This is to certify that the

thesis entitled

STATE FORMULATION OF LARGE-SCALE LINEAR
TIME-INVARIANT BOND GRAPH MODELS

presented by

Benjamin Moultrie

has been accepted towards fulfillment
of the requirements for

Ph.D. degree in Mechanical Engineering

RC Rosenberg

Major professor

Date 5/3/79

O-7639

STATE FORMULATION OF LARGE-SCALE LINEAR TIME-INVARIANT BOND GRAPH MODELS

By

Benjamin Moultrie

A DISSERTATION

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

DOCTOR OF PHILOSOPHY

Department of Mechanical Engineering

1979

ABSTRACT

STATE FORMULATION OF LARGE-SCALE LINEAR
TIME-INVARIANT BOND GRAPH MODELS

by

Benjamin Moultrie

In this dissertation, topology-based equations are developed which give the effort-flow basis order for the juncture structure transformation of an arbitrary weighted junction structure. These equations are used to develop three upper bounds for the numberof distinct sets of port variables which can be used to specify weighted junction structure input-output relations. Each successive bound is shown to be numerically smaller and computationally more expensive than its predecessor. Examples are given which use the established bounds.

Also, a causal assignment procedure is specified which simplifies the state model formulation process for linear time-invariant bond graphs. This result is used to develop an efficient computer implemented state model formulator for linear time-invariant bond graphs. The key storage features and novel matrix manipulation procedures of this state model formulator are explored, and key computer subroutines are

given. The enhanced performance characteristics of this formulator are validated by computer test results.

ACKNOWLEDGMENTS

With apologies to Buffon:

For in those few men whose head is steady, whose heart is compassionate, and whose sense is exquisite - there is substance, thought and reason; there is the art of speaking to the mind.

Thanks Doc.

TABLE OF CONTENTS

Chapter Page

Appendix Page

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

A_1 - number of external 1-junctions

A_0 - number of external 0-junctions

B_1 - sum of the degrees of the 1-junctions

B_0 - sum of the degrees of the 0-junctions

E - number of independent effort variables

F - number of independent flow variables

N_1 - number of 1-junctions

N_0 - number of 0-junctions

N_B - number of bonds

N_E - number of independent (port) effort inputs

N_F - number of independent (port) flow inputs

N_I - number of internal bonds

N_P - number of port bonds

N_T - number of TF multiports

P_1 - number of port bonds incident to 1-junctions

P_0 - number of port bonds incident to 0-junctions

Note that $A_1 \leq P_0$ and $A_0 \leq P_0$.

I. INTRODUCTION

1.1 Background

Although very few dynamic systems are truly linear, they are frequently adequately approximated by linear models. This serves to decrease the system analysis effort, while yielding acceptable design results. In general, as dynamic systems increase in size and complexity, the creation and analysis of even linear models becomes an arduous and tedious task. Thus, the development of new and more powerful tools which can be used in the modeling and analysis process is necessarily a continuing effort.

Digital computers and linear simulation programs are proving to be essential tools in the designing of dynamic systems. The simulation of a system can be described as a procedure which begins with the development of a system model and continues with the "processing" of the model in order to infer the performance characteristics of the system under study [1].

In many engineering activities, the word "model" has come to mean the description of a system in mathematical terms. Historically, the techniques used to obtain mathematical models have been given prominence in accordance with the engineering discipline of the system analyst. Consequently, numerous digital simulation programs have been developed which require as input a particular energy domain

description of the system to be studied. In order to be used for multiple energy domain systems, such programs generally require the development of system element analogies. This need to "reason by analogy" tends to make single-energy-domain simulation programs undesirable for the analysis and simulation of complex multiple-energy-domain systems. Representative examples of single-energy-domain digital simulation programs are NET-2 [2] and SPICE [3] for electrical circuits and systems, and DRAM [4] and MEDUSA [5] for dynamic mechanical systems. Additional examples of such specialized programs can be found in a 1975 volume of Shock and Vibration [6].

In contrast to the number of developed single-energy-domain digital simulation programs, comparatively few simulation programs have been developed which accept as input a multiple-energy-domain description of a system. In developing multiple-energy-domain simulation programs, two approaches may be used.

The first approach is to develop a program which accepts more than one single-energy-domain element type as input. The SUPER*SCEPTRE program is a digital simulation program which uses this approach [7]. It accepts electrical system and mechanical system element types as inputs. In general, a desire for easy program implementation as well as constraints on program size limits the number of single-energy-domain element types which can be included as admissible inputs. Therefore, the value of this approach is limited.

The second approach is to develop a program based on a process which describes many physical systems and uses a small number of basic elements. The ENPORT-4 program uses this approach [8].

ENPORT-4 is a digital simulation program which accepts as input a linear, time-invariant, multiple-energy-domain description of a physical system, and serves as a state model formulator-analyzer. The foundation for ENPORT-4 is the generalized energy-based modeling technique of bond graphs. This provides ENPORT-4 with its very desirable ability to treat all energy domains uniformly. An undesirable feature of ENPORT-4 is its generally large processing time requirement. This is largely due to ENPORT-4's inefficient state model formulator. This issue is addressed in this research.

1.2 The Status Of Bond Graph Theory And Practice

Traditionally, for physical systems, the concepts of energy and energy-flow have been important considerations in the development of system models. In 1960, Paynter introduced a novel multiport approach by which a system's energy characteristics can be explicitly exhibited and a system state model can be systematically obtained [9]. This is the method of bond graphs. A history of the process leading to its development is contained in Karnopp and Rosenberg [10]. The current situation regarding theory and practice is indicated by a bond graph bibliography compiled by Gebben [11].

The majority of bond graph oriented research has been

in the area of applicability. The concerns of this dissertation are in the areas of theory and methodology rather than applicability.

Works which relate directly to this research are the junction structure studies by Nobuhide [12], Ort and Martens [13], and Perelson [14]. Each study has as a prime objective the establishment of conditions for the solvability of junction structure algebraic loops.

In the Nobuhide study, a matrix representation of the junction structure is developed with the aid of junction structure power information, but without the aid of causal information. Specific blocks of this matrix are then manipulated to establish an algorithm by which loop solvability can be determined for a restricted class of junction structures.

Ort and Martens rely on junction structure power information and causal information to develop a junction structure matrix representation which is used to establish loop solvability conditions. They also establish an orthogonality relationship between particular types of junction structure equations.

Perelson also relies on junction structure power information and causal information to establish loop solvability conditions. His results are more extensive than those achieved by Ort and Martens. They are used in the state model formulation procedure which is discussed in Chapter IV.

In the process of deriving loop solvability conditions,

each of the above works partially realizes the "basis order rules" developed in this research. The broader results presented here are complete and developed without the aid of junction structure power information or causal information.

An important work to be noted is the state model formulation algorithm by Rosenberg [15]. This algorithm is the foundation for the state model formulator in Chapter IV.

An additional work to be noted is the publication by van Dixhoorn which demonstrates how block-diagram-oriented digital simulation languages can be adapted for the interactive simulation of nonlinear bond graphs on minicomputers [16]. For linear or nonlinear bond graphs, the procedure described requires that the analyst augment the graph in terms of power and causality, resolve algebraic loops and uncertainties, and define specific element blocks. For the analyst, this graph analysis problem increases in difficulty as the bond graph increases in size and complexity. Although it processes only linear-time invariant bond graphs, ENPORT-4 does not require the analyst to analyze the graph. For this reason and because of the great utility of linear simulation programs, methods for increasing the efficiency of ENPORT-4 were of concern in this research.

1.3 Research Highlights And Dissertation Organization

The results of this research can be divided into the categories of (1) bond graph theory and (2) bond graph methodology. The first category encompasses the results

in Chapters II and III, and the second category encompasses the results in Chapter IV.

The key research results can be highlighted by delineating the major aspects of each chapter. The major aspects of Chapter II are

1) the graph theoretic development of junction structures (from the node set {0,1,TF,GY}) and introduction to standard junction structure concepts;
2) the introduction of new results for weighted junction structures (the basis order rules and an upper bound for the number of distinct bases for a junction structure transformation);
3) the precise defining of causal concepts;
4) the precise defining of causal complexes in bond graph terms.

The major aspects of Chapter III are

1) the specification of SSCAP (the standard sequential causality assignment procedure);
2) the verification of the simplifying influence which SSCAP has on the form of the reduced junction matrix.

The major aspects of Chapter IV are

1) the presentation of a new sparse-matrix-based state model formulator which implements some results achieved in Chapters II and III, and employs the novel sparse matrix inversion subroutine INPRD (INverse-PRoDuct);
2) the results of a performance study (for the new state

model formulator) which considers computer storage and processing time requirements for three test bond graphs. The research results are summarized in Chapter V, and all proofs and computer subroutines are in the appendices for the sake of brevity.

II. BOND GRAPH JUNCTION STRUCTURES

Bond graphs are graphs whose nodes are called multiports and whose edges are called bonds. The terms "node" and "multiport" will be used interchangeably in the subsequent development. The principal categories of bond graph multiports are shown in Figure 2.1. A discussion on the field multiports (sources, storages, and dissipators) is deferred until Chapter III. This chapter develops standard bond graph junction structure terminology and notation, and extends such where necessary. In this development, graph theory concepts and terminology are employed. Since there is a broad range of terminology in the graph theory literature, textbooks by Busacker and Saaty [17] and Harary [18] are cited as references.

2.1 Junction Structure Terminology And Notation

A junction structure is comprised of bond graph multiports which represent the features of a dynamic system which neither store energy, supply power, or dissipate power. In standard bond graph notation, these multiports are represented by the elements of the set {0,1,TF,GY}. The multiport "0" is called "zero" of "0-junction" (zero junction). The multiport "1" is called a "one" or "1-junction" (one-junction). The "TF" multiport is called a "transformer", and the "GY" multiport is called a "gyrator". As a mathematical

convenience, the node "EN" is introduced and will be added to the set {0,1,TF,GY}; it will be called the "environment node". The elements of the set {0,1,TF,GY,EN} will be generally referred to as "junction structure nodes".

Definition: A bond is an unordered pair $b=(v,w)=(w,v)$ where v and w are distinct junction structure nodes.

This definition restricts self-loops from being classified as bonds, e.g., (v,v) is not a bond. For the purpose of model analysis, it is useful to distinguish between types of bonds.

Definition: A port bond is a bond which is incident to an EN-node.

A port bond is also called an "external" bond.

Definition: An internal bond is a bond which is not incident to an EN-node.

The concept of a junction structure can now be defined.

Definition: A junction structure G is a finite set, V_G, of junction structure nodes together with a set of bonds, X_G, such that if $b \in X_G$, then there exist $v,w \in V_G$ so that $b=(v,w)$.

The shorthand notation "JS" will substitute frequently for the phrase "junction structure" in the balance of this dissertation. Various types of JS's can be defined.

Definition: A standard junction structure is a JS in which (1) every junction (0-junction or 1-junction) has degree greater than or equal to two, (2)

every TF-node and GY-node has degree equal to two, and (3) every EN-node has degree equal to one and is adjacent to a nonenvironment JS node.

Hereafter, unless otherwise stated, all JS's will be considered to be standard.

Definition: A simple junction structure is a JS which contains only elements from the set {0,1,EN} and their incident bonds.

A simple junction structure will be denoted by "SJS". Two subclasses of SJS's are "tripartite" and "proper".

Definition: A tripartite simple junction structure is a SJS in which every internal bond b has the form b=(1,0), and every external bond d has the form d=(1,EN) or d=(0,EN).

Definition: A proper simple junction structure is a SJS which is tripartite and standard.

Definition: A weighted junction structure is a JS which contains only elements from the set {0,1,EN,TF} and their incident bonds.

A weighted junction structure will be denoted by "WJS". Examples of JS's are shown in Figure 2.2 Properties of JS's can be obtained from several publications [12-14, 19-24].

2.2 Power Orientation

Two conjugate variables or signals are associated with each bond in every JS [25]. These are known as an "effort" variable and a "flow" variable, and are denoted by the symbols

"e" and "f" respectively. The effort and flow variables are scalar functions of an independent variable, taken to be time.

Definition: The power associated with the bond b is the function given by

$$P(t) = e(t) \cdot f(t)$$

where e and f are the respective effort and flow variables of b.

The bond variables e and f are frequently called "power variables".

Definition: The power orientation of a bond b is the sense of direction (with respect to b's incident nodes) of b's power function.

The power orientation of a bond is indicated graphically by placing half of an arrow-head on the bond-end which is defined by the node to which the positive sense of power is directed. This graphical procedure is illustrated by example in Figure 2.3. The interpretation of the graphical procedure is identical to that illustrated for any pair of JS nodes.

2.3 Analytic Properties Of Junction Structure Nodes

Except for the EN-node, which is used as a bond terminator, each JS node algebraically constrains the effort and flow variables of its incident bonds. At a 0-junction, effort variables are identical and flow variables sum to zero; this is analogous to vertex relations in electrical circuit analysis. At a 1-junction, flow variables are

identical and effort variables sum to zero; this is analogous to loop relations in electrical circuit analysis. The algebraic signs in the variable constraint equations are determined by the power orientation associated with each bond [25]. A detail description of all JS node variable relations is contained in Table 1.

2.4 Basis Order Rules

In studying systems using bond graphs, it is important to develop a well-defined analytic input-output relation for the JS. This relation is referred to as the "JS transformation". Conditions for its existence have been developed by Nobuhide [12], Ort and Martens [13], Perelson [14], and Rosenberg and Andry [19,20].

Before the JS transformation can be analytically defined, a basis for it must be specified. Algebraic equations which determine the number and types of basis variables suitable for expressing the JS transformation for a WJS have resulted from this research. These WJS topology-based equations are called the "basis order rules". Employing the given nomenclature, the basis order rules are presented here as Theorem 1 and Theorem 2 for the cases of SJS's and WJS's respectively. Also, associated corollaries are given.

Theorem 1: Every standard SJS satisfies the relations

$$\text{(i)} \quad E = N_B + N_0 - B_0 - N_1$$

and

$$\text{(ii)} \quad F = N_B + N_1 - B_1 - N_0 .$$

Corollary 1.1: Every proper SJS satisfies the relations

(i) $E=N_0+P_1-N_1$

and

(ii) $F=N_1+P_0-N_0$.

Corollary 1.2: Every standard SJS satisfies the relations

(i) $N_E=N_B+N_0-B_0-N_1$

and

(ii) $N_F=N_B+N_1-B_1-N_0$.

Theorem 2: Every standard WJS satisfies the relations

(i) $E=N_B+N_0-B_0-N_1-N_T$

and

(ii) $F=N_B+N_1-B_1-N_0-N_T$

Corollary 2.1: Every standard WJS satisfies the relations

(i) $N_E=N_B+N_0-B_0-N_1-N_T$

and

(ii) $N_F=N_B+N_1-B_1-N_0-N_T$.

The above theorems and corollaries are formally developed in Appendix A.

In Appendix B, the basis order rules are used to develop three predictors of the number of distinct basis variable sets which a WJS transformation may possess. The most accurate of these predictors is given here in Theorem 3.

Theorem 3: The number of distinct basis variable sets for a WJS transformation is bounded above by

$$U_2 = \sum_{e=L_E}^{M_E} \binom{A_0}{e} \binom{A_1}{N_F - P_0 + e} = \sum_{f=L_F}^{M_F} \binom{A_0}{N_E - P_1 + f} \binom{A_1}{f}$$

where $M_E = \min(N_E, A_0)$, $L_E = \max(0, N_E - P_1)$, $M_F = \min(N_F, A_1)$, and $L_F = \max(0, N_F - P_0)$.

Application of the basis order rules yields the number of effort inputs and flow inputs which can be independently specified for a given WJS. Rosenberg has shown that a JS which does not contain an "essential" gyrator is equivalent to a WJS by a transformation process [24]. Thus, the basis order rules can be extended to any JS which does not contain an essential gyrator.

Consider the application of Corollary 1.2 to an arbitrary proper SJS cycle, say C, with incident port bonds. Then the following interpretation (which employs the concept of causality) of the resulting numbers N_E and N_F is based on the publications of Ort and Martens [13] and Perelson [14]; although their results are for proper SJS cycles, the results can be readily extended to include a union of WJS cycles.

The following notation is used by Perelson for cycle quantities:

$N_f \equiv$ the number of linearly independent flow equations;

$J_+ \equiv$ the number of junction causal port bonds;

$J_- \equiv$ the number of environment causal port bonds.

Note that $J_+ + J_- = N_p$. From the above publications, $(N_f)_{max} = J_+ + N_I$ and $(N_f)_{max} = N_0 + B_1 - N_1$, where C is an

n-port if and only if $N_f=(N_f)_{max}$, i.e., the JS transformation for C exists if and only if $N_f=(N_f)_{max}$.

Suppose $N_F<0$. Then

$$N_F<0 \rightarrow N_0+B_1-N_1>N_B \rightarrow N_f<(N_f)_{max}=N_0+B_1-N_1,$$

since $N_f \leq N_B$. Therefore, $N_f<0$ implies that C is not an n-port. Suppose $N_F=0$ where $N_f=(N_f)_{max}$. Then

$$N_F=0 \rightarrow 0=N_B-(N_0+B_1-N_1)=N_B-(J_++N_I)$$

$$=(P_1+P_0)-J_+=N_P-J_+ \rightarrow J_+=N_P \rightarrow J_-=0$$

Therefore, $N_F=0$ implies that C is not an n-port (this conclusion represents an exclusion of a case where C acts like a source of flow to its environment [26]). Similarly, $N_E \leq 0$ implies that C is not an n-port. Thus, in order for C to be an n-port, it is necessary that $0<N_E$, $N_F<N_P$. In this context, the basis order rules can be applied as a preliminary test to determine if a bond graph model may have a "physical realization".

2.5 Causal Concepts

In addition to power orientations, a JS can be further augmented by associating with each bond an "input-output" notion of directed flow and effort variables; this is the concept of "causality" [25].

Definition: The causal orientation of a bond b is the sense of causality associated with b.

A causally oriented bond identifies its effort variables as an input to one incident node, and identifies its flow

variable as an input to the remaining incident node; thus, it provides bi-directional input (and output) information.

Definition: The causal form of a JS is the assemblage of the causal orientations of its bonds.

2.5.1 Causal Assignment

"Assigning causality" to a JS is the graphical process of causally orienting its bonds. A bond's causal orientation is indicated graphically by placing a short stroke (called a "causal stroke") perpendicular to the bond at the bond-end incident to the node of effort variable input. Thus, the bond-end without the causal stroke is incident to the node of flow variable input.

2.5.2 Causal Completeness

Definition: A bond is acausal if it has not been causally oriented.

Definition: A JS is acausal if all of its bonds are acausal.

Definition: A JS node is causally complete if none of its incident bonds are acausal.

Definition: A JS is causally complete if all of its nodes are causally complete.

2.5.3 Causal Consistency

Definition: The causal form of a node is consistent if it does not violate any of the constraints defined by the node's algebraic properties.

Definition: The causal form of a node is inconsistent or in conflict if it is not consistent.

Definition: The causal form of a JS is consistent if the the causal form of each of its nodes is consistent.

Definition: The causal form of a JS in inconsistent or in conflict if it is not consistent.

The consistent causal forms for JS nodes are given in Figure 2.4. Examples of node inconsistent causal forms are given in Figure 2.5.

Remark 2.1: The causal form of a node is inconsistent if and only if the number of flow inputs (outputs) or effort inputs (outputs) specified by it is in violation of the node's algebraic properties.

2.5.4 Causal Extension

Prior to the discussion on the extension of causality, the preliminary concept of "causal implication" is introduced.

Definition: Given a node v and incident bond b, b has strong causal implication (with respect to v) if its causal orientation specifies an effort input to v where v is a 0-junction, or a flow input to v where v is a 1-junction, or either input to a TF-node or GY-node.

Definition: A causally oriented bond has weak causal implication (with respect to a given incident node) if it does not have strong causal implication.

The concept of causal implication proves to be useful when considering causal extension procedures.

Remark 2.2: For any node having only acausal incident bonds, if any bond is given a causal orientation with strong causal implication, or if all bonds but one are given causal orientations with weak causal implication, then the JS node can be causally completed in a consistent manner by observing the node's algebraic or causal assignment properties.

The causal extension concept can now be introduced. Consider a JS node which is causally completed by an effort or a flow input. This node defines an input to each of its adjacent nodes. In turn, these inputs contribute to the causal completion of additional nodes, and may result in additional causally complete nodes. Thus, a causally complete node results in the propagation of causal information.

Definition: The extension of causality or causal extension process is the propagation of causal information in accordance with the algebraic properties of the JS nodes.

It should be emphasized that the causal extension process implies the propagation of causal information until no additional nodes can be causally completed using the effort and flow inputs which are known.

Remark 2.1 and the causal assignment properties of JS nodes reveal two properties of the causal extension process as applied to a causal orientation of an external bond b.

Property 2.1: If the initial causal orientation and subsequent causal extension of b results in a node causal conflict, then the reversal of b's causal orientation (followed by causal extension) does not yield a node causal conflict.

Property 2.2: If the initial causal orientation and subsequent causal extension of b results in a node causal conflict, then sufficient prior input information was available to determine the variables associated with b in an implicit manner.

Property 2.1 is proven in Appendix C. Property 2.2 follows directly from Remark 2.1 and Property 2.1.

Additional properties of the causal extension process can also be given; each is stated without remark.

Property 2.3: In JS trees the extension of a causal orientation never yields causally inconsistent nodes, since there is a unique path between distinct nodes in a tree graph.

Property 2.4: If the causal extension process terminates at a nonenvironment JS node without causally completing the given JS node, then the node is a junction (0 or 1) of degree greater than two; it has at least two incident acausal bonds after the termination of the causal

extension process.

Henceforth, unless otherwise stated, all causal orientations will be assumed to be followed by the causal extension process.

2.6 Causal Complexes

A measure of the versatility of a simulation program is its ability to identify and resolve algebraic loops. When using a bond graph as a modeling tool, algebraic loops appear graphically as "causal complexes" in the junction structure.

Consider an arbitrary JS, say G. Suppose that G has a complete and consistent causal form which can be realized by a sequential causality assignment process in which the causal orientation of external bonds is given priority. Assume the causal orientation (by the sequential process) of the external bonds of G does not causally complete G.

Definition: A causal complex in G is a set, C, of acausal bonds and their incident nodes such that every pair of distinct nodes in C is joined by a path in C.

Definition: A causal complex is maximal if it is not contained in any distinct causal complex.

In bond graph literature, a causal complex is called a "causal loop" if it is a cycle. For the sake of brevity, all causal complexes will be considered to be maximal. An example of a causal complex is given in Figure 2.6.

In mathematical terms, the occurrence of a causal complex

represents an implicit relationship between JS variable sets, i.e., an algebraic loop.

Definition: A causal complex is solvable if it can be represented by a nonsingular matrix of the form (I-L) where I is the identity matrix and L is a matrix determined by the algebraic constraint equations of the nodes in the complex.

The solvability of causal complexes has been studied by several workers [12-14,19,20]. It has been shown by Ort and Martens that the solvability of causal complexes is a necessary and sufficient condition for the existence of the JS transformation. This condition is employed in the state model formulation procedure discussed in Chapter IV.

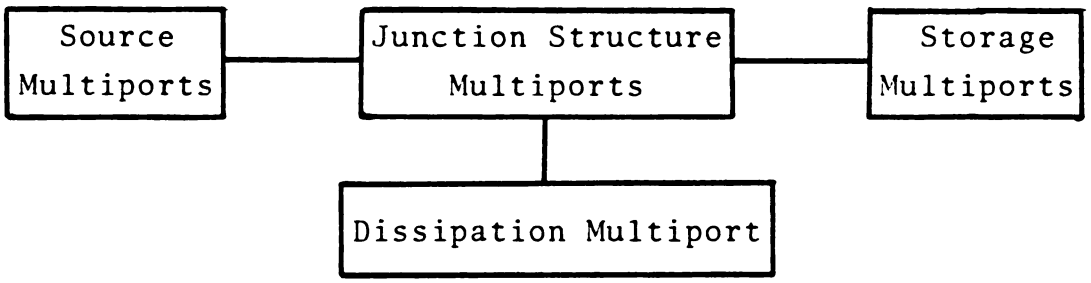


Figure 2.1. Generic categories of bond graph multiports.

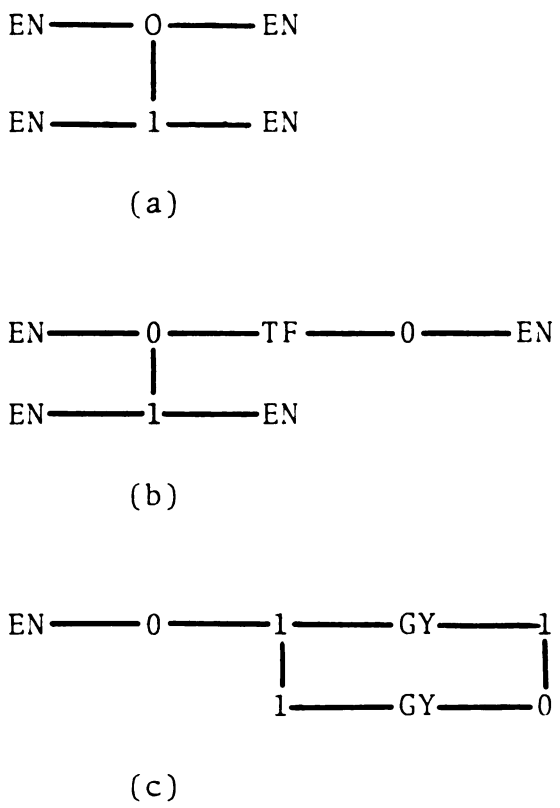
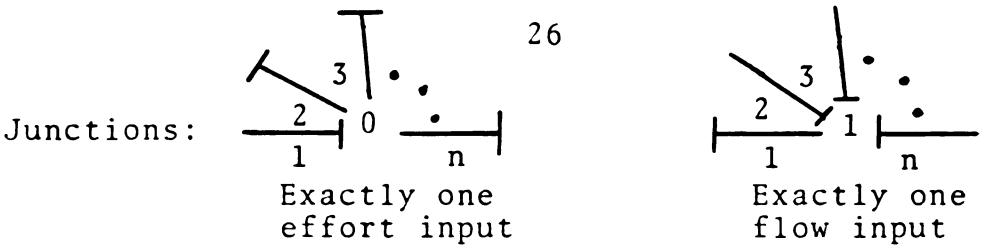


Figure 2.2. Junction structure examples. (a) SJS tree. (b) WJS tree. (c) JS containing a cycle.

