





This is to certify that the  
thesis entitled

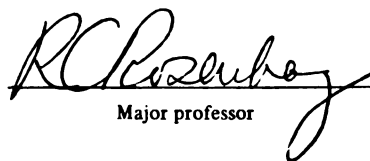
DIVERSE PLATFORM MODELING OF DYNAMICAL SYSTEMS

presented by

Robert Alex Mitchell

has been accepted towards fulfillment  
of the requirements for

Master of Science degree in Mechanical Engineering

  
Major professor

Date 11/12/91

# LIBRARY

## Michigan State University

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

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

**DIVERSE PLATFORM MODELING OF DYNAMICAL SYSTEMS**

**By**

**Robert Alex Mitchell**

**A THESIS**

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

**MASTER OF SCIENCE**

**Department of Mechanical Engineering**

**1991**

A current  
dynamically  
software  
solution  
proposed  
platform  
bonding  
system  
a Stand

684-1764

## **ABSTRACT**

### **DIVERSE PLATFORM MODELING OF DYNAMICAL SYSTEMS**

**BY**

**Robert Alex Mitchell**

A current problem in engineering simulation is the difficulty with which the various dynamical components of a system may be conveniently modeled using a single software platform. The Diverse Platform Modeling (DPM) technique proposes a solution path for this modeling difficulty. Two general strategies for application of the proposed technique, using the bond-graph modeling domain as a "host" simulation platform, are presented. A software utility has been created for ENPORT, an existing bond graph modeling and simulation platform, that allows externally created dynamical system models of a specified class to be injected into the ENPORT system graph using a Standard Model Description (SMD) file.

during

invalu

at Ros

much a

a stude

eventual

R

commen

the phy

formidat

## **ACKNOWLEDGEMENTS**

**I would like to thank my advisor, Dr. Ronald Rosenberg, for his guidance during the development of this thesis. I would also like to thank Dr. Rosenberg for the invaluable skills I have attained in the area of software development under his direction at Rosencode Associates Inc. The experience I have gained in this area has been as much a part of my education as the course work.**

**I would like to thank my family for their support throughout my long carrier as a student at Michigan State University. They have continually assured me that eventually all the struggles and sacrifices would pay off.**

**Finally, I would like to thank my wife Robin for her suggestions and helpful comments during the writing of this thesis. There is a bright carrier waiting for her in the physical sciences. Also, I would like to acknowledge her blossoming as a formidable contender in the arena of miniature basketball.**



## **TABLE OF CONTENTS**

	<b>Page</b>
<b>LIST OF TABLES</b>	<b>vi</b>
<b>LIST OF FIGURES</b>	<b>vii</b>
<b>NOMENCLATURE</b>	<b>viii</b>
<b>Chapter</b>	
1. INTRODUCTION	1
2. THE DIVERSE PLATFORM MODELING TECHNIQUE	4
3. THE BOND GRAPH MODELING DOMAIN AS HOST FOR DPM	6
4. TOPOLOGICAL APPROACH TO DPM WITH ENPORT AS HOST	8
4.1 Multiport Configuration for Injected MCK Models	8
4.2 Modal Configuration for Injected MCK Models	11
5. SYSTEM EQUATION APPROACH TO DPM WITH ENPORT AS HOST	15
5.1 The Macro Dynamic System as Applied to MCK Models	15
5.2 A Class Specific Macro Dynamic Node Type	17
6. EXAMPLE SYSTEM	19
6.1 The Finite Element Model	19
6.2 Multiport Expansion	22
6.3 Modal Expansion	23
6.4 MCK Macro Dynamic System	24
6.5 Numerical Results	25
7. SUMMARY AND RECOMMENDATIONS	26
7.1 Summary	26
7.2 Recommendations for Future Development	26
<b>LIST OF REFERENCES</b>	<b>28</b>

## **APPENDICES**

<b>A. USER DOCUMENTATION</b>	<b>29</b>
<b>A1. SAMPLE MCK STANDARD MODEL DESCRIPTION FILE</b>	<b>30</b>
<b>A2. USER ACCESS FROM ENPORT</b>	<b>32</b>
<b>B. THE EXAMPLE SYSTEM</b>	<b>34</b>
<b>B1. MCK FILE FOR THE EXAMPLE SYSTEM</b>	<b>35</b>
<b>B2. ENPORT MODEL FILE FOR EXAMPLE PROBLEM</b>	<b>37</b>
<b>(MULTIPOINT)</b>	
<b>B3. ENPORT MODEL FILE FOR EXAMPLE PROBLEM</b>	<b>40</b>
<b>(MODAL)</b>	
<b>C. ORGANIZATION OF THE FORTRAN SOURCE CODE</b>	<b>46</b>
<b>C1. SUBPROGRAM CALLING TREE</b>	<b>47</b>
<b>C2. SUBPROGRAM LIST</b>	<b>48</b>
<b>C3. FORTRAN SOURCE CODE LISTINGS</b>	<b>49</b>

Table

Table

Table

Table

Table

## LIST OF TABLES

	Page
Table B-1. MCK file for the example system.	35
Table B-2. ENPORT model file for multiport example.	37
Table B-3. ENPORT model file for modal example.	40
Table C-1. The subprogram calling tree.	47
Table C-2. The subprogram list and associated source files.	48

Figure

Figure

Figure

Figure

Figure

Figure

Figure

Figure

Figure

Figure

Figure

Figure

Figure

Figure

Figure

Figure

Figure

Figure

## LIST OF FIGURES

	Page
Figure 1-1 Graphical representation of DPM.	2
Figure 2-1 Diagram of the Diverse Platform Modeling technique.	4
Figure 3-1 Topological expansion of SMDs with bond graph domain as host.	6
Figure 3-2 Direct equation generation from SMDs with bond graph domain as host.	7
Figure 4-1 Multiport configuration for a MCK dynamical system model.	9
Figure 4-2 Multiport configuration for a MCK dynamical system model with causality visible.	11
Figure 4-3 Modal configuration for a MCK dynamical system model.	13
Figure 4-4 Modal configuration for a MCK dynamical system with causality visible.	14
Figure 5-1 The Macro Dynamic System for an n-dimensional MCK model.	17
Figure 6-1 Dynamical system employed in numerical study.	19
Figure 6-2 Schematic diagram of FE nodal degrees of freedom.	20
Figure 6-3 Example system modeled using multiport expansion.	22
Figure 6-4 Example system modeled using modal expansion.	23
Figure 6-5 Example system modeled using a MCK Macro Dynamic System node.	24
Figure 6-6 Response of the momentum of the load mass vs. time for sinusoidal forcing.	25
Figure A-1 Sample MCK Standard Description File for two DOF System.	31
Figure A-2 ENPORT directory path for MCK model injection utility.	32
Figure A-3 Sample ENPORT dialog screen for MCK model injection.	33

C

C'

d

E

K

K'

M

M'

V

X

Z

Greek S

φ

η

## NOMENCLATURE

<b>C</b>	damping matrix
<b>C'</b>	modal damping matrix
<b>d</b>	translational displacement
<b>E</b>	external effort vector
<b>K</b>	stiffness matrix
<b>K'</b>	modal stiffness matrix
<b>M</b>	mass matrix
<b>M'</b>	modal mass matrix
<b>V</b>	velocity state vector
<b>X</b>	state vector
<b>Z</b>	physical coordinate vector

### Greek Symbols

<b><math>\phi</math></b>	rotational displacement
<b><math>\eta</math></b>	modal coordinate vector



simulat

(FE) sc

compo

electron

simulat

systems

mention

subsystem

system

Diverse

modelin

subsystem

The mo

of the sy

## Chapter 1

### INTRODUCTION

The practicing engineer has an arsenal of powerful software available for simulating the behavior of dynamical system components. For example, Finite Element (FE) software may be used to develop accurate dynamical models of intricate structural components of a system. Similarly, circuit analysis software may be used for modeling electronic components of a system. Thermal, hydraulic, and other domain-specific simulation software is also available. It is typical, however, for large dynamical systems to be composed of elements or subsystems from a variety of the previously mentioned modeling domains as well as others. The interactions between these subsystems define the dynamical behavior of the conglomerate system.

A current problem in engineering simulation is the difficulty with which the system as a whole may be conveniently modeled using a single software platform. The Diverse Platform Modeling (DPM) technique proposes a solution path for this modeling difficulty. Proper application of DPM will allow the behavior of each subsystem to be modeled using the domain specific platform best suited for the task. The models may then be placed into a common, or "host", platform for the simulation of the system as a whole. This is represented graphically in Figure 1-1.

