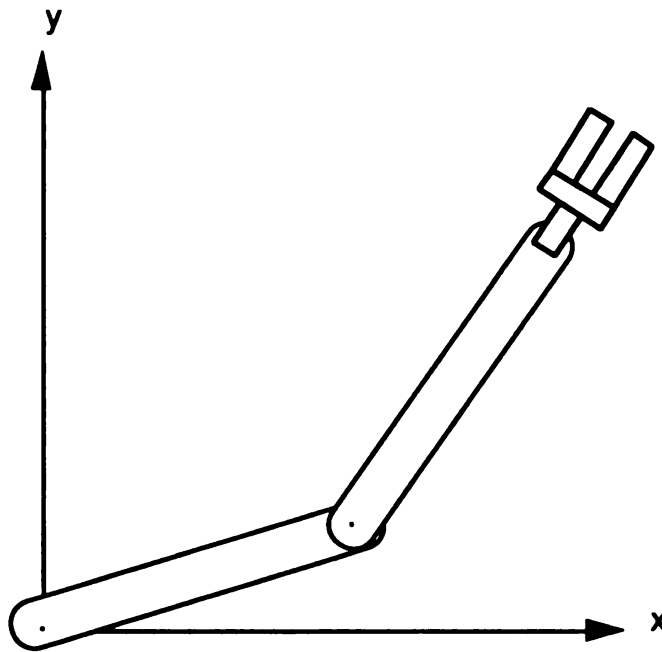


Figure B-1. Robot Arm in Vertical Plane

The bond graph of this model is shown in Figure B-2 on the following page. The model yielded derivative causality when the pinned body and the unconstrained body were attached.

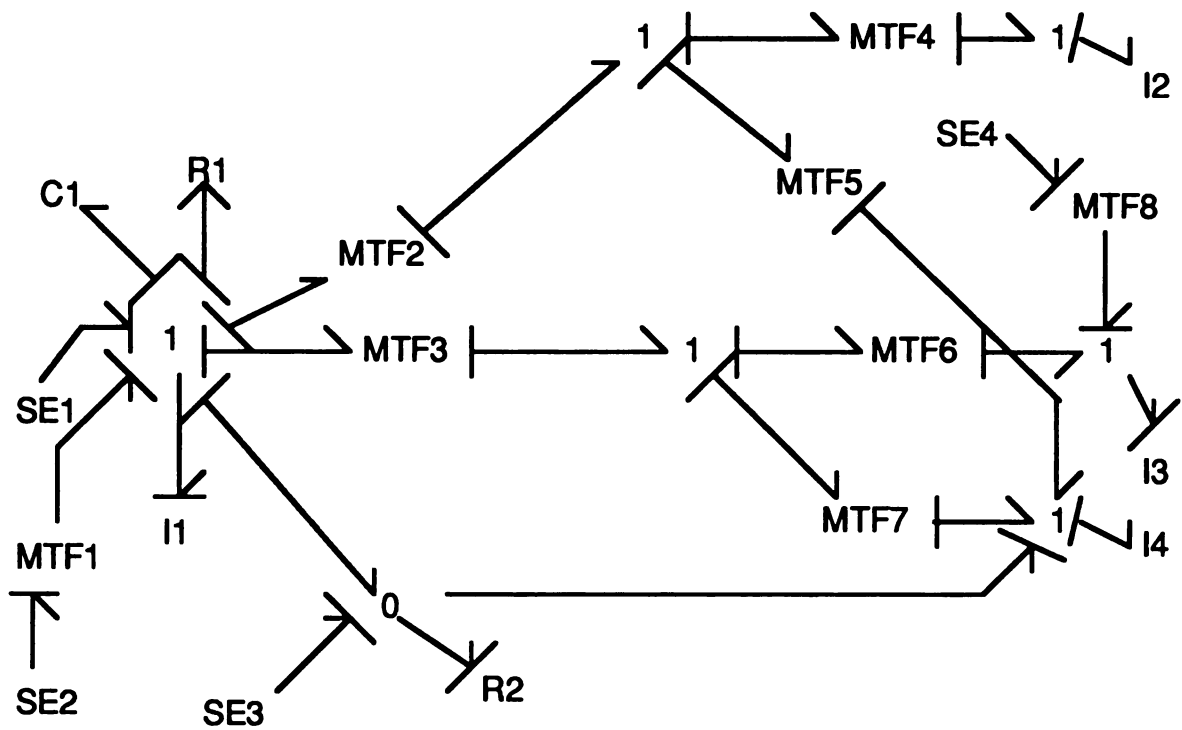


Figure B-2. Bond Graph Model of Robot Arm

The causality was corrected by dynamic augmentation; the insertion of stiff spring elements, used to represent the connection point of the two arms. One spring was inserted for the vertical velocity component and one stiff spring for the horizontal velocity. The augmented bond graph is shown in Figure B-3 on the following page.