Figure 2.3. Example of power orientation. Positive sense of power is directed from the 1-junction to the 0-junction.

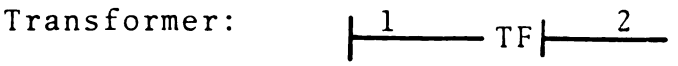
Table 1. Analytic Properties of Junction Structure Multiports.

Multiport	Degree	Properties
1	$m \geq 2$	(i) Admits exactly one flow variable as an input. (ii) If f_k is the single flow variable input, then $f_i = f_k$, where $i=1, 2, \ldots, m$ and $i \neq k$. (iii) If f_k is the flow variable input, then $e_k = \sum_{\substack{i=1 \\ i \neq k}}^{m} \sigma_i e_i$ where $\sigma_i = \pm 1$ depending upon the power orientation of bond b_i.
0	$m \geq 2$	(i) Admits exactly one effort variable as an input. (ii) If e_k is the single effort variable input, then $e_i = e_k$, where $i=1, 2, \ldots, m$ and $i \neq k$. (iii) If e_k is the effort variable input, then $f_k = \sum_{\substack{i=1 \\ i \neq k}}^{m} \sigma_i f_i$, where $\sigma_i = \pm 1$ depending upon the power orientation of bond b_i.
TF	$m=2$	There exists a single-valued function Ξ such that $e_1 = e_2 \Xi$ and $f_2 = f_1 \Xi$.
GY	$m=2$	There exists a single-valued function ψ such that $e_1 = f_2 \psi$ and $e_2 = f_1 \psi$.
EN	$m=1$	No analytic properties; serves as a bond terminator.

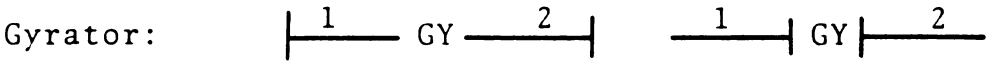


Exactly one effort input

Exactly one flow input



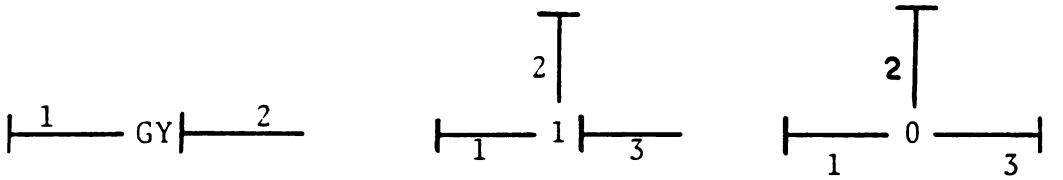
Exactly one flow input and one effort input.



Exactly two flow inputs

Exactly two effort inputs

Figure 2.4. All consistent causal forms for junction structure multiports.



Defines one flow input and one effort input

Defines two flow inputs

Does not define an effort input

Figure 2.5. Examples of junction structure multiports with inconsistent causal forms.

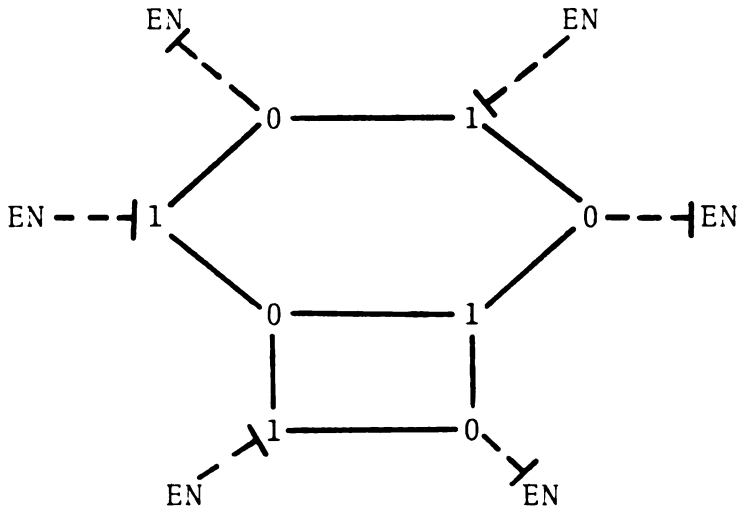


Figure 2.6. Example of a causal complex (shown in solid lines).

III. STATE MODEL FORMULATION FOR LINEAR TIME-INVARIANT BOND GRAPHS

Bond graph digital simulation programs have proven to be particularly important contributions to the arsenal of the design engineer. Among their many attractive features, such programs treat all energy domains uniformly. This feature significantly reduces the design effort for multiple-energy-domain systems.

Currently, there are two computer procedures for bond graph simulation which have appeared in bond graph literature. The first procedure is the ENPORT program [8]. ENPORT is a digital program for the simulation of linear time-invariant bond graphs. It is a state model formulator-analyzer. Presently, it is in use at several academic and industrial installations in various localized versions. The second procedure is based on the adaptation of a block-diagram-oriented digital simulation language so that it can be used to interactively simulate nonlinear bond graphs. It is described in a publication by van Dixhoorn [16]. The procedure requires the system analyst to augment the graph in terms of power and causality, resolve algebraic loops and uncertainties, and define specific element blocks. For the analyst, this graph analysis process increases in difficulty as the bond graph increases in size and complexity. The ENPORT program does not require the analyst to analyze the graph. For this reason

and because of the great utility of linear time-invariant simulation programs, only the ENPORT program will be considered here.

In the computer simulation of systems, the fundamental phase is the development of a mathematical model. For bond graph modeled systems, an algorithm for the formulation of a mathematical model (in state-space form) has been developed by Rosenberg [15]. Prior to the discussion of the algorithm for linear time-invariant bond graphs, it is necessary to introduce the bond graph field multiports, and define a causality assignment procedure.

3.1 Field Multiports

Through its multiports and bonds, a bond graph model provides a graphical representation of a physical system's power exchanges. The multiports used in the modeling process are contained in the set {0, 1, TF, GY, SE, SF, C, I, R} where SE, SF, C, I, and R topologically replace the environment node in the formation of external bonds. The multiports "SE" and "SF" represent independent power suppliers; they are called "effort source" and "flow source" respectively. The multiports "C" and "I" represent energy storages; they are called "capacitance" and "inertance" respectively. The mulport "R" is called a "resistance" and represents power dissipation. It will be assumed that no two field multiports are adjacent in bond graph models. An analytic description of each field multiport is contained in Table 2.

Frequently, an external bond is referred to by the type of field multiport to which it is incident, e.g., an external bond which is incident to a storage multiport is referred to as a "storage" bond. This convention will be assumed here.

Examples of consistent causal forms for each field multiport are given in Figure 3.1. The only consistent causal orientations for source bonds are those illustrated in Figure 3.1. In general, any causal form is consistent for the multiports C, I, and R.

Definition: The causal orientation of a storage bond is integral if it specifies a flow input to a C-multiport or an effort input to an I-multiport.

Definition: The causal orientation of a storage bond is derivative if it is not integral.

In the state model formulation algorithm, only those storage bonds with an integral causal orientation are identified with system state variables; it is these variables which contribute to a basis for the JS transformation. See Karnopp and Rosenberg for a more detailed discussion [25]. Now that the set of bond graph multiports has been completed, a causality assignment procedure can be formalized.

3.2 The Standard Sequential Causality Assignment Procedure

The manner in which a bond graph is causally completed can influence the selection of system state variables and the process by which a system state model is derived. A systematic causal completion procedure is given in Figure 3.2 which simplifies the state model formulation procedure. It will be referred to as the "standard sequential causal assignment procedure" (SSCAP) and may be found in Karnopp and Rosenberg [25]. The diagram in Figure 3.2 assumes that each causal orientation is followed by the causal extension process, and the reversal of a bond's causal orientation is preceded by the restoration of the bond graph to the causal form possessed prior to the initial causal orientation of that bond.

By its design, SSCAP assures that energy related variables are given priority over other system physical variables for consideration as system state variables. Another important feature of SSCAP is the significant impact it has on the form of the JS transformation which is often referred to as the "reduced junction matrix". The reduced junction matrix equation and associated vector definitions are given in Figure 3.3. It is shown in Appendix D that the S_{22}, S_{23}, and S_{32} blocks of the reduced junction matrix are zero as a consequence of Property 2.2 and SSCAP. This extends and validates conjectures made by Rosenberg [15]. These results simplify the state model formulation algorithm for linear time-invariant systems.

3.3 The State Model Formulation Algorithm

The state model formulation stage is the central phase in the computer simulation of bond graphs. The current formulation procedure for the ENPORT program is based on the state model formulation algorithm developed by Rosenberg [15]. This algorithm also serves as the foundation for the formulation procedure discussed in Chapter IV. Referring to the matrix equation in Figure 3.3, the state model formulation algorithm can be summarized as defining a procedure by which the reduced junction matrix equation can be resolved to an equation which expresses $\dot{\underline{X}}_i$ in terms of $\underline{X}_i$, $\underline{U}$, and $\dot{\underline{U}}$ where $\dot{\underline{X}}_i$ and $\dot{\underline{U}}$ are the time derivatives of $\underline{X}_i$ and $\underline{U}$ respectively.

Prior to the development of the reduced junction matrix it is necessary to construct a matrix from which all junction structure node equations can be obtained. This is accomplished with the aid of junction structure causal and power information, and by ordering the junction structure effort variables and flow variables. The resulting matrix is called the "junction" matrix, its implied equation is given in Figure 3.4 with associated vector definitions. The reduced junction matrix is obtained by expressing $\underline{V}_{out}$ in terms of $\underline{V}_{in}$, i.e., eliminating $\underline{V}_{int}$ from the junction matrix equation. This process is illustrated in Figure 3.5. The matrix $[S_1 + S_2 (I - S_4)^{-1} S_3]$ is then partitioned to give to the form of the reduced junction matrix illustrated in Figure 3.3.

The reduced junction equation can be expanded into the following equations,

$$\underline{\dot{X}}_i = S_{11}\underline{Z}_i + S_{12}\underline{\dot{X}}_d + S_{13}\underline{D}_{out} + S_{14}\underline{U} \tag{4}$$

$$\underline{Z}_d = S_{21}\underline{Z}_i + S_{24}\underline{U} \tag{5}$$

$$\underline{D}_{in} = S_{31}\underline{Z}_i + S_{33}\underline{D}_{out} + S_{34}U \tag{6}$$

where the expression for $\underline{V}$ is not of interest. Coupled to the above equations are relations for the field multiports which are obtained from parameter and causal information,

$$\underline{Z}_i = F_{11}\underline{X}_i + F_{12}\underline{X}_d \tag{7}$$

$$\underline{Z}_d = F_{12}\underline{X}_i + F_{22}\underline{X}_d \tag{8}$$

$$\underline{D}_{out} = LD_{in} \tag{9}$$

The first step in the process of reducing these equations to state form is to replace $\underline{Z}_i$ in (4), (5), and (6) by (7). Also, replace $\underline{Z}_d$ in (5) by (8). Then collecting terms in (4) and (6), and solving for $\underline{X}_d$ in (5) yields

$$\underline{\dot{X}}_i = S_{11}F_{11}\underline{X}_i + S_{11}F_{12}\underline{X}_d + S_{12}\underline{\dot{X}}_d + S_{13}\underline{D}_{out} + S_{14}\underline{U} \tag{10}$$

$$\underline{X}_d = T_1\underline{X}_i + T_2\underline{U} \tag{11}$$

$$\underline{D}_{in} = S_{31}F_{11}\underline{X}_i + S_{31}F_{12}\underline{X}_d + S_{33}\underline{D}_{out} + S_{34}\underline{U} \tag{12}$$

where

$$T_1 = (F_{22}-S_{21}F_{12})^{-1}(S_{21}F_{11}-F_{21}) \tag{13}$$

and

$$T_2 = (F_{22}-S_{21}F_{12})^{-1}S_{24}. \tag{14}$$

Next replace $\underline{X}_d$ and $\underline{D}_{out}$ in (10) and (12) by (11) and (9) respectively. This yields the following equations for $\underline{X}_i$ and $\underline{D}_{in}$,

$$\dot{\underline{X}}_i = S_{11}(F_{11}+F_{12}T_1)\underline{X}_i+S_{12}\dot{\underline{X}}_d+S_{13}L\underline{D}_{in}+(S_{11}F_{12}T_2+S_{14})\underline{U} \quad (15)$$

$$\underline{D}_{in} = T_3\underline{X}_i + T_4U \quad (16)$$

where

$$T_3 = (I - S_{33}L)^{-1}(F_{11} + F_{12}T_1) \quad (17)$$

and

$$T_4 = (I - S_{33}L)^{-1}(S_{31}F_{12}T_2 + S_{34}). \quad (18)$$

For the final step, replace $\underline{D}_{in}$ in (15) by (16), and use (11) to eliminate $\dot{\underline{X}}_d$ in (15). Then solving for $\dot{\underline{X}}_i$ yields the system state model,

$$\dot{\underline{X}}_i = A\underline{X}_i + B\underline{U} + E\dot{\underline{U}} \quad (19)$$

where

$$A = (I-S_{12}T_1)^{-1}[S_{11}(F_{11}+F_{12}T_1)+S_{13}LT_3] \quad (20)$$

$$B = (I-S_{12}T_1)^{-1}(S_{11}F_{12}T_2+S_{13}LT_4+S_{14}) \quad (21)$$

and

$$E = (I-S_{12}T_1)^{-1}S_{12}T_2. \quad (22)$$

Some important aspects of the computer implemention of this state model formulation algorithm are explored in Chapter IV.

Table 2. Analytic Properties of Field Multiports (with exactly one incident bond)

Multiport	Properties
SE	Defines the associated effort variable as independent, i.e., $e=e(t)$.
SF	Defines the associated flow variable as independent, i.e., $f=f(t)$.
C	Defines the associated effort variable as $e=\Phi(q)$ for *$q(t)=q(t_0)+\int_{t_0}^{t} f(\lambda)d\lambda$ where f is the flow variable associated with C.
I	Defines the associated flow variable as $f=\Psi(q)$ for *$p(t)=p(t_0)+\int_{t_0}^{t} e(\lambda)d$ where e is the effort variable associated with I.
R	For the associated effort and flow variables, $\Xi(e,f)=0$.

*q is called a "generalized displacement", and p is called a "generalized momentum".

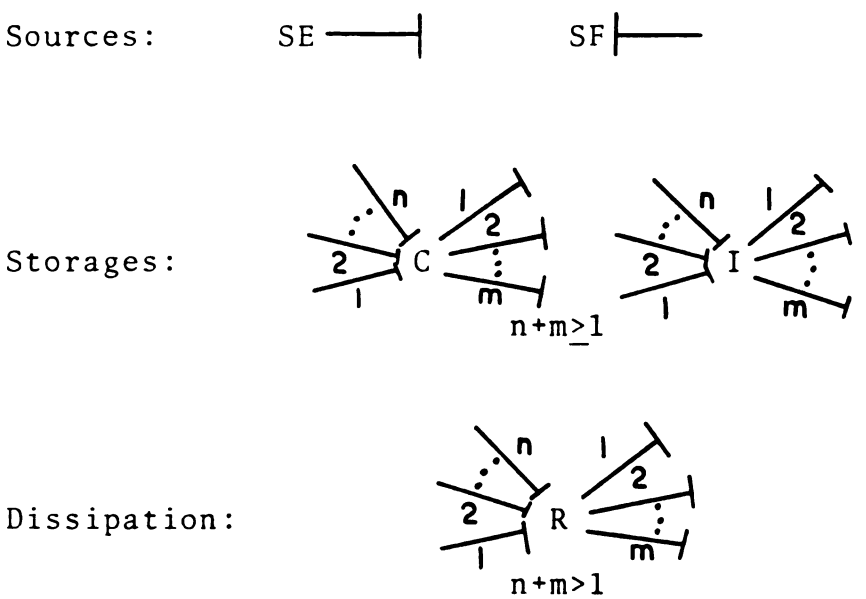


Figure 3.1. Consistent causal forms for field multiports.

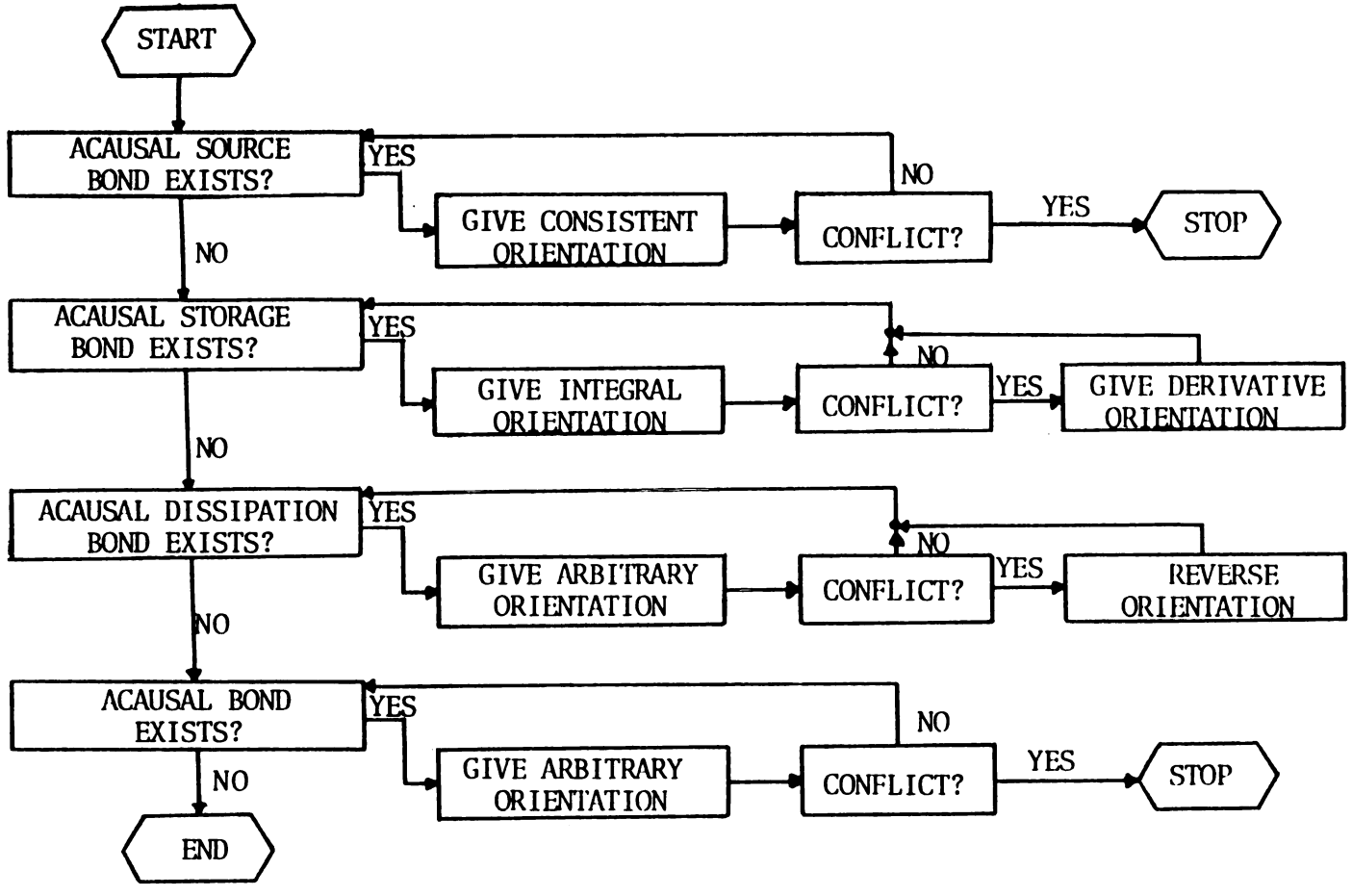


Figure 3.2. Diagram of the standard sequential causal assignment procedure

$$\begin{bmatrix} \dot{\underline{X}}_i \\ \underline{Z}_d \\ \underline{D}_{in} \\ \underline{V} \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} & S_{13} & S_{14} \\ S_{21} & S_{22} & S_{23} & S_{24} \\ S_{31} & S_{32} & S_{33} & S_{34} \\ S_{41} & S_{42} & S_{43} & S_{44} \end{bmatrix} \begin{bmatrix} \underline{Z}_i \\ \dot{\underline{X}}_d \\ \underline{D}_{out} \\ \underline{U} \end{bmatrix}$$

$\dot{\underline{X}}_i$ - vector of inputs to independent storage elements

$\underline{Z}_d$ - vector of inputs to dependent storage elements

$\underline{D}_{in}$ - vector of inputs to the dissipation elements

$\underline{V}$ - vector of inputs to the source elements

$\underline{Z}_i$ - vector of outputs from the independent storage elements

$\dot{\underline{X}}_d$ - vector of outputs from the dependent storage elements

$\underline{D}_{out}$ - vector of outputs from the dissipation elements

$\underline{U}$ - vector of outputs from the source elements

Figure 3.3. The reduced junction matrix equation.

$$\begin{bmatrix} \underline{v}_{out} \\ \underline{v}_{int} \end{bmatrix} = \begin{bmatrix} S_1 & S_2 \\ S_3 & S_4 \end{bmatrix} \begin{bmatrix} \underline{v}_{in} \\ \underline{v}_{int} \end{bmatrix}$$

$\underline{v}_{in}$ - vector of all inputs to the junction structure

$\underline{v}_{int}$ - vector of all junction structure internal variables

$\underline{v}_{out}$ - vector of all outputs from the junction structure

Figure 3.4. The junction matrix equation

$$\underline{v}_{out} = S_1\underline{v}_{in} + S_2\underline{v}_{int}$$

$$\underline{v}_{int} = S_3\underline{v}_{in} + S_4\underline{v}_{int}$$

(a)

$$\underline{v}_{int} = (I - S_4)^{-1}S_3\underline{v}_{in}$$

(b)

$$\underline{v}_{out} = [S_1 + S_2(I-S_4)^{-1}S_3]\underline{v}_{in}$$

(c)

Figure 3.5. Derivation of the reduced junction matrix. (a) The expanded junction matrix equation. (b) Internal variables in terms of inputs. (c) The unpartitioned reduced junction matrix equation.

IV. A COMPUTER IMPLEMENTED STATE MODEL FORMULATOR OF INCREASED EFFICIENCY

The ENPORT-4 program is a powerful tool for the modeling, analysis, and simulation of multiport systems. When given a bond graph description of a system, ENPORT-4 selects physically-meaningful state variables and derives the system state model, eigenvalues, and time response. The many additional features, available options and outputs, and the structure of ENPORT-4 are discussed in the program's documentation [8].

Although ENPORT-4 provides the system analyst with a broad array of system information, it has significant inadequacies which have a profound affect on program performance. Most of these inadequacies are revealed in the graph reduction and state model formulation procedures. In the following sections, deficiencies are identified in the ENPORT-4 graph reduction and state model formulation procedures, and modifications are discussed which increase overall program efficiency. These modifications have been implemented in the ENPORT-5 program which is currently under development. Key ENPORT-5 graph reduction and state model formulation subroutines are listed in Appendix F.

4.1 Design Features

4.1.1 Data Structures

At various stages in the bond graph processing procedure assorted graph parameter and structural information must be retained or manipulated. In general, when interpreted in matrix form, this information results in a sparse matrix analogous to a graph incidence or adjacency matrix [27]. A major deficiency of ENPORT-4 is its use of full storage (storage which includes all matrix zero entries) in multi-dimensional arrays for the retention and manipulation of bond graph information.

A major improvement in efficiency is realized in ENPORT-5 by minimizing data storage requirements through the use of a sparse-matrix-based storage format. This is achieved by using push-down stacks and linked data structures [28-30]. In particular, the ENPORT-5 graph reduction and state model formulation procedures employ simple lists for the retention and manipulation of data. These lists are grouped in pairs, where the entries of each list are ordered. In each list pair, one list contains the nonzero entries of an implied matrix of known dimensions, and the other list contains the coordinates of the matrix entries where each coordinate pair is converted to a unique number. The conversion of a coordinate pair is accomplished by representing the position of a matrix entry as the entry's column coordinate added to the product of the matrix column dimension and one less than the

entry's row coordinate. An example of a list pair is given in Figure 4.1.

4.1.2 Causal Assignment

The assignment of causality is an important stage in the processing of a bond graph. The ENPORT-4 program uses a causal assignment scheme which is a modification of SSCAP (the standard sequential causality assignment procedure) in that the scheme gives priority to user specified causal orientations. Although this feature provides the knowledgeable user with a great deal of flexibility, the unwary user may specify causal orientations which may violate system constraints (such as constraints imposed by sources), give a false indication of system order, or create uncertainty in the state model formulation procedure.

The causal assignment scheme employed by the ENPORT-5 program is a direct implementation of SSCAP in which the user cannot specify causal orientations until after all source bonds and storage bonds have been causally oriented. This scheme not only eliminates the difficulties discussed above, but also guarantees that if a reduced junction matrix exists, then it has the simplified form identified in section 3.2.