Elec



The  
broad spe  
been focu  
document  
for additi  
technolog

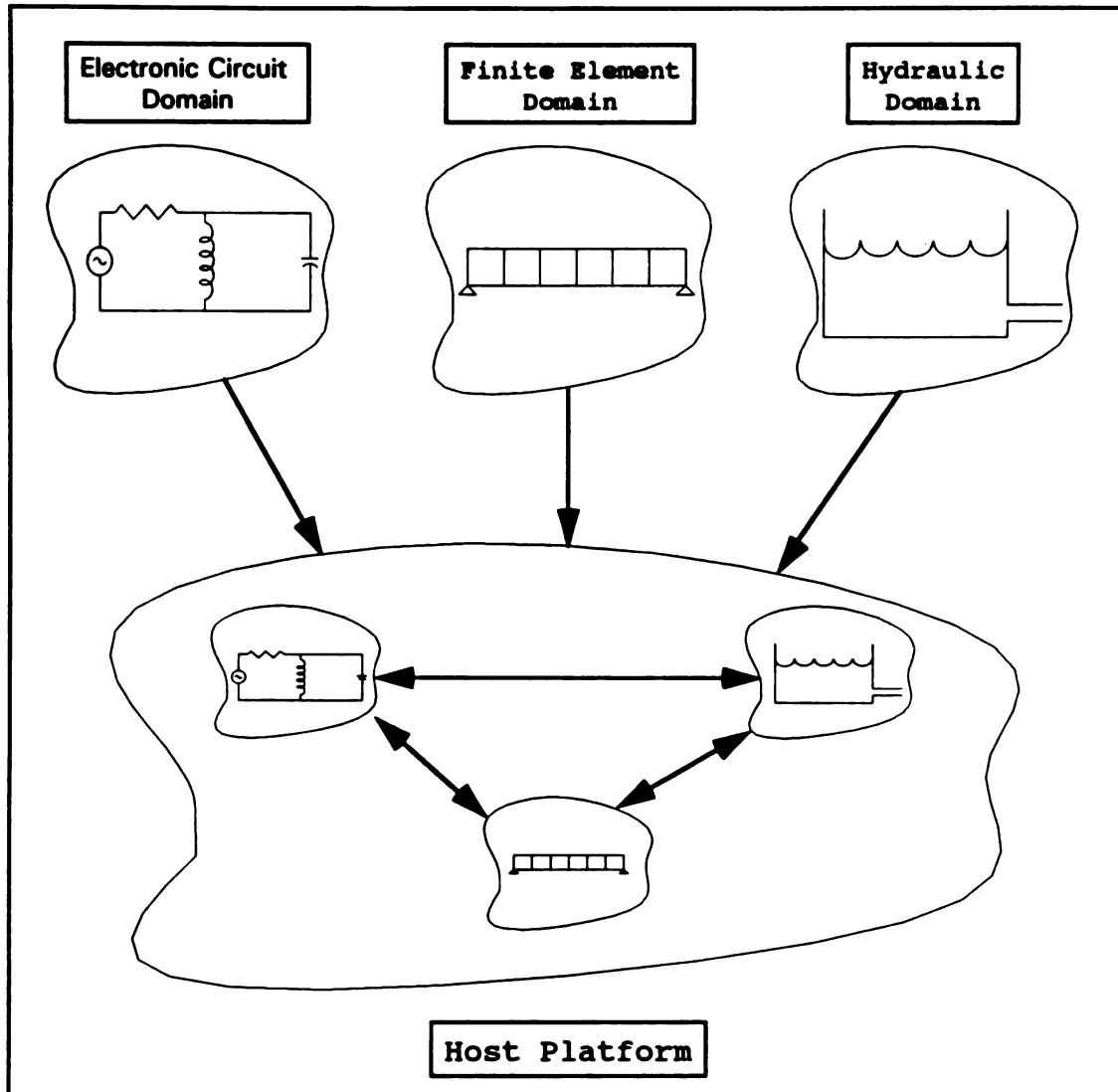


Figure 1-1. Graphical representation of DPM.

The DPM concepts presented in this thesis are general and may be applied to a broad spectrum of modeling problems. The software development effort, however, has been focused on a single common class of dynamical system models. This thesis documents the current state of the proposed modeling technique and provides guidance for additional development. Illustrative examples, borrowed from existing modeling technology, lend support to the feasibility of DPM.

T

abstract f

the bond

first app

approach

DPM wa

simulation

platform

presented

The DPM technique is first presented as a system modeling alternative in abstract form. Two general strategies for application of the proposed technique, using the bond graph modeling domain as a host simulation platform, are presented. The first approach may be considered as a topological expansion technique. The second approach may be considered as a system equation technique. A software realization of DPM was created for use within ENPORT, an existing bond graph modeling and simulation package [1]. An illustrative example of DPM, using ENPORT as the host platform, is provided. Finally, a summary of the proposed modeling technique is presented with recommendations for additional development effort.

Th  
provided  
states and  
data state

Th  
dynamica  
Dynamica  
paper tech  
capable o

## Chapter 2

### THE DIVERSE PLATFORM MODELING TECHNIQUE

The general concepts of the DPM technique are illustrated by the diagram provided in Figure 2-1. The blocks shown in the figure represent three distinct data states and the arrows indicate a process by which data states are transformed. Each data state indicated in Figure 2-1 is discussed briefly in the following paragraphs:

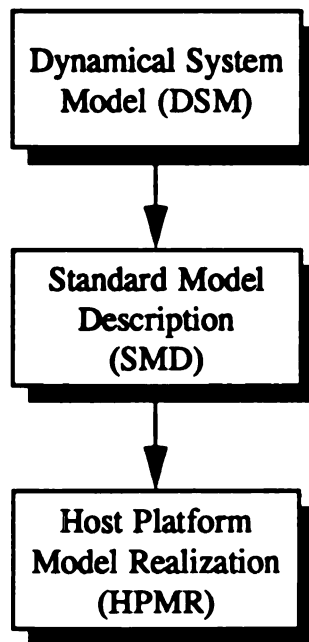


Figure 2-1. Diagram of the Diverse Platform Modeling technique.

The "Dynamical System Model" (DSM) is any reasonable representation of a dynamical system. Two common forms for a DSM are graphical and numerical. Dynamical models may be generated using diverse platforms ranging from pencil and paper techniques to sophisticated software packages. Each modeling platform must be capable of creating a standard model description to communicate modeling information.



commun

essential

simulatio

expressib

for conv

increasin

commun

It

utilities t

Realizatio

of transla

platform

most usef

In

system to

may then

has the re

Interaction

prescribed

whole may

A "Standard Model Description" (SMD) is a standardized format for communicating parameters of a specific problem class. The development of SMDs is essential for communication between diverse modeling platforms and the host simulation environment. Ideally, several classes of dynamical system models should be expressible using SMDs. A communication medium, such as the data file, is required for convenient transport of SMDs between modeling platforms. As a result of the increasing popularity of windowing software, a standardized protocol for "clipboard" communication would also prove useful as a medium for data exchange.

It is the responsibility of the host simulation platform to provide the necessary utilities to convert SMDs into an internally useful format or a "Host Platform Model Realization" (HPMR). It is not necessary that the host simulation platform be capable of translating all classes of dynamical systems represented by SMDs. The host platform capable of the greatest number of translations, however, is potentially the most useful.

In summary, the DPM technique allows individual components of a dynamical system to be modeled using the platform best suited for the task. The resulting models may then be injected into a host platform through the use of SMDs. The host platform has the responsibility of translating SMDs into a locally useful format or HPMR. Interactions between the components of the dynamical system are achieved in a manner prescribed by the host platform. Finally, numerical simulation of the system as a whole may be performed.

## Chapter 3

### THE BOND GRAPH MODELING DOMAIN AS HOST FOR DPM

A feasibility study of DPM was conducted using the bond graph modeling domain as the host environment. The term "system graph" will be used to indicate a bond-graph/block-diagram representation of a dynamical system. Two general strategies for incorporating SMDs into a system graph were investigated. The first strategy, illustrated in Figure 3-1, may be considered a topological expansion approach. Using this approach, a complete system graph representation of the injected model is generated and available for inspection. System equations are then generated from the system graph using standard bond graph and block diagram modeling techniques [2].

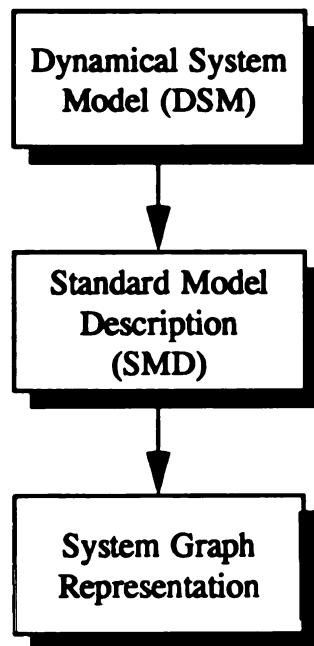


Figure 3-1. Topological expansion of SMDs with bond graph domain as host.

The second model injection strategy, illustrated in Figure 3-2, may be considered a system equation level approach. This method absorbs details of the injected model into

the equi  
of gene  
contain  
system n

Figure 2

Fi  
classes of  
The topol  
implement  
of DPM  
problem c  
class select

the equivalent of a new system graph node type. The new node type must be capable of generating platform-specific system equations directly from the model information contained in the SMD. In this approach, the topological details are reduced to a single system node.

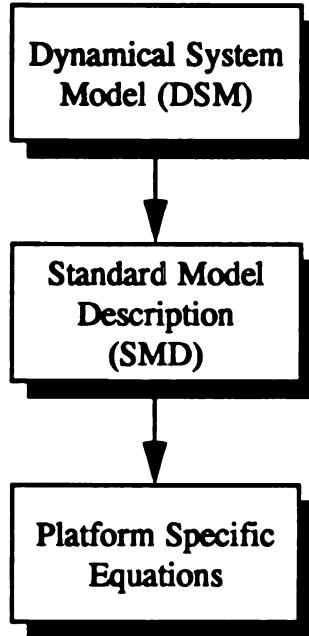


Figure 3-2. Direct equation generation from SMDs with bond graph domain as host.

Figure 3-1 and Figure 3-2 illustrate two general strategies for injecting various classes of externally created dynamical system models into a bond graph host domain. The topological expansion approach was employed in the development of a software implementation of DPM using ENPORT as the host platform. Although the concepts of DPM are not limited to a specific class of dynamical system models, a single problem class was selected for the present software development effort. The problem class selected is the topic of Chapter 4.

modeled  
equation  
technique  
Equation

In Equati  
and stiffn  
velocity,  
vector. I  
common  
following  
presented  
but numer

4.1 Multi

Th  
concepts c  
organizing  
structure c

## Chapter 4

### TOPOLOGICAL APPROACH TO DPM WITH ENPORT AS HOST

The behavior, or local behavior, of many dynamical systems may be accurately modeled using systems of second-order, linear, constant-coefficient, differential equations. These equations may be generated using any of several analytical modeling techniques or computer software platforms and are of the classical form provided in Equation 4-1.

$$M\ddot{\mathbf{z}}(t) + C\dot{\mathbf{z}}(t) + K\mathbf{z}(t) = \mathbf{E}(t) \quad (4-1)$$

In Equation (4-1),  $M$ ,  $C$ , and  $K$  are  $(n \times n)$  real, constant-coefficient mass, dissipation, and stiffness matrices respectively;  $\ddot{\mathbf{z}}$ ,  $\dot{\mathbf{z}}$ , and  $\mathbf{z}$  are  $(n \times 1)$  time-dependent acceleration, velocity, and displacement vectors respectively, and  $\mathbf{E}$  is a  $(n \times 1)$  time dependent effort vector. For the purpose of the present investigation, dynamical system models of this common class (MCK) will be injected into the ENPORT modeling domain. The following articles discuss a software utility developed to achieve this result. The utility presented uses the system graph expansion technique to create two topologically unique but numerically equivalent representations of the injected MCK model.