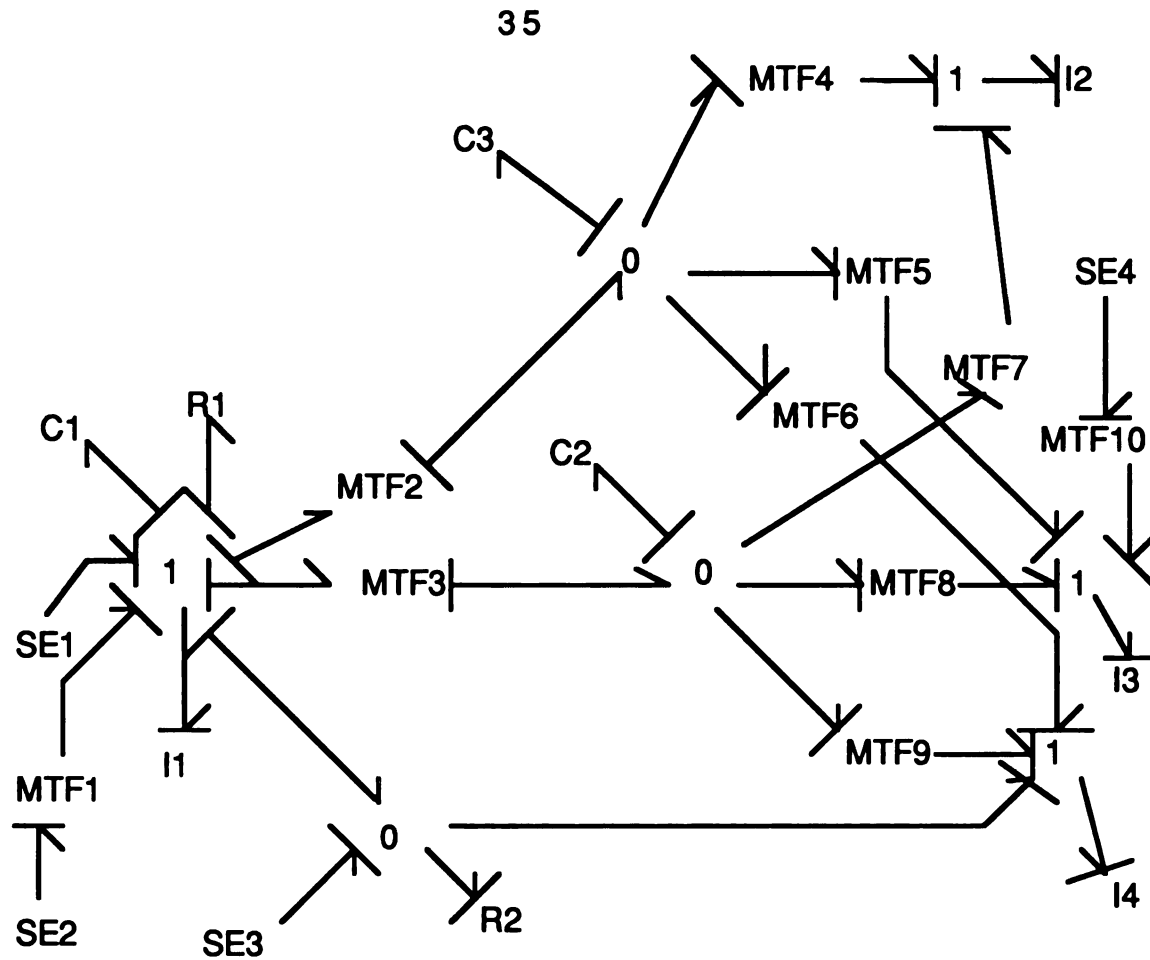


Figure 18. Augmented Bond Graph Model of Robot Arm

This dynamic augmentation methodology was discovered in Zeid's work [5]. The substitution served to alleviate the derivative causality, as Figure B-3 shows, and facilitate the formulation of state equations. The modified system model was then examined using ENPORT and the simulation results compared favorably with those available in the literature.



## APPENDIX C

# THE FORTRAN ALGORITHM

[illegible]

```

C *****
C
C
C      INTEGER CB,DE,CE,OE,IEQ,IFTP,CT,ID2(5)
C      INTEGER I1,I2,CNT,CH1,CH2,ID(5),PATH(5,20)
C      DOUBLE PRECISION PC(5),SC(5)
C      CHARACTER PPT*1,SPT*1,NTYPE*4,FNAM*4
C      LOGICAL FLAG
C
C
C      INCLUDE 'SIZEBK.CBK'
C      INCLUDE 'GREDBK.CBK'
C      INCLUDE 'FUNCBK.CBK'
C      INCLUDE 'UTILBK.CBK'
C
C
C      EXTERNAL BLNKLN,PROMPT,WRTSTR,NEWL
C
C *** T B A C K *****
C
C
C...INITIALIZE COUNTER FOR NUMBER OF PATHS
C
10    CONTINUE
      CNT = 1
C
C...USE CAUSAL PATH SUBROUTINES TO IDENTIFY THE DERIVATIVE
C CAUSALITY AND, WHERE LINEAR ELEMENTS EXIST, CALCULATE
C COEFFICIENTS AND ELIMINATE DERIVATIVE CAUSALITY FROM
MODEL
C
C
C      DO 2000 I=1,INELS
C
C...IDENTIFY ELEMENT TYPE
C
C      NTYPE = NBXTP(I)(2:2)
C
C...I ELEMENT
C
C      IF (NTYPE.EQ.'I') THEN
C
C...OBTAIN THIS ELEMENT'S CONNECTOR(S) AND ASSIGN CB

```

```

C
    I1 = IBDPTR(I)
    I2 = IBDPTR(I+1)-1
C
C...IF MULTIPOINT ELEMENT FOUND SKIP THIS PATH
C
    IF (I1.NE.I2) THEN
        CALL BLNKLN
        CALL WRTSTR('  A multiport I element was found')
        CALL BLNKLN
        GOTO 1900
    ENDIF
    CB = IBDSEL(I1)
C
C...CHECK CAUSALITY FOR I ELEMENT
C
    IF (BNDCAU(CB).NE.I) THEN
        DE = I
C
C...FIND EQUATION NUMBER
C
    IEQ = INEQP(I)
C
C...FIND EQUATION TYPE
C
    IFTP = IFCNTP(IEQ)
    FNAM = FNNAM1(IFTP)
C
C...IF EQUATION TYPE IS NOT ATTENUATE OR GAIN SKIP THIS PATH
C IF EQUATION IS OK, START TRACE
C
    IF ((FNAM.EQ.'GAIN').OR.(FNAM.EQ.'ATT')) THEN
        FLAG = .FALSE.
    ELSE
        CALL BLNKLN
        CALL WRTSTR('  A nonlinear I element was found')
        CALL BLNKLN
        GOTO 1900
    ENDIF
C
C...INITIALIZE FOR TRACE
C
    CALL INIT(DE,PPT,SPT,PC,SC,CNT)

```

```

C
C...RUN TRACE AND CHECK FOR PUNTS
C
    OE = DE
    CALL TRACE(CB,PPT,SPT,PC,SC,OE,FLAG,PATH,CNT,ID,ID2)
    IF (FLAG.EQ..TRUE.) GOTO 1900
    CNT = CNT + 1
C
C
    ENDIF
C
C...C ELEMENT
C
    ELSE IF (NTYPE.EQ.'C') THEN
C
C...OBTAIN THIS ELEMENT'S CONNECTOR(S)
C
    I1 = IBDPTR(I)
    I2 = IBDPTR(I+1)-1
C
C...IF MULTIPOINT ELEMENT FOUND SKIP THIS PATH
C
    IF (I1.NE.I2) THEN
        CALL BLNKLN
        CALL WRTSTR('  A multiport C element was found')
        CALL BLNKLN
        GOTO 1900
    ENDIF
    CB = IBDSEL(I1)
C
C...CHECK CAUSALITY FOR C ELEMENT
C
    IF (BNDCAU(CB).EQ.I) THEN
        DE = I
C
C...FIND EQUATION NUMBER
C
    IEQ = INEQP(I)
C
C...FIND EQUATION TYPE
C
    IFTP = IFCNTP(IEQ)
    FNAM = FNNAM1(IFTTP)