4.1.3 Determination Of Junction Structure Reducibility

The ENPORT-4 procedure for determining the reducibility of the junction matrix represents an additional area of inefficiency. The junction matrix equation was given in

Figure 3.4. As illustrated in Figure 3.5, the junction matrix is reducible if the matrix $(I-S_4)$ is nonsingular. In ENPORT-4, junction matrix reducibility only can be determined during the process of attempting to invert the matrix $(I-S_4)$.

Based on previous work, the reducibility of the junction matrix depends on the solvability of causal complexes [12-14,19,20]. Stating the case more explicitly, the junction matrix is reducible if and only if each causal complex is solvable. As an explicit step in the ENPORT-5 causal assignment procedure, causal complexes are identified and their graph locations are communicated to the user. Prior to the formulation of the junction matrix, each causal complex is tested for solvability in order to determine junction matrix reducibility. If any causal complex is determined to be unsolvable, then bond graph processing aborts and the user is notified of all unsolvable causal complexes.

4.1.4 Junction Matrix Formulation

As an intermediate step in the formulation of the junction matrix in ENPORT-4, a matrix equation is explicitly formed for each junction structure node. The entries of each node matrix are then placed in the junction matrix in accordance with bond classifications and orderings.

The ENPORT-5 program does not explicitly form a matrix equation for each junction structure node. Instead, the

junction matrix is constructed directly by using node causal forms, bond power orientations, and graph model parameters to obtain the coefficients of the summation, identity, and proportionality output equations for each junction structure node. Specifically, causal, power, and parameter information is used to identify the flow output variable and the coefficients of the corresponding flow input variables for each 0-junction, the effort output variable and the coefficients of the corresponding effort input variables for each 1-junction, the identity relations for each 0-junction and 1-junction, and the proportionality relationships for each transformer and gyrator. Once determined, each of the above coefficients (except zeros) is directly stored in a compact junction matrix where each entry position is determined by bond classifications and orderings, and the dimensions of the implied full storage junction matrix.

4.1.5 Matrix Inversion

As illustrated in section 3.3, several calculations in the state model formulation algorithm require the computation of a matrix inverse. The matrix inversion routine used by ENPORT-4 is a Gauss-Jordan procedure which selects a matrix entry of greatest magnitude for the pivot at each

stage of the deflation process. In ENPORT-4, the selection of a pivot requires a row and column scan of mostly zero entries since each matrix is generally sparse and in a full storage format.

A result of this research was the development of the sparse matrix inversion subroutine which is employed by the ENPORT-5 program. This subroutine is called "INPRD" (INverse-PRoDuct). Its development was motivated by the lack of a matrix inversion routine which can take advantage of the special features of the equations in the state model formulation algorithm.

A very important consideration in the development of any sparse matrix inversion routine is the possible increase in the storage requirements for the inverse of a sparse matrix [31]. The INPRD subroutine effectively eliminates the problem of storage growth by taking advantage of the features of the matrix calculations in the graph reduction and state model formulation procedures. In these procedures, matrix inverses in calculations appear in the form $A^{-1}B$ where the matrix product $A^{-1}B$ relates sets of junction structure variables. INPRD is a Gauss-Jordan type procedure which controls storage requirements by accepting the generally sparse matrices A and B as inputs and returning the generally sparse matrix $A^{-1}B$ as output. Note that A^{-1} is not explicitly computed unless B is the compatible identity matrix. INPRD applies directly to B a set of

transformations which represent elementary row operations for the reduction of A to the identity matrix. In order to minimize round-off errors, a matrix entry of greatest magnitude in A is selected as the pivot at each stage in the process of deflating A. The benefits of INPRD are evidenced by the performance characteristics of the ENPORT-5 program.

4.2 State Model Formulator Computer Test Results

In this section, some performance aspects of the ENPORT-4 and ENPORT-5 state model formulators will be considered. In particular, processing times and junction matrix storage requirements will be assessed for three test examples interactively processed on the CDC 6500 computer.

The processing time will be interpreted as the CP (central processor) execution time consumed from the point of parameter input to the point of state model output. Storage considerations are limited to the junction matrix, since it is the largest system matrix in the formulation process. For each example considered, the processing time and junction matrix storage space requirements of ENPORT-4 will be used as benchmarks.

The first test example is a structural model of a lever mechanism with inertia load (see Figure 4.2). The second test example is a structural model of a beam-block transducer system (see Figure 4.3). The final test example is a structural model of a radar pedestal position control system (see Figure 4.4). Each of the above examples may be

found in the user's manual for the ENPORT-4 program, where each is studied in detail [8].

For each test example, the computational results are contained in Table 3 and Table 4 for storage requirements and processing time respectively. From Table 3, it is observed that the ENPORT-5 formulator requires significantly less storage (as typified by the junction matrix) than does the ENPORT-4 formulator for a given bond graph model. The ENPORT-4 storage requirement for the junction matrix is given by $(N_p+2N_I)^2$. The ENPORT-5 storage requirement for the junction matrix is given by $4(N_p+2N_I-1)$. Thus, the difference between the bond graph model storage demands of the ENPORT-4 and ENPORT-5 formulators becomes increasingly dramatic as the number of bonds in a graph model increases.

From Table 4, it is observed that the ENPORT-5 formulator provides a significant savings in processing time for the given test examples. In general, the size (and sign) of this savings is a function of several variables, e.g., the number of causal complexes, and the density and dimensions of matrices to be manipulated. As an explicit case, consider the multiplication of an (nxm) matrix by an (mxp) matrix, neither of which contains a zero entry. In a full storage format, this matrix multiplication requires nmp scalar multiplications and n(m-1)p scalar additions. In ENPORT-5, this matrix multiplication requires nmp scalar multiplications, nmp scalar additions, and 2nm(p+1) element

comparisons. Note that full storage matrix multiplication requires the same number of scalar operations irrespective of the sparsity of either matrix factor. In general, the above matrix multiplication in ENPORT-5 requires n(m-r)(p-s) scalar multiplications, n(m-r)(p-s) scalar additions, and 2nm(p+1)-rn(p+2)-2ns(m-r/2) element comparisons where r is the average number of zeros per row in the (nxm) matrix and s is the average number of zeros per row in the (mxp) matrix. Note that the worst case is given for the number of element comparisons. It is seen that as r and s increase, matrix multiplication in a full storage format rapidly becomes computationally more demanding than matrix multiplication in ENPORT-5. Thus, although it is possible for the processing time required by ENPORT-5 to exceed the processing time required by ENPORT-4, this possibility is minimized by the general sparsity of system matrices and the "inverse-multiplication" feature of the INPRD subroutine.

In conclusion, the combined results of Table 3 and Table 4 suggest that the ENPORT-5 state model formulator not only enhances the processing performance and capabilities of the ENPORT-5 program, but also contributes to a reduced dollar cost for the operation of a linear time-invariant bond graph simulation program.

$$\begin{bmatrix} 2 & 0 & -1 & 0 \\ 0 & 5 & 0 & 3 \\ 0 & 0 & 0 & -4 \end{bmatrix}$$

A 3x4 matrix

$$\begin{bmatrix} 2 \\ -1 \\ 5 \\ 3 \\ -4 \end{bmatrix} \qquad \begin{bmatrix} 1 \\ 3 \\ 6 \\ 8 \\ 12 \end{bmatrix}$$

Entry list Position list

Figure 4.1. Example of a list pair with the corresponding matrix.

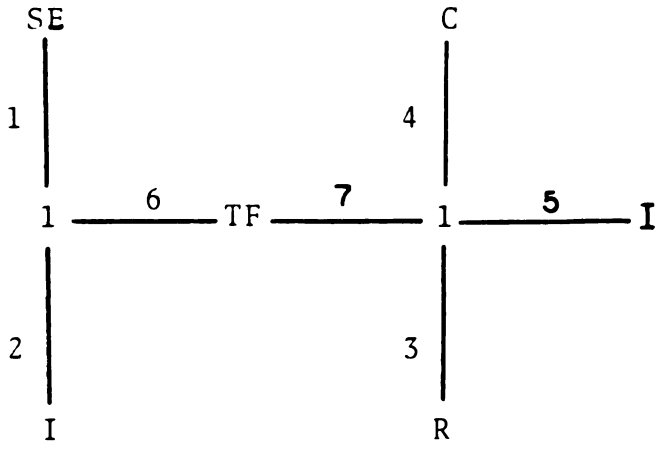


Figure 4.2. Test example 1

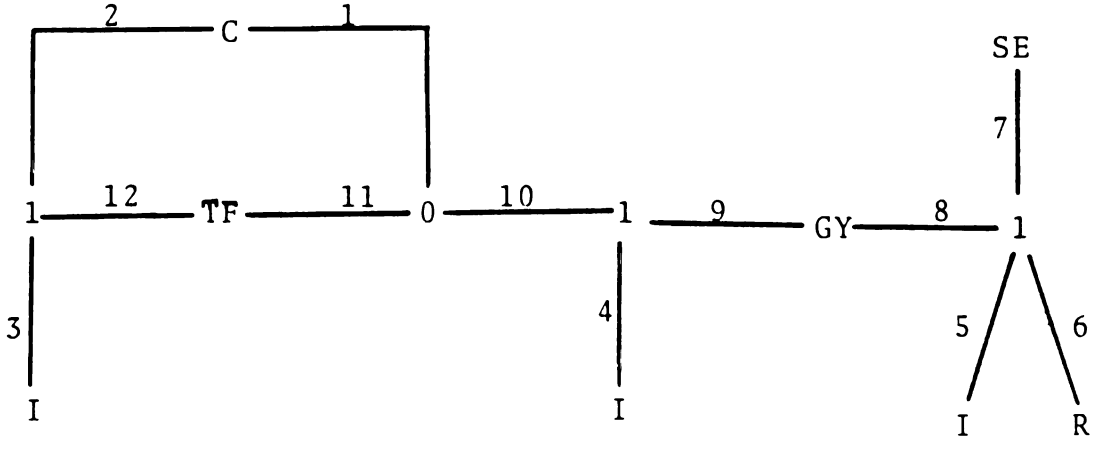


Figure 4.3. Test example 2

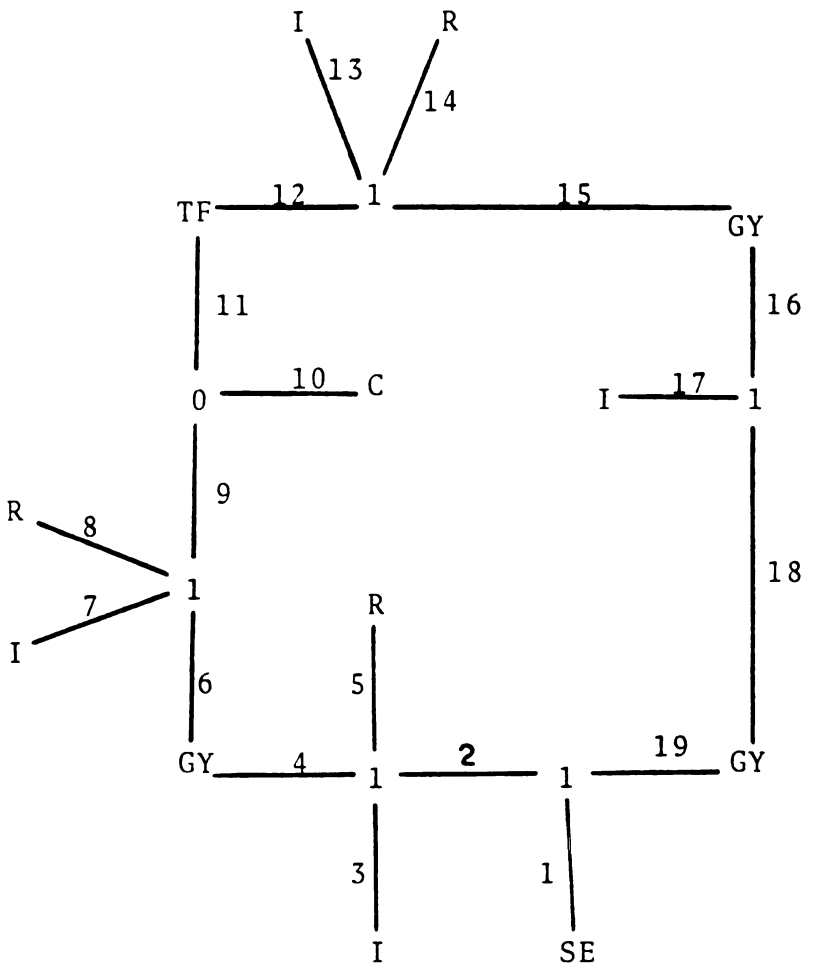


Figure 4.4. Text example 3

Table 3. Junction Matrix Storage For Test Examples

Example	(1) W4	(2) W5	W5/W4
1	81	32	.395
2	289	64	.221
3	841	112	.133

(1) W4≡Number of storage words for junction matrix in ENPORT-4.
(2) W5≡Number of storage words for junction matrix in ENPORT-5.

Table 4. Processing Time for State Model Formulation

Example	(1) PT4	(2) PT5	PT5/PT4
1	0.604	0.138	0.228
2	0.905	0.346	0.382
3	1.420	0.927	0.653

(1) PT4≡Processing time (in seconds) for ENPORT-4.
(2) PT5≡Processing time (in seconds) for ENPORT-5.

V. SUMMARY

The basis order rules are among the most significant results achieved in this investigation. For weighted junction structures, these topology-based formulations provide the bond graph analyst with the composition of a basis for the junction structure transformation. In addition, when used with Theorem 3, the basis order rules provide a "good" estimate of the number of distinct basis variable sets for the junction structure transformation.

Herein, it was shown that the standard sequential causality assignment procedure assures that the S_{23}, S_{32}, and S_{33} blocks of the reduced junction matrix are zero for a reducible junction structure. This resulted in a major simplification in Rosenberg's state model formulation algorithm which serves as a model for the ENPORT-5 state model formulator [15].

As indicated by the computer results in Chapter IV, the ENPORT-5 state model formulator provides heretofore unrealized efficiency and speed in the automated processing of linear time-invariant bond graphs. The key features of this formulator are (1) the use of sparse-matrix-based storage, (2) the determination of junction structure reducibility prior to the formation of the junction matrix, (3) the direct construction of the junction matrix, and (4) the

versatile sparse-matrix-based INPRD subroutine. As a result of the storage and processing efficiency of the ENPORT-5 state model formulator, the ENPORT-5 program has a greatly enhanced capacity for the processing of large bonds graphs. Future advances in graph processing efficiency can be achieved by the development of a general technique which does not require matrix inversions for the determination of junction structure reducibility. A step in this direction can be made by the development of "basis order rules" which are applicable to any junction structure containing a gyrator. Such formulations would offer necessary conditions for the reducibility of an arbitrary junction structure, as well as serve as aids for the determination of a basis for the junction structure transformation.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Chorafas, D.N., Systems and Simulation, Academic Press, 1965.

[2] Malmberg, A.F., "NET-2 Network Analysis Program-User's Manual Release 9", HDL-050-1, Braddock, Dunn, and McDonald, Inc., El Paso, Texas, 1973.

[3] Nagel, L.W., and Pederson, D.O., "SPICE (Simulation Program with Integrated Circuit Emphasis)", ERL-M382, Electronics Research Laboratory, College of Engineering, Univ. of California, Berkeley, Ca., 94720, April, 1973.

[4] Chace, M.A., and Angell, J.C., "Users Guide to DRAM (Dynamic Response of Articulated Machinery)", Design Engineering Computer Aids Laboratory, Department of Mechanical Engineering, University of Michigan, Ann Arbor, Mich., Feb., 1976.

[5] Dix, R.C., "A Users Manual for MEDUSA (MEchanism-Dynamics-Universal System Analyzer)", Mechanics, Mechanical and Aerospace Engineering Department, Illinois Institute of Technology, Chicago, ILL., Jan., 1975.

[6] Pilkey, W., and Pilkey, B., eds, Shock and Vibration Computer Programs - Reviews and Summaries, SVM-10, The Shock and Vibration Information Center, Naval Research Laboratory, Washington, D.C., 1975.

[7] Bowers, J.C., O'Reilly, J.E., and Shaw, G.A., "SUPER* SCEPTRE - User's Manual", DAAA-21-73-C-0655, AMC, Picatinny Arsenal, Dover, N.J., 1975.

[8] Rosenberg, R.C., "A User's Guide to ENPORT-4", John Wiley, Inc., 1974.

[9] Paynter, H.M., Analysis and Design of Engineering Systems, M.I.T. Press, 1960.

[10] Karnopp, D.C., and Rosenberg, R.C., Analysis and Simulation of Multiport Systems - The Bond Graph Approach to Physical System Dynamics, M.I.T. Press, 1968 (68A35012).

[11] Gebben, V.D., "Bond Graph Bibliography for 1961-1976", Trans. ASME, J. Dyn. Sys., Meas., and Control, 99(1977) pp. 143-145.

[12] Nobuhide, S., "Bond Graphs: Structural Properties and Augmentation Algorithm", Presented at the Conference on System Control Theory and its Applications to Dynamic Economic Models, Nagoya City University, Mar. 27-29, 1976.

[13] Ort, J.R., and Marten, H.R., "The Properties of Bond Graph Junction Structure Matrices", Trans. ASME, J. Dyn. Sys., Meas., and Control, 95(1973) pp. 362-367.

[14] Perelson, A.S., "Bond Graph Junction Structures", Trans. ASME, J. Dyn. Sys., Meas., and Control, 97(1975) pp. 189-195.

[15] Rosenberg, R.C., "State-Space Formulation for Bond Graph Models of Multiport Systems", J. Dyn. Sys., Meas., and Control, Trans. ASME, 93(1971) pp. 35-40.

[16] Dixhoorn, J.J. van, "Simulation of Bond Graphs on Minicomputers", J. Dyn. Sys., Meas., and Control, Trans. ASME, 99(1977) pp. 9-14.

[17] Busacker, R.G., and Saaty, T.L., Finite Graphs and Networks - An Introduction with Applications, McGraw-Hill, 1965.

[18] Harary, F., Graph Theory, Addison-Wesley, 1972.

[19] Rosenberg, R.C., and Andry, A.N., Jr., "Solvability of Bond Graph Junction Structures with Loops", IEEE Trans. on Circuits and Systems, 26(1979) pp. 130-137.

[20] Rosenberg, R.C., and Andry, A.N., Jr., "Solvability of Certain Classes of Bond Graph Junction Structures", Proceedings of the Second International Symposium on Large Engineering Systems, Univ. of Waterloo, Waterloo, Ontario, Canada, May, 1978, pp. 537-541.

[21] Ort, J.R., and Martens, H.R., "A Topological Procedure for Converting a Bond Graph to a Linear Graph", Trans. ASME, J. Dyn. Sys., Meas., and Control, 96(1974) pp. 307-314.

[29] Perelson, A.S., Discussion re [13], Trans. Asme, J. Dyn. Sys., Meas., and Control, 98(1976) pp. 209-210.

[30] Perelson, A.S., and Oster, G.F., "Bond Graphs and Linear Graphs", J. Franklin Institute, 302(1976)159-185.

[31] Karnopp, D.C., and Rosenberg, R.C., System Dynamics: A Unified Approach, John Wiley Inc., 1974.

[32] Rosenberg, R.C., "Essential Gyrators and Reciprocity in Junction Structures", (accepted for publication) J. Franklin Institute.

[33] Karnopp, D.C., "Some Bond Graph Identities Involving Junction Structures", Trans. ASME, J. Dyn. Sys., Meas., and Control, 97(1975) pp. 439-440.

[34] Korfhage, R.R., Discrete Computational Structures, Academic Press, 1974.

[35] Berztiss, A.T., Data Structures - Theory and Practice, Academic Press, 1971.

[36] Knuth, D.E., The Art of Computer Programming, Vol. I, Chap. 2, Addison-Wesley, 1973.

[37] Stone, H.S., Introduction to Computer Organization and Data Structures, McGraw-Hill, 1972.

[38] Tewarson, R.P., Sparse Matrices, Academic Press, 1973.

[39] Riordan, J., Combinatorial Identities, Wiley, 1968.

APPENDIX A

DERIVATION OF THE BASIS ORDER RULES

A.1 Basis Order and Simple Junction Structures

In this section we derive a pair of general computational rules for predicting the order and variable-type composition of an N-port SJS basis. An N-port JS has exactly N EN-nodes. It is common usage to refer to bonds (0,EN) or (1,EN) as port bonds.

Motivation for these rules is derived by considering the number of free variables which remain following the imposition of a set of independent constraint equations on a set of system variables. Several types of proper SJS's are studied first; then the results are extended to standard SJS's. In passing, alternate forms of the order rules for proper SJS's are given. The order rules are presented here as Theorem 1.

Theorem 1: Every standard SJS satisfies the relations

$$(i) E = N_B + N_0 - B_0 - N_1$$

and

$$(ii) F = N_B + N_1 - B_1 - N_0.$$

A.1.1 Basis Order for Proper Simple Junction Structure Forests

Initially we establish Theorem 1 for an arbitrary proper SJS tree G by demonstrating that G can be obtained

from a forest of separate 1-junctions and 0-junctions by a series of subgraph concatenations. It is observed that junctions satisfy the order rules. Following the assumption that G contains more than a single junction, a 0-junction in G is identified as a base node to which is added an appropriate number of 1-junctions and 0-junctions, of specified degrees, which yields a SJS equivalent to G. Additionally, it is noted that the concatenation of a junction to a proper SJS tree yields a proper SJS tree which satisfies the order rules, thus yielding the results for G. Finally, by considering the order rules for each component, the results are extended to an arbitrary proper SJS forest.

Lemma A.1: Every proper SJS forest satisfies the relations

$$E = N_B + N_0 - B_0 - N_1 \text{ and } F = N_B + N_1 - B_1 - N_0.$$

Prior to proving Lemma A.1, two definitions are needed. First we state that two distinct SJS's are <u>conformable</u> if one contains a 0 and the other contains a 1.

We now define the graph <u>concatenation operator</u> C where C(G,H) = K is a binary operation performed on two proper SJS's (G and H) which are conformable to yield a third proper SJS (K). Let G be a connected proper SJS and H be a connected proper SJS where $V_G \cap V_H = \emptyset$. Also, let u be a 1-junction and u_{EN} be an EN-node in V_G, and v be a 0-junction and v_{EN} be an EN-node in V_H, where $(u,u_{EN}) \varepsilon X_G$ and $(v,v_{EN}) \varepsilon X_H$. Then C[G(u), H(v)] will denote the connected proper SJS K where

$V_K \equiv [V_G \cup V_H] - \{v_{EN}, u_{EN}\}$ and

$X_K \equiv [X_G \cup X_H \cup \{(v,u)\}] - \{(v,v_{EN}),(u,u_{EN})\}$.

Note that if $K = C[G(u), H(v)]$ and $K' = C[G(u), H(v)]$ (different EN-nodes are removed) then K and K' are isomorphic. For $C[G(u), H(v)]$, it will be said that G and H are "concatenated". Note that $C[H(v), G(u)] = C[G(u), H(v)]$. We now proceed with the proof of Lemma A.1.

Proof: Let $G_1^{(m)}$ denote a connected proper SJS such that $V_{G_1}(m)$ consists of exactly one 1-junction and exactly m EN-nodes, where $m \geq 2$. Thus $G_1^{(m)}$ has the form shown in Figure A.1.

Observe that the definition of a 1-junction applied to $G_1^{(m)}$ yields the results

$E = N_B + N_0 - B_0 - N_1 = (m) + (0) - (0) - (1) = m - 1$ and

$F = N_B + N_1 - B_1 - N_0 = (m) + (1) - (m) - (0) = 1$.

These results agree with Lemma A.1.

Let $G_0^{(n)}$ denote a connected proper SJS such that $V_{G_0}(n)$ consists of exactly one 0-junction and exactly n EN-nodes, where $n \geq 2$. Thus, $G_0^{(n)}$ has the form shown in Figure A.2.

Observe that the definition of a 0-junction applied to $G_0^{(n)}$ yields the results

$E = N_B + N_0 - B_0 - N_1 = (n) + (1) - (n) - (0) = 1$ and

$F = N_B + N_1 - B_1 - N_0 = (n) + (0) - (0) - (1) = n - 1$.

These results agree with Lemma A.1.

Note that Lemma A.1 applies to a forest with an arbitrary number of components, each of which is a 0- or 1-junction together with a set of EN-nodes.

Now consider a concatenation involving $G_1^{(m)}$ for $m \geq 2$. Let G be an arbitrary connected proper SJS where V_G contains at least one 0-junction, say v, and let E_G and F_G be given. Observe that $K = C[G(v), G_1^{(m)}]$, where $m \geq 2$, contains one less effort variable and one less flow variable than $G \cup G_1^{(m)} = \langle V_G \cup V_{G_1}^{(m)}, X_G \cup X_{G_1}^{(m)} \rangle$ since $o(X_K) = 0(X_G, \cup X_{G_1}^{(m)}) - 1$. ("o" denotes "order of".) Also, K and $G \cup G_1^{(m)}$ yield the same number of independent flow constraint equations, since the number of junctions and junction degrees are unchanged. Then, clearly

$$E_K = E_G + E_{G_1}(m) - 1 \text{ and } F_K = F_G + F_{G_1}(m) - 1.$$

Let $\Delta E_1^{(m)} \equiv E_K - E_G$ and $\Delta F_1^{(m)} \equiv F_K - F_G$.

Then $\Delta E_1^{(m)} = E_{G_1}(m) - 1 = m - 2$ and $\Delta F_1^{(m)} = F_{G_1}(m) - 1 = 0$.

where $m \geq 2$.

That is, as a result of a concatenation involving $G_1^{(m)}$ the incremental changes in E and F are known.