#### 4.1 Multiport Configuration for Injected MCK Models

The first of the of the two topological expansion techniques exploits multiport concepts of bond graph modeling. Multiport nodes provide a convenient method for organizing the energy of a system into lumped storage and dissipation fields. The structure of Equation (4-1) suggests an organization of this type and lends itself

natural

referred

F

system'

the C n

C; and

constitu

software

indicate

form pr

matrix e

The 1-ju

locations



naturally to the multiport configuration shown in Figure 4-1. The interested reader is referred to reference [2] for a more complete coverage of multiport systems.

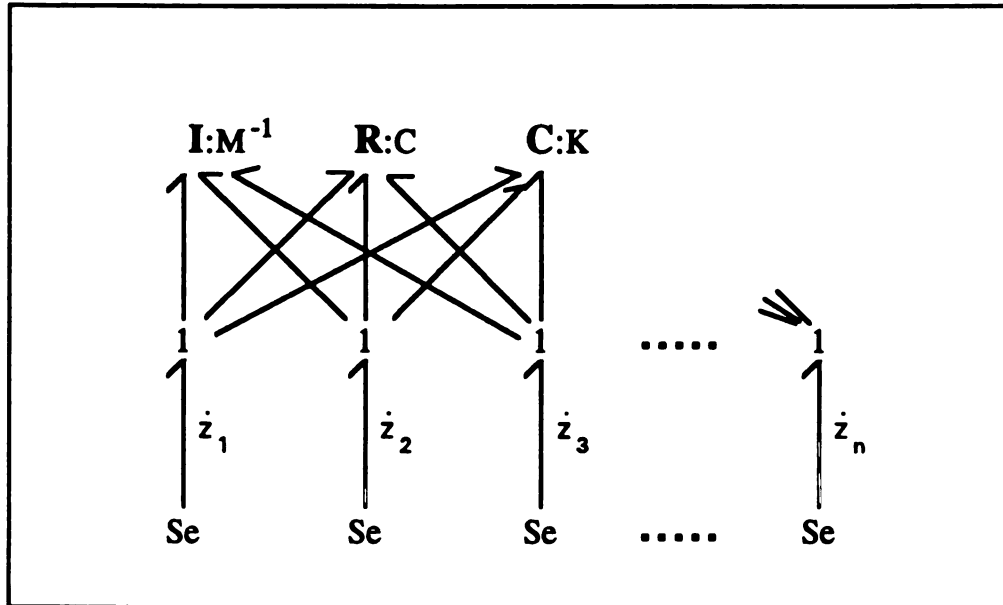


Figure 4-1. Multiport configuration for a MCK dynamical system model.

The multiport I, R, and C nodes shown in Figure 4-1 represent the dynamical system's dissipation, potential energy, and inertia fields respectively. Associated with the C node is the stiffness matrix  $K$ ; associated with the R node is the damping matrix  $C$ ; and associated with the I node is the inverse of the mass matrix  $M$ . The node constitutive equations can be defined by the ENPORT function type "MATRIX". A software utility was created to facilitate automatic system graph generation of the form indicated in Figure 4-1. The utility reads the MCK model description from a file of the form provided in Appendix A1. A suitably-sized IRC bond graph is generated. The matrix elements are placed into ENPORT as indicated in Figure 4-1 ( $:M^{-1}$ ,  $:C$ ,  $:K$ ). The 1-junctions shown in the figure mark the geometric coordinates and provide input locations for external effort sources. Each 1-junction corresponds to a single degree of

freedom

for illus

be prov

only if

dynamic

This iss

of the pr

inherent

preferred

the causa

Procedur

nodes in

of these

implicit i

integratio

freedom in the MCK model. The effort sources (Se) shown in the figure are provided for illustrative purposes. In practice, effort inputs may be of the form indicated or may be provided by connections to other nodes in the system graph. An "Se" is needed only if the input effort is different from zero or the velocity is of interest.

At this point it is important to address the issue of interfacing the injected dynamical system models to other system components in the host simulation platform. This issue is not unique to a specific problem class, to the host platform representation of the problem class, or even to the host simulation platform selected. This issue is an inherent component of dynamical system modeling. Inspection of Figure 4-2 reveals a preferred causality at the interface ports (Se locations). The term "preferred" refers to the causal markings which result from application of the Standard Causality Assignment Procedure (SCAP) [2]. From these markings, the causal requirements at the interface nodes in Figure 4-2 are apparent. Although an alternative causality assignment at any of these locations is physically reasonable, the equations for the system would become implicit in that case. A suitable solver in the nonlinear case would require an implicit integration method.

Figure

4.2 Me

Equati  
equati  
throug

$\bar{U}$  is th  
natural  
and st.

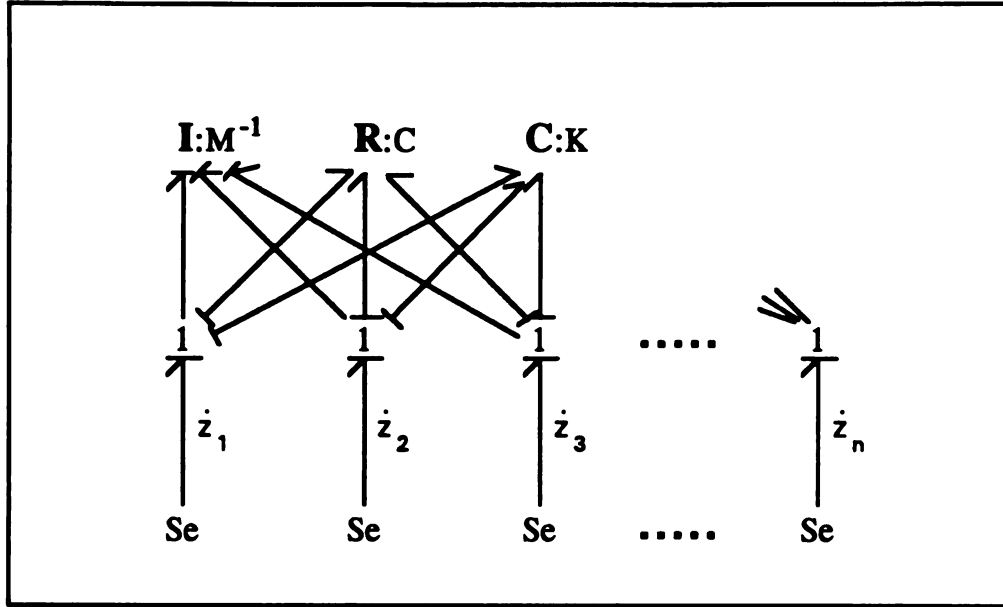


Figure 4-2. Multiport configuration for a MCK dynamical system model with causality visible.

#### 4.2 Modal Configuration for Injected MCK Models

The second topological expansion technique requires that the MCK model of Equation (4-1) be transformed into a set of  $n$  decoupled second-order differential equations. This is accomplished using the transformations provided in Equations (4-2) through (4-4).

$$M' = U^T M U \quad (4-2)$$

$$C' = U^T C U \quad (4-3)$$

$$K' = U^T K U \quad (4-4)$$

$U$  is the  $n \times n$  modal matrix composed of  $n$  column eigenvectors corresponding to the  $n$  natural frequencies of the system.  $M'$ ,  $C'$ , and  $K'$  are the  $(n \times n)$  modal mass, damping, and stiffness matrices respectively. It is important to note that the modal damping

matrix

special

software

for this

defined

Equation

Equation

different

found in

alternat

matrix will only be decoupled by the technique presented in Equation (4-3) for the special but important case of a modally damped system. The current state of the software utility developed for modal expansion of injected MCK models does not check for this condition and should be used with caution. If the displacement vector is defined as

$$\mathbf{z}(t) = \mathbf{U}\boldsymbol{\eta}(t) \quad (4-5)$$

Equation (4-1) may be written as:

$$\mathbf{M}'\ddot{\boldsymbol{\eta}}(t) + \mathbf{C}'\dot{\boldsymbol{\eta}}(t) + \mathbf{K}'\boldsymbol{\eta}(t) = \mathbf{U}^T\mathbf{E}(t) \quad (4-6)$$

Equation (4-6) is a decoupled system of linear, second-order, constant-coefficient differential equations. More comprehensive coverage of modal decomposition may be found in Reference [3] or any standard vibrations text.

The transformation of the MCK model into its modal form facilitates an alternate bond graph representation as illustrated in Figure 4-3.

The  $n^{\text{th}}$   
and has  
defined  
values w  
follows  
introduc  
model a  
bond gra

I  
locations  
ports of



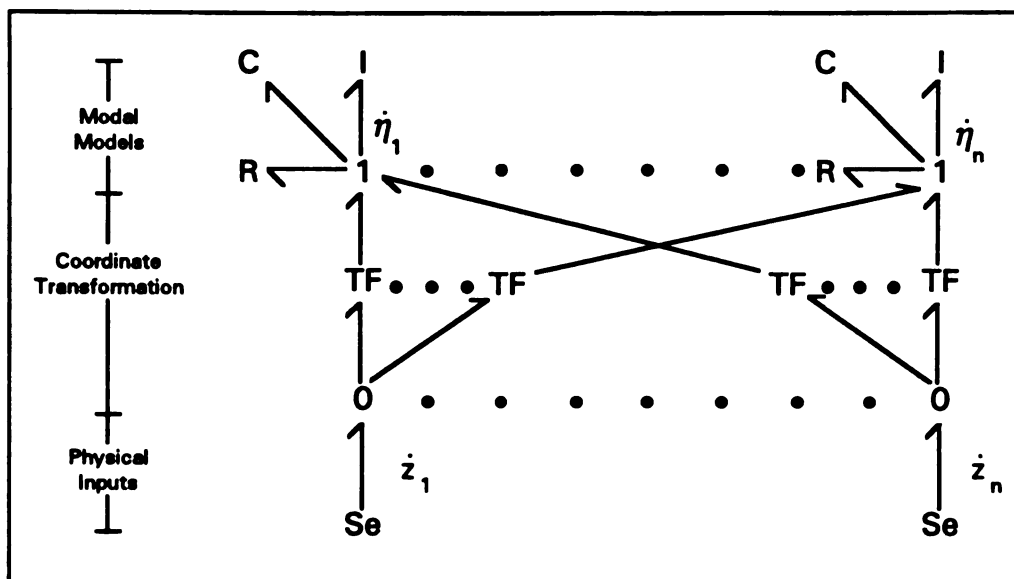


Figure 4-3. Modal configuration for a MCK dynamical system model.