```

```

C
C...IF EQUATION IS NOT ATTENUATE OR GAIN -- PUNT!
C IF EQUATION IS OK, START TRACE
C
    IF ((FNAM.EQ.'GAIN').OR.(FNAM.EQ.'ATT')) THEN
        FLAG = .FALSE.
    ELSE
        CALL BLNKLN
        CALL WRTSTR('  A nonlinear C element was found')
        CALL BLNKLN
        GOTO 1900
    ENDIF

C
C...INITIALIZE FOR TRACE
C
    CALL INIT(DE,PPT,SPT,PC,SC,CNT)

C
C...RUN TRACE AND CHECK FOR PUNTS
C
    OE = DE
    COUNT = 1
    CALL TRACE(CB,PPT,SPT,PC,SC,OE,FLAG,PATH,CNT,ID,ID2)
    IF (FLAG.EQ..TRUE.) GOTO 1900
    CNT = CNT + 1

C
    ENDIF

C
C
C
    ENDIF

C
1900    CONTINUE
C
2000    CONTINUE
C
C...EXIT IF MODEL HAS NOT BEEN CHANGED
C
    IF (CNT.EQ.1) THEN
        CALL BLNKLN
        CALL WRTSTR('  No derivative causality was found')
        CALL BLNKLN
        RETURN
    ENDIF

C

```



```

C *****
C
C  VARIABLE LIST:
C
C    CNT:  NUMBER OF PATHS IN MODEL
C    DE:   DERIVATIVE ELEMENT
C    NTYPE: ELEMENT TYPE
C    PC:   PRIMARY COEFFICIENT
C    PPT:  PRIMARY PATH TRACE
C    SC:   SECONDARY COEFFICIENT
C    SPT:  SECONDARY PATH TRACE
C
C *****
C
C
C    INTEGER DE,CNT
C    CHARACTER NTYPE*4,PPT*1,SPT*1
C    DOUBLE PRECISION PC(5),SC(5)
C
C    INCLUDE 'SIZEBK.CBK'
C    INCLUDE 'GREDBK.CBK'
C    INCLUDE 'FUNCBK.CBK'
C    INCLUDE 'UTILBK.CBK'
C
C *****
C ** I N I T *****
C
C
C...ASSIGN INITIAL VALUES DEPENDING ON FORM OF DE
C
C    NTYPE = NBXTP(DE)(2:2)
C    IF (NTYPE.EQ.'C') THEN
C      PPT = 'E'
C      SPT = 'F'
C
C
C
C    ELSE
C      PPT = 'F'
C      SPT = 'E'
C    ENDIF
C
C
C
C

```

## C...INITIALIZE PC, AND SC

C

**PC(CNT) = 1.0D0**

**SC(CNT) = 1.0D0**

C

C

## RETURN

**END**

C

C

C>>>>>

C

C

c

[illegible]

C

C

SUBROUTINE TRACE(CB,PPT,SPT,PC,SC,OE,FLAG,PATH,CNT,ID,ID2)

C

C

\*\*\*\*\*

C

**C VARIABLE LIST:**

C

**C AC: ACTUAL CONNECTOR (IDENTIFICATION TOOL)**

C BNDTYP: CONNECTOR TYPE

**C      CB:      CURRENT CONNECTOR NUMBER IN TRACE**

**C**      **CE:**      **CURRENT ELEMENT IN TRACE**

**C**    **CNT:**    **NUMBER OF TRACE PATHS IN MODEL**

**C COUNT: NUMBER OF ELEMENTS IN TRACE**

**C**      **ENAM:**      **ELEMENT NAME IN SYSTEM GRAPH**

**C      FLAG:      EXIT SIGNAL (TRACE FAILED IF FLAG=.TRUE.)**

**C**    **FNAM:**    **NAME OF FUNCTION ASSOCIATED WITH IEQ**

**C 11: CONNECTOR LIST POINTER**

**C 12: CONNECTOR LIST POINTER**

**C ID: PATH IDENTIFICATION**

**C ID2: ARRAY STORING NUMBER OF ELEMENTS IN PATH**

C IEQ: EQUATION NUMBER ASSOCIATED WITH CURRENT  
C ELEMENT

**C**    **IFTP:**    **INTEGER FUNCTION TYPE OF IEQ**

**C MGY: GYRATOR MODULUS FOUND IN TRACE**

**C MTF: TRANSFORMER MODULUS FOUND IN TRACE**

C	NTYPE:	ELEMENT TYPE
---	--------	--------------



C	OE:	PREVIOUS ELEMENT IN TRACE
C	PATH:	ARRAY CONTAINING ELEMENT NUMBERS IN TRACE
C	PC:	PRIMARY COEFFICIENT
C	POWER1:	DIRECTION OF CONNECTOR BETWEEN OE AND CE
C	POWER2:	DIRECTION OF CONNECTOR BETWEEN CE AND NE
C	PPT:	PRIMARY PATH TRACE
C	SC:	SECONDARY COEFFICIENT
C	SCOUNT:	NUM. OF PRIMARY PATH'S EXITING A 0 OR 1 ELEMENT
C		(NOTE: IF SCOUNT > 1 THE MODEL CANNOT BE
C		ADJUSTED WITH THIS METHOD)
C	SPT:	SECONDARY PATH TRACE
C	STRK1:	ELEMENT NUMBER AT CAUSAL STROKE END OF CB
C	STRK2:	ELEMENT NUMBER AT CAUSAL STROKE END OF NB
C	TEST:	COMPLETION CHECK (COMPLETE IF TEST=.TRUE.)