Now consider a concatenation involving $G_0^{(n)}$ for $n \geq 2$. Let G be an arbitrary connected proper SJS where V_G contains at least one 1-junction, say u, and let E_G and F_G be given. Observe that $K = C[G(u), G_0^{(n)}]$, where $n \geq 2$

contains one less effort variable and one less flow variable than $G \cup G_0^{(n)}$. Also, K and $G \cup G_0^{(n)}$ yield the same number of independent effort constraint equations and the same number of independent flow constraint equations. Therefore,

$$E_K = E_G + E_G(n) - 1 \text{ and } F_K = F_G + F_G(n) - 1.$$

Let $\Delta E_0^{(n)} \equiv E_K - E_G$ and $F_0^{(n)} \equiv F_K - F_G$. Then

$$\Delta E_0^{(n)} = E_{G_0}(n) - 1 = 0 \text{ and } \Delta F_0^{(n)} = F_{G_0}(n) - 1 = n - 2,$$

where $n \geq 2$.

We now establish lemma A.1 for an arbitrary proper SJS tree. Let G be an arbitrary proper SJS tree. Suppose G contains N_1 1-junctions and N_0 0-junctions; not both N_1 and N_0 are zero, since G is proper. If $N_0=0$, G is a 1-junction component; if $N_1=0$, G is a 0-junction component. In either case, we are done. Therefore, assume $N_1>0$ and $N_0>0$.

Enumerate the 0-junctions in V_G by $v_1, v_2, \ldots, v_{N_0}$, and the 1-junctions in V_G by $v_{N_0+1}, v_{N_0+2}, \ldots, v_{N_0+N_1}$. Let $V_G^{(0)} = \{v_1, v_2, \ldots, v_{N_0}\}$ and $V_G^{(1)} = \{v_{N_0+1}, v_{N_0+2}, \ldots, v_{N_0+N_1}\}$. Then $V_G^{(1)} \cap V_G^{(0)} = \emptyset$ and $V_G^{(1)} \cup V_G^{(0)}$ contains all junctions in G. The junctions in G will now be partitioned according to their distances from v_1. Let $S_{-1} = \emptyset$ and $S_i = \{v \in V_G^{(1)} \cup V_G^{(0)} \mid d(v_1, v) = i\}$, where $i = 0,1,2,\ldots$ Note that $S_i \cap S_j = \emptyset$ if $i \neq j$. The order of $V_G^{(1)} \cup V_G^{(0)}$ is finite and G is a tree imply that there exists a smallest

positive $k \leq N_1 + N_0 - 1$ such that $d(v_1, v) \leq k$ for all $v \in V_G^{(1)} \; V_G^{(0)}$. Without loss of generality, assume k is odd. Then

$$\bigcup_{i=0}^{(\frac{k-1}{2})} S_{2i} = V_G^{(0)} \text{ and } \bigcup_{i=0}^{(\frac{k-1}{2})} S_{2i+1} = V_G^{(1)}.$$

Relabel the elements in S_i so that $v_{i,j}$ is the jth element in S_i, where $0 \leq i \leq k$ and $1 \leq j \leq o(S_i)$. Let $\alpha_{i,j} = \deg(v_{i,j})$ for $v_{i,j}$ in G. Also, let $G_{0,0} = G_0^{(\deg v_1)}$, and let $G_{i,j}$ be the proper SJS tree corresponding to $v_{i,j}$ where $G_{i,j}$ is either $G_1^{(m)}$ or $G_0^{(n)}$ if i is odd or even, respectively, and m or $n = \alpha_{i,j}$.

Now we will reconstruct G from a proper SJS forest of N_0 0-junction and N_1 1-junctions by using the internal bonds of G as a directory. Note that the concatenation of two conformable proper SJS trees yields a proper SJS tree.

Let $G_{1,0} = G_{0,0}$, and let $G_{1,1} = G_{1,o(S_0)}$ be the proper SJS tree obtained from the series of concatenations of $G_{1,0}$ with $G_{1,j}$ at $v_{0,1}$ for each j such that $(v_{o,1}, v_{1,j}) \in X_G$; i.e., $1 \leq j \leq o(S_1)$. Let $G_{2,0} = G_{1,o(S_0)}$, and let $G_{2,1}$ be the proper SJS tree obtained from the series of concatenations of $G_{2,0}$ with $G_{2,j}$ at $v_{1,1}$ for each j such that $(v_{1,1}, v_{2,j}) \in X_G$. Let $G_{2,2}$ be the proper SJS tree obtained from the series of concatenations of $G_{2,1}$ with $G_{2,j}$ at $v_{1,2}$ for each j such that $(v_{1,2}, v_{2,j}) \in X_G$.

In general, let $G_{i,n}$ be the proper SJS tree obtained from the series of concatenations of $G_{i,n-1}$ with $G_{i,j}$ at $v_{i-1,n}$ for each j such that $(v_{i-1,n}, v_{i,j}) \in X_G$, where $G_{i,0} = G_{i-1,o(S_{i-2})}$, $1 \leq i \leq k$ and $1 \leq n \leq o(S_{i-1})$. Then $G_{k,o(S_{k-1})} = G$.

An example of the construction procedure is given in Figure A.3.

The construction procedure involves N_0-1 concatenations of proper SJS trees having the form $G_0^{(n)}$, and N_1 concatenations of proper SJS trees having the form $G_1^{(m)}$.

Therefore,

$$E_G = E_{G_{k,o(S_{k-1})}} = E_{G_{0,0}} + \sum_{i=2}^{N_0} \Delta E_0(\deg v_i) + \sum_{i=N_0+1}^{N_0+N_1} \Delta E_1(\deg v_i)$$

$$= (1) + (0) + \sum_{i=N_0+1}^{N_0+N_1} (\deg v_i - 2) = \sum_{i=N_0+1}^{N_0+N_1} \deg v_i - 2N_1 + 1 \quad \text{and}$$

$$F_G = F_{G_{k,o(S_{k-1})}} = F_{G_{0,0}} + \sum_{i=2}^{N_0} \Delta F_0(\deg v_i) + \sum_{i=N_0+1}^{N_0+N_1} \Delta F_1(\deg v_i) =$$

$$(\deg v_1 - 1) + \sum_{i=2}^{N_0} (\deg v_i - 2) + 0 = \sum_{i=1}^{N_0} \deg v_i - 2N_0 + 1.$$

Observe that (i) $B_0 = \sum_{i=1}^{N_0} \deg v_i$,

(ii) $B_1 = \sum_{i=N_0+1}^{N_0+N_1} \deg v_i$,

(iii) $N_I = N_1 + N_0 - 1$ (Euler's Rule),

(iv) $N_B = B_1 + B_0 - N_I$,

(v) $P_0 = B_0 - N_I$, and

(vi) $P_1 = B_1 - N_1$.

Therefore, if G is a proper SJS tree, then

$$E = \sum_{i=N_0+1}^{N_0+N_1} \deg v_i - 2N_1 + 1 = N_B + N_0 - B_0 - N_1$$

$$F = \sum_{i=1}^{N_0} \deg v_i = 2N_0 + 1 = N_B + N_1 - B_1 - N_0.$$

Note that $\deg v_i \geq 2$, $1 \leq i \leq N_0$, implies that $E \geq 1$ and $F \geq 1$. We now extend the results to an arbitrary proper SJS forest G. Let $G_1, G_2, \ldots, G_n$ be the components of G. Then each component G_i is a proper SJS tree.

Therefore,

$$E_{G_i} = N_{BG_i} + N_{0G_i} - N_{1G_i} \quad \text{and}$$

$$F_{G_i} = N_{BG_i} + N_{1G_i} - B_{1G_i} - N_{0G_i}, \text{ where } 1 \leq i \leq n.$$

Then

$$E_G = \sum_{i=1}^{n} E_{G_i} = \sum_{i=1}^{n} (N_{BG_i} + N_{0G_i} - B_{0G_i} - N_{1G_i})$$

$$= \sum_{i=1}^{n} N_{BG_i} + \sum_{i=1}^{n} N_{0G_i} - \sum_{i=1}^{n} B_0G_i - \sum_{i=1}^{n} N_{1G_i} = N_{BG} + N_{0G} - B_{0G} - N_{1G},$$

and

$$F_G = \sum_{i=1}^{n} F_{G_i} = \sum_{i=1}^{n} (N_{BG_i} + N_{1G_i} - B_{1G_i} - N_{0G_i})$$

$$= \sum_{i=1}^{n} N_{BG_i} + \sum_{i=1}^{n} N_{1G_i} - \sum_{i=1}^{n} N_{0G_i} = N_{BG} + N_{1G} - B_{1G} - N_{0G},$$

where $E_G \geq n$ and $F_G \geq n$.

Hence, if G is a proper SJS forest, then G satisfies the relations $E = N_B + N_0 - B_0 - N_1$ and $F = N_B + N_1 - B_1 - N_0$.

Recall that B_0 is the total number of (external and internal) bonds incident to 0-junctions. Observe that in a proper SJS every internal bond is incident to exactly one 1-junction and exactly one 0-junction; thus, in a proper SJS, N_I is included in B_0. Therefore, for a proper SJS, $N_B = P_1 + B_0$. Similarly, for a proper SJS, $N_B = P_0 + B_1$. These results and Lemma A.1 validate the following corollary.

Corollary 1.1: Every proper SJS satisfies the relations

(i) $E = N_0 + P_1 - N_1$

(ii) $F = N_1 + P_0 - N_0$.

A.1.2 Basis Order for Proper Simple Junction Structures

Now Theorem 1 will be established for an arbitrary connected proper SJS G. This will be accomplished by removing cycle bonds from G until a spanning tree is obtained, and then demonstrating that the previous relations remain valid when these bonds are replaced. These results will then be extended to an arbitrary proper SJS.

Lemma A.2: Every proper SJS satisfies the relations

$E = N_B + N_0 - B_0 - N_1$ and

$F = N_B + N_1 - B_1 - N_0$.

Prior to proving Lemma A.2 we define a transformation T which removes cycles from a connected proper SJS, and define a transformation S which creates a cycle in a connected proper SJS.

To obtain T, let G be an arbitrary connected proper SJS. Assume G contains at least one cycle, say C. Let b be an arbitrary bond in C, and let v_{EN} and u_{EN} be EN-nodes. Then there exists a unique 1-junction $v \varepsilon V_G$ and a unique 0-junction $u \varepsilon V_G$ such that b = (v, u). Let T [G(v, u)] denote the SJS H where $V_H \equiv V_G \cup \{v_{EN}, u_{EN}\}$ and $X_H \equiv [X_G \cup \{(v, v_{EN}), (u, u_{EN})\}] - \{(v,u)\}$. Then, clearly, H is a connected proper SJS which contains one less cycle than G. Observe that $N_{1H} = N_{1G}$, $N_{0H} = N_{0G}$, $P_{1H} = P_{1G} + 1$, and $P_{0H} = P_{0G} + 1$.

To obtain S, let G be an arbitrary connected proper SJS. Suppose there exist 1-junction $v \varepsilon V_G$ and 0-junction $u \varepsilon V_G$ such that $(v,u) \notin X_G$. Also, assume there exist EN-nodes v_{EN} and u_{EN} in V_G such that (v, v_{EN}) and (u, u_{EN}) are in X_G. Then let S (G,v,u) denote the SJS H where $V_H \equiv V_G - \{v_{EN}, u_{EN}\}$ and $X_H \equiv [X_G \cup \{(v,u)\}] - \{(v, v_{EN}), (u, u_{EN})\}$.

Then H is unique (to an isomorphism), and H is a connected proper SJS which contains one more cycle than G. Observe that S(G,v,u) contains one less flow variable and one less effort variable than G, where S(G,v,u) and G yield the same number of independent flow constraint equations and the same number of independent effort constraint equations. Therefore, if H = S(G,v,u), then

$$E_H = E_G - 1 \text{ and } F_H = F_G - 1.$$

Let $\Delta E_S \equiv E_H - E_G$ and $\Delta F_S \equiv F_H - F_G$.

Then $\Delta E_S = -1$ and $\Delta F_S = -1$.

We now proceed with the proof of Lemma A.2.

Proof: Since for a proper SJS we have $N_B = P_1 + B_0$ and $N_B = P_0 + B_1$, it is sufficient to show that every proper SJS satisfies the relations $F = N_1 + P_0 - N_0$ and $E = N_0 + P_1 - N_1$.

Consider an arbitrary proper SJS G. If G is a forest, then, done, by Lemma A.1. Therefore, assume G contains at least one cycle. Let G be connected and let $G_0 \equiv G$. Also, let v_i and u_i denote 1-junctions and 0-junctions respectively, for all i. Then G_0 contains some cycle C_1. Let $b_1 = (v_1, u_1)$ be an arbitrary bond in C_1, and set $G_1 = T[G_0,(v_1,u_1)]$. In general, if G_{i-1} contains some cycle C_i, let $b_i=(v_i,u_i)$ be an arbitrary bond in C_i, and set $G_i=T[G_{i-1}, (v_i,u_i)]$. Then the order of X_G finite implies that G contains a finite number of cycles, which implies that there exists a smallest positive integer k such that G_k is a spanning tree. Note that $N_{0G_k} = N_{0G}$, $N_{1G_k} = N_{1G}$, $P_{0G_k} = P_{0G} + k$, and $P_{1G_k} = P_{1G} + k$. G_k is a proper SJS tree. Therefore,

$$E_{G_k} = N_{0G_k} + P_{1G_k} - N_{1G_k} \text{ and } F_{G_K} = N_{1G_k} + P_{0G_k} - N_{0G_k}.$$

Observe that $S[G_i,v_i,u_i] = G_{i-1}$, i=k,k-1,...,1; i.e., k applications of S to G_k yields G_0.

Therefore,

$$E_G = E_{G_k} + k\,(\Delta E_S) = (N_{0G_k} + P_{1G_k} - N_{1G_k}) + k(-1)$$

$$= (N_{0G} + P_{1G} + k - N_{1G}) - k = N_{0G} + P_{1G} - N_{1G} \text{ and}$$

$$F_G = F_{G_k} + k\,(\Delta F_S) = (N_{1G_k} + P_{0G_k} - N_{0G_k}) + k(-1)$$

$$= (N_{1G} + P_{0G} + k - N_{0G}) - k = N_{1G} + P_{0G} - N_{0G}.$$

Thus, if G is a connected proper SJS, then G satisfies $E = N_0 + P_1 - N_1$ and $F = N_1 + P_0 - N_0$. Assume G is not connected. Let $G_1, G_2, \ldots, G_n$ be the components of G. Then each eomponent of G is a connected proper SJS which implies that

$$E_{G_i} = N_{0G_i} + P_{1G_i} - N_{1G_i} \text{ and } F_{G_i} = N_{1G_i} + P_{0G_i} - N_{0G_i},$$

where $1 \leq i \leq n$.

Therefore,

$$E_G = \sum_{i=1}^{n} F_{G_i} = \sum_{i=1}^{n} N_{0G_i} + \sum_{i=1}^{n} P_{1G_i} - \sum_{i=1}^{n} N_{1G_i} = N_{0G} + P_{1G} - N_{1G},$$

and $F_G = \sum_{i=1}^{n} F_{G_i} = \sum_{i=1}^{n} N_{1G_i} + \sum_{i=1}^{n} P_{0G_i} - \sum_{i=1}^{n} N_{0G_i} = N_{1G} + P_{0G} - N_{0G}$.

Hence, if G is a proper SJS, then G satisfies the relations $E = N_0 + P_1 - N_1$ and $F = N_1 + P_0 - N_0$, and thus, G satisfies the relations $E = N_B + N_0 - B_0 - N_1$ and $F = N_B + N_1 - B_1 - N_0$.

A.1.3 Basis Order for Standard Simple Junction Structures

In this section we prove Theorem A.1.

Theorem 1: Every standard SJS satisfies the relations

$$E = N_B + N_0 - B_0 - N_1 \text{ and}$$

$$F = N_B + N_1 - B_1 - N_0.$$

The proof of Theorem 1 is preceeded by the definition of the transformation T which reduces the number of bonds formed by nodes of the same type in a standard SJS.

Let G be an arbitrary standard SJS. Assume G is not proper, i.e., G contains a bond of the form (v_1,v_2) where v_1 and v_2 are nodes of the same type.

Then for $(v_1,v_2)\varepsilon X_G$, where v_1 and v_2 are nodes of the same type, let u be a junction of a node-type distinct v_1 and v_2. Without loss of generality, if v_1 and v_2 are EN-nodes, then let u be a 1-junction. Then $T(G,v_1,v_2)$ will denote the SJS H where $V_H \equiv V_G \cup \{u\}$ and $X_H \equiv [X_G - \{(v_1,v_2)\}] \cup \{(v_1,u),(u,v_2)\}$. Observe that H is a standard SJS which contains one less bond of the form (1,1), (0,0) or (EN,EN) than G (see Figure A.4).

Note that $H = T(G,v_1,v_2)$ contains one more effort variable and yields one more independent effort constraint equation than G. Therefore, $E_H=F_G$. Similarly, H contains one more flow variable and yields one more independent flow constraint equation than G. Thus, $F_H=F_G$. Observe that if v_1 and v_2 are 1-junctions, then $N_{BH}=N_{BG}+1$, $N_{1H}=N_{1G}$, $N_{0H}=N_{0G}+1$, $B_{1H}=B_{1G}$, and $B_{0H}=B_{0G}+2$. Also, if v_1 and v_2 are 0-junctions or EN-nodes, then $N_{BH}=N_{BG}+1$, $N_{1H}=N_{1G}+1$, $N_{0H}=N_{0G}$, $B_{1H}=B_{1G}+2$, and $B_{0H}=B_{0G}$.

We are now prepared to establish the order rules for an arbitrary standard SJS.

Proof: Let G be an arbitrary standard SJS. If G is a proper SJS, then done, by Lemma A.2. Therefore, assume G is not proper. Let k_0,k_1, and k_2 be the number of bonds in G of the forms (0,0),(1,1) and (EN,EN) respectivelv. Let $k=k_0+k_1+k_2<\infty$. Without loss of generality, assume $k_0\geq 1$. Let (v_{1i},v_{2i}) denote bonds of the form (0,0) where $1\leq i\leq k_0$; (v_{1i},v_{2i}) denote bonds

of the form (1,1) where $k_0 + 1 \le i \le k_0 + k_1$; and (v_{1i}, v_{2i}) denote bonds of the form (EN,EN) where $k_0 + k_1 + 1 \le i \le k$. Also, let $G_0 \equiv G$ and $G_i \equiv T(G_{i-1}, v_{1i}, v_{2i})$ where $i = 1,2,\ldots,k$. Observe that G_k is a proper SJS. Therefore, by Lemma A.2 G_k satisfies the relations

$$E = N_B + N_0 - B_0 - N_1 \quad \text{and}$$

$$F = N_B + N_1 - B_1 - N_0 \quad \text{where} \quad N_{BG_k} = N_{BG} + k,$$

$N_{0G_k} = N_{0G} + k_1$, $N_{1G_k} = N_{1G} + k_0 + k_2$, $B_{1G_k} = B_{1G} + 2(k_0 + k_2)$, and $B_{0G_k} = B_{0G} + 2k_1$.

Recall that for standard SJS H containing adjacent nodes v_1 and v_2 of the same node-type, $E_H = E_{T(H,v_1,v_2)}$ and $F_H = F_{T(H,v_1v_2)}$.

Therefore,

$$E_G = E_{G_0} = E_{G_k} = N_{BG_k} + N_{0G_k} - B_{0G_k} - N_{1G_k}$$

$$= (N_{BG} + k_0 + k_1 + k_2) + (N_{0G} + k_1) - (B_{0G} + 2k_1) - (N_{1G} + k_0 + k_2)$$

$$= N_{BG} + N_{0G} - B_{0G} - N_{1G}, \quad \text{and}$$

$$F_G = F_{G_0} = F_{G_k} = N_{BG_k} + N_{1G_k} - B_{1G_k} - N_{0G_k}$$

$$= (N_{BG} + k_0 + k_1 + k_2) + (N_{1G} + k_0 + k_2) - (B_{1G} + 2k_0 + 2k_2) - (N_{0G} + k_1)$$

$$= N_{BG} + N_{1G} - B_{1G} - N_{0G}.$$

Hence, every standard SJS satisfies the relations

$E = N_B + N_0 - B_0 - N_1$ and $F = N_B + N_1 - B_1 - N_0$.

A.1.4 Port Basis Order for Standard Simple Junction Structures

We now show that the number of independent effort/flow variables corresponding to a given standard SJS is its

number of independent port efforts/flow; i.e., $N_E = E$ and $N_F = F$.

Corollary 1.2: Every standard SJS satisfies the relations

$$\text{(i)} \quad N_E = N_B + N_0 - B_0 - N_1$$

and

$$\text{(ii)} \quad N_F = N_B + N_1 - B_1 - N_0.$$

Proof: Let G be an arbitrary standard SJS. By Theorem G satisfies

$$E = N_B + N_0 - B_0 - N_1 \quad \text{and} \quad F = N_B + N_1 - B_1 - N_0.$$

Observe that in a node-bond incidence count, each internal bond is counted twice and each external bond is counted once in $B_0 + B_1$. Therefore,

$$(B_0 + B_1) + N_P = 2N_B; \quad \text{i.e., } N_P = 2N_B - (B_0 + B_1).$$

Then $E + F = (N_B + N_0 - B_0 - N_1) + (N_B + N_1 - B_1 - N_0)$

$= 2N_B - (B_0 + B_1) = N_P$.

Note that $N_E \leq E$, $N_F \leq F$, and $N_E + N_F = N_P$.

Assume $N_E < E$. Then $N_F \leq F$ and $N_E < E$ imply that $N_P = N_E + N_F \leq N_E + F < E + F = N_P$; i.e., $N_P < N_P$.

Thus, $N_E = E$. Similarly, $N_F = F$.

Hence, every standard SJS satisfies the relations

$$N_E = N_B + N_0 - B_0 - N_1 \quad \text{and} \quad N_F = N_B + N_1 - B_1 - N_0.$$

A.2 Basis Order and Weighted Junction Structures

Section A.2 is devoted to the development of basis order rules for standard weighted junction structures.

A.2.1 Basis Order for Standard Weighted Junction Structures

The basis order rules for a standard WJS are presented here in the form of Theorem 2.

Theorem 2: Every standard WJS satisfies the relations

$$E = N_B + N_0 - B_0 - N_1 - N_T \quad \text{and}$$

$$F = N_B + N_1 - B_1 - N_0 - N_T.$$

The proof of Theorem 2 is preceded by the definition of a transformation T which removes TF-nodes from standard WJS's. Let G be an arbitrary standard WJS. Assume G contains at least one TF-node u. Let nodes v_1 and v_2 be adjacent to u in G. Then T(G,u) will denote the WJS H where $V_H \equiv V_G - \{u\}$ and $X_H \equiv [X_G \cup \{(v_1,v_2)\}] - \{(v_1,u),(u,v_2)\}$. Note that H is a standard WJS which contains one less TF-node and one less internal bond than G.

Observe that H = T(G,u) contains one less effort variable and yields one less independent effort constraint equation than G. Therefore, $E_H = E_G$. Similarly, H contains one less flow variable and yields one less independent flow constraint equation than G. Thus, $F_H = F_G$. Note that $N_{BH} = N_{BG} - 1$, $N_{0H} = N_{0G}$, $N_{1H} = N_{1G}$, $B_{0H} = B_{0G}$, and $B_{1H} = B_{1G}$.

We are now prepared to establish the order rules for an arbitrary standard WJS.

Proof: Let G be an arbitrary standard WJS. If G does not contain a TF-node ($N_T = 0$), then G is a standard SJS and we are done, by Theorem 1. Therefore, assume G contains at least one TF-node. Let N_T be the number

of TF-nodes in G. Enumerate the TF-nodes in G $u_1, u_2, \ldots, u_{N_T}$. Let $G_0 \equiv G$ and $G_i = T(G_{i-1}, u_i)$ where $1 \leq i \leq N_T$.

Observe that G_{N_T} is a standard SJS. Therefore, G_{N_T} satisfies $E = N_B + N_0 - B_0 - N_1$ and $F = N_B + N_1 - B_1 - N_0$ where $N_{BG_{N_T}} = N_{BG} - N_T$, $N_{0G_{N_T}} = N_{0G}$, $N_{1G_{N_T}} = N_{1G}$, $B_{0G_{N_T}} = B_{0G}$, and $B_{1G_{N_T}} = B_{1G}$. Recall that if $H = T(G,u)$, then $E_G = E_H$ and $F_G = F_H$. Therefore,

$$E_G = E_{G_0} = E_{G_{N_T}} = N_{BG_{N_T}} + N_{0G_{N_T}} - B_{0G_{N_T}} - N_{1G_{N_T}}$$

$= (N_{BG} - N_T) + (N_{0G}) - (B_{0G}) - (N_{1G}) = N_{BG} + N_{0G} - B_{0G} - N_{1G} - N_T$ and

$$F_G = F_{G_0} = F_{G_{N_T}} = N_{BG_{N_T}} + N_{1G_{N_T}} - B_{1G_{N_T}} - N_{0G_{N_T}}$$

$= (N_{BG} - N_T) + (N_{1G}) - (B_{1G}) - (N_{0G}) = N_{BG} + N_{1G} - B_{1G} - N_{0G} - N_T$.

Hence, every standard WJS satisfies the relations

$E = N_B + N_0 - B_0 - N_1 - N_T$ and $F = N_B + N_1 - B_1 - N_0 - N_T$.

A.2.2 Port Basis Order For Standard Weighted Junction Structures

We now show that the number of independent effort/flow variables corresponding to a given standard WJS is its number of independent port efforts/flows.

Corollary 2.1: Every standard WJS satisfies the relations

(i) $N_E = N_B + N_0 - B_0 - N_1 - N_T$

and

(ii) $N_F = N_B + N_1 - B_1 - N_0 - N_T$.