The  $n^{\text{th}}$  I, R, and C nodal grouping shown in Figure 4-3 represents a "modal oscillator" and has parameter values which correspond to the  $n^{\text{th}}$  diagonal elements of the matrices defined in Equation (4-6). The  $n$  transformers (TF) incident to the  $i^{\text{th}}$  0-junction have values which correspond to the  $n^{\text{th}}$  row of the modal transformation matrix ( $U$ ). This follows from Equation (4-6). The 0-junction nodes provide locations for the introduction of external effort sources for each of the  $n$  degrees of freedom in the model and for observation of the resulting motions. More detailed coverage of modal bond graph representation is provided in references [4] and [5].

Inspection of Figure 4-4 reveals a preferred causality at the interface ports (Se locations). This observation is consistent with the discussion regarding the interface ports of the multiport topology presented in article 4.1.



Figure

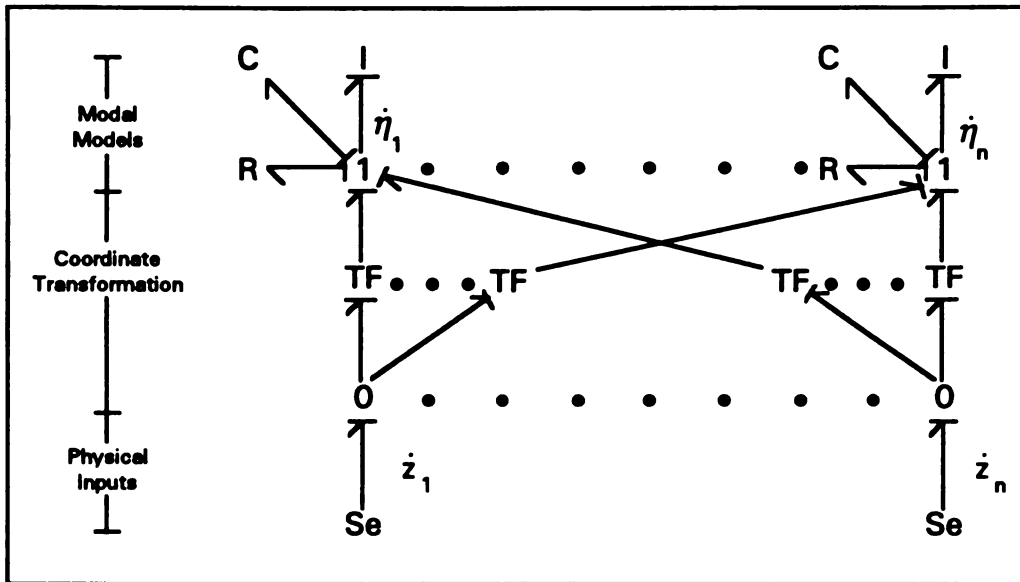


Figure 4-4. Modal configuration for a MCK dynamical system with causality visible.

S

strategy

equation

multiple

dynam

first te

FORT

ENPO

Dynar

system

dynam

indica

5.1 T

devel

Mich

type

is de

code

MDS

equa

## Chapter 5

### SYSTEM EQUATION APPROACH TO DPM WITH ENPORT AS HOST

The system equation approach is presented as an alternative model injection strategy to topological expansion. This approach builds platform-specific system equations from a SMD. The resulting set of equations are represented with a single multiport node in the system graph. Two system-equation level techniques for injecting dynamical system models into ENPORT are presented in the following articles. The first technique, the "Macro Dynamic System" or MDS, requires the creation of a FORTRAN problem description module which must be compiled and linked with the ENPORT software package. The second technique, the "Class Specific Macro Dynamic Node Type", involves creating a new ENPORT node type for each dynamical system class of interest. Although both strategies may be applied to a broad class of dynamical system models, the following articles concentrate on models of the form indicated in Equation (4-1).

#### 5.1 The Macro Dynamic System as Applied to MCK Models

The Macro Dynamic System (MDS) is an ENPORT utility currently under development by Yan Ying Wang, a mechanical engineering doctoral student at Michigan State University. This utility facilitates the creation of a "black box" node type in which the relationship between an arbitrary number of input and output signals is defined by a FORTRAN problem description of a specified format. The FORTRAN code must be compiled and linked with the ENPORT software package. Creating a MDS from a MCK model description requires the formulation of a set of state equations which may be integrated by ENPORT. To this end, Equation (4-1) may be

transfo

Equatic

Equatic

The app

Equatic

expres

Equatic

numen

macro

variab

transformed into a system of  $2n$  state equations by introducing a state variable as in Equation (5-1).

$$V \equiv \dot{Z} \quad (5-1)$$

Equation (4-1) may now be expressed as follows:

$$\dot{V} = -M^{-1}CV - M^{-1}KZ + M^{-1}E \quad (5-2)$$

The appropriate state vector is provided below.

$$X_{2n \times 1} = \begin{bmatrix} V_{n \times 1} \\ Z_{n \times 1} \end{bmatrix} \quad (5-3)$$

Equations (5-1) through (5-3) may be combined to form the compact state space expression shown in Equation (5-4).

$$\begin{bmatrix} \dot{V}_{n \times 1} \\ \dot{Z}_{n \times 1} \end{bmatrix} = \begin{bmatrix} -M^{-1}C & -M^{-1}K \\ I & 0 \end{bmatrix} \begin{bmatrix} V_{n \times 1} \\ Z_{n \times 1} \end{bmatrix} + \begin{bmatrix} M^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} E_{n \times 1} \\ 0 \end{bmatrix} \quad (5-4)$$

Equation (5-4) defines  $2n$  first-order differential equations which are required for numerical integration within ENPORT. The inputs required for the MCK dynamic macro are the applied forces,  $E$ . The output variables may be selected from the state variable list. A graphical representation of the MDS is shown in Figure 5-1.

Fi

The sta  
to the M  
(5-4), a  
output  
an itera  
one tim

5.2 A

platfo  
incon  
assoc  
modu  
Thiro  
MDS  
enca  
Fina



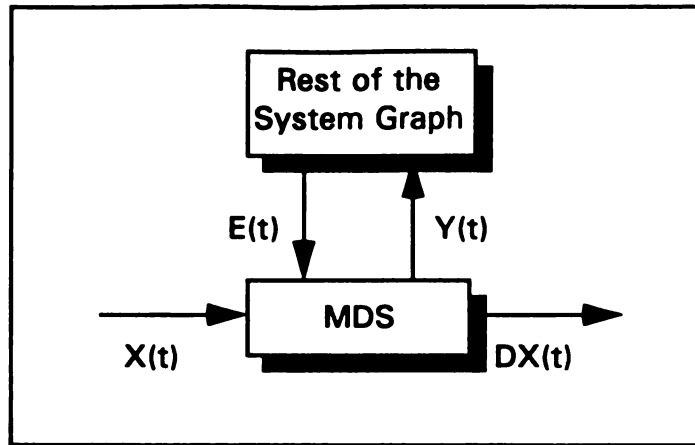


Figure 5-1. The Macro Dynamic System for an n-dimensional MCK model.

The state variables,  $X(t)$ , and the effort variables,  $E(t)$ , shown in Figure 5-1 are passed to the MDS module. The derivatives of the state variables,  $DX(t)$ , defined by Equation (5-4), are calculated by the MDS and returned to the ENPORT solver for storage. The output variables,  $Y(t)$ , are available for use by other nodes in the system graph. This is an iterative process which serves to advance the state of the numerical simulation by one time step.

## 5.2 A Class Specific Macro Dynamic Node Type

The MDS utility presented in the preceding article provides an excellent test platform for a variety of dynamical systems. Its generality, however, presents some inconveniences. First, knowledge of the FORTRAN programming language and associated computer skills are required. Second, compiling a problem description module and linking it with the ENPORT software package is a time consuming process. Third, development of problem description modules is prone to error. Fourth, the MDS is a "black box" representation of a system. Thus, the dynamics of the encapsulated systems must be known to assure proper numerical integration parameters. Finally, supplying users with complete object code for a commercial software package

is not

part, b

softwa

followi

applied

state sy

ENPOR

equival

identica

class sy

system

the sys

node t

descrip

exists

equatic

the sys

may be

is not wise as a business consideration. These difficulties may be overcome, at least in part, by the inclusion of problem class specific MDS node types within the ENPORT software package. The creation of a MCK specific node types is the topic of the following discussion and is the recommended path for future DPM development effort.

Inspection of Equations (5-1) through (5-4) reveals an algorithm which may be applied to an MCK model description for automatic formulation of ENPORT solvable state space equations. The equations generated may be placed directly into the ENPORT equation data base for integration. This development path creates the equivalent of a new node type for ENPORT. The topology of the new node type is identical to the MDS node and is functionally equivalent. The development of system class specific node types for ENPORT has many benefits. For example, dynamical system models could easily be injected into ENPORT using SMDs and represented in the system graph as a single node. In addition, by developing system class specific node types, equation generation is performed automatically and FORTRAN problem description modules are not necessary. One example of this type of node that currently exists in ENPORT is the transfer function node type. This node generates system equations directly from polynomial coefficients supplied when defining parameters for the system graph nodes. Further information regarding the transfer function node type may be found in Reference [1].

DPM f

The

(FE

me

the

pr

6.

c

v

## Chapter 6

### EXAMPLE SYSTEM

The dynamical system illustrated in Figure 6-1 will be used to demonstrate DPM for the class of problems indicated in Equation (4-1).

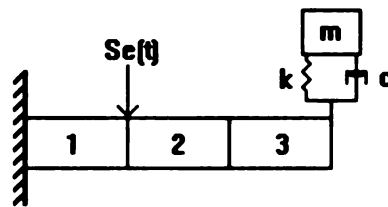


Figure 6-1. Dynamical system employed in numerical study.

The beam indicated in Figure 6-1 will be constructed using the Finite Element Method (FEM). The resulting model will be injected into the ENPORT software platform by means of a SMD. The damped spring mass oscillator will be modeled and attached to the beam within ENPORT. The source of effort will be a simple sine wave also provided within the ENPORT simulation platform.

#### 6.1 The Finite Element Model

The three-element beam structure illustrated in Figure 6-1 will be used to demonstrate DPM. Each node of the finite element mesh contains two degrees of freedom (translational and rotational) as indicated in Figure 6-2.