C  
C \*\*\*\*\*

```

C
C
INTEGER CB,CE,OE,I1,I2,AC,STRK1,STRK2,SCOUNT,IEQ,IFTP
INTEGER COUNT,CNT,ID(5),PATH(5,20),ID2(5)
CHARACTER PPT*1,SPT*1,NTYPE*4,FNAM*8,BNDTYP*4
DOUBLE PRECISION PC(5),SC(5),POWER1,POWER2,MGY,MTF
LOGICAL TEST,FLAG

```

```
C
C
C      INCLUDE 'SIZEBK.CBK'
C      INCLUDE 'GREDBK.CBK'
C      INCLUDE 'FUNCBK.CBK'
C      INCLUDE 'UTILBK.CBK'
```

```
C
C
C      EXTERNAL BLNKLN,WRTSTR,NEWL,PROMPT
```

```
C
C**TRACE*****
```

C  
C  
C...INITIALIZE TEST, FLAG, SCOUNT, COUNT, ID AND PATH

```
C
FLAG = .FALSE.
ID(CNT) = 1
TEST = .FALSE.
SCOUNT = 0
COUNT = 1
```

```

      PATH(CNT,COUNT) = OE
C
C
10  CONTINUE
C
C...IDENTIFY CURRENT ELEMENT, CE, FROM CURRENT CONNECTOR, CB,
C AND OLD ELEMENT, OE. SIMILIARLY, IDENTIFY THE POWER
C...DIRECTION WITH +1 INDICATING THAT POWER IS CONSERVED
C (CB GOES FROM OE TO CE).
C
C
      CE = IELSBD(CB,1)
      POWER1 = -1.0D0
      IF (CE.EQ.OE) THEN
      CE = IELSBD(CB,2)
      POWER1 = 1.0D0
      ENDIF
      STRK1 = BNDCAU(CB)
C
C...IDENTIFY ELEMENT TYPE OF CE
C
      NTYPE = NBXTP(CE)(2:2)
C
C
C *****
C *****
C
C
      IF (NTYPE.EQ.'0') THEN
C
C...IDENTIFY THE CONNECTOR LIST FOR THIS ELEMEMENT
C
      I1 = IBDPTR(CE)
      I2 = IBDPTR(CE+1)-1
C
C...IF SIGNAL EXISTS IN CONNECTOR LIST -- PUNT!
C
      DO 20 J=I1,I2
      AC = IBDSEL(J)
      BNDTYP = BDTP(AC)(1:1)
      IF(BNDTYP.EQ.'SIGNAL') THEN
      ID(CNT) = 3
      GOTO 10100

```

```

        ENDIF
20    CONTINUE
C
C...EXAMINE EACH CONNECTOR ON THE ELEMENT
C
        DO 100 I=I1,I2
        AC = IBDSEL(I)
C
C...FIND POWER DIRECTION OF AC
C
        IF (IELSBD(AC,1).EQ.CE) THEN
            POWER2 = 1.0D0
        ELSE
            POWER2 = -1.0D0
        ENDIF
C
C...IDENTIFY THE CAUSAL STROKE END OF AC
C
        STRK2 = BNDCAU(AC)
C
C...IF THE PRIMARY PATH IS FLOW, CHECK FOR SPLIT PATH,
C ASSIGN NEW CONNECTOR NUMBER, AND ADJUST PC & SC
C
        IF ((PPT.EQ.'F').AND.(STRK2.NE.CE).AND.(AC.NE.CB)) THEN
C
C
            SCOUNT = SCOUNT + 1
C
C...USE CAUSAL INFO. ALONG WITH POWER DIRECTIONS OF
C AC AND CB TO DETERMINE ADJUSTMENTS TO SC AND PC
C
            IF (POWER1.EQ.POWER2) THEN
                PC(CNT) = PC(CNT)
            ELSE
                PC(CNT) = -1.0D0*PC(CNT)
            ENDIF
            SC(CNT) = SC(CNT)
C
C...ASSIGN NEXT CONNECTOR
C
        NB = AC
C
C...IF PPT IS EFFORT: ASSIGN NEW CON. NUMBER AND ADJUST SC

```

```

C
    ELSE IF ((PPT.EQ.'E').AND.(STRK2.EQ.CE)) THEN
C
C...USE CAUSAL INFO. ALONG WITH POWER DIRECTIONS OF
C AC AND CB TO DETERMINE ADJUSTMENTS TO SC AND PC
C
    IF (POWER1.EQ.POWER2) THEN
        SC(CNT) = SC(CNT)
    ELSE
        SC(CNT) = -1.0D0*SC(CNT)
    ENDIF
    PC(CNT) = PC(CNT)
C
C...ASSIGN NEXT CONNECTOR
C
    NB = AC
C
    ENDIF
100  CONTINUE
C
C...ASSIGN VALUES FOR NEXT STEP IN TRACE
C
    COUNT = COUNT + 1
    PATH(CNT,COUNT) = CE
    OE = CE
    CB = NB
    TEST = .FALSE.
C
C...NOTE SPLIT PATH
C
    IF (SCOUNT.GT.1) THEN
        CALL BLNKLN
        CALL WRTSTR(' Split path at 0 junction ')
        CALL WRTSTR(' Trace failed... ')
        CALL BLNKLN
        ID(CNT) = 2
        GOTO 10000
    ENDIF
C
C
C *****
C *****
C

```

```

C
    ELSE IF (NTYPE.EQ.'1') THEN
C
C...IDENTIFY THE CONNECTOR LIST FOR THIS ELEMENT
C
    I1 = IBDPTR(CE)
    I2 = IBDPTR(CE+1)-1
C
C...IF SIGNAL EXISTS IN CONNECTOR LIST -- PUNT!
C
    DO 150 J=I1,I2
    AC = IBDSEL(J)
    BNDTYP = BDTP(AC)(1:1)
    IF(BNDTYP.EQ.'SIGNAL') THEN
        ID(CNT) = 3
        GOTO 10100
    ENDIF
150    CONTINUE
C
C...EXAMINE EACH CONNECTOR ON THE ELEMENT
C
    DO 200 I=I1,I2
    AC = IBDSEL(I)
C
C...FIND POWER DIRECTION OF AC
C
    IF (IELSBD(AC,1).EQ.CE) THEN
        POWER2 = 1.0D0
    ELSE
        POWER2 = -1.0D0
    ENDIF
C
C...IDENTIFY THE CAUSAL STROKE END OF AC
C
    STRK2 = BNDCAU(AC)
C
C...IF THE PRIMARY PATH IS EFFORT, CHECK FOR SPLIT PATH,
C ASSIGN NEW CONNECTOR NUMBER, AND ADJUST PC & SC
C
    IF ((PPT.EQ.'E').AND.(STRK2.EQ.CE).AND.(AC.NE.CB)) THEN
C
C