Proof: Let G be an arbitrary standard WJS. By Theorem 2, G satisfies $E = N_B + N_0 - B_0 - N_1 - N_T$ and $F = N_B + N_1 - B_1 - N_0 - N_T$. In the proof of Theorem 2, it was observed that a standard WJS can be "reduced" to a standard SJS by a series of applications of the transformation T. Also, it was noted that each application of T decreases the TF-node count by one and decreases the bond count by one. Recall that for a standard SJS, $2N_b = N_P + (B_0 + B_1)$. Then for a standard WJS, the above observations yield

$$2N_B = N_P + (B_0 + B_1) + 2N_T; \text{ i.e., } N_P = 2N_B - (B_0 + B_1) - 2N_T.$$

Then $E + F = (N_B + N_0 - B_0 - N_1 - N_T) + (N_B + N_1 - B_1 - N_0 - N_T)$

$$= 2N_B - (B_0 + B_1) - 2N_T = N_P.$$

Note that $N_E \leq E$, $N_F \leq F$, and $N_E + N_F = N_P$.

Assume $N_E < E$. Then $N_F \leq F$ and $N_E < E$ imply that $N_P = N_E + N_F \leq N_E + F < E + F = N_P$; i.e., $N_P < N_P$.

Therefore, $N_E = E$. Similarly, $N_F < F$ implies that $N_P < N_P$.

Thus, $N_F = F$.

Hence, every standard WJS satisfies the relations $N_E = N_B + N_0 - B_0 - N_1 - N_T$ and $N_F = N_B - N_1 - B_1 - N_0 - N_T$.

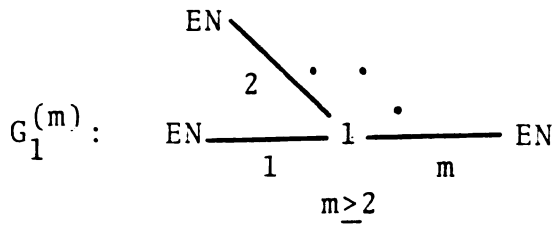


Figure A.1. A 1-junction proper SJS.

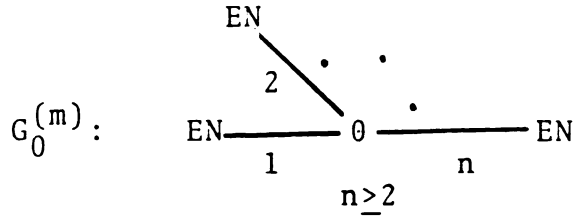
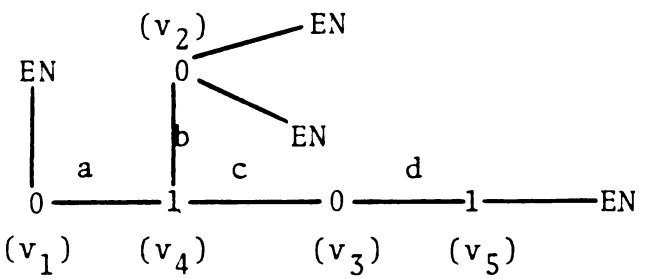


Figure A.2. A 0-junction proper SJS.

G:



$V_G^{(1)} = \{v_4, v_5\}$, $V_G^{(0)} = \{v_1, v_2, v_3\}$.

Internal Bonds - $\{(v_1,v_4),(v_2,v_4),(v_3,v_4),(v_3,v_5)\}$

$k = 3$

$S_0 = \{v_1\}$, $S_1 = \{v_4\}$, $S_2 = \{v_2, v_3\}$, $S_3 = \{v_5\}$

Junctions - $v_{0,1} \equiv v_1; v_{1,1} \equiv v_4; v_{2,1} \equiv v_2; v_{2,2} \equiv v_3; v_{3,1} \equiv v_5$.

Degrees - $\alpha_{0,1} = 2; \alpha_{1,1} = 3; \alpha_{2,1} = 3; \alpha_{2,2} = 2; \alpha_{3,1} = 2$.

$G_0^{(2)}$: EN—0—EN (v_1), a

$G_{1,1}$: EN—a—1—c—EN, b, EN (v_4)

$G_{2,1}$: EN—b—0—EN, EN (v_2)

$G_{2,2}$: EN—c—0—d—EN (v_3)

$G_{3,1}$: EN—d—1—EN (v_5)

$G_{0,0}$: EN—0—EN, a (v_1) (same as $G_0^{(2)}$)

Figure A.3. Example of Lemma A.1 construction procedure

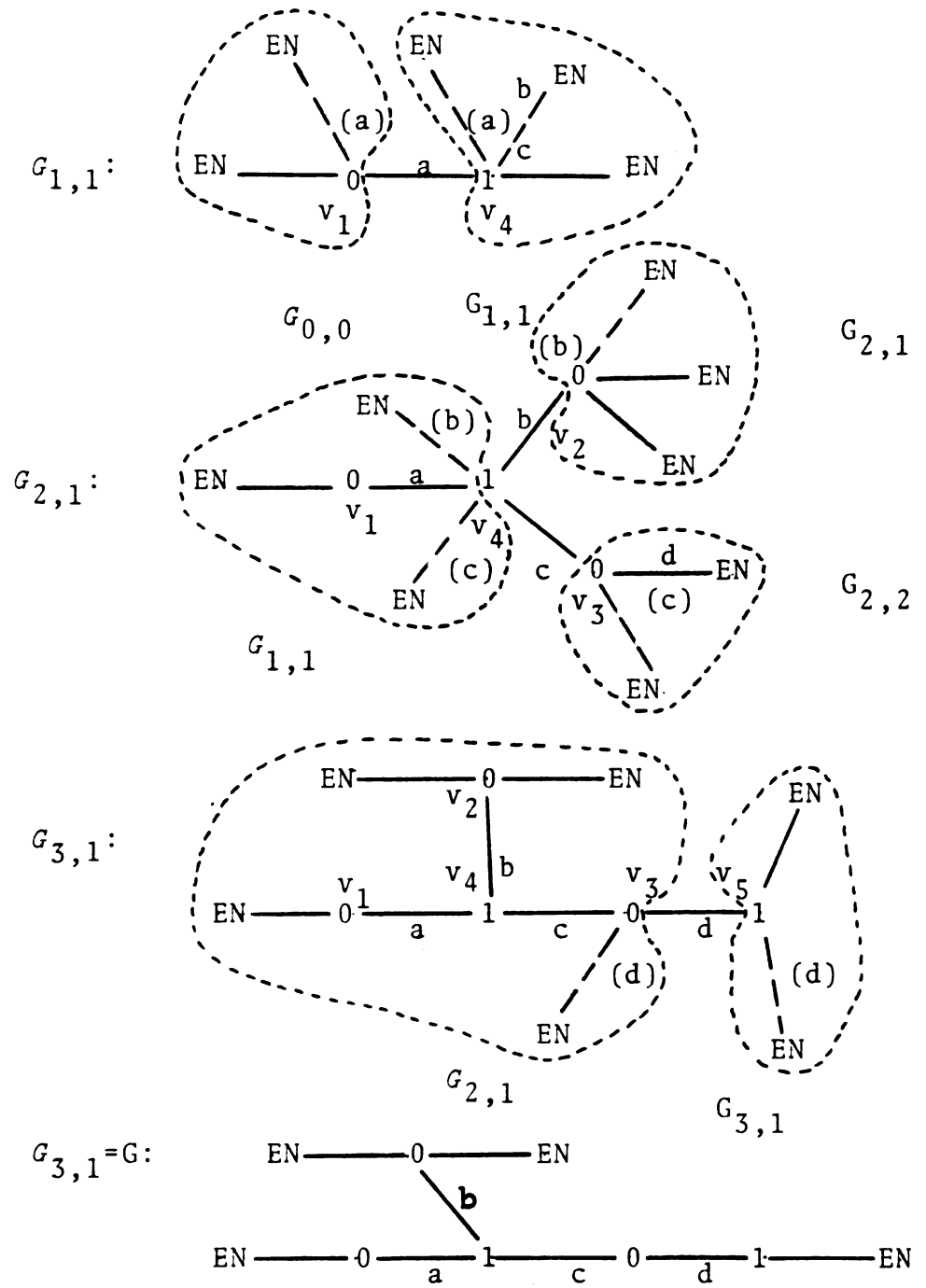


Figure A.3 (cont'd.).

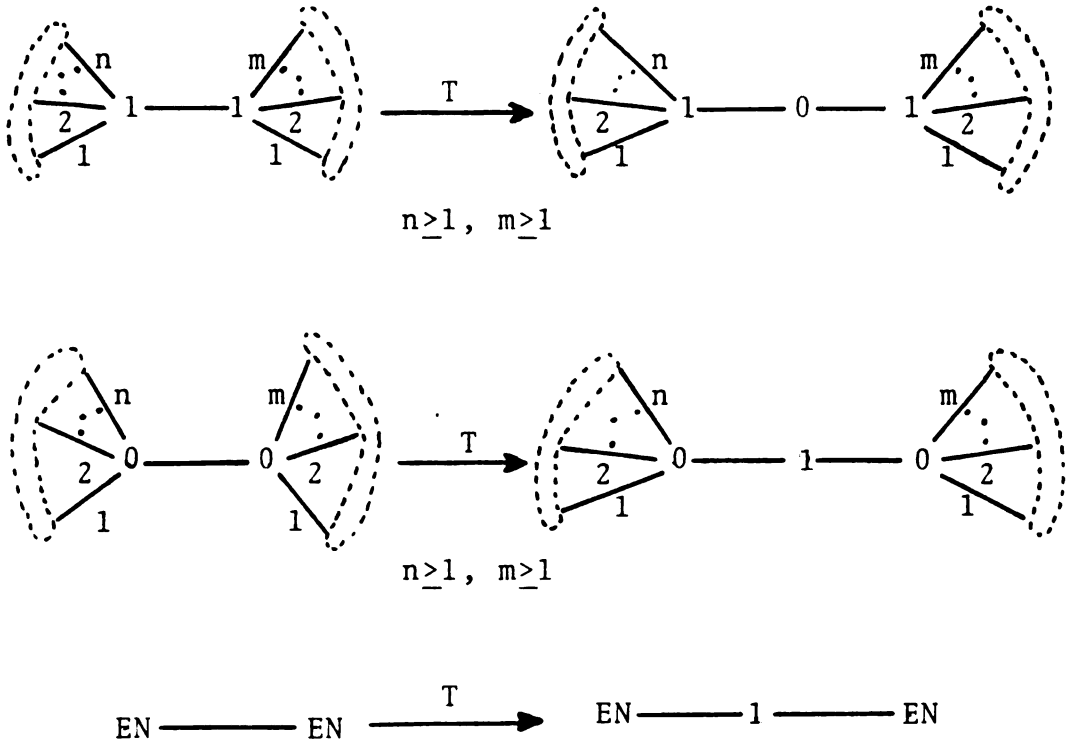


Figure A.4. Transformation for converting a standard SJS to a proper SJS

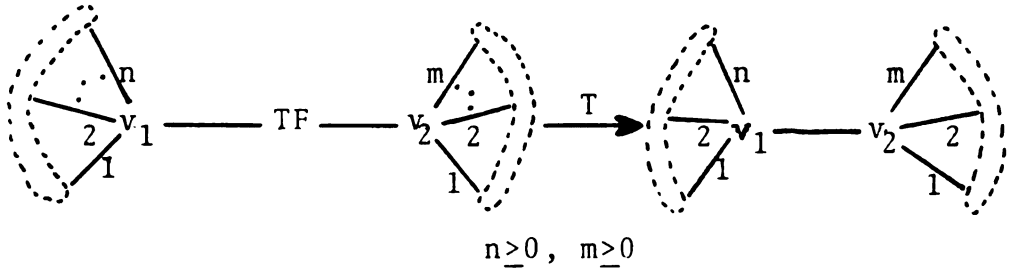


Figure A.5. Transformation for converting a WJS to a SJS.

APPENDIX B

AN APPLICATION OF THE BASIS ORDER RULES

Definition: For a WJS (weighted junction structure), a causal form is <u>feasible</u> if it does not violate any 1-junction, 0-junction, or TF node constraint, and every port bond is causally oriented.

The effort-flow variable composition of a basis can be determined from the topological properties of a WJS. The composition is given by the <u>basis order rules</u>, the general froms of which are

$$N_E = N_B + N_0 - B_0 - N_1 - N_T$$

and (1)

$$N_F = N_B + N_1 - B_1 - N_0 - N_T \ .$$

<u>Example 1</u>

The WJS in Figure B.1 has the following topological properties: $N_B = 12$, $N_0 = 2$, $N_1 = 3$, $N_T = 2$, $B_0 = 5$, $B_1 = 9$.

The basis order rules yield

$$N_E = 12 + 2 - 5 - 3 - 2 = 4$$

and

$$N_F = 12 + 3 - 9 - 2 - 2 = 2$$

This input pattern is illustrated by the causally augmented WJS in Figure B.2.

Henceforth, all weighted junction structures will be considered as n-port structures.

In the process of obtaining a "good" upper bound for the number of unlabelled feasible port bond causal orientations, three expressions for upper bounds will be derived where each successive expression requires greater knowledge of the junction structure and yields a smaller upper bound.

B.1 An Upper Bound For C As A Direct Application Of The Basis Order Rules.

A question of particular interest concerns the number of distinct port-variable bases which an n-port possesses. The basis order rules yield an upper bound for the number of such bases.

From the fact that every port bond can accept exactly two causal orientations it is easily seen that

$$C \leq U_0 \qquad (2)$$

where C is the total number of unlabelled feasible port bond causal orientation and $U_0 = 2^{N_P}$.

However, an improvement over U_0 can be obtained by a direct application of the basis order rules.

Given N_P port bond with N_E efforts as inputs, a (generally) smaller upper bound can be expressed as

$$C \leq U_1 \tag{3}$$

where

$$U_1 = \binom{N_P}{N_E} \tag{4}$$

and

$$\binom{n}{m} = \begin{cases} \frac{n!}{m!(n-m)!} & \text{if } 0 \leq m \leq n, \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

for integers n and m.

Notice that U_1 represents a significant reduction from the coarse upper bound U_0.

Noting that

$$N_P = N_E + N_F, \tag{6}$$

one obtains the related form

$$\binom{N_P}{N_E} = \frac{N_P!}{N_E!N_F!} = \binom{N_P}{N_F} . \tag{7}$$

As would be expected, the results are symmetric with respect to effort and flow variables.

The inequality $U_1 \leq U_0$ can be demonstrated by expressing 2^{N_P} as a binomial expansion.

Recalling the Binomial Expansion Theory,

$$(a + b)^n = \sum_{k=0}^{n} \binom{n}{k} a^{n-k} b^k,$$

let $a = 1$, $b = 1$, and $n = N_P$. Then

$$2^{N_P} = (1 + 1)^{N_P} = \sum_{k=0}^{N_P} \binom{N_P}{k} . \qquad (8)$$

Then for $0 \leq N_E \leq N_P$ and $0 \leq N_F \leq N_P$, $\binom{N_P}{N_E}$ and $\binom{N_P}{N_F}$ are merely symmetrical terms in (8). Thus $U_1 \leq U_0$. In particular, if $N_P \geq 1$, then $U_1 < U_0$.

Example 2

Referring to Figure B.1,

$$N_P = 6, \; N_E = 4, \text{ and } \; N_F = 2.$$

Then

$$U_0 = 2^6 = 64$$

and

$$U_1 = \binom{6}{4} = \frac{6!}{4!2!} = 15 \; .$$

Thus, $C \leq 15 < 64$.

B.2 A Refined Upper Bound for C

In general, U_1 can be improved upon considering the constraint equations associated with WJS elements. It will be assumed that the WJS of interest contains no external TF elements. This assumption results in no loss of generality, due to the causal properties of TF element [25].

Let e be the number of port effort inputs to external 0-junctions. Similarly, let f be the number of port flow inputs to external 1-junctions.

For a given e, if "e" port efforts are inputs to the A_0 external 0-junctions, then $P_0 - e$ port flows are inputs to the remaining $P_0 - e$ port bonds which are incident to 0-junctions. Thus, $N_F - (P_0 - e)$ port flows are inputs to the A_1 external 1-junctions, and $N_E - e$ port efforts are inputs to the remaining $P_1 - (N_F - P_0 + e)$ port bonds which are incident to 1-junctions.

Note that

$$f = N_F - P_0 + e \tag{9}$$

and

$$P_1 - (N_F - P_0 + e) = (P_1 + P_0) - N_F - e$$

$$= N_P - N_F - e = N_E - e \ .$$

Then

$$C \leq U_2 \tag{10}$$

where

$$U_2 = \sum_{e=L_E}^{M_E} \binom{A_0}{e}\binom{A_1}{N_F-P_0+e} \tag{11}$$

for some M_E and L_E.

If $N_E \leq A_0$, then $M_E = N_E$. If $N_E > A_0$, then $M_E = A_0$. Thus,

$$M_E = \min(N_E, A_0) \ . \tag{12}$$

Suppose $P_1 \leq N_E$. Then there are at least $N_E - P_1$ port effort inputs to the A_0 external 0-junctions. Then $L_E = N_E - P_1$. Observe that $f = 0$ when $e = N_E - P_1$.

Suppose $P_1 > N_E$. Then $N_F > P_0$, and there can be a minimum of zero port efforts inputs to the A_0 external 0-junctions. Then $L_E = 0$. Observe that $f = N_F - P_0 > 0$ when $e = 0$. Thus,

$$L_E = \max\ (0,\ N_E - P_1) \tag{13}$$

Equation (11) is symmetric in e and f. Thus, U_2 can be formulated in terms of specified port flow inputs, if desired.

By (9) we have $e = N_E - P_1 + f$, since $N_E + N_F = P_0 + P_1$. Then, by (5),

$$\sum_{e=L_E}^{M_E} \binom{A_0}{e}\binom{A_1}{N_F-P_0+e} = \sum_{e=L_E}^{N_E} \binom{A_0}{e}\binom{A_1}{N_F-P_0+e} = \sum_{f=L_F}^{P_1} \binom{A_0}{N_E-P_1+f}\binom{A_1}{f}$$

$$= \sum_{f=L_F}^{A_1} \binom{A_1}{N_E-P_1+f}\binom{A_1}{f} = \sum_{f=L_F}^{M_F} \binom{A_0}{N_E-P_1+f}\binom{A_1}{f}$$

where $M_F = \min (N_F, A_1)$ and $L_F = \max (0, N_F - P_0)$.

It will now be shown that $U_2 \leq U_1$.

Suppose e port efforts are inputs to the A_0 external 0-junctions. Then there are at most $\binom{A_1}{N_F-P_0+e}$ causally consistent input assignments of $N_E - e$ port efforts to the P_1 port bonds incident to 1-junctions, given causally consistent input assignments of $N_F - P_0 + e$ port flows to the P_1 port bonds incident to 1-junctions.

Observe that $\binom{P_1}{N_E-e}$ is the number of <u>unrestricted</u> input assignments of $N_E - e$ port efforts to the P_1 port bonds incident to 1-junctions. Therefore,

$$\binom{A_1}{N_F-P_0+e} \leq \binom{P_1}{N_E-e} \qquad (14)$$

Therefore, by (5) and (14),

$$\sum_{e=L_E}^{M_E} \binom{A_0}{e}\binom{A_1}{N_F-P_0+e} = \sum_{e=0}^{N_E} \binom{A_0}{e}\binom{A_1}{N_F-P_0+e}$$

$$\leq \sum_{e=0}^{N_E} \binom{A_0}{e}\binom{P_1}{N_E-e} \leq \sum_{e=0}^{N_E} \binom{P_0}{e}\binom{P_1}{N_E-e} = \binom{P_0+P_1}{N_E} = \binom{N_P}{N_E} .$$

Therefore, $U_2 \leq U_1$. In particular, whenever $A_1 < P_1$ or $A_0 < P_0$, U_2 is a significant improvement over U_1.

Example 3

Again referring to Figure B.1, $A_0 = 0$, $A_1 = 3$, $P_0 = 0$, $P_1 = 6$, $N_E = 4$, and $N_F = 2$. Therefore, $M_E = \min(4, 0) = 0$ and $L_E = \max(0, 4-6) = 0$. Thus,

$$C \leq U_2 = \binom{0}{0}\binom{3}{2} = 3,$$

which is much lower than the upper bound of 15 obtained in example 2. In addition, the junction structure in Figure B.1 has exactly three unlabelled causal orientations which are given in Figure B.3, hence, $C = U_2$ in this example.

The significance of U_2 is captured by the following theorem.

Theorem 3: The number of distinct basis variable sets for a WJS transformation is bounded above by

$$U_2 = \sum_{e=L_E}^{M_E} \binom{A_0}{e}\binom{A_1}{N_F-P_0+e} = \sum_{f=L_F}^{M_F} \binom{A_0}{N_E-P_1+f}\binom{A_1}{f}$$

where $M_E = \min(N_E, A_0)$, $L_E = \max(0, N_E - P_L)$, $M_F = \min(N_F, A_1)$, and $L_F = \max(0, N_F - P_0)$.

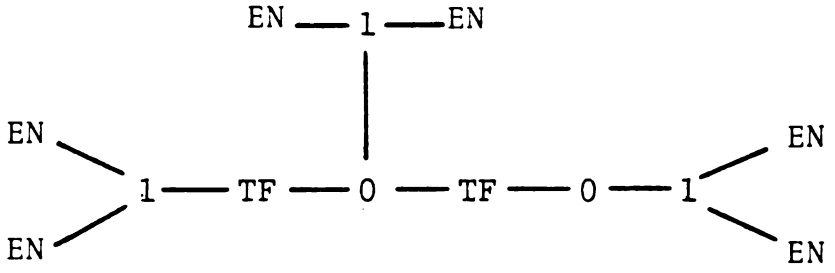


Figure B.1. Example of a weighted junction structure.

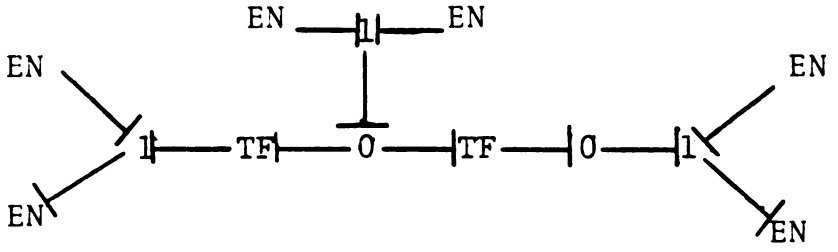


Figure B.2. Causally augmented weighted junction structure.

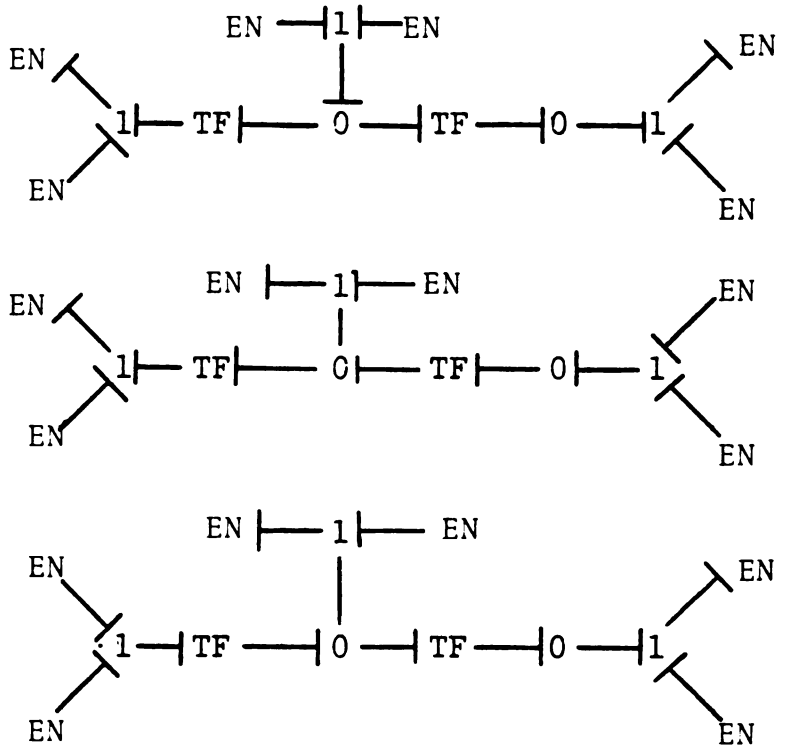


Figure B.3. Unlabelled causal orientations of a weighted junction structure.

APPENDIX C

THE RESOLUTION OF A CONFLICT RESULTING FROM A PORT BOND CAUSAL ORIENTATION

Property 2.1: Let G be a junction structure with a port bond b. Suppose a causal orientation (and subsequent causal extension) of b results in a causal conflict, then the opposite causal orientation (and subsequent causal extension) of b will not yield a causal conflict.

Proof:

It will be assumed that each causal orientation of b is followed by the causal extension process, and that G is acausal (since the pruning process of Appendix E can be initially applied to remove all causally oriented bonds and causally completed nodes).

G contains a node of degree greater than two since a causal orientation of b results in a causal conflict. From a graph theoretic perspective, a causal orientation of b defines a "walk" of G [18]. Let EN_o be the field-node incident to b. Then there exists a shortest path in G joining EN_o and a JS-node, V_o, of degree greater than two such that (1) every JS-node (exclusively) between EN_o and V_o has degree two, and (2) there exists no shorter path in G joining EN_o and a JS-node of degree greater than two.

Let d denote the path bond incident to V_o. A causal orientation of b resulting in a causal conflict implies that the resulting causal orientation of d gives d strong causal implication with tespect to V_o. The reversal of the causal orientation of b results in a reversal of d's causal orientation, since EN_o and V_o are joined by a sequence of JS-nodes of degree two. This gives d weak causal implication with respect to V_o, which leaves V_o causally incomplete; consequently, no causal information propagates beyond V_o and no causal conflict can result. Hence, if a causal orientation of b yields a causal conflict, then the opposite causal orientation of b will not yield a causal conflict.