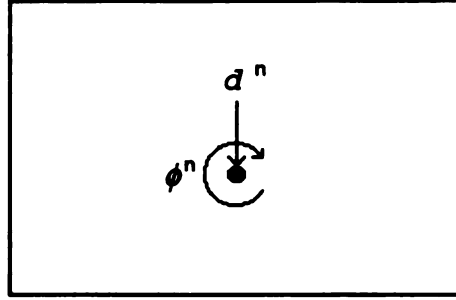


Figure 6-2. Schematic diagram of FE nodal degrees of freedom.

The elemental coefficient matrices are provided in Equations (6-1) and (6-2).

$$M_e = \frac{\rho A L}{420} \begin{bmatrix} 156 & 22L & 54 & -13L \\ 22L & 4L^2 & 13L & -3L^2 \\ 54 & 13L & 156 & -22L \\ -13L & -3L^2 & -22L & 4L^2 \end{bmatrix} \quad (6-1)$$

$$K_e = \frac{EI}{L^3} \begin{bmatrix} 12 & 6L & -12 & 6L \\ 6L & 4L^2 & -6L & 2L^2 \\ -12 & -6L & 12 & -6L \\ 6L & 2L^2 & -6L & 4L^2 \end{bmatrix} \quad (6-2)$$

$M_e$  is a consistent mass matrix,  $K_e$  is the stiffness matrix,  $E$  is the material modulus of elasticity,  $I$  is the moment of inertia,  $\rho$  is the material density,  $A$  is the cross-sectional area, and  $L$  is the elemental length. The elemental mass and stiffness matrices are combined using standard FE techniques to form the global system of equations for the beam. The resulting (6x6) equation is of the following form,

$$M\ddot{z} + Kz = E \quad (6-3)$$

The global  $M$  and  $K$  matrices indicated in Equation (6-3) are provided in Appendix B1 in the form of a SMD file and  $z$  is defined below.

In  
dis  
ve  
ex

$$z = \begin{bmatrix} d^1 \\ \phi^1 \\ d^2 \\ \phi^2 \\ d^3 \\ \phi^3 \end{bmatrix} \quad (6-4)$$

In Equation (6-4),  $d$  indicates translational displacement and  $\phi$  indicates rotational displacement. Superscripts indicate the node number (see Figure 6-2). The effort vector provided in Equation (6-5) indicates the nodal coordinates which are stimulated externally.

$$E = \begin{bmatrix} E_1(t) \\ 0 \\ 0 \\ 0 \\ E_5(t) \\ 0 \end{bmatrix} \quad (6-5)$$

The external effort sources indicated in Equation (6-5) are not necessarily defined explicitly. As previously mentioned,  $E_1(t)$  is defined as a simple sine wave for this example. The effort applied to the fifth degree of freedom,  $E_5(t)$ , is the reaction force generated by the damped spring mass oscillator and may not be assigned explicitly. More detailed discussion regarding the formulation of FE models is provided in References [6] and [7]. The SMD created for the FE model is provided in Appendix B1. The following articles illustrate three unique yet numerically equivalent system graph models of the dynamical system shown in Figure 6-1.



6.2 M

multip

graph i

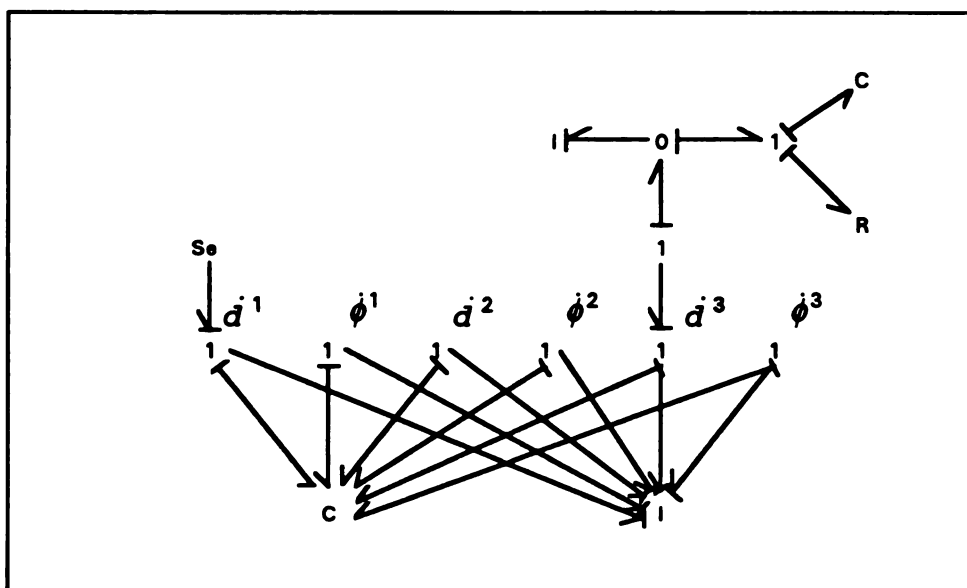
Care

causa

prop

## 6.2 Multiport Expansion

The dynamical system shown in Figure 6-1 was modeled in ENPORT using the multiport expansion approach to represent the beam component. The resulting system graph is shown in Figure 6-3.



**Figure 6-3. Example system modeled using multiport expansion.**

Careful inspection of the causal markings shown in Figure 6-3 indicates that integral causality has been maintained throughout the system graph. Thus, the model is properly posed for the ENPORT solver.

6.3 M

repres

shown

As in

through

### 6.3 Modal Expansion

6.4 M

the dy

Figur

As ind  
inputs  
that w  
require  
conside

## 6.4 MCK Macro Dynamic System

The system graph shown in Figure 6-5 represents the hypothetical appearance of the dynamical system as modeled using a MCK macro dynamic system node.

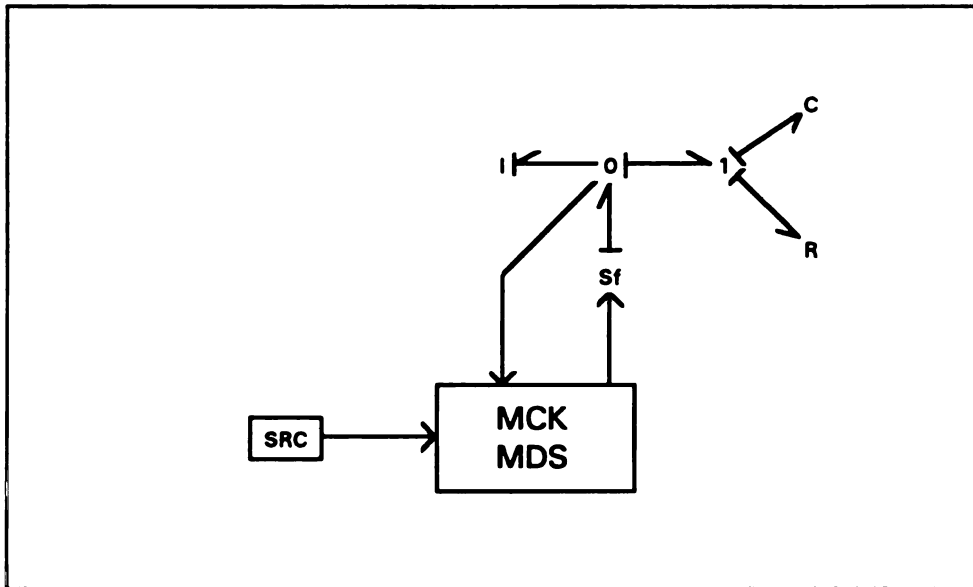


Figure 6-5. Example system modeled using a MCK Macro Dynamic System node.

As indicated in Figure 6-5, the MCK Macro Dynamic System node requires signals as inputs and provides signals as outputs. There are, however, no fundamental difficulties that would prevent this hypothetical dynamic node type from being developed to require power bonds to communicate information. This design issue is left for the consideration of future ENPORT developers.

## 6.5 Numerical Results

The numerical results obtained during the numerical simulation of the system shown in Figure 6-1 were identical for the two system graph configurations tested (see Figure 6-3 and Figure 6-4). Thus, a single figure, representative of both topologies, is provided in Figure 6-6.

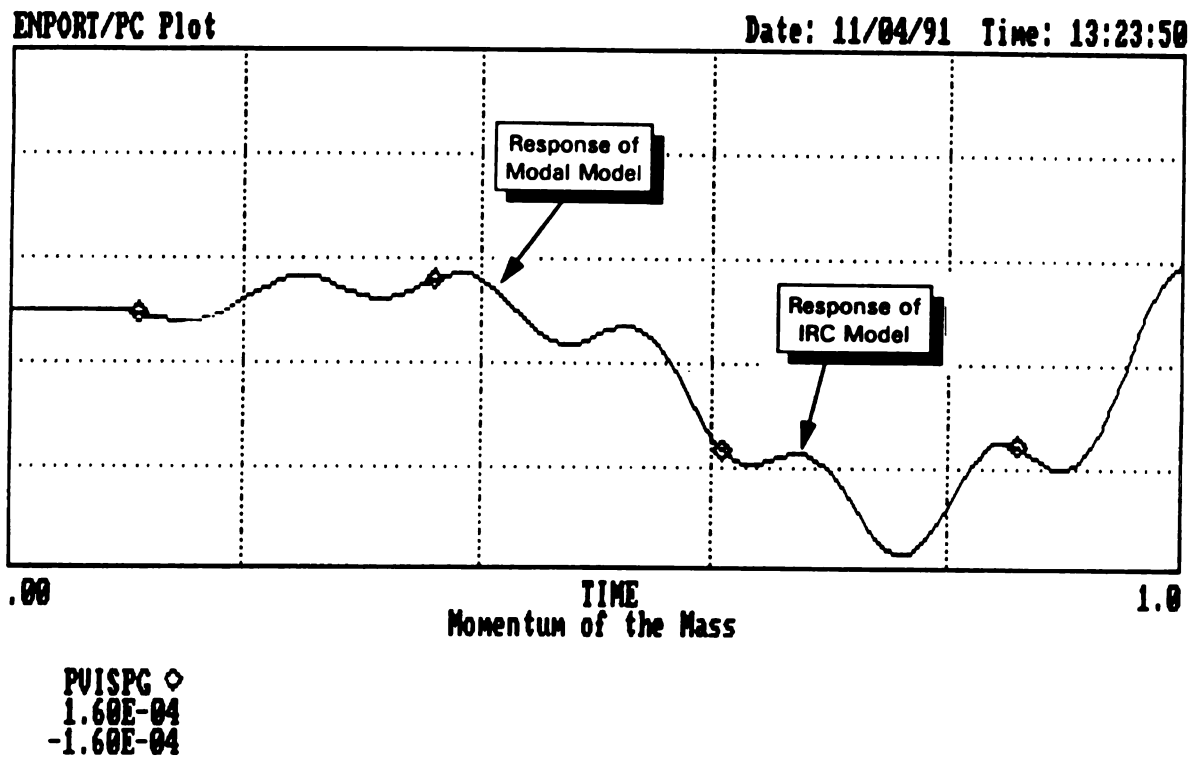


Figure 6-6. Response of the momentum of the load mass vs. time for sinusoidal forcing.