```

```

SCOUNT = SCOUNT + 1
C
C...USE CAUSAL INFO. ALONG WITH POWER DIRECTIONS OF
C AC AND CB TO DETERMINE ADJUSTMENTS TO SC AND PC
C
    IF (POWER1.EQ.POWER2) THEN
        PC(CNT) = PC(CNT)
    ELSE
        PC(CNT) = -1.0D0*PC(CNT)
    ENDIF
    SC(CNT) = SC(CNT)
C
C...ASSIGN NEXT CONNECTOR
C
    NB = AC
C
C...IF PPT IS FLOW: ASSIGN NEW CONNECTOR NUMBER AND ADJUST SC
C
    ELSE IF ((PPT.EQ.'F').AND.(STRK2.NE.CE)) THEN
C
C...USE CAUSAL INFO. ALONG WITH POWER DIRECTIONS OF
C AC AND CB TO DETERMINE ADJUSTMENTS TO SC AND PC
C
    IF (POWER1.EQ.POWER2) THEN
        SC(CNT) = SC(CNT)
    ELSE
        SC(CNT) = -1.0D0*SC(CNT)
    ENDIF
    PC(CNT) = PC(CNT)
C
C...ASSIGN NEXT CONNECTOR
C
    NB = AC
C
C
    ENDIF
200 CONTINUE
C
C...ASSIGN VALUES FOR NEXT STEP IN TRACE
C
    COUNT = COUNT + 1
    PATH(CNT,COUNT) = CE

```

```

      OE = CE
      CB = NB
      TEST = .FALSE.
C
C...NOTE SPLIT PATH
C
      IF (SCOUNT.GT.1) THEN
        CALL BLNKLN
        CALL WRTSTR(' Split path found at 1 junction ')
        CALL WRTSTR(' Trace failed...' )
        CALL BLNKLN
        ID(CNT) = 2
        GOTO 10000
      ENDIF
C
C
C *****
C *****
C
C
      ELSE IF (NTYPE.EQ.'T') THEN
C
C...IDENTIFY MOD.TF FROM CE
C
C...FIND EQUATION NUMBER FOR CE
C
      IEQ = INEQP(CE)
C
C...FIND EQUATION TYPE OF CE
C
      IFTP = IFCNTP(IEQ)
      FNAM = FNNAM1(IFTP)
C
C...IF MTF IS NOT A CONSTANT -- PUNT!
C
      IF (FNAM.EQ.'CON') THEN
        MTF = EQPAR(IEQPTR(IEQ))
      ELSE
        ID(CNT) = 3
        GOTO 10200
      ENDIF
C
C...FIND CONNECTOR LIST FOR THIS ELEMENT

```

```

C
    I1 = IBDPTR(CE)
    I2 = IBDPTR(CE+1)-1
C
C...IF SIGNAL EXISTS IN CONNECTOR LIST -- PUNT!
C
    DO 250 J=I1,I2
    AC = IBDSEL(J)
    BNDTYP = BDTP(AC)(1:1)
    IF(BNDTYP.EQ.'SIGNAL') THEN
        ID(CNT) = 3
        GOTO 10100
    ENDIF
250    CONTINUE
C
C...FIND NEXT CONNECTOR
C
    DO 300 I = I1,I2
    AC = IBDSEL(I)
    IF (AC.NE.CB) THEN
        NB = AC
C
C...CHECK CAUSAL DIRECTION
C
        IF (BNDCAU(NB).EQ.CE) THEN
C
C...ADJUST PC AND SC
C
            PC(CNT) = MTF*PC(CNT)
            SC(CNT) = (1.0D0/MTF)*SC(CNT)
        ELSE
            PC(CNT) = (1.0D0/MTF)*PC(CNT)
            SC(CNT) = MTF*SC(CNT)
        ENDIF
    ENDIF
300    CONTINUE
C
C...ASSIGN VALUES FOR NEXT STEP IN TRACE
C
    COUNT = COUNT + 1
    PATH(CNT,COUNT) = CE
    OE = CE
    CB = NB

```



```

      TEST = .FALSE.C
C
C *****
C *****
C
C
      ELSE IF (NTYPE.EQ.'G') THEN
C
C   IDENTIFY MGY FROM CE
C
C...FIND EQUATION NUMBER OF CE
C
      IEQ = INEQP(CE)
C
C...FIND EQUATION TYPE OF CE
C
      IFTP = IFCNTP(IEQ)
      FNAM = FNNAM1(IFTP)
C
C...IF MGY IS NOT A CONSTANT -- PUNT!
C
      IF (FNAM.EQ.'CON') THEN
        MGY = EQPAR(IEQPTR(IEQ))
      ELSE
        ID(CNT) = 3
        GOTO 10200
      ENDIF
C
C...FIND CONNECTOR LIST FOR THIS ELEMENT
C
      I1 = IBDPTR(CE)
      I2 = IBDPTR(CE+1)-1
C
C...IF SIGNAL EXISTS IN CONNECTOR LIST -- PUNT!
C
      DO 350 J=I1,I2
      AC = IBDSEL(J)
      BNDTYP = BDTP(AC)(1:1)
      IF(BNDTYP.EQ.'SIGNAL') THEN
        ID(CNT) = 3
        GOTO 10100
      ENDIF
350  CONTINUE

```

```

C
C...FIND NEXT CONNECTOR
C
    DO 400 I = I1,I2
    AC = IBDSEL(I)
    IF (AC.NE.CB) THEN
        NB = AC
C
C...CHECK CAUSAL DIRECTION
C
    IF (BNDCAU(NB).EQ.CE) THEN
C
C...SWITCH PATHS (ENTERED GY ON PRIMARY FLOW)
C
        PPT = 'E'
        SPT = 'F'
C
C...ADJUST PC AND SC
C
        PC(CNT) = (1.0D0/MGY)*PC(CNT)
        SC(CNT) = MGY*SC(CNT)
C
        ELSE
C
C...SWITCH PATHS (ENTERED GY ON PRIMARY EFFORT)
C
        PPT = 'F'
        SPT = 'E'
C
C...ADJUST PC AND SC
C
        PC(CNT) = MGY*PC(CNT)
        SC(CNT) = (1.0D0/MGY)*SC(CNT)
        ENDIF
        ENDIF
400    CONTINUE
C
C...ASSIGN VALUES FOR NEXT STEP IN TRACE
C
    COUNT = COUNT + 1
    PATH(CNT,COUNT) = CE
    OE = CE
    CB = NB