APPENDIX D

THE IMPACT OF THE STANDARD SEQUENTIAL CAUSALITY ASSIGNMENT PROCEDURE (SSCAP) ON THE REDUCED JUNCTION MATRIX

Theorem D.1: Let G be a bond graph with an algebraically reducible junction structure. Assume that G can be completely and consistently causally oriented by SSCAP. Then dependent storage field inputs are determined by source field and independent storage field outputs. Moreover, no dissipation field input is determined by a dependent storage field output.

Proof:

It will be assumed that (1) a causal assignment is always followed by causal extension, (2) each field multiport is a one-port, and (3) each field multiport is adjacent to a 0-junction or 1-junction. It has been shown that a linear time-invariant field multiport with n ports can be replaced by n one-port field multiports [25]. Thus, for this reason and due to the causal characteristics of junction structure nodes, the above assumptions result in no loss of generality.

Consider the acasual representation of $G_1 = G$ given in Figure D.1 where n_1 is the number of sources, m_1 is the number of storage multiports, and p_1 is the number of dissipation multiports. Without loss of generality, assume $n_1 \geq 1$.

In the following discussion only consistent causal orientations of source bonds will be considered.

Causally orient the bond incident to an arbitrary source in G_1. Let u_1 denote the output from this source. Prune (see Appendix E) from G_1 all resulting causally oriented bonds and causally completed nodes. (Note that no causal conflicts can result from source bond orientations due to the theorem's hypothesis). Let G_2 be the bond graph obtained from G_1 by the pruning procedure (see Figure D.2). If any dependent storage field bond or any dissipation field bond is pruned from G_1, then each corresponding dependent storage field input or dissipation field input is determined by u_1 together with prior information. If $G_2 \neq \emptyset$, then done. Assume $G_2 \neq \emptyset$. Then G_2 has the acausal form given in Figure D.1 where n_1, m_1, and p_1 are replaced by n_2, m_2, and p_2 respectively. Observe that if $n_2 \geq 1$, then the source bond causal orientation process together with the bond graph pruning procedure can be repeated (at most n_1 times) until all source bonds of G have been causally oriented. Then there exists a smallest positive integer $k \leq n_1+1$ such that G_k contains no sources. (If $n_1=0$, then $k=1$). $n_1 \geq 1 \Rightarrow k \geq 2$. Then each dependent storage input and each dissipation input specified in G_h is determined by $u_1, u_2, \ldots, u_h$ where $1 \leq h \leq k-1$ and u_h is defined similar to u_1.

Consider G_k. G_k has the acausal form given in Figure D.3. If $m_k=0$, then done. Assume $m_k>0$. Assign integral causality to an arbitrary storage bond, b, in G_k. If a causal conflict results, then restore G_k to its acausal form and select a different storage bond to causally orient. In this case, property 2.2 of the causal extension process indicates that sufficient input information was available to determine the variables associated with bond b, i.e., the storage element incident to b is dependent and its input is determined by $u_1, u_2, \ldots, u_{k-1}$. Suppose the integral causal orientation of b does not result in a causal conflict in G_k. Let x_1 be the resulting output of the storage node incident to the oriented b. Let G_{k+1} be the bond graph obtained from G_k by the pruning procedure. If any dependent storage bond or any dissipation bond is pruned from G_k, then each corresponding dependent storage input or dissipation input isdetermined by x_1 in G_k, and therefore, is determined by $u_1, u_2, \ldots, u_{k-1}, x_1$ in G.

G_{k+1} has the acasual form given in Figure D.3 with k replaced by k+1. If $m_{k+1}=0$, then done. If $m_{k+1}\neq 0$, then the above storage bond integral orientation and graph pruning process can be repeated (at most m_k times) until the integral orientation of any remaining storage bond yields a causal conflict, i.e., there is a smallest integer $\ell\geq 0$ such that either the integral causal orientation of each storage bond in $G_{k+\ell}$ results in a causal inconsistency or $m_{k+\ell}=0$ where $\ell\leq m_k$, As noted above, if the integral causal orientation of

a storage bond results in a causal conflict, then the bond's associated variables are determined by previously specified source and (independent) storage outputs. Then each dependent storage input and each dissipation input specified in G_h is determined by

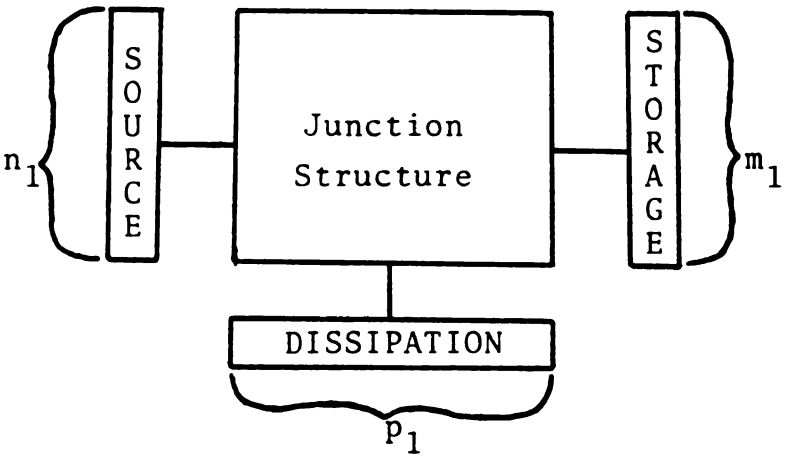
$u_1, \ldots, u_h$ if $1 \leq h \leq k-1$

$u_1, \ldots, u_{k-1}$ if $\ell = 0$

$u_1, \ldots, u_{k-1}, x_1, \ldots, x_{h-k+1}$

if $k \leq h \leq k+\ell-1$ and $\ell > 0$

where each x_i is defined similar to x_1. Thus, all dependent storage inputs are determined by source and independent storage outputs, and no dissipation inputs is determined by a dependent storage output.



n_1, m_1, and p_1 are nonnegative integers

Figure D.1. Symbolic bond graph representation with fields identified.

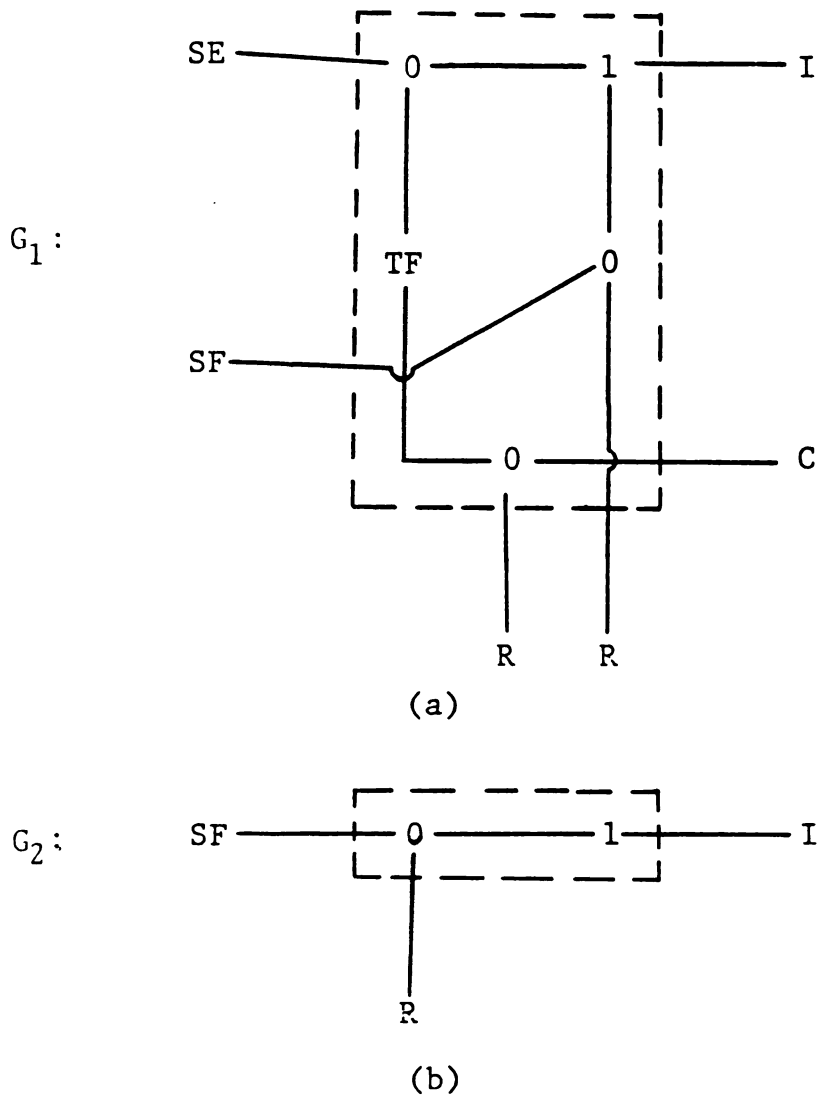
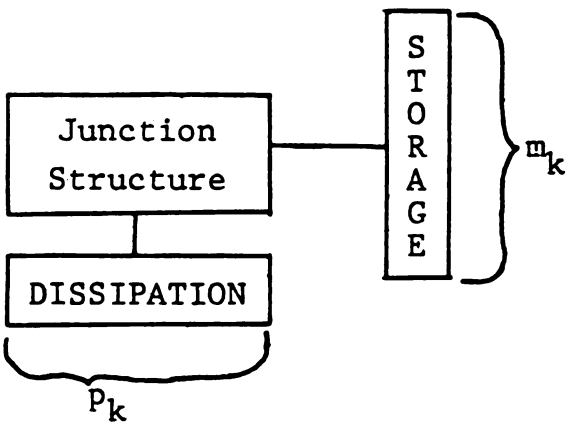


Figure D.2. Example of subgraph generation by pruning. (a) Bond graph G_1 with source bond causally oriented. (b) Bond graph G_2 obtain from G_1 by the pruning process.



m_k and p_k are nonnegative integers

Figure D.3. Symbolic bond graph representation with source field pruned.

APPENDIX E

PRUNING AND SOME ASPECTS OF JUNCTION STRUCTURES

Let G be a junction structure and d be an arbitrary bond in G. Assume that G is acasual. Then a casual set with respect to d is the set of casually oriented bonds and causally completed nodes in G which result from the casual extension of a casual orientation of d. "S(d)" will denote a casual set with respect to d; this notation assumes that the casual orientation of d is known.

Consider the JS given in Figure E.1. If bond b_{67} is causally oriented so that the effort variable is an input to node 6, then $S(b_{67}) = \{b_{67}\}$. If bond b_{67} is causally oriented so that the effort variable is an input to node 7, then $S(b_{67}) = V_G \cup X_G$ where V_G is the node set of G and X_G is the bond set of G.

Note that (1) a casual set does not contain any casually incomplete notes; and (2) there is at least one and at most two casual sets with respect to bond b in G, since b has exactly two casual orientations and the casual extension process implies the unique and exhaustive propagation of input information.

Remark E.1: If G is a tree JS, then all causally complete nodes in S(b) are consistent since there is a unique path between distinct nodes in a tree graph.

Let G be a JS and d be an arbitrary bond in G.

Arbitrarily, causally oriented d. Then "prune S(d) from G" will mean delete from G the bonds and nodes in S(d). "G-S(d)" will denote the structure obtained by pruning S(d) from G.

Remark E.2: Pruning does not introduce additional port-bonds since it does not introduce additional EN-nodes.

Remark E.3: The removal or "pruning" of causally oriented bonds from a causally incomplete junction does not alter the input information which is required to causally complete the junction. That is, a causally incomplete junction has $n \geq 2$ inputs to be determined (i.e., bonds to be causally oriented) of which exactly one must have strong causal implication. Thus, once an input of strong causal implication is known or n-1 inputs of weak causal implication are known, the junction can be causally completed in a consistent fashion.

Remark E.4: It follows from Property 2.4 of the causal extension process that if G is a JS then G-S(b) is a JS, where G-S(b) will be called the "null bond graph" if G-S(b) is empty. A property of JS trees can now be given.

Theorem E.1: Every JS tree can be causally completed in a consistent fashion by the sequential causal orientation (and causal extension) of its port-bonds.

Proof:

In the subsequent discussion, it will be assumed that (1) each bond causal orientation is followed by the causal extension process, and (2) for each set S(p), the bond p has been arbitrarily causally oriented.

Let T_1 be an arbitrary JS tree and R_1 be the set of all port bonds in T_1. Then $2 \leq o(R_1) < \infty$. Let $p_1 \in R_1$. If $R_1 \cap S(p_1) = R_1$ (i.e. all port bonds of T_1 are in $S(p_1)$), then done by the properties of pruning and the causal extension process.

Assume $R_1 \cap S(p_1) \neq R_1$. Let $T_2 = T_1 - S(p_1)$ and $R_2 = R_1 - [R_1 \cap S(p_1)]$. By the properties of the pruning process, T_2 is a JS forest. Therefore, $o(R_2)$ is greater than or equal to twice the number of components of T_2. Let $p_2 \in R_2$. Observe that $S(p_1) \cap S(p_2) = \emptyset$.

In general, if $R_i \cap S(p_i) \neq R_i$, then let $T_{i+1} = T_i - S(p_i)$ and $R_{i+1} = R_1 - [R_1 \cap \bigcup_{j=i}^{i} S(p_j)]$ where $p_j \in R_j$ for $1 \leq j \leq i$ and $i \geq 1$.

T_i is a JS forest and $o(R_i)$ is greater than or equal to twice the number of components of T_i for each i.

$o(R_1) < \infty$ implies there exists a positive integer K such that $R_K \cap S(p_K) = R_K$; i.e., after the $K^{\underline{th}}$ pruning, all port bonds (and thus, all bonds) have been pruned from T_1. This occurs if and only if all bonds of T_1 have been causally oriented.

Recall that the causal extension process results in consistent causal assignments in JS trees, and pruning does not affect the consistency of causal assignments.

Then $V \cup X = \bigcup_{i=1}^{K} S(p_i)$ where V is the node set of T_1, X is

the bond set of T_1, $S(p_i) \cap S(p_j) = \emptyset$ if $i \neq j$, and each $S(p_i)$ is a subset of bonds and nodes in T_1 which have been causally assigned in a consistent fashion.

Hence, T_1 has been causally completed in a consistent fashion by the sequential causal orientation of its port-bonds.

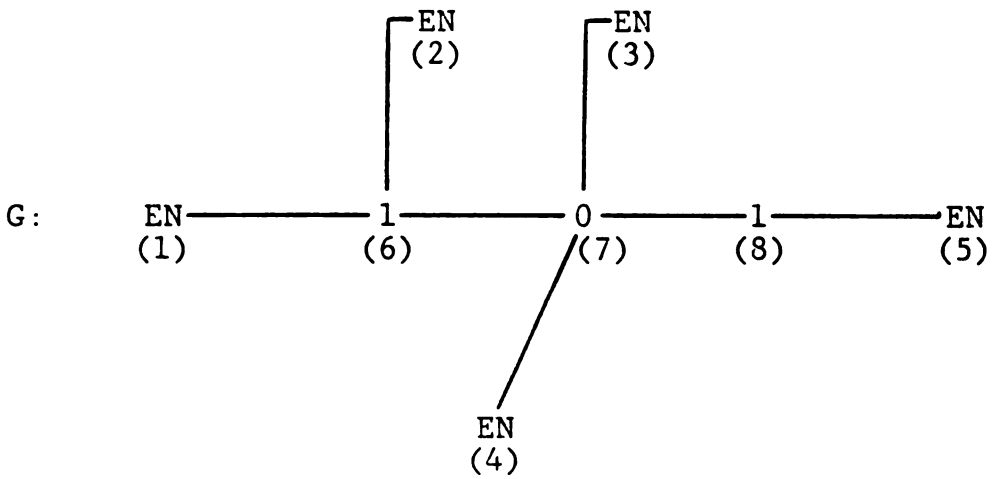
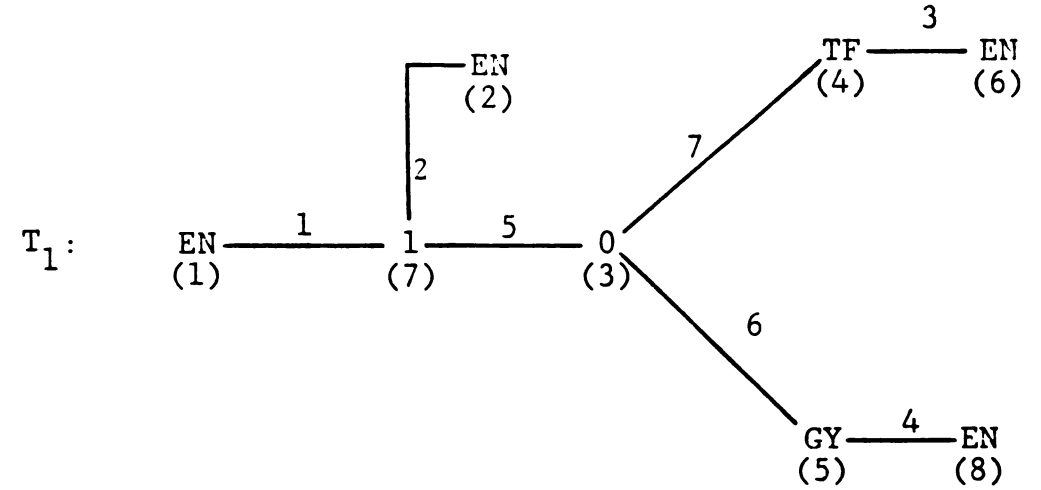
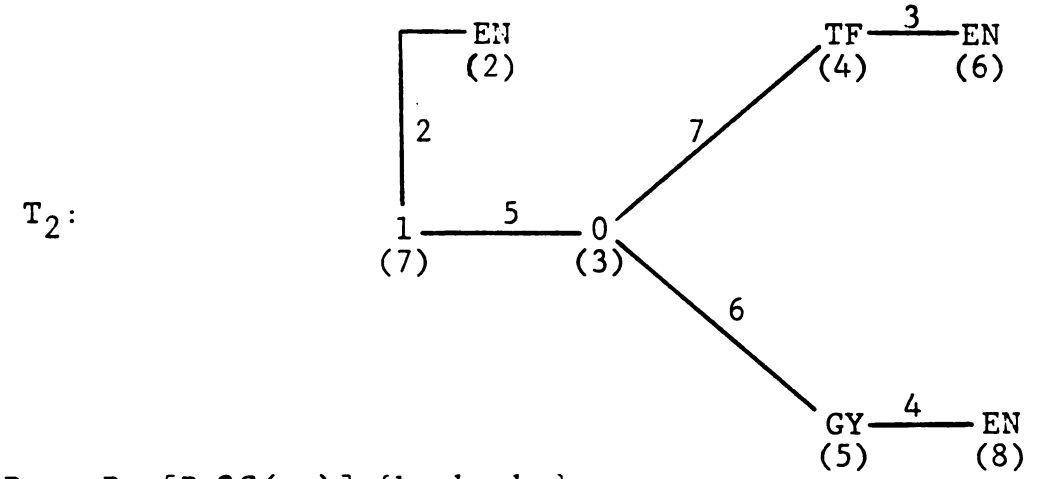


Figure E.1. Junction structure with labelled nodes.



$R_1 = \{b_1, b_2, b_3, b_4\}$
$p_1 = b_1$ with effort input to v_7
$S(p_1) = \{b_1, v_1\}$



$R_2 = R_1 - [R_1 \cap S(p_1)] = \{b_2, b_3, b_4\}$
$p_2 = b_2$ with effort input to v_7
$S(p_2) = \{b_2, b_3, b_4, b_5, b_6, b_7, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$

Figure E.2. Example of the pruning procedure.

$b_i \equiv$ bond i and $v_i \equiv$ node i

APPENDIX F

COMPUTER SUBROUTINES

Table 5. Subroutine For Causal Complex Identification

```
      SUBROUTINE CYCCMPX
      INTEGER CMX
      COMMON /BK1/IN,OUT,IERRF,IPRLST(20),HEAD(20)
      COMMON/BK3/NEL,NBD,NFBG,NLBG,IELLST(50),IELNAM(50),NBIMX(100),
     +IBMX(50,2),NPTR(51),NMCR,NMCPT,NTYP(9)
      COMMON/BK5/CMX(100),MELPNT(10),MBND(10),MIT(10),NTEMP(30),MNSTKM,
     +MNSTKN,MTOP,NTOP,LEVEL,IBLIS(13),ICCNODE(13),IBPNT(4),NODLIS(4),
     +JPNT,KBOND,INODE,MAXBD,MAXND
C*****************************************************************************
C CYCCMPX IDENTIFIES CAUSAL COMPLEXES AND STORES THEIR ORDERED BONDS IN IBLIS
C AND STORES THEIR NODES IN ICCNODE. NTEMP STACKS CAUSAL COMPLEX NODES DURING
C PROCESSING. IBPNT PARTITIONS IBLIS INTO COMPLEXES. NODLIS PARTITIONS ICCNODE
C INTO COMPLEXES. JPNT IS THE NUMBER OF CAUSAL COMPLEXES FOUND. KBOND IS THE
C NUMBER OF BONDS IN IBLIS. INODE IS THE NUMBER OF NODES IN ICCNODE. MAXBD,
C MAXND, AND MNSTKN ARE THE DIMENSIONS OF IBLIS, ICCNODE, AND NTEMP RESPECTIVELY
C THE DIMENSIONS OF IBPNT AND NODLIS ARE ONE PLUS ONE-FOURTH OF THE DIMENSION
C OF ICCNODE.
C
C      IERRF=500 STACK OVERFLOW ON IBLIS
C            510 STACK OVERFLOW ON NTEMP
C            520 STACK OVERFLOW ON ICCNODE
C*****************************************************************************
C
C FIND INITIAL ACAUSAL BOND
      NBD2=2*NBD
      DO 90 I=1,NBD2
      ID=I
      IF(CMX(I).NE.1)GOTO 90
      JPNT=JPNT+1
      IBPNT(JPNT)=KBOND+1
      ITOP=0
      NODLIS(JPNT)=INODE+1
C ADD ACAUSAL BOND TO IBLIS
   10 KBOND=KBOND+1
      IF(KBOND.GT.MAXBD)GOTO 130
      NP=NBIMX(ID)
      IBLIS(KBOND)=NP
C REMOVE BOND FROM ACAUSAL BOND LIST
      DO 20 K=1,NBD2
   20 IF(NBIMX(K).EQ.NP)CMX(K)=-CMX(K)
C PUSH INCIDENT NODES ONTO STACK AND STORE
      IF(ITOP.EQ.0)GOTO 40
      DO 30 K=1,ITOP
   30 IF(NTEMP(K).EQ.IBMX(NP,1))GOTO 50
   40 ITOP=ITOP+1
      INODE=INODE+1
      IF(ITOP.GT.MNSTKN)GOTO 140
      IF(INODE.GT.MAXND)GOTO 150
```

Table 5. (continued)

```
      NTEMP(ITOP)=IBMX(NP,1)
      ICCNODE(INODE)=IBMX(NP,1)
   50 CONTINUE
      DO 60 K=1,ITOP
   60 IF(NTEMP(K).EQ.IBMX(NP,2))GOTO 70
      ITOP=ITOP+1
      INODE=INODE+1
      IF(ITOP.GT.MASTKN)GOTO 140
      IF(INODE.GT.MAXND)GOTO 150
      NTEMP(ITOP)=IBMX(NP,2)
      ICCNODE(INODE)=IBMX(NP,2)
C POP FIRST NODE FROM STACK
   70 NODE=NTEMP(ITOP)
C FIND INCIDENT ACAUSAL BOND
      N1=NPTR(NODE)
      N2=NPTR(NODE+1)-1
      DO 80 K=N1,N2
      IF(CMX(K).NE.1)GOTO 80
      ID=K
      GOTO 10
   80 CONTINUE
      ITOP=ITOP-1
      IF(ITOP.EQ.0)GOTO 90
      GOTO 70
   90 CONTINUE
C RESET CMX
      DO 100 I=1,NBD2
  100 CMX(I)=IABS(CMX(I))
C UPDATE POINTER VECTORS
      IBPNT(JPNT+1)=KBOND+1
      NODLIS(JPNT+1)=INODE+1
C SORT BONDS INTO ASCENDING ORDER WITHIN EACH CAUSAL COMPLEX
      DO 110 I=1,JPNT
      K1=IBPNT(I)+1
      K2=IBPNT(I+1)-1
      DO 110 K=K1,K2
      J1=IBPNT(I)
      J2=K-1
      DO 110 J=J1,J2
      IF(IBLIS(K).GE.IBLIS(J))GOTO 11
      IHOLD=IBLIS(J)
      IBLIS(J)=IBLIS(K)
      IBLIS(K)=IHOLD
  110 CONTINUE
```

Table 5. (continued)

```
C PRINT CAUSAL COMPLEXES
      DO 120 I=1,JFNT
      K1=IEPNT(I)
      K2=IBPNT(I+1)-1
      WRITE 1000,I
      WRITE 2000,(IBLIS(J),J=K1,K2)
  120 CONTINUE
      RETURN
  130 IERRF=500
      GOTO 160
  140 IERRF=510
      GOTO 160
  150 IERRF=520
  160 WRITE 3000,IERRF
      RETURN
 1000 FORMAT(*  THE BONDS OF CAUSAL COMPLEX NUMBER *,I2,* ARE :*)
 2000 FORMAT(2X,20I4,/,2X,20I4)
 3000 FORMAT(*0 ERROR NUMBER *,I4,* IN CYCCMPX*)
      END
```