7.1 S

of too

develo

ENPO

means

Injecte

other

compl

do pr

Recon

7.2 R

inform

injecti

system

the in

recom



## Chapter 7

### SUMMARY AND RECOMMENDATIONS

#### 7.1 Summary

The Diverse Platform Modeling Technique addresses the need for a coherent set of tools when modeling dynamical systems. Specifically, two software utilities were developed that facilitate the injection of MCK dynamical system models into the ENPORT modeling platform. MCK system models are communicated to ENPORT by means of a Standard Model Description file developed specifically for the task. Injected MCK models are represented in the ENPORT system graph and interact with other system nodes according to the standard rules of bond graph modeling.

The utilities created to facilitate DPM within ENPORT do not represent a complete set of tools which will solve all modeling difficulties. The utilities, however, do provide support for further development of the proposed modeling technique. Recommendations for development are provided in the following article.

#### 7.2 Recommendations for Future Development

Topological expansion of MCK dynamical system models provided useful information pertaining to the development of system graph "macros" and model injection strategies. First, the two topological representations of MCK dynamic systems investigated (multiport and modal form) indicate a specific causality forced at the input sites using the Standard Causality Assignment Procedure [2]. Thus, it is recommended that future development include a method for enforcing or "locking"

propo-

simul

produ

be an

useful

direct

advan

experi

capabi

increa

that se

topolog

proper causality at the input sites of macro nodes during the model building stage of the simulation process. Second, the topological expansion of even small MCK models produced unreasonably large system graph representations. There does not appear to be any strong evidence that these visual representations are necessary or particularly useful for the user. Thus, it is recommended that future development be focused on the direct formulation of system equations from the SMDs. This approach has the advantage of reducing topological details to a single node in the system graph. Finally, experience suggests that the platform best suited for the exploitation of graphical macro capabilities are windowing environments. This type of environment has become increasingly popular among software users and developers. Thus, it is recommended that software design and development of advanced ENPORT capabilities, such as topological macros, be restricted to environments of this type.

## **LIST OF REFERENCES**

## LIST OF REFERENCES

1. Rosenberg, R. C., 'The ENPORT Reference Manual', Rosencode Associates Inc., Lansing, Michigan, 1990.
2. Rosenberg, R. C. and Karnopp, D. C., INTRODUCTION TO PHYSICAL SYSTEM DYNAMICS, McGraw-Hill, New York, 1983.
3. Meirovitch, L., ANALYTICAL METHODS IN VIBRATIONS, Macmillian, New York, 1967.
4. Margolis, D. L. and Young, G. E., 'Reduction of Models of Large Scale Lumped Structures Using Normal Modes and Bond Graphs', J. Franklin Inst., Vol. 304, No. 1, pp. 65-79, July 1977.
5. Margolis, D. L., 'A Survey of Bond Graph Modeling for Interacting Lumped and Distributed Systems', J. Franklin Inst., Vol. 319, No. 1/2, pp. 125-135, January/February 1985.
6. Reddy, J. N., AN INTRODUCTION TO THE FINITE ELEMENT METHOD, McGraw-Hill, New York, 1984.
7. Logan, D. L., A FIRST COURSE IN THE FINITE ELEMENT METHOD, PWS, Boston, 1986.
8. Press, W. H. and Flannery, B. P. and Teukolsky, S. A. and Vetterling, W. T., NUMERICAL RECIPES - THE ART OF SCIENTIFIC COMPUTING, Cambridge University Press, Cambridge, 1986.
9. Shigley, J. E. and Mitchell, L. D., MECHANICAL ENGINEERING DESIGN, Fourth Edition, McGraw-Hill, New York, 1983.

## **APPENDICES**

S  
W  
i

## **APPENDIX A**

### **USER DOCUMENTATION**

**The information in this appendix pertains to the use of the MCK model injection software utility created for ENPORT. A sample SMD file is presented in Appendix A1 with discussion regarding format requirements. An abbreviated ENPORT session is included in Appendix A2 that demonstrates user access to the new utility.**



## APPENDIX A1

### SAMPLE MCK STANDARD MODEL DESCRIPTION FILE

A sample SMD file for a two degree of freedom MCK model is shown in Figure 11. Note that text may not be placed in the first column of the file. Data entered in the "FILE", "NAME", "TITLE", and "DESCRIPTION" fields is not used internally by ENPORT. These fields are provided for user documentation and future internal use. The dimension of the MCK model is placed in the "DIMENSION" field. The current maximum value for this parameter is ten. The "FORCE INFORMATION" field contains two columns for information. The first column must be an integer value equal to zero or one. If a value of one is entered in this column, an effort source is placed in the system graph with the value indicated in the second column of the corresponding row. A value of zero in the first field prevents the placement of an effort source in the system graph for the associated degree of freedom. Thus, if a particular degree of freedom in the MCK model is of interest, a value of one must be placed in the first column. The proper format for data entry into the "MASS MATRIX", "STIFFNESS MATRIX", and "DAMPING MATRIX" fields is provided in Figure A-1. A "\*" in the second column of the file indicates a comment line. The delimiters for a matrix entry are "[" and "]". The delimiter for a matrix row is ";". These delimiters are required. The mass matrix and stiffness matrix are required. The damping matrix, however, is not required. Blank lines are ignored by the processor. Finally, the file is terminated with an "END-FILE" statement.

**HEADING**

```

FILE          (Not used internally)
  VIBS2.MCK

NAME          (Eight chars max - for future macro use)
  VIBS2

TITLE        (Not used internally)
  MCK test file.

DESCRIPTION   (10 lines max - Not used internally)
  This is a Standard Model Description file for a two degree of
  freedom MCK system.

DIMENSION     (10 is max for now)
  0002

FORCE INFORMATION (0 = ignore, 1 = include)
[
  1          1.0000E+00
  1          1.0000E+00
]

MASS MATRIX   (Required - Format is 5x,1p,E13.4,1x....)
* bbsx.xxxxE+xx bbsx.xxxxE+xx bbsx.xxxxE+xx bbsx.xxxxE+xx bbsx.xxxxE+xx
[
  1.0000E+00    0.0000E+00;
  0.0000E+00    1.0000E+00;
]

STIFFNESS MATRIX (Required - Format is 5x,1p,E13.4,1x....)
[
  2.0000E+00   -1.0000E+00;
 -1.0000E+00    1.0000E+00;
]

DAMPING MATRIX (Values optional - heading and [] required.)
[
]

END-FILE

```

Figure A-1. Sample MCK Standard Description File for two DOF System.

## APPENDIX A2

### USER ACCESS FROM ENPORT

Figure 12 indicates the menu path required to gain access to ENPORT's MCK model injection utility.

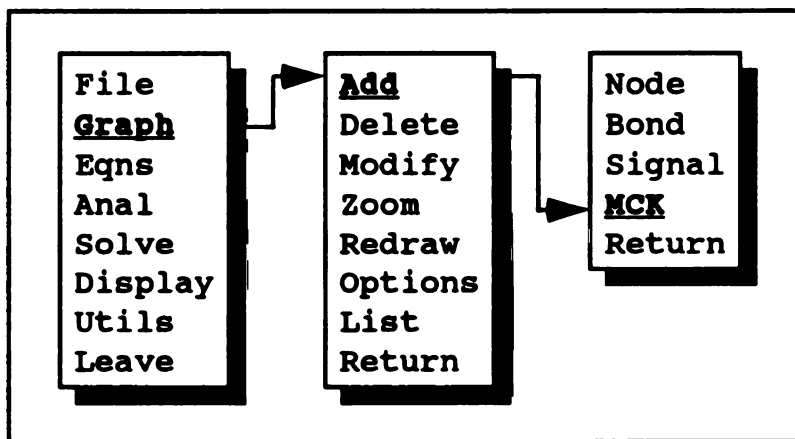


Figure A-2. ENPORT directory path for MCK model injection utility.

Once the MCK option indicated in Figure 12 has been selected, a small amount of user input is required. User input is achieved using the ENPORT dialog screen shown in Figure A-3.

Enter file name :		VIBS3.MCK	
Processing file command:			
HEADING			
DIMENSION	2		
FORCE INFORMATION	1	1.0000E+00	
MASS MATRIX	4	1.0000E+00	1.0000E+00
STIFFNESS MATRIX	4	2.0000E+00	1.0000E+00
DAMPING MATRIX	0		
END FILE			
Macro Type?		<input type="checkbox"/> Modal	<input checked="" type="checkbox"/> Multiport
Nodes required		:	6
Connectors required		:	6

Figure A-3. Sample ENPORT dialog screen for MCK model injection.

The shadowed boxes in Figure A-3 indicate the user input portions of the dialog screen. All other fields shown in the figure are status and debug information. The two numerical fields following the "FORCE INFORMATION" heading are the values found in the first row of the corresponding heading of associated MCK data file (see Figure 11). Similarly, the numerical fields following the "MASS MATRIX", "STIFFNESS MATRIX", and "DAMPING MATRIX" headers, are the number of elements read, first element received, and last element received respectively. After all appropriate information has been entered, ENPORT returns the user to the system graph. A location for topological expansion of the MCK data is requested and the appropriate system graph model is constructed.

## **APPENDIX B**

### **THE EXAMPLE SYSTEM**

**This Appendix contains additional information regarding the example system presented in Chapter 6.**

## APPENDIX B1

### MCK FILE FOR THE EXAMPLE SYSTEM

Table B-1. MCK file for the example system.

#### HEADING

##### FILE

FEM10.MCK

##### NAME

FEM10 (eight chars max - for future macro use)

##### TITLE

FEM test file.