```

**TEST = .FALSE.**

C

C

\*\*\*\*\*

\*\*\*\*\*

C

C

```
ELSE IF (NTYPE.EQ.'I') THEN
```

C

### C...FIND EQUATION NUMBER

C

**IEQ = INEQP(CE)**

C

### C...FIND EQUATION TYPE

C

**IFTP = IFCNTP(IEQ)**

**FNAM = FNNAM1(IFTF)**

C

**C...IF EQUATION TYPE IS NOT ATT OR GAIN -- PUNT!**

C

**IF ((FNAM.EQ.'ATT').OR.(FNAM.EQ.'GAIN')) THEN**

**FLAG = .FALSE.**

## ELSE

**ID(CNT) = 3**

**GOTO 10200**

**ENDIF**

**C**

### C...FIND CONNECTOR LIST FOR THIS ELEMENT

**C**

**I1 = IBDPTR(CE)**

$$I2 = IBDPTR(CE+1)-1$$

**C**

**C...IF SIGNAL EXISTS IN CONNECTOR LIST -- PUNT!**

**C**

**DO 450 J=I1,I2**

**AC = IBDSEL(J)**

**BNDTYP = BDTP(AC)(1:1)**

```
IF(BNDTYP.EQ.'SIGNAL') THEN
```

**ID(CNT) = 3**

# GOTO 10100

**ENDIF**

**450 CONTINUE**

**C**

C...IDENTIFY THIS ELEMENT IN PATH

C

```

      ID(CNT) = 1
      COUNT = COUNT + 1
      PATH(CNT,COUNT) = CE
      ID2(CNT) = COUNT
      TEST = .TRUE.

```

C

C

C\*\*\*\*\*

C\*\*\*\*\*

C

C

```

      ELSE IF (NTYPE.EQ.'C') THEN

```

C

C...FIND EQUATION NUMBER

C

```

      IEQ = INEQP(CE)

```

C

C...FIND EQUATION TYPE

C

```

      IFTP = IFCNTP(IEQ)
      FNAM = FNNAM1(IFTP)

```

C

C...IF FUNCTION IS NOT ATT OR GAIN -- PUNT!

C

```

      IF ((FNAM.EQ.'ATT').OR.(FNAM.EQ.'GAIN')) THEN
        FLAG = .FALSE.
      ELSE
        ID(CNT)=3
        GOTO 10200
      ENDIF

```

C

C...FIND CONNECTOR LIST FOR THIS ELEMENT

C

```

      I1 = IBDPTR(CE)
      I2 = IBDPTR(CE+1)-1

```

C

C...IF SIGNAL EXISTS IN CONNECTOR LIST -- PUNT!

C

```

      DO 550 J=I1,I2
      AC = IBDSEL(J)
      BNDTYP = BDTP(AC)(1:1)

```

56

```
      IF(BNDTYP.EQ.'SIGNAL') THEN
        ID(CNT) = 3
        GOTO 10100
      ENDIF
550    CONTINUE
C
C...IDENTIFY THIS ELEMENT IN PATH
C
      ID(CNT) = 1
      COUNT = COUNT + 1
      PATH(CNT,COUNT) = CE
      ID2(CNT) = COUNT
      TEST = .TRUE.
C
C
C *****
C *****
C
C
      ELSE IF (NTYPE.EQ.'E') THEN
C
C...PRINT ERROR MESSAGE AND RETURN
C
      CALL BLNKLN
      CALL WRTSTR(' Trace found <Se> as forcing element')
      CALL WRTSTR(' Trace failed...' )
      CALL BLNKLN
      ID(CNT) = 3
      FLAG = .TRUE.
      RETURN
C
C
C *****
C *****
C
C
      ELSE IF (NTYPE.EQ.'F') THEN
C
C...PRINT ERROR MESSAGE AND RETURN
C
      CALL BLNKLN
      CALL WRTSTR(' Trace found <Sf> as forcing element')
      CALL WRTSTR(' Trace failed...' )
```

```
CALL BLNKLN
ID(CNT) = 3
FLAG = .TRUE.
RETURN
```

C

C

**C** ★★★★★★★★★★★★

[illegible]

C

C

```
ELSE IF (NTYPE.EQ.'R') THEN
```

**C**

### C...PRINT ERROR MESSAGE AND RETURN

C

**CALL BLNKLN**

**CALL WRTSTR(' Trace found <R> as forcing element')**

CALL WRTSTR(' Trace failed...')

**CALL BLNKLN**

**ID(CNT) = 3**

**FLAG = .TRUE.**

## RETURN

C

C

C ★★★★★★★★

C

C

C

**ELSE**

C

**C...TRACE HAS FOUND AN ILLEGAL ELEMENT TYPE**

C

C

**CALL BLNKLN**

**CALL WRTSTR(' Trace has found an illegal element')**

CALL WRTSTR(' Trace failed...')

**CALL BLNKLN**

**ID(CNT) = 3**

```
FLAG = .TRUE.
```

## RETURN

C

## C...SPLIT PATH

C

10000 CONTINUE

```

C
    FLAG = .TRUE.
    RETURN
C
C...SIGNAL FOUND
C
10100    CONTINUE
C
    CALL BLNKLN
    CALL WRTSTR(' A signal was found in the path')
    CALL WRTSTR(' Trace failed...      ')
    CALL BLNKLN
    FLAG = .TRUE.
    RETURN
C
C...ILLEGAL FUNCTION TYPE
C
10200    CONTINUE
C
    CALL BLNKLN
    CALL WRTSTR(' Illegal function type found')
    CALL WRTSTR(' Trace failed...      ')
    CALL BLNKLN
    FLAG = .TRUE.
    RETURN

C
C
C*****
C*****
C
C
C
    ENDIF
C
C
    IF (TEST.EQ..FALSE.) GOTO 10
C
C
    RETURN
    END
C
C
C>>>>>

```

[illegible]