Table 6. Subroutine For The Determination Of Causal Complex Solvability

```
      SUBROUTINE TESTPLX
      INTEGER CMX
      COMMON /BK1/IN,OUT,IERRF,IPRLST(20),HEAD(20)
      COMMON/BK3/NEL,NBD,NPBG,NLBG,IELLST(50),IELMAP(50),NBIMX(100),
     +IBMX(50,2),NPTR(51),NMCR,NMCPT,NTYP(9)
      COMMON/BK5/CMX(100),MELPNT(10),MBND(10),MIT(10),NTEMP(30),MNSTKM,
     +MASTKN,MTOP,NTCP,LEVEL,IBLIS(13),ICCNODE(13),IBPNT(4),NODLIS(4),
     +JPNT,KBCND,INODE,MAXBD,MAXND
      COMMON /BK6/PAR(100),IPTR(51),IPFLG,NPAR
      COMMON/BK8/A(60),IAPTR(60),LENA,MAXA,ASUM(50),IASUM,B(10),
     +IBPTR(10),LENB,MAXB,BSUM(50),DET,E( 5),IEPTR( 5),LENE,MAXE,
     +FL(19),IFLPTR(19),LENFL,MAXFL,FS(25),IFSPTR(25),LENFS,MAXFS,
     +IFS12,IFS21,IFS22,KL(50),KM(50),SDATA(148),IPS(148),JS12,JS13,
     +JS14,JS21,JS24,JS31,JS32,JS34,JS41,LENS,MAXDATA,RMAT1( 98),
     +IRMAT1( 98),IRMAT2( 98),JS42,JS43,JS44,
     +LENMAT1,MATDIM1,RMAT2( 98),LENMAT2,MATDIM2,T(150),ITPTR(150),
     +IT0,IT1,IT2,IT3,IT4,MAXT,WK( 98),IWK( 98),MAXWK

C*****************************************************************************
C--TESTPLX DETERMINES IF CAUSAL COMPLEXES ARE ALGEBRAICALLY REDUCIBLE. SDATA/IPS
C--IS USED AS A WORKING ARRAY. TESTPLX CALLS INPRD.
C
C-----IERRF=200 IRREDUCIBLE CAUSAL COMPLEX
C           210 OVERFLOW IN SDATA ARRAY
C           220 OVERFLOW IN RMAT1 ARRAY
C*****************************************************************************
C
C NO CAUSAL COMPLEXES FOUND
      IF(JPNT.LE.0)RETURN
C
C IDENTIFY COMPLEX BONDS IN CMX
      NBD2=2*NBD
      DO 10 I=1,KBCND
      DO 10 J=1,NBD2
   10 IF(NBIMX(J).EQ.IBLIS(I))CMX(J)=-CMX(J)
C
C FIND NEXT CAUSAL COMPLEX
      DO 150 I=1,JPNT
      LIB1=IBPNT(I)
      LIB2=IBPNT(I+1)-1
      LIN1=NODLIS(I)
      LIN2=NODLIS(I+1)-1
      LENG=LIB2-LIB1+1
      LEN=LENG+LENG
      NBIN2=LEN
```

Table 6. (continued)

```
C FIND NEXT CALSAL COMPLEX NODE
      IPUT=0
      DO 120 J=LIN1,LIN2
      NODE= ICCNODE(J)
      NTAP=IELLST(NODE)
      IF(NTAP.GE.8)GOTO 90
      INDEX1=NPTR(NODE)
      INDEX2=NPTR(NODE+1)-1
      IT=-3
      IF(NTAP.EQ.7)IT=-6
C RESET ACTIVATION FLAG
      IACT=0
      DO 20 J1=INDEX1,INDEX2
   20 IF(CMX(J1).EQ.IT)GOTO 30
C ACTIVATION FOUND--SUPPRESS IDENTITY RELATIONS
      IACT=1
      DO 25 J1=INDEX1,INDEX2
   25 IF(CMX(J1).EQ.(IT+1))GOTO 30
C SETUP FOR JUNCTIONS
   30 CONTINUE
      IT=8+IT
      CMX(J1)=-CMX(J1)
      IOUT=NBIMX(J1)
      IDIREC=1
      IF(IBMX(IOUT,1).NE.NODE)IDIREC=2
C OBTAIN ROW INDEX FOR OUTPUT FROM JUNCTION
      DO 4  L1=LIB1,LIB2
   40 IF(ICUT.EQ.IBLIS(L1))GOTO 50
   50 IRODEX=L1+LENG
      IF(NTAP.NE.6)IRODEX=L1

C LOCATE SETUP VECTOR
      DO 80 L1=INDEX1,INDEX2
      IF(NBIMX(L1).EQ.IOUT)GOTO 80
      IF(CMX(L1).GE.0)GOTO 80
      CMX(L1)=-CMX(L1)
      IBCN=NBIMX(L1)
      IF(CMX(L1).EQ.IT)GOTO 55
      IPUT=IPUT+1
      IF(IPUT.GT.MAXDATA)GOTO 170
      SDATA(IPUT)=1.
      IF(IBMX(IBCN,ICIREC).NE.IBMX(ICUT,IDIREC))SDATA(IPUT)=-1.
   55 CONTINUE
```

Table 6. (continued)

```
C OBTAIN COLUMN INDEX
      DO 60 M=LIB1,LIB2
   60 IF(IBON.EQ.IBLIS(M))GOTO 70
   70 CONTINUE
      IF(CMX(L1).EQ.IT)GOTO 75
      IPS(IPUT)=(IRODEX-1)*LEN+M+LENG
      IF(NTAP.NE.6)IPS(IPUT)=IPS(IPUT)-LENG
   75 CONTINUE
      IF(IACT.EQ.1)GOTO 80
C INSERT IDENTITY ENTRIES
      IPLT=IPLT+1
      IF(IPUT.GT.MAXDATA)GOTO 170
      SDATA(IPLT)=-1.
      IPS(IPUT)=(M-1)*LEN+IRODEX-LENG
      IF(NTAP.EQ.6)GOTO 80
      IPS(IPUT)=(M+LENG-1)*LEN+IRODEX+LENG
   80 CONTINUE
      GOTO 120
C
C SETUP FOR TF-NODE OR GY-NODE
   90 CONTINUE
C FIND INCIDENT BONDS
      N1=NPTR(NODE)
      N2=NBIMX(N1+1)
      N1=NBIMX(N1)
C OBTAIN SETUP MATRIX INDICES
      DO 110 L1=LIB1,LIB2
      IF(N1.NE.IBLIS(L1))GOTO 100
      M1=L1
  100 IF(N2.NE.IBLIS(L1))GOTO 110
      M2=L1
  110 CONTINUE
C
C UPDATE SETUP VECTOR
      VAL=PAR(IPTR(NODE))
      IF(VAL.EQ.0.)GOTO 120
      L1=NPTR(NODE)
      CMX(L1)=-CMX(L1)
      CMX(L1+1)=-CMX(L1+1)
      IF(CMX(L1).GT.3)GOTO 115
      IF(CMX(L1).GT.2)GOTO 112
      IPLT=IPLT+1
      IF(IPUT.GT.MAXDATA)GOTO 170
      SDATA(IPLT)=-1./VAL
      IPS(IPUT)=(M2-1)*LEN+M1
      IF(NTAP.EQ.9)IPS(IPUT)=(M2+LENG-1)*LEN+M1
```

Table 6. (continued)

```
112 CONTINUE
    IF(NTAP.EQ.8.AND.CMX(L1+1).EQ.5)GOTO 120
    IF(NTAP.EQ.9.AND.CMX(L1+1).EQ.2)GOTO 120
    IPUT=IPUT+1
    IF(IPUT.GT.MAXDATA)GOTO 170
    SDATA(IPUT)=-1./VAL
    IPS(IPUT)=(M1+LENG-1)*LEN+M2+LENG
    IF(NTAP.EQ.9)IPS(IPUT)=IPS(IPUT)-LENG
    GOTO 120
115 CONTINUE
    IF(NTAP.EQ.8.AND.CMX(L1+1).EQ.2)GOTO 117
    IF(NTAP.EQ.9.AND.CMX(L1+1).EQ.5)GOTO 117
    IPUT=IPUT+1
    IF(IPUT.GT.MAXDATA)GOTO 170
    SDATA(IPUT)=-VAL
    IPS(IPUT)=(M1-1)*LEN+M2
    IF(NTAP.EQ.9)IPS(IPUT)=IPS(IPUT)+LENG
117 CONTINUE
    IF(CMX(L1).EQ.5)GOTO 120
    IPUT=IPUT+1
    IF(IPUT.GT.MAXDATA)GOTO 170
    SDATA(IPUT)=-VAL
    IPS(IPUT)=(M2+LENG-1)*LEN+M1+LENG
    IF(NTAP.EQ.9)IPS(IPUT)=(M2-1)*LEN+M1+LENG
120 CONTINUE
```

Table 6. (continued)

```
C--COMPLTE DETERMINANT OF CAUSAL COMPLEX
      LENMAT1=IPLT*LEN
      IF(LENMAT1.GT.MAXWK)GOTO 900
      LENMAT2=
      MATDIM1=LEN
      MATDIM2=LEN
      INDEX1=-LEN
      DO 141 IK=1,LEN
      INDEX1=INDEX1+LEN+1
      RMAT1(IK)=1.
  141 IRMAT1(IK)=INDEX1
      DO 142 IK=1,IPLT
      RMAT1(LEN+IK)=SDATA(IK)
  142 IRMAT1(LEN+IK)=IPS(IK)
C--SORT ARRAY
      IBCN=LENMAT1-1
      DO 143 IK=1,IBCN
      DO 143 JK=IK,LENMAT1
      IF(IRMAT1(JK).GT.IRMAT1(IK))GOTO 143
      IOUT=IRMAT1(IK)
      IRMAT1(IK)=IRMAT1(JK)
      IRMAT1(JK)=IOUT
      VAL=RMAT1(IK)
      RMAT1(IK)=RMAT1(JK)
      RMAT1(JK)=VAL
  143 CONTINUE
      CALL IMPRD
      IF(DET.NE.0.)GOTO 150
      WRITE 1000,I
      IERRF=200
  150 CONTINUE
      IF(IERRF.NE.0)GOTO 999
      RETURN
  170 IERRF=210
      GOTO 999
  900 IERRF=220
  999 WRITE 2000,IERRF
      RETURN
 1000 FORMAT(*  CAUSAL COMPLEX NUMBER *,I2,* IS ALGEBRAICALLY IRREDUCIBL
     *E*)
 2000 FORMAT(*0 ERROR NUMBER *,I4,* IN TESTPLX*)
      END
```

Table 7. Subroutine For The Computation Of The Reduced Junction Matrix

```
      SUBROUTINE SMATX
      COMMON /BK1/ITN,OUT,IERRF,IPRLST(20),HEAD(20)
      COMMON /BK3/NEL,NBC,NPBG,NLBG,IELLST(50),IELNAM(50),NBIMX(10),
     + IBMX(50,2),NPTR(51),NMCR,NMCPT,NTYP(9)
      COMMON /BK7/IBEG(50),IBT(50),NBEX,NFI,NFD,NFL,NFS,NBIN
      COMMON/BK8/A(60),IAPTR(60),LENA,MAXA,ASUM(50),IASUM,B(10),
     +IBPTR(10),LENB,MAXB,BSUM(50),DET,E(5),IEPTR(5),LENE,MAXE,
     +FL(19),IFLPTR(19),LENFL,MAXFL,FS(25),IFSPTR(25),LENFS,MAXFS,
     +IFS12,IFS21,IFS22,KL(50),KM(50),SDATA(148),IPS(148),JS12,JS13,
     +JS14,JS21,JS24,JS31,JS33,JS34,JS41,LENS,MAXDATA,RMAT1(98),
     +IRMAT1(98),IRMAT2(98),JS42,JS43,JS44,
     +LENMAT1,MATDIM1,RMAT2(98),LENMAT2,MATDIM2,T(150),ITPTR(150),
     +IT0,IT1,IT2,IT3,IT4,MAXT,WK(98),IWK(98),MAXWK
C
C--*****************************************************************************
C--SMATX COMPUTES THE JUNCTION STRUCTURE TRANSFORMATION (STORED IN ARRAY
C--SDATA/IPS OF LENGTH LENS) AND SORTS IT INTO WORKING BLOCKS. SMATX STORES
C--THE TRANSFORMATION (OF LENGHT IT0) FROM INPUT VARIABLES TO INTERNAL VARIABLES
C--IN ARRAY T. THE TAILS OF SDATA/IPS , ASUM, KL, AND KM ARE USED AS WORKING
C--STORAGE. SMATX CALLS TESTPLX, JSGET, AND INPRD.
C
C-----IERRF=200 OVERFLOW IN SDATA ARRAY
C           220 IRREDUCIBLE JUNCTION STRUCTURE
C           230 JUNCTION STRUCTURE LACKS EXTERNAL BONDS
C           240 OVERFLOW IN T ARRAY
C           250 OVERFLOW IN RMAT1 OR RMAT2 ARRAY
C           260 NBEX EXCEEDS IASUM--INCREASE ASUM/BSUM
C--*****************************************************************************
C
C CHECK JUNCTION STRUCTURE SIZE AGAINST SDATA SIZE.
      NJS=0
      DO 5 I=1,NEL
    5 IF(IELLST(I).GE.6)NJS=NJS+1
      IF(4+NBIN+2*(NBEX-NJS).GT.MAXDATA)GOTO 900
C
C CHECK FOR IRREDUCIBLE CAUSAL COMPLEX
      CALL TESTPLX
      IF(IERRF.NE.0)RETURN
C JUNCTION STRUCTURE IS REDUCIBLE
C
C INIALIZE
      IF(NBEX.GT.IASUM)GOTO 960
      NBIN2=NBIN+NBIN
      LENA=NBEX+NBIN2
      IT0=0
```

Table 7. (continued)

```
C FIND NUMBER OF EXTERNAL EFFORT OUTPUTS
      IEFOUT=0
      IF(NBEX.EQ.0)GOTO 930
      DO 10 I=1,NBD
      IF(IBT(I).EQ.5)GOTO 10
      IF(IBT(I).GT.0)IEFOUT=IEFOUT+1
   10 CONTINUE
C
C DEFINE THE S-MATRIX
   20 IPUT=0
      CALL JSGET(LEN,NBIN,IEFOUT,IPUT)
      IF(IERRF.NE.0)RETURN
C SORT BY ROWS
      IPASS=0
   55 CONTINUE
      LIN1=IPUT-1
      DO 100 I=1,LIN1
      DO 100 J=I,IPUT
      IF(IPS(J).GT.IPS(I))GOTO 100
      IHOLD=IPS(I)
      IPS(I)=IPS(J)
      IPS(J)=IHOLD
      HOLD=SDATA(I)
      SDATA(I)=SDATA(J)
      SDATA(J)=HOLD
  100 CONTINUE
      IF(IPASS.NE.0)GOTO 295
      IF(NBIN.EQ.0)GOTO 500
C
C FIND THE FIRST NONZERO ELEMENT OF THE S3-MATRIX
C IS3 DENOTES THE INDEX OF THE FIRST NONZERO ELEMENT OF THE S3-MATRIX
      LIN1=NBEX*LEN+1
      DO 120 I=1,IPUT
  120 IF(IPS(I).GE.LIN1)GOTO 130
  130 IS3=I
C
C--SORT US ARRAY INTO S1, S2, S3, S4 BLOCKS
C--IS1 DENOTES THE LAST NONZERO ELEMENT IN THE S1-MATRIX
C IS4 DENOTES THE INDEX OF THE FIRST NONZERO ELEMENT OF THE S4-MATRIX
      KPASS=1
      M1=1
      M2=LEN
      IP1=1
      IP2=NBEX
      IP3=NBEX
      LIN1=IS3-1
      M1=(LEN+1)*LEN
  125 K=M1
```

Table 7. (continued)

```
      INDEX1=IP1-M2
      INDEX2=IP2-M2
      DO 132 I=1,IP3
      INDEX1=INDEX1+M2
      INDEX2=INDEX2+M2
      DO 131 J=K,LIN1
      IF(IPS(J).GE.INDEX1.AND.IPS(J).LE.INDEX2)IPS(J)=IPS(J)-M1
  131 IF(IPS(J).GT.INDEX2)GOTO 132
  132 K=J
C
  137 INDEX1=LIN1-1
      DO 133 I=N1,INDEX1
      DO 133 J=I,LIN1
      IF(IPS(J).GT.IPS(I))GOTO 133
      IHOLD=IPS(I)
      IPS(I)=IPS(J)
      IPS(J)=IHOLD
      HOLD=SDATA(I)
      SDATA(I)=SDATA(J)
      SDATA(J)=HOLD
  133 CONTINUE
      IF(IPASS.EQ.2)GOTO 525
C
      KOUNT=N1-1
      DO 134 I=N1,LIN1
      IF(IPS(I).LT.0)KOUNT=KOUNT+1
  134 IF(IPS(I).LT.0)IPS(I)=IPS(I)+M1
      IF(KPASS.GT.6)GOTO 501
      GOTO(135,136,351,352,353,355)KPASS
  501 KPASS=KPASS-6
      GOTO(357,358,371 ,372,373)KPASS
  135 IS1=KOUNT
      N1=IS3
      IP1=NBEX*LEN+1
      IP2=IP1+NBEX-1
      IP3=NBIN2
      LIN1=IPUT
      KPASS=2
      GOTO 129
  136 IS4=KOUNT+1
C
C--COMPUTE INV(I-S4)*S3 AND STORE IN TO
      INDEX1=(NBEX-1)*LEN+NBEX
      LENMAT1=IPUT-IS4+NBIN2+1
      IF(LENMAT1.GT.MAXWK)GOTO 950
      LENMAT2=IS4-IS3
      IF(LENMAT2.GT.MAXWK)GOTO 950
      MATDIM1=NBIN2
      MATDIM2=NBEX
```

Table 7. (continued)

```
      IF(IS4.GT.IPLT)GOTO 515
      DO 510 I=IS4,IPUT
  510 SDATA(I)=-SDATA(I)
  515 DO 520 I=1,NBIN2
      INDEX1=INDEX1+LEN+1
      IPLT=IPLT+1
      IF(IPUT.GT.MAXDATA)GOTO 900
      IPS(IPUT)=INDEX1
  520 SDATA(IPLT)=1.
      LIN1=IPUT
      N1=IS4
      IPASS=2
      GOTO 137
  525 K=IS4
      IPASS=0
      INDEX1=NBEX+1
      INDEX2=-NBIN2
      LIB1=(NBEX-1)*LEN+NBEX
      DO 530 I=INDEX1,LEN
      LIB1=LIB1+LEN
      LIB2=LIB1+NBIN2
      INDEX2=INDEX2+NBIN2
      DO 527 J=K,IPUT
      IF(IPS(J).GT.LIB2)GOTO 530
      IRMAT1(J-IS4+1)=IPS(J)-LIB1+INDEX2
  527 RMAT1(J-IS4+1)=SDATA(J)
  530 K=J
      LIB1=(NBEX-1)*LEN
      INDEX2=-NBEX
      LIN1=IS4-1
      K=IS3
      DO 540 I=INDEX1,LEN
      LIB1=LIB1+LEN
      LIB2=LIB1+NBEX
      INDEX2=INDEX2+NBEX
      DO 535 J=K,LIN1
      IF(IPS(J).GT.LIB2)GOTO 540
      IRMAT2(J-IS3+1)=IPS(J)-LIB1+INDEX2
  535 RMAT2(J-IS3+1)=SDATA(J)
  540 K=J
      CALL INPRD
      IF(DET.EQ. .)GOTO 920
      IF(LENMAT2.GT.MAXT)GOTO 940
      IT =LENMAT2
      IF(LENMAT2.EQ. )GOTO 555
      DO 550 I=1,LENMAT2
      T(I)=RMAT2(I)
  550 IPTR(I)=IRMAT2(I)
  555 CONTINUE
```

Table 7. (continued)

```
C--COMPUTE S2*INV(I-S4)*S3
      IPUT=IS3-1
      ISMARK=IS3-1
      INDEX1=IS3-1
      LIN1=IS1+1
      N1=-LEN+NBEX
      N2=0
      DO 730 L=1,NBEX
      N1=N1+LEN
      N2=N2+LEN
      DO 660 I=1,NBEX
  660 ASUM(I)=0.
      IF(ITO.EQ.0.OR.IS1.EQ.INDEX1)GOTO 695
      DO 680 J=LIN1,INDEX1
      IF(IPS(J).LE.N1)GOTO 680
      IF(IPS(J).GT.N2)GOTO 690
      M1=IPS(J)-N1
      LIB1=(M1-1)*NBEX
      LIB2=LIB1+NBEX
      DO 670 I=1,ITO
      IF(ITPTR(I).LE.LIB1)GOTO 670
      IF(ITPTR(I).GT.LIB2)GOTO 680
      M2=ITPTR(I)-LIB1
      ASUM(M2)=ASUM(M2)+SDATA(J)*T(I)
  670 CONTINUE
  680 CONTINUE
  690 LIN1=J
C
C--ADD S1
  695 LIB1=N1-NBEX
      LIB2=N1
      IF(IS1.EQ.0)GOTO 710
      DO 700 I=1,IS1
      IF(IPS(I).LE.LIB1)GOTO 700
      IF(IPS(I).GT.LIB2)GOTO 710
      M2=IPS(I)-LIB1
      ASUM(M2)=ASUM(M2)+SDATA(I)
  700 CONTINUE
  710 M1=(L-1)*NBEX
C
C--STORE NONZERO ENTRIES
      DO 720 J=1,NBEX
      IF(ASUM(J).EQ.0.)GOTO 720
      IPUT=IPUT+1
      IF(IPUT.GT.MAXDATA)GOTO 900
      IPS(IPUT)=M1+J
      SDATA(IPUT)=ASUM(J)
  720 CONTINUE
  730 CONTINUE
```

Table 7. (continued)

```
C SHIFT DATA IN S-MA RIX TO PLACE REDUCED JS-MATRIX IN POSITIONS 1 TO IPUT
      INDEX1=IPUT-ISMARK
      IPUT=INDEX1
      DO 220 I=1,INDEX1
      SDATA(I)=SDATA(ISMARK+I)
  220 IPS(I)=IPS(ISMARK+I)
  500 CONTINUE
C
C--STORE PRESENT ORDERING OF OUTPUT VECTOR
      INDEX1=0
      INDEX2=IEFOUT
      DO 240 I=1,NBD
      IF(IBT(I).EQ.5)GOTO 240
      IF(IBT(I).LT.0)GOTO 230
      INDEX1=INDEX1+1
      KL(INDEX1)=I
      GOTO 240
  230 INDEX2=INDEX2+1
      KL(INDEX2)=I
  240 CONTINUE
C
C--STORE FIELD ORDERING OF OUTPUT VECTOR
      INDEX1=0
      DO 250 I=1,4
      DO 250 J=1,NBD
      IF(IBT(J).NE.I.AND.IBT(J).NE.(-I))GOTO 250
      INDEX1=INDEX1+1
      KM(INDEX1)=J
  250 CONTINUE
      DO 254 I=1,NBEX
      DO 254 J=1,NBEX
  254 IF(KL(I).EQ.KM(J))KL(NBEX+I)=J
C
C--REORDER JS-TRANSFORMATION
      K=1
      LIB1=-NBEX
      DO 270 I=1,NBEX
      LIB1=LIB1+NBEX
      LIB2=LIB1+NBEX
      M1=(KL(NBEX+I)-1)*NBEX
      DO 260 J=K,IPUT
      IF(IPS(J).GT.LIB2)GOTO 270
      M2=IPS(J)-LIB1
  260 IPS(J)=M1+KL(NBEX+M2)
  270 K=J
```

Table 7. (continued)

```
C--SORT BY ROWS
      IPASS=1
      GOTO 95
  255 CONTINUE
C
C--PRINT JS-TRANSFORMATION
      DO 300 I=1,NBEX
      KM(I+NBEX)=1HE
      KL(I)=1HF
      IF(IBT(KM(I)).LT.0)KM(I+NBEX)=1HF
  300 IF(IBT(KM(I)).LT.0)KL(I)=1HE

C PRINT JS-TRANSFORMATION
      WRITE 6000
      DO 320 I=1,NBEX,5
      M1=I+4
      IF(M1.GT.NBEX)M1=NBEX
      WRITE 6200,(KL(J),KM(J),J=I,M1)
      K=1
      INDEX1=M1-I+1
      LIB1=I-NBEX
      LIB2=M1-NBEX
      DO 320 IK=1,NBEX
      LIB1=LIB1+NBEX
      LIB2=LIB2+NBEX
      LIN1=LIB1-I
      DO 305 J=1,INDEX1
  305 ASUM(J)=0.
      DO 310 J=K,IPUT
      IF(IPS(J).LT.LIB1)GOTO 310
      IF(IPS(J).GT.LIB2)GOTO 315
      ASUM(IPS(J)-LIB1+1)=SDATA(J)
  310 CONTINUE
  315 K=J
  320 WRITE 6400,KM(NBEX+IK),KM(IK),(ASUM(J),J=1,INDEX1)
C
C--SORT JS ARRAY
C--JS12,JS13,JS14,JS21,JS24,JS31,JS33,JS34,AND JS41 DENOTE THE INDICES OF THE
C--FIRST NONZERO ELEMENTS IN S12,S13,S14,S21,S24,S31,S33,S34, AND S41 .
C--JS42,JS43, AND JS44 DENOTE THE INDICES OF THE FIRST NONZERO ELEMENTS IN
C--S42,S43, AND S44.
      LENS=IPUT
      LGHF=NFI+NFD
      LIB1=NFI+NBEX
      DO 325 I=1,IPUT
  325 IF(IPS(I).GT.LIB1)GOTO 330
  330 JS21=I
      LIB1=LGHF+NBEX
      DO 335 I=JS21,IPUT
  335 IF(IPS(I).GT.LIB1)GOTO 340
```