##### DESCRIPTION

This file contains the mass and stiffness matrices for a 3-element 2-DOF FE model. The boundary nodes (1 and 2) are not included. The following are the physical parameters:

Material : Steel

$E = 30 \times 10^6 \text{ lb/in}^2$

$I = b \cdot h^3 / 12$

$p = 484 \text{ lb/ft}^3$

$b = 1" \quad h = 1" \quad l = 10" \text{ per element.}$

##### DIMENSION

(10 is max for now)

0006

##### FORCE INFORMATION (0 = Ignore, 1 = Include)

```
[
  1      1.0000E+01
  0      0.0000E+00
  0      0.0000E+00
  0      0.0000E+00
  1      1.0000E+02
  0      0.0000E+00
]
```

##### MASS MATRIX (Format is 5x,1p,E13.4,1x....)

\* bbsx.xxxxE+xx bbsx.xxxxE+xx bbsx.xxxxE+xx bbsx.xxxxE+xx bbsx.xxxxE+xx

```
[
  3.5880E+03    0.0000E+00    6.2100E+02   -1.4950E+03    0.0000E+00
  0.0000E+00;
  0.0000E+00    9.2000E+03    1.4950E+03   -3.4500E+03    0.0000E+00
  0.0000E+00;
  6.2100E+02    1.4950E+03    3.5880E+03    0.0000E+00    6.2100E+02
 -1.4950E+03;
 -1.4950E+03   -3.4500E+03    0.0000E+00    9.2000E+03    1.4950E+03
 -3.4500E+03;
  0.0000E+00    0.0000E+00    6.2100E+02    1.4950E+03    1.7940E+03
 -2.5300E+03;
  0.0000E+00    0.0000E+00   -1.4950E+03   -3.4500E+03   -2.5300E+03
  4.6000E+03;
]
```

Table B-1 (cont'd).

## STIFFNESS MATRIX

```

[
  6.0000E+04    0.0000E+00   -3.0000E+04    1.5000E+05    0.0000E+00
  0.0000E+00;
  0.0000E+00    2.0000E+06   -1.5000E+05    5.0000E+05    0.0000E+00
  0.0000E+00;
 -3.0000E+04   -1.5000E+05    6.0000E+04    0.0000E+00   -3.0000E+04
  1.5000E+05;
  1.5000E+05    5.0000E+05    0.0000E+00    2.0000E+06   -1.5000E+05
  5.0000E+05;
  0.0000E+00    0.0000E+00   -3.0000E+04   -1.5000E+05    3.0000E+04
 -1.5000E+05;
  0.0000E+00    0.0000E+00    1.5000E+05    5.0000E+05   -1.5000E+05
  1.0000E+06;
]

```

## DAMPING MATRIX (OPTIONAL)

```

[
]

```

END-FILE

## APPENDIX B2

### ENPORT MODEL FILE FOR EXAMPLE PROBLEM (MULTIPOINT)

Table B-2. ENPORT model file for multiport example.

#### HEADING

```
FILE    MODEL                                15:41:19  11/11/91  ENPORT/PC 4.01
FEMMUL.ENP                                Model_name: mul port
```

#### TITLE

Multiport expansion of 6 DOF MCK with spring mass oscillator.

#### DESCRIPTION

This is a 6 degree of freedom FE model with a spring mass attached.  
The multiport expansion technique has been used to represent the FE model.

#### SYSTEM GRAPH DESCRIPTION

NODE	TYPE	XLOC	YLOC	ACT	MACRO
SV101	MEGV	-400.	200.	T	
1V101	M1GV	-400.	100.	T	
1V102	M1GV	-300.	100.	T	
1V103	M1GV	-200.	100.	T	
1V104	M1GV	-100.	100.	T	
1V105	M1GV	0.	100.	T	
1V106	M1GV	100.	100.	T	
I1	M1GV	-400.	-100.	T	
C1	MCGV	-200.	-100.	T	
1J1	M1GV	0.	400.	T	
O2	MOGV	0.	600.	T	
1J2	M1GV	0.	800.	T	
1J3	M1GV	200.	600.	T	
CSPG	MCGV	400.	500.	T	
ISPG	M1GV	200.	900.	T	
RSPG	MRGV	400.	700.	T	

CONNECTOR	TYPE	FROM	TO	VERTICES
SB101	BG V	SV101	1V101	
IB101	BG V	1V101	I1	
IB102	BG V	1V102	I1	
IB103	BG V	1V103	I1	
IB104	BG V	1V104	I1	
IB105	BG V	1V105	I1	
IB106	BG V	1V106	I1	
CB101	BG V	1V101	C1	
CB102	BG V	1V102	C1	
CB103	BG V	1V103	C1	
CB104	BG V	1V104	C1	
CB105	BG V	1V105	C1	
CB106	BG V	1V106	C1	



Table B-2 (cont'd).

V22	BG V	1J1	02
VDIF2	BG V	02	1J3
VSPG2	BG V	1J3	CSPG
V3	BG V	02	1J2
VISPG	BG V	1J2	ISPG
VSPG	BG V	1J3	RSPG
V1	BG V	1J1	1V105

## NODE EQUATIONS

Named\_parameters: 0

Number of outputs: 16

Node: SV101      Connectors: SB101

Equation:  $Y = \text{SIN} (X, P)$       1 1 2

Y list	X list	Parameters	Index
ESB101	TIME	1.0000E+00	0
		4.2190E+01	0

Node: I1      Connectors: IB101      IB102      IB103

Equation:  $Y = \text{MATRIX} (X, P)$       6 6 0

	PIB101	PIB102	PIB103
FIB101	3.2510E-04	4.2076E-05	-4.7933E-05
FIB102	4.2076E-05	1.4687E-04	-4.5737E-05
FIB103	-4.7933E-05	-4.5737E-05	3.8578E-04
FIB104	9.1433E-05	8.1985E-05	4.5209E-05
FIB105	6.7466E-05	5.9117E-05	2.3810E-04
FIB106	9.0102E-05	7.9139E-05	2.9024E-04

	PIB104	PIB105	PIB106
FIB101	9.1433E-05	6.7466E-05	9.0102E-05
FIB102	8.1985E-05	5.9117E-05	7.9139E-05
FIB103	4.5209E-05	2.3810E-04	2.9024E-04
FIB104	2.3822E-04	2.6082E-04	3.3681E-04
FIB105	2.6082E-04	2.8643E-03	1.8484E-03
FIB106	3.3681E-04	1.8484E-03	1.5809E-03

Node: C1      Connectors: CB101      CB102      CB103

Equation:  $Y = \text{MATRIX} (X, P)$       6 6 0

	QCB101	QCB102	QCB103
ECB101	6.0000E+04	0.0000E+00	-3.0000E+04
ECB102	0.0000E+00	2.0000E+06	-1.5000E+05
ECB103	-3.0000E+04	-1.5000E+05	6.0000E+04
ECB104	1.5000E+05	5.0000E+05	0.0000E+00
ECB105	0.0000E+00	0.0000E+00	-3.0000E+04
ECB106	0.0000E+00	0.0000E+00	1.5000E+05

	QCB104	QCB105	QCB106
ECB101	1.5000E+05	0.0000E+00	0.0000E+00
ECB102	5.0000E+05	0.0000E+00	0.0000E+00
ECB103	0.0000E+00	-3.0000E+04	1.5000E+05
ECB104	2.0000E+06	-1.5000E+05	5.0000E+05
ECB105	-1.5000E+05	3.0000E+04	-1.5000E+05
ECB106	5.0000E+05	-1.5000E+05	1.0000E+06

Table B-2 (cont'd).

Node: CSPG	Connectors: VSPG2			
Equation:	Y = GAIN ( X, P )	1	1	1
Y list	X list	Parameters	Index	
ĒVSPG2	Q̄VSPG2	5.0000E+02	0	
Node: ISPG	Connectors: VISPG			
Equation:	Y = ATT ( X, P )	1	1	1
Y list	X list	Parameters	Index	
F̄VISPG	P̄VISPG	1.0000E+01	0	
Node: RSPG	Connectors: VSPG			
Equation:	Y = GAIN ( X, P )	1	1	1
Y list	X list	Parameters	Index	
ĒVSPG	F̄VSPG	1.0000E+01	0	

END-FILE

## APPENDIX B3

### ENPORT MODEL FILE FOR EXAMPLE PROBLEM (MODAL)

Table B-3. ENPORT model file for modal example.

#### HEADING

```
FILE    MODEL                                15:45:18  11/11/91  ENPORT/PC 4.01
FEMMOD.ENP                                Model_name: fem mode
```

#### TITLE

This is a 6 DOF FE model with spring mass oscillator attached.

#### DESCRIPTION

This is a 6 DOF Finite Element Model injected into ENPORT using the modal expansion technique. A damped spring mass oscillator has been attached to the injected Finite Element model.

#### SYSTEM GRAPH DESCRIPTION

NODE	TYPE	XLOC	YLOC	ACT	MACRO
SM101	MEGV	-400.	200.	T	
OM101	MOGV	-400.	100.	T	
OM105	MOGV	2400.	100.	T	
TF101	MTGV	-400.	0.	T	
TF102	MTGV	-300.	0.	T	
TF103	MTGV	-200.	0.	T	
TF104	MTGV	-100.	0.	T	
TF105	MTGV	0.	0.	T	
TF106	MTGV	100.	0.	T	
TF125	MTGV	2400.	0.	T	
TF126	MTGV	2500.	0.	T	
TF127	MTGV	2600.	0.	T	
TF128	MTGV	2700.	0.	T	
TF129	MTGV	2800.	0.	T	
TF130	MTGV	2900.	0.	T	
1M101	M1GV	-400.	-200.	T	
1M102	M1GV	300.	-200.	T	
1M103	M1GV	1000.	-200.	T	
1M104	M1GV	1700.	-200.	T	
1M105	M1GV	2400.	-200.	T	
1M106	M1GV	3100.	-200.	T	
IM101	M1GV	-400.	-300.	T	
IM102	M1GV	300.	-300.	T	
IM103	M1GV	1000.	-300.	T	
IM104	M1GV	1700.	-300.	T	
IM105	M1GV	2400.	-300.	T	
IM106	M1GV	3100.	-300.	T	
CM101	MCGV	-300.	-300.	T	
CM102	MCGV	400.	-300.	T	
CM103	MCGV	1100.	-300.	T	
CM104	MCGV	1800.	-300.	T	

Table B-3 (cont'd).