**VARIABLE LIST:**

ANS:	ANSWER
CNT:	PATH NUMBER
COUNT:	ELEMENT NUMBER
DE:	DERIVATIVE ELEMENT
DFLT:	DEFAULT ANSWER
DL:	ELEMENT NUMBER
EC:	EQUIVALENT COEFFICIENT
EL:	ELEMENT NUMBER
ENAM1:	ELEMENT NAME
ENAM2:	ELEMENT NAME
FE:	FORCING ELEMENT
FLAG:	ERROR INDICATOR
I1:	POINTER
IB:	POINTER
ID2:	ARRAY STORING NUMBER OF ELEMENTS IN PATH
IEQ:	EQUATION NUMBER
K:	STIFFNESS PARAMETER
Mt	INERTIAL PARAMETER
Nt	ELEMENT NAME
NAM:	ELEMENT NAME
NAME:	ELEMENT NAMES
OC:	ORIGINAL COEFFICIENT
PATH:	MATRIX CONTAINING ELEMENTS IN TRACE
PC:	PRIMARY COEFFICIENT
SC:	SECONDARY COEFFICIENT
Z:	POINTER

[illegible]

```
INTEGER PATH(5,20),CNT,ID2(5),DE,FE,IEQ,EL,DL,I1,IB,Z
```



```

    INTEGER COUNT
    DOUBLE PRECISION PC(5),SC(5),OC,EC,K,M
    CHARACTER NTYPE*4,ENAM1*8,ENAM2*8,ANS*8,STRING*40
    CHARACTER NAM*8,NAME(20)*8,DFLT*8,N*8
    LOGICAL FLAG

C
    EXTERNAL WRTSTR,PROMPT,GETANS,BLNKLN,NEWL,MENSET,GOON
C
    INCLUDE 'SIZEBK.CBK'
    INCLUDE 'GREDBK.CBK'
    INCLUDE 'FUNCBK.CBK'
    INCLUDE 'UTILBK.CBK'

C
C**REDUCE*****
C
    COUNT = ID2(CNT)
    CALL MENSET(.TRUE.)
    FULL = MMFULL

C
C...INFORM USER OF PATH CONTENTS
C
    DO 4 I=1,COUNT
        Z = PATH(CNT,I)
        NAME(I) = ELNAM(Z)
4    CONTINUE
C
    CALL BLNKLN
    CALL WRTSTR('  The path contains the following elements:')
    CALL BLNKLN
    WRITE(STRING,5)(NAME(I),I=1,COUNT)
5    FORMAT(4X,6(1X,A3))
    CALL WRTSTR(STRING)
    CALL BLNKLN

C
C...OBTAIN REDUCTION CHOICES
C
    DE = PATH(CNT,1)
    FE = PATH(CNT,COUNT)
    ENAM1 = ELNAM(DE)
    ENAM2 = ELNAM(FE)

C
C...PROMPT USER FOR REDUCTION CHOICE (DEFAULT = DE)
C

```

```

C
10  CONTINUE
    CALL BLNKLN
    CALL WRTSTR('    Which element would you like to reduce?')
    CALL BLNKLN
    WRITE(STRING,100)ENAM1,ENAM2
100  FORMAT(5X,A8,'or ',5X,A8)
    CALL WRTSTR(STRING)
    CALL BLNKLN
    STRING='    Enter selection ('//ENAM1//'):'
    CALL PROMPT(STRING)

```

```

C
C...READ AND INTERPRET ANSWER
C

```

```

    ANS = '#'
    CALL GETANS(ANS)
    IF ((ANS.EQ.'#').OR.(ANS.EQ.' ')) THEN
        ANS = ENAM1
    ENDIF

```

```

C
C...ASSIGN VALUES FOR REDUCTION
C

```

```

    IF (ANS.EQ.ENAM1) THEN
        EL = FE
        DL = DE
    ELSE IF (ANS.EQ.ENAM2) THEN
        EL = DE
        DL = FE
    ELSE
        CALL BLNKLN
        CALL WRTSTR('    Bad entry...Please try again')
        CALL BLNKLN
        GOTO 10
    ENDIF

```

```

C
C...FIND OC
C

```

```

    NTYPE = NBXTP(DL)(2:2)
    IEQ = INEQP(DL)
    IF (NTYPE.EQ.'I') THEN
        OC = EQPAR(IEQPTR(IEQ))
        IF (OC.NE.0.0D0) THEN
            OC = 1.0D0/OC

```

```

ENDIF
ELSE
  OC = EQPAR(IEQPTR(IEQ))
ENDIF
C
C...OBTAIN CORRECT COEFFICIENTS
C
  NTYPE = NBXTP(EL)(2:2)
C
C...IDENTIFY CONNECTOR LIST
C
  I1 = IBDPTR(EL)
  IB = IBDSEL(I1)
C
C
  IF (NTYPE.EQ.'I') THEN
C
C...FIND EQUATION NUMBER
C
  IEQ = INEQP(EL)
C
C...FIND COEFFICIENT
C
  M = EQPAR(IEQPTR(IEQ))
  IF (M.NE.0.0D0) THEN
    M = 1.0D0/M
  ENDIF
C
C...CALCULATE EQUIVALENT COEFFICIENT
C
  EC = M-PC(CNT)*SC(CNT)*OC
  IF (EC.NE.0.0D0) THEN
    EC = 1.0D0/EC
  ENDIF
C
C
  ELSE IF (NTYPE.EQ.'C') THEN
C
C...FIND EQUATION NUMBER
C
  IEQ = INEQP(EL)
C
C...FIND COEFFICIENT

```

```

C
    K = EQPAR(IEQPTR(IEQ))
C
C...CALCULATE EQUIVALENT COEFFICIENT
C
    EC = K - PC(CNT)*SC(CNT)*OC
C
C
    ELSE
C
C...ERROR IN LIST
C
    CALL BLNKLN
    CALL WRTSTR('    An error in the procedure has occurred')
    CALL BLNKLN
    FLAG = .TRUE.
    RETURN
    ENDIF
C
C...ELIMINATION
C
    CALL ELIM(PATH,DL,CNT,ID2)
    DO 200 I=2,5
200    E7CFLG(I) = .FALSE.
        CALL REASSIGN(.TRUE.)
C
C...CORRECT THE EQUATION
C
    DO 1000 I=1,INELS
        NAM = ELNAM(I)
        IF (ENAM1.EQ.NAM) THEN
            DE = I
            IEQ = INEQP(DE)
            N = NAM
        ELSE IF (ENAM2.EQ.NAM) THEN
            FE = I
            IEQ = INEQP(FE)
            N = NAM
        ENDIF
1000    CONTINUE
C
C...ASSIGN CORRECT COEFFICIENT
C