Table 7. (continued)

```
  340 JS31=I
      LIB1=(LGHF+NFL)*NBEX
      DO 345 I=JS31,IPUT
  345 IF(IFS(I).GT.LIB1)GOTO 350
  350 JS41=I
C
      M1=(NBEX+1)*NBEX
      M2=NBEX
      IF(NFI.EQ.0)GOTO 354
      N1=1
      LIN1=JS21-1
      IP1=1
      IP2=NFI
      IP3=NFI
      KPASS=3
      GOTO 129
  351 JS12=KOUNT+1
      N1=JS12
      IP1=IP2+1
      IP2=IP2+NFD
      KPASS=4
      GOTO 129
  352 JS13=KOUNT+1
      N1=JS13
      IP1=IP2+1
      IP2=IP2+NFL
      KPASS=5
      GOTO 129
  353 JS14=KOUNT+1
  354 IF(NFD.EQ.0)GOTO 356
      N1=JS21
      LIN1=JS31-1
      IP1=NFI*NBEX+1
      IP2=IP1+NFI-1
      IP3=NFD
      KPASS=6
      GOTO 129
  355 JS24=KOUNT+1
C
  356 IF(NFL.EQ.0)GOTO 370
      N1=JS31
      LIN1=JS41-1
      IP1=LGHF*NBEX+1
      IP2=IP1+NFI-1
      IP3=NFL
      KPASS=7
      GOTO 129
  357 JS33=KOUNT+1
```

Table 7. (continued)

```
      N1=JS33
      IP1=IP2+NFD+1
      IP2=IP2+NFL
       KPASS=8
      GOTO 129
  358 JS34=KOUNT+1
  370 IF(NFS.EQ.0)GOTO 359
      N1=JS41
      LIN1=LENS
      IP1=(LGHF+NFL)*NBEX+1
      IP2=IP1+NFI-1
      IP3=NFS
      KPASS=9
      GOTO 129
  371 JS42=KOUNT+1
      N1=JS42
      IP1=IP2+1
      IP2=IP2+NFD
      KPASS=10
      GOTO 129
  372 JS43=KOUNT+1
      N1=JS43
      IP1=IP2+1
      IP2=IP2+NFL
      KPASS=11
      GOTO 129
  373 JS44=KOUNT+1
  359 IF(NFI.NE.0)GOTO 360
      JS12=JS21
      JS13=JS21
      JS14=JS21
  360 IF(NFD.NE.0)GOTO 365
      JS24=JS31
  365 IF(NFL.NE.0)GOTO 374
      JS33=JS41
      JS34=JS41
  374 IF(NFS.NE.0)RETURN
      JS42=LENS
      JS43=LENS
      JS44=LENS
      RETURN
```

Table 7. (continued)

```
    910 IERRF=200
        GOTO 999
    920 WRITE 6500
        IERRF=220
        GOTO 999
    930 WRITE 5500
        IERRF=230
        GOTO 999
    940 IERRF=240
        GOTO 999
    950 IERRF=250
        GOTO 999
    960 IERRF=260
    999 WRITE 3000,IERRF
        RETURN
   3000 FORMAT(* ERROR NUMBER *,I4,* IN SMATX*)
   5500 FORMAT(*0 JUNCTION STRUCTURE LACKS EXTERNAL BONDS*)
   6100 FORMAT(*- JS-TRANSFORMATION--OUTPUTS IN COLUMN--INPUTS IN ROW*)
   6200 FORMAT(*0*,10X,5(A1,I2,9X))
   6400 FORMAT(2X,A1,I2,2X,5(E11.4,1X))
   6500 FORMAT(*0 JUNCTION STRUCTURE IS IRREDUCIBLE*)
        END
```

Table 8. Subroutine For The Construction Of The Junction Matrix

```
      SUBROUTINE JSGET(LEN,NBIN,IEFOLT,IPUT)
      COMMON /BK1/IN,OUT,IERRF,IPRLST(20),HEAD(20)
      COMMON /BK3/NEL,NBD,NPBG,NLDG,IELLST(50),IELNAM(50),NBIMX(100),
     + IEMX(50 ,2),NPTR(51),NMCR,NMCPT,NTYP(9)
      COMMON/BK5/CMX(100),MELMNT(10),MBND(10),MIT(10),NTEMP(30),MNSTKM,
     +MASTKN,MTCP,NTCP,LEVEL,IBLIS(13),ICCNODE(13),IBPNT(4),NODLIS(4),
     +JPNT,KBCND,INODE,MAXBD,MAXND
      COMMON /BK6/PAR(100),IPTR(51),IPFLG,NPAR
      COMMON/BK8/A(60),IAPTR(60),LENA,MAXA,ASUM(50),IASUM,B(10),
     +IBPTR(10),LENB,MAXB,BSUM(50),DET,E( 5),IEPTR( 5),LENE,MAXE,
     +FL(19),IFLPTR(19),LENFL,MAXFL,FS(25),IFSPTR(25),LENFS,MAXFS,
     +IFS12,IFS21,IFS22,KL(50),KM(50),SDATA(148),IPS(148),JS12,JS13,
     +JS14,JS21,JS24,JS31,JS33,JS34,JS41,LENS,MAXDATA,RMAT1( 96),
     +IRMAT1( 98),IRMAT2( 98),JS42,JS43,JS44,
     +LENMAT1,MATDIM1,RMAT2( 98),LENMAT2,MATDIM2,T(150),ITPTR(150),
     +ITC,IT1,IT2,IT3,IT4,MAXT,WK( 98),IWK( 98),MAXWK
      INTEGER CMX
C
C*****************************************************************************
C--JSGET COMPLIES THE UNORDERED JUNCTION STRUCTURE MATRIX (STORED IN SDATA/IPS).
C--JSGET CALLS FINDEX.
C
C-----IERRF=200 OVERFLOW IN SDATA ARRAY
C*****************************************************************************
C
      NBEX=NBD-NBIN
      DO 90 I=1,NEL
      NTAP=IELLST(I)
      IF(NTAP.LE.5)GOTO 90
      IF(NTAP.GE.8)GOTO 60
C
C S-MATRIX ENTRIES FOR JUNCTIONS
      INDEX1=NPTR(I)
      INDEX2=NPTR(I+1)-1
      IT=3
      IF(NTAP.EQ.7)IT=6
C RESET ACTIVATION FLAG
      IACT=0
C
C FIND BOND OF STRONG INPUT
      DO 30 J1=INDEX1,INDEX2
   30 IF(CMX(J1).EQ.IT)GOTO 40
C ACTIVATION FOUND
      IACT=1
      DO 35 J1=INDEX1,INDEX2
   35 IF(CMX(J1).EQ.(IT-1))GOTO 40
   40 CONTINUE
      IOUT=NBIMX(J1)
```

Table 8. (continued)

```
C OBTAIN POWER DIRECTION
      IDIREC =1
      IF(IBMX(ICUT,1).NE.I)ICIREC=2
C
C FIND ROW COORDINATE
      IRODEX=1
      CALL FINDEX(IOUT,IEFOUT,IT,IRODEX)
      M2=IRODEX
      IF(IRODEX.LE.NBEX)GOTO 45
      M2=IRODEX-NBIN
      IF(NTAP.EQ.7)M2=IRODEX+NBIN
   45 CONTINUE
C
C UPDATE S-MATRIX
      IT=6
      IF(NTAP.EQ.7)IT=3
      DO 50 L1=INDEX1,INDEX2
      IF(NBIMX(L1).EQ.IOUT)GOTO 50
      IBUN=NBIMX(L1)
      IF(CMX(L1).EQ.(8-IT))GOTO 46
      IPUT=IPUT+1
      IF(IPUT.GT.MAXDATA)GOTO 900
      SDATA(IPUT)=-1.
      IF(IBMX(IBUN,ICIREC).NE.IBMX(ICUT,IDIREC))SDATA(IPUT)=1.
   46 CONTINUE
C FIND COLUMN COORDINATE AND STORE
      ICODEX=
      CALL FINDEX(IBUN,IEFOUT,IT,ICODEX)
      M1=ICODEX
      IF(ICODEX.LE.NBEX)GOTO 48
      M1=ICODEX-NBIN
      IF(NTAP.EQ.7)M1=ICODEX+NBIN
   48 CONTINUE
      IF(CMX(L1).EQ.(8-IT))GOTO 49
      IPS(IPUT)=(IRODEX-1)*LEN+ICODEX
   49 CONTINUE
      IF(IACT.EQ.1)GOTO 50
C
C INSERT IDENTITY ENTRIES
      IPUT=IPUT+1
      IF(IPUT.GT.MAXDATA)GOTO 900
      SDATA(IPUT)=1.
      IPS(IPUT)=(M1-1)*LEN+M2
   50 CONTINUE
      GOTO 9(
```

Table 8. (continued)

```
C S-MATRIX ENTRIES FOR TF-NODES OR GY-NODES
   60 CONTINUE
C FIND INCIDENT BOND
      N1=NPTR(I)
      N2=NBIMX(N1+1)
      N1=NBIMX(N1)
C
C UPDATE S-MATRIX
      VAL=FAR(IPTR(I))
      IF(VAL.EQ.0.)GOTO 90
      M1=IEFOUT
      M2=NBIN
      IF(NTAP.EQ.8)GOTO 70
      M1=-M1
      M2=-M2
   70 CONTINUE
      L1=NPTR(I)
      IF(CMX(L1).GT.3)GOTO 80
      IT=3
      IRODEX=1
      CALL FINDEX(N1,IEFOUT,IT,IRODEX)
      INDEX1=IRODEX-IEFOUT
      IF(IRODEX.GT.NBEX)INDEX1=IRODEX-NBIN
      IF(NTAP.EQ.8)IT=6
      ICODEX=1
      CALL FINDEX(N2,IEFOUT,IT,ICODEX)
      INDEX2=ICODEX
      IF(ICODEX.GT.NBEX)INDEX2=ICODEX+M2
      IF(CMX(L1).EQ.2)GOTO 75
      IPUT=IPUT+1
      IF(IPUT.GT.MAXDATA)GOTO 900
      SDATA(IPUT)=1./VAL
      IPS(IPUT)=(IRODEX-1)*LEN+INDEX2
   75 CONTINUE
      IF(NTAP.EQ.8.AND.CMX(L1+1).EQ.5)GOTO 90
      IF(NTAP.EQ.9.AND.CMX(L1+1).EQ.2)GOTO 90
      IPUT=IPUT+1
      IF(IPUT.GT.MAXDATA)GOTO 900
      SDATA(IPUT)=1./VAL
      IPS(IPUT)=(ICODEX-1)*LEN+INDEX1
      GOTO 90
```

Table 8. (continued)

```
   80 CONTINUE
      IT=6
      IRCDEX=1
      CALL FINDEX(N1,IEFOUT,IT,IRCDEX)
      INDEX1=IRODEX+IEFOUT
      IF(IRODEX.GT.NBEX)INDEX1=IRCDEX+NBIN
      IF(NTAP.EQ.8)IT=3
      ICCDEX=1
      CALL FINDEX(N2,IEFOUT,IT,ICCDEX)
      INDEX2=ICODEX
      IF(ICODEX.GT.NBEX)INDEX2=ICODEX-M2
      IF(NTAP.EQ.8.AND.CMX(L1+1).EQ.2)GOTO 85
      IF(NTAP.EQ.9.AND.CMX(L1+1).EQ.5)GOTO 85
      IPUT=IPUT+1
      IF(IPUT.GT.MAXDATA)GOTO 900
      SDATA(IPUT)=VAL
      IPS(IPUT)=(IRODEX-1)*LEN+INDEX2
   85 CONTINUE
      IF(CMX(L1).EQ.5)GOTO 90
      IPUT=IPUT+1
      IF(IPUT.GT.MAXDATA)GOTO 900
      SDATA(IPUT)=VAL
      IPS(IPUT)=(ICODEX-1)*LEN+INDEX1
   90 CONTINUE
      RETURN
  900 IERRF=200
      WRITE 9000,IERRF
      RETURN
 9000 FORMAT(*0 ERROR NUMBER *,I4,* IN JSGET*)
      END
```

Table 9. Subroutine For The Determination Of Junction Matrix Entry Position

```
      SUBROUTINE FINDEX(IBOND,IEFOUT,IT,IDEX)
      COMMON/BK7/IBEQ(50),IBT(50),NBEX,NFI,NFD,NFL,NFS,NBIN
C
C*****************************************************************************
C--FINDEX ASSIGNS A MATRIX COORDINATE POSITION TO EACH JUNCTION STRUCTURE
C--VARIABLE.
C*****************************************************************************
C
      IPOS=0

C DETERMINE IF IBOND IS INTERNAL BOND
      IF(IBEQ(IBOND).GT.NBEX)GOTO 40
      IF(IT.EQ.3)GOTO 20

C EXTERNAL EFFORT
      DO 10 I=1,IBOND
      IF(IBT(I).EQ.5)GOTO 10
      IF(IBT(I).GT.0)IPOS=IPOS+1
   10 CONTINUE
      IDEX=IPOS
      RETURN

C EXTERNAL FLOW
   20 DO 30 I=1,IBOND
      IF(IBT(I).LT.0)IPOS=IPOS+1
   30 CONTINUE
      IDEX=IPOS+IEFOUT
      RETURN

C INTERNAL BOND
   40 DO 50 I=1,IBOND
      IF(IBT(I).EQ.5)IPOS=IPOS+1
   50 CONTINUE
      IPOS=IPOS+NBEX
      IF(IDEX.EQ.1.AND.IT.EQ.3)IPOS=IPOS+NBIN
      IF(IDEX.EQ.2.AND.IT.EQ.3)IPOS=IPOS+NBIN
      IDEX=IPOS
      RETURN
      END
```

Table 10. Subroutine For Matrix Inversion

```
      SUBROUTINE INPRD
      COMMON /BK1/IN,OUT,IERRF,IPRLST(20),HEAD(20)
      COMMON/BK8/A(60),IAPTR(60),LENA,MAXA,ASUM(50),IASUM,B(10),
     +IBPTR(10),LENB,MAXB,BSLM(50),DET,E( 5),IEPTR( 5),LENE,MAXE,
     +FL(19),IFLPTR(19),LENFL,MAXFL,FS(25),IFSPTR(25),LENFS,MAXFS,
     +IFS12,IFS21,IFS22,KL(50),KM(50),SDATA(148),IPS(148),JS12,JS13,
     +JS14,JS21,JS24,JS31,JS33,JS34,JS41,LENS,MAXDATA,RMAT1( 98),
     +IRMAT1( 98),IRMAT2( 98),JS42,JS43,JS44,
     +LENMAT1,MATDIM1,RMAT2( 98),LENMAT2,MATDIM2,I(150),ITPTR(150),
     +IT0,IT1,IT2,IT3,IT4,MAXT,WK( 98),IWK( 98),MAXWK
C
C*****************************************************************************
C--INPRD COMPUTES THE PRODUCT OF THE INVERSE OF RMAT1 TIMES RMAT2. THE PRODUCT
C--IS RETURNED IN RMAT2. RMAT1 IS A MATDIM1 BY MATDIM1 MATRIX (OF DENSE LENGTH
C--LENMAT1) AND RMAT2 IS A MATDIM1 BY MATDIM2 MATRIX (OF DENSE LENGTH LENMAT2).
C--THE INVERSE OF RMAT1 IS RETURNED IF RMAT2 IS INITIALLY THE IDENTITY MATRIX OF
C--DIMENSION MATDIM1. THE DETERMINANT OF RMAT1 IS RETURNED IN ALL CASES.
C--INPRD USES ASUM, BSUM, KL, KM, AND WK/IWK AS WORKING AREAS.
C
C-----IERRF=200 OVERFLOW IN WK ARRAY
C           210 MATDIM1 OR MATDIM2 EXCEEDS IASUM
C*****************************************************************************
C
C--INITIALIZE
      IF(MATDIM1.GT.IASUM.OR.MATDIM2.GT.IASUM)GOTO 910
      DET=1.
C
      DO 2 I=1,MATDIM1
    2 KM(I)=I
C--FIND LARGEST PIVOT
      DO 100 IROW=1,MATDIM1
      N1=(IROW-1)*MATDIM1+IROW
      PIV=0.
      IPIV=
      SIGNP=1.
      IF(LENMAT1.EQ.0)GOTO 11
      DO 3 I=1,MATDIM1
    3 KL(I)=KM(I)
      DO 5 I=1,LENMAT1
    5 IF(IRMAT1(I).GE.N1)GOTO 6
    6 K=I
      IF(K.GT.LENMAT1)K=LENMAT1
      IF(IRMAT1(K).LT.N1)GOTO 11
      N1=N1-IROW*MATDIM1
      IF(IRMAT1(K).GT.N1)GOTO 11
```

Table 10. (continued)

```
      DO 10 I=K,LENMAT1
      HOLD=RMAT1(I)
      IF(HOLD.LT.0.)HOLD=-HOLD
      IF(HOLD.LE.PIV)GOTO 10
      PIV=HOLD
      SIGNP=RMAT1(I)/HOLD
      IPIV=IRMAT1(I)
   10 CONTINUE
      PIV=PIV*SIGNP
   11 DET=DET*PIV
C--CHECK FOR ZERO DETERMINANT
      IF(DET.EQ.0.)RETURN
C
C--IDENTIFY ROWS AND COLUMNS TO BE INTERCHANGED
      N1=N1-MATDIM1
      NEWR=IROW-1
      DO 15 I=IROW,MATDIM1
      NEWR=NEWR+1
      N2=N1+MATDIM1
      IF(IPIV.GT.N1.AND.IPIV.LE.N2)GOTO 20
   15 N1=N1+MATDIM1
   20 NEWC=IPIV-N1
      IF(IROW.EQ.NEWR)GOTO 16
      HOLD=KL(IROW)
      KL(IROW)=KL(NEWR)
      KL(NEWR)=HOLD
      DET=-DET
   16 IF(IROW.EQ.NEWC)GOTO 18
      HOLD=KM(IROW)
      KM(IROW)=KM(NEWC)
      KM(NEWC)=HOLD
      DET=-DET
   18 CONTINUE
C
C--INTERCHANGE COLUMNS
      N1=-MATDIM1
      LIM1=1
      DO 23 I=1,MATDIM1
      N1=N1+MATDIM1
      N2=N1+MATDIM1
      DO 21 J=LIM1,LENMAT1
      IF(IRMAT1(J).GT.N2)GOTO 23
      M1=IRMAT1(J)-N1
      IF(M1.EQ.IROW)IRMAT1(J)=IRMAT1(J)-IROW+NEWC
   21 IF(M1.EQ.NEWC)IRMAT1(J)=IRMAT1(J)-NEWC+IROW
   23 LIM1=J
```

Table 10. (continued)

```
C--INTERCHANGE ROWS
      N1=(IROW-1)*MATDIM1
      N2=N1+MATDIM1
      M1=(NEWR-1)*MATDIM1
      M2=M1+MATDIM1
      DO 25 I=K,LENMAT1
      IF(IRMAT1(I).LE.N1.OR.IRMAT1(I).GT.N2)GOTO 22
      IRMAT1(I)=IRMAT1(I)-N1+M1
      GOTO 25
   22 IF(IRMAT1(I).LE.M1.OR.IRMAT1(I).GT.M2)GOTO 25
      IRMAT1(I)=IRMAT1(I)-M1+N1
   25 CONTINUE
C
C--SORT ARRAY
      LIM1=LENMAT1-1
      DO 28 I=1,LIM1
      DO 28 J=I,LENMAT1
      IF(IRMAT1(J).GT.IRMAT1(I))GOTO 28
      IHOLD=IRMAT1(I)
      IRMAT1(I)=IRMAT1(J)
      IRMAT1(J)=IHOLD
      HOLD=RMAT1(I)
      RMAT1(I)=RMAT1(J)
      RMAT1(J)=HOLD
   28 CONTINUE
C
C--DEFINE TRANSFORMATION FACTOR OF INV(RMAT1)
      DO 30 I=1,MATDIM1
   30 ASUM(I)=0.
      N1=IROW-MATDIM1
      K=1
      DO 40 I=1,MATDIM1
      N1=N1+MATDIM1
      DO 35 J=K,LENMAT1
      IF(IRMAT1(J).GT.N1)GOTO 38
   35 IF(IRMAT1(J).EQ.N1)GOTO 37
      GOTO 40
   37 ASUM(I)=-RMAT1(J)
   38 K=J
   40 CONTINUE
      ASUM(IROW)=1.
C
C--COMPUTE MODIFIED RMAT1 ARRAY
      M2=(IROW-1)*MATDIM1
      DO 44 I=1,LENMAT1
   44 IF(IRMAT1(I).GT.M2)GOTO 45
   45 LIM1=I
```

Table 10. (continued)

```
      N1=M2+MATDIM1
      DO 46 I=LIM1,LENMAT1
   46 IF(IRMAT1(I).GE.N1)GOTO 47
   47 LIM2=I-1
      IF(I.GT.LENMAT1)I=LENMAT1
      IF(IRMAT1(I).EQ.N1)LIM2=I
      K=1
      N1=-MATDIM1
      NTOP=0
      DO 65 L=1,MATDIM1
      N1=N1+MATDIM1
       N2=N1+MATDIM1
      DO 48 I=1,MATDIM1
   48 BSUM(I)=0.
      IF(L.EQ.IROW)GOTO 57
      DO 50 I=K,LENMAT1
      IF(IRMAT1(I).LE.N1)GOTO 50
      IF(IRMAT1(I).GT.N2)GOTO 52
      M1=IRMAT1(I)-N1
      BSUM(M1)=BSUM(M1)+RMAT1(I)
   50 CONTINUE
   52 K=I
      IF(ASUM(L).EQ.0.)GOTO 54
      DO 53 I=LIM1,LIM2
      M1=IRMAT1(I)-M2
   53 BSUM(M1)=BSUM(M1)+ASUM(L)*RMAT1(I)/PIV
C--STORE NONZERO ENTRIES
   54 DO 55 J=1,MATDIM1
      IF(BSUM(J).EQ.0.)GOTO 55
      NTOP=NTOP+1
      IF(NTOP.GT.MAXLK)GOTO 900
      WK(NTOP)=BSUM(J)
      IWK(NTOP)=N1+J
   55 CONTINUE
      GOTO 65
   57 DO 60 I=K,LENMAT1
      IF(IRMAT1(I).LE.N1)GOTO 60
      IF(IRMAT1(I).GT.N2)GOTO 62
      NTOP=NTOP+1
      IF(NTOP.GT.MAXLK)GOTO 900
      WK(NTOP)=RMAT1(I)*ASUM(IROW)/PIV
      IWK(NTOP)=IRMAT1(I)
   60 CONTINUE
   62 K=I
   65 CONTINUE
      LENMAT1=NTOP
      DO 70 I=1,NTOP
      RMAT1(I)=WK(I)
   70 IRMAT1(I)=IWK(I)
```

Table 10. (continued)

```
C--COMPLTE UPDATED PRODUCT
      IF(LENMAT2.EQ.0)GOTO 100
      NTOP=0
      LIM1=(KL(IROW)-1)*MATDIM2
      LIM2=LIM1+MATDIM2
      DO 90 L=1,MATDIM1
      DO 72 I=1,MATDIM2
   72 BSUM(I)=0.
      IHOLD=(KM(L)-1)*MATDIM2
      IF(L.EQ.IROW)GOTO 82
      N1=(KL(L)-1)*MATDIM2
      N2=N1+MATDIM2
      DO 75 I=1,LENMAT2
      IF(IRMAT2(I).LE.N1.OR.IRMAT2(I).GT.N2)GOTO 74
      M1=IRMAT2(I)-N1
      BSUM(M1)=RMAT2(I)+BSUM(M1)
      GOTO 75
   74 IF(IRMAT2(I).LE.LIM1.OR.IRMAT2(I).GT.LIM2)GOTO 75
      M1=IRMAT2(I)-LIM1
      BSUM(M1)=BSUM(M1)+ASUM(L)*RMAT2(I)/PIV
   75 CONTINUE
C--STORE NONZERO ENTRIES
      DO 80 J=1,MATDIM2
      IF(BSUM(J).EQ.0.)GOTO 80
      NTOP=NTOP+1
      IF(NTOP.GT.MAXWK)GOTO 900
      WK(NTOP)=BSUM(J)
      IWK(NTOP)=IHOLD+J
   80 CONTINUE
      GOTO 90
   82 DO 85 I=1,LENMAT2
      IF(IRMAT2(I).LE.LIM1.OR.IRMAT2(I).GT.LIM2)GOTO 85
      NTOP=NTOP+1
      IF(NTOP.GT.MAXWK)GOTO 900
      WK(NTOP)=RMAT2(I)*ASUM(IROW)/PIV
      IWK(NTOP)=IHOLD+IRMAT2(I)-LIM1
   85 CONTINUE
   90 CONTINUE
      LENMAT2=NTOP
      DO 96 I=1,NTOP
      RMAT2(I)=WK(I)
   96 IRMAT2(I)=IWK(I)
  100 CONTINUE
```

Table 10. (continued)

```
C--SORT RMAT2 ARRAY
      LIM1=LENMAT2-1
      DO 110 I=1,LIM1
      DO 110 J=I,LENMAT2
      IF(IRMAT2(J).GT.IRMAT2(I))GOTO 110
      IHOLD=IRMAT2(I)
      IRMAT2(I)=IRMAT2(J)
      IRMAT2(J)=IHOLD
      HOLD=RMAT2(I)
      RMAT2(I)=RMAT2(J)
      RMAT2(J)=HOLD
  110 CONTINUE
      RETURN
  900 IERRF=200
      GOTO 999
  910 IERRF=210
  999 PRINT 9999,IERRF
 9999 FORMAT(*0 ERROR NUMBER *,I4,* IN INPRD*)
      RETURN
      END
```