CM105	MCGV	2500.	-300.	T
CM106	MCGV	3200.	-300.	T
1	M1GV	2400.	300.	T
02	M0GV	2400.	400.	T
1J2	M1GV	2400.	500.	T
ISPG	M1GV	2600.	500.	T
1J3	M1GV	2300.	400.	T
RSPG	MRGV	2200.	500.	T
CSPG	MCGV	2200.	300.	T

CONNECTOR	TYPE	FROM	TO	VERTICES
SB101	BG V	SM101	OM101	
TB101	BG V	OM101	TF101	
TB102	BG V	OM101	TF102	
TB103	BG V	OM101	TF103	
TB104	BG V	OM101	TF104	
TB105	BG V	OM101	TF105	
TB106	BG V	OM101	TF106	
TB125	BG V	OM105	TF125	
TB126	BG V	OM105	TF126	
TB127	BG V	OM105	TF127	
TB128	BG V	OM105	TF128	
TB129	BG V	OM105	TF129	
TB130	BG V	OM105	TF130	
1B101	BG V	TF101	1M101	
1B102	BG V	TF102	1M102	
1B103	BG V	TF103	1M103	
1B104	BG V	TF104	1M104	
1B105	BG V	TF105	1M105	
1B106	BG V	TF106	1M106	
1B125	BG V	TF125	1M101	
1B126	BG V	TF126	1M102	
1B127	BG V	TF127	1M103	
1B128	BG V	TF128	1M104	
1B129	BG V	TF129	1M105	
1B130	BG V	TF130	1M106	
IB101	BG V	1M101	IM101	
IB102	BG V	1M102	IM102	
IB103	BG V	1M103	IM103	
IB104	BG V	1M104	IM104	
IB105	BG V	1M105	IM105	
IB106	BG V	1M106	IM106	
CB101	BG V	1M101	CM101	
CB102	BG V	1M102	CM102	
CB103	BG V	1M103	CM103	
CB104	BG V	1M104	CM104	
CB105	BG V	1M105	CM105	
CB106	BG V	1M106	CM106	
V22	BG V	1	02	
VDIF2	BG V	02	1J3	
V3	BG V	02	1J2	
VISPG	BG V	1J2	ISPG	
VSPG	BG V	1J3	RSPG	
VSPG2	BG V	1J3	CSPG	
V1	BG V	1	OM105	

Table B-3 (cont'd).

## NODE EQUATIONS

Named\_parameters: 0

Number of outputs: 28

Node: SM101	Connectors: SB101			
Equation: Y = SIN	( X, P )	1 1 2		
Y_list	X_list	Parameters	Index	
ESB101	TIME	1.0000E+00	0	
		4.2190E+01	0	
Node: TF101	Connectors: TB101	1B101		
FTB101	= MTF101	* F1B101		
E1B101	= MTF101	* ETB101		
Equation: Y = CON	( X, P )	1 0 1		
Y_list	X_list	Parameters	Index	
MTF101		1.0000E+00	0	
Node: TF102	Connectors: TB102	1B102		
FTB102	= MTF102	* F1B102		
E1B102	= MTF102	* ETB102		
Equation: Y = CON	( X, P )	1 0 1		
Y_list	X_list	Parameters	Index	
MTF102		1.0000E+00	0	
Node: TF103	Connectors: TB103	1B103		
FTB103	= MTF103	* F1B103		
E1B103	= MTF103	* ETB103		
Equation: Y = CON	( X, P )	1 0 1		
Y_list	X_list	Parameters	Index	
MTF103		1.0000E+00	0	
Node: TF104	Connectors: TB104	1B104		
FTB104	= MTF104	* F1B104		
E1B104	= MTF104	* ETB104		
Equation: Y = CON	( X, P )	1 0 1		
Y_list	X_list	Parameters	Index	
MTF104		1.0000E+00	0	
Node: TF105	Connectors: TB105	1B105		
FTB105	= MTF105	* F1B105		
E1B105	= MTF105	* ETB105		
Equation: Y = CON	( X, P )	1 0 1		
Y_list	X_list	Parameters	Index	
MTF105		1.0000E+00	0	
Node: TF106	Connectors: TB106	1B106		
FTB106	= MTF106	* F1B106		
E1B106	= MTF106	* ETB106		
Equation: Y = CON	( X, P )	1 0 1		
Y_list	X_list	Parameters	Index	
MTF106		1.0000E+00	0	

Table B-3 (cont'd).

Node: TF125	Connectors: TB125	1B125	
FTB125	= MTF125	* F1B125	
E1B125	= MTF125	* ETB125	
Equation: Y = CON	( X, P )	1 0 1	
Y_list	X_list	Parameters	Index
MTF125		8.3144E+00	0
Node: TF126	Connectors: TB126	1B126	
FTB126	= MTF126	* F1B126	
E1B126	= MTF126	* ETB126	
Equation: Y = CON	( X, P )	1 0 1	
Y_list	X_list	Parameters	Index
MTF126		-3.7879E+00	0
Node: TF127	Connectors: TB127	1B127	
FTB127	= MTF127	* F1B127	
E1B127	= MTF127	* ETB127	
Equation: Y = CON	( X, P )	1 0 1	
Y_list	X_list	Parameters	Index
MTF127		-3.9697E+00	0
Node: TF128	Connectors: TB128	1B128	
FTB128	= MTF128	* F1B128	
E1B128	= MTF128	* ETB128	
Equation: Y = CON	( X, P )	1 0 1	
Y_list	X_list	Parameters	Index
MTF128		1.3440E+00	0
Node: TF129	Connectors: TB129	1B129	
FTB129	= MTF129	* F1B129	
E1B129	= MTF129	* ETB129	
Equation: Y = CON	( X, P )	1 0 1	
Y_list	X_list	Parameters	Index
MTF129		-1.6953E+00	0
Node: TF130	Connectors: TB130	1B130	
FTB130	= MTF130	* F1B130	
E1B130	= MTF130	* ETB130	
Equation: Y = CON	( X, P )	1 0 1	
Y_list	X_list	Parameters	Index
MTF130		6.0412E+00	0
Node: IM101	Connectors: IB101		
Equation: Y = ATT	( X, P )	1 1 1	
Y_list	X_list	Parameters	Index
FIB101	PIB101	5.1961E+04	0
Node: IM102	Connectors: IB102		
Equation: Y = ATT	( X, P )	1 1 1	
Y_list	X_list	Parameters	Index
FIB102	PIB102	3.8072E+04	0
Node: IM103	Connectors: IB103		
Equation: Y = ATT	( X, P )	1 1 1	
Y_list	X_list	Parameters	Index
FIB103	PIB103	4.8831E+04	0

Table B-3 (cont'd).

Node: IM104	Connectors: IB104			
Equation:	Y = ATT ( X, P )	1 1 1		
Y_list	X_list	Parameters	Index	
FIB104	PIB104	6.4808E+03	0	
Node: IM105	Connectors: IB105			
Equation:	Y = ATT ( X, P )	1 1 1		
Y_list	X_list	Parameters	Index	
FIB105	PIB105	1.0284E+04	0	
Node: IM106	Connectors: IB106			
Equation:	Y = ATT ( X, P )	1 1 1		
Y_list	X_list	Parameters	Index	
FIB106	PIB106	1.3215E+05	0	
Node: CM101	Connectors: CB101			
Equation:	Y = GAIN ( X, P )	1 1 1		
Y_list	X_list	Parameters	Index	
ECB101	QCB101	9.2496E+07	0	
Node: CM102	Connectors: CB102			
Equation:	Y = GAIN ( X, P )	1 1 1		
Y_list	X_list	Parameters	Index	
ECB102	QCB102	1.7051E+07	0	
Node: CM103	Connectors: CB103			
Equation:	Y = GAIN ( X, P )	1 1 1		
Y_list	X_list	Parameters	Index	
ECB103	QCB103	6.1746E+06	0	
Node: CM104	Connectors: CB104			
Equation:	Y = GAIN ( X, P )	1 1 1		
Y_list	X_list	Parameters	Index	
ECB104	QCB104	1.6159E+05	0	
Node: CM105	Connectors: CB105			
Equation:	Y = GAIN ( X, P )	1 1 1		
Y_list	X_list	Parameters	Index	
ECB105	QCB105	3.2115E+04	0	
Node: CM106	Connectors: CB106			
Equation:	Y = GAIN ( X, P )	1 1 1		
Y_list	X_list	Parameters	Index	
ECB106	QCB106	1.0442E+04	0	
Node: ISPG	Connectors: VISPG			
Equation:	Y = ATT ( X, P )	1 1 1		
Y_list	X_list	Parameters	Index	
FVISPG	PVISPG	1.0000E+01	0	
Node: RSPG	Connectors: VSPG			
Equation:	Y = GAIN ( X, P )	1 1 1		
Y_list	X_list	Parameters	Index	
EVSPG	FVSPG	1.0000E+01	0	

Table B-3 (cont'd).

<b>Node:</b>	<b>CSPG</b>	<b>Connectors:</b>	<b>VSPG2</b>		
<b>Equation:</b>	<b>Y = GAIN</b>	<b>( X, P )</b>	<b>1 1 1</b>		
<b>Y_list</b>	<b>X_list</b>	<b>Parameters</b>	<b>Index</b>		
<b>ĒVSPG2</b>	<b>Q̄VSPG2</b>	<b>5.0000E+02</b>	<b>0</b>		

**END-FILE**



## APPENDIX C

### ORGANIZATION OF THE FORTRAN SOURCE CODE

The FORTRAN source code has been placed into three files. The first file, "MBMCK.FOR", contains file processing, flow control, and user dialog support subroutines. The second file, "MBMCKG.FOR", contains the subroutines which control the creation of the modal and multiport topologies. The third file, "MCKMTH.FOR", contains the subroutines which perform the mathematical manipulations necessary to construct the modal and multiport topologies. The subroutines contained within the three files are supported by a common-block structure contained in the files "MCKHDR.CBK" and "MCKSIZ.CBK". The source code developed for this study may be considered as a "stand alone" module. The module has a single entry point (subroutine MCKLOD) which is called from a single location within the ENPORT source code (subroutine GFGNOC). Further information regarding the organization of the source code is provided in Appendix C1 and Appendix C2. A complete listing of the FORTRAN source code is provided in Appendix C3.

## APPENDIX C1

### SUBPROGRAM CALLING