```

```

      EQPAR(IEQPTR(IEQ)) = EC
C
C...PROMPT USER FOR EQUATION TYPE
C
1400   CONTINUE
      CALL BLNKLN
      IF (NTYPE.EQ.'I') THEN
        DFLT = 'GAIN'
      ELSE IF (NTYPE.EQ.'C') THEN
        DFLT = 'ATT'
      ENDIF
      WRITE(STRING,1450)N
1450   FORMAT(5X,'Select equation type for ',A8)
      CALL WRTSTR(STRING)
      CALL BLNKLN
      STRING='      (//DFLT//):'
      CALL PROMPT(STRING)
C
C...READ AND INTERPRET ANSWER
C
      ANS = '#'
      CALL GETANS(ANS)
      IF ((ANS.EQ.'#').OR.(ANS.EQ.' ')) THEN
        ANS = DFLT
      ENDIF
      IF ((ANS.NE.'ATT').OR.(ANS.NE.'GAIN')) THEN
        IFTP = IFCNTP(IEQ)
        FNNAM1(IFTP) = ANS
      ELSE
        CALL BLNKLN
        CALL WRTSTR('  Bad entry please try again...')
        GOTO 1400
      ENDIF
C
C
      RETURN
      END
C
C
C>>>>>
C
C
C

```



```

C
  IF (DE.EQ.PATH(CNT,1)) THEN
    A = 2
    B = COUNT-2
    C = 1
  ELSE IF (DE.EQ.PATH(CNT,COUNT)) THEN
    A = COUNT-1
    B = 3
    C = -1
  ENDIF

C
C
  DO 100 I = A,B,C
C
C...FIND TYPE OF NEXT ELEMENT IN TRACE
C
  CE = PATH(CNT,I)
  NTYPE = NBXTP(CE)(2:2)

C
C
  IF ((NTYPE.EQ.'T').OR.(NTYPE.EQ.'G')) THEN
C
C...CHECK NUMBER OF CONNECTORS
C
    I1 = IBDPTR(CE)
    I2 = IBDPTR(CE+1)-1
    IF (I1.EQ.I2) THEN
C
C...IF ONLY ONE CONNECTOR EXISTS, DELETE CE AND ITS CONNECTOR
C
      CALL DELND(CE)
    ENDIF

C
C
    ELSE IF ((NTYPE.EQ.'0').OR.(NTYPE.EQ.'1')) THEN
C
C...IDENTIFY NUMBER OF CONNECTORS
C
      I1 = IBDPTR(CE)
      I2 = IBDPTR(CE+1)-1
C
C...IF ONLY ONE CONNECTOR EXISTS, DELETE CE AND ITS CONNECTOR
C

```

[illegible]



EXTERNAL GRPROC, MENSET, BLNKLN, WRTSTR, PROMPT, GETANS  
 EXTERNAL LSTLB1, LSTEQS, SETFCN, RHFILE, INVOPT, DEFEQS  
 EXTERNAL DMPFLG, REDOEQ, GOON, LSTDQS, PARDRV

C

C\*\*\*REASSIGN\*\*\*\*\*

C

CALL MENSET(.TRUE.)  
 FULL=MMFULL

C

C...Check on causality status first...

C

BGPART= NEL.GT.0  
 IF (BGPART) THEN  
 IF ((.NOT.E7SFLG(2)).OR.(.NOT.E7CFLG(2))) THEN

C

C...Assign auto causality to the entire graph...

C

DO 20 I= 3,5  
 20 E7CFLG(I)=.FALSE.  
 AUTO=.TRUE.  
 CALL GRPROC( AUTO,OKAY,CHANGE )  
 E7CFLG(2)= OKAY  
 IF (OKAY) THEN  
 E7SFLG(2)=.TRUE.  
 ELSE

C

C...Let the user try now...

C

CALL BLNKLN  
 CALL WRTSTR(' Automatic causality assignment produced')  
 CALL WRTSTR(' a system graph with derivative causality.')  
 AUTO = .FALSE.  
 CALL GRPROC( AUTO,OKAY,CHANGE )  
 E7CFLG(2)= OKAY  
 IF (OKAY) THEN  
 E7SFLG(2)=.TRUE.  
 ENDIF  
 ENDIF  
 ENDIF  
 ELSE  
 E7SFLG(2)=.TRUE.  
 E7CFLG(2)=.TRUE.  
 ENDIF

C IF (DBGFLG) CALL DMPFLG('FUNCD2')

C

C...Check on equation status next...

C

IF (.NOT.E7SFLG(3)) THEN

C

C...Set all node equations to their default status...

C

E7CFLG(4)=.FALSE.

E7CFLG(5)=.FALSE.

I= 1

CALL DEFEQS( I,OKAY )

E7SFLG(3)= OKAY

CALL DEFEQS( I,OKAY )

E7SFLG(3)= OKAY

E7CFLG(3)= OKAY

ELSE

IF (.NOT.E7CFLG(3)) THEN

C

C...Recover the equations to the maximum extent possible...

C

E7CFLG(4)=.FALSE.

E7CFLG(5)=.FALSE.

CALL REDOEQ( OKAY )

E7CFLG(3)= OKAY

ENDIF

ENDIF

C

C

RETURN

END

C

C

C>>>>

[EOB]

## LIST OF REFERENCES

## LIST OF REFERENCES

1. Rosenberg, R. and Karnopp D. An Introduction to Physical System Dynamics. 1st edition, New York, New York, McGraw-Hill, 1983.
2. ROSENCODE Associates, Inc. The ENPORT Reference Manual, ROSENCODE Associates, Inc., Lansing, MI, 1990, Vol. 7.3.1.
3. Emery, S. "Roller Coaster Analysis." Michigan State University, Department of Mechanical Engineering, ME 851 Final Project, Dec. 1989.
4. Tarokh, M. "Simulation analysis of dynamics and decentralized control of robot manipulators." Technical Article, Simulation, October 1989, pp 169-176.
5. Zeid, A. and Chang D. "A Modular Computer Model for the Design of Vehicle Dynamics Control Systems." Vehicle System Dynamics, 18 (1989), pp. 201-221.
6. Zeid, A. "Bond Graph Modeling of Planar Mechanisms With Realistic Joint Effects." Journal of Dynamic Systems, Measurement, and Control, March 1989, Vol. 111, pp. 15-23.
7. Karnopp, D. and Margolis, D. "Analysis and Simulation of Planar Mechanism Systems Using Bond Graphs." Journal of Mechanical Design, April 1979, Vol. 101, pp. 187-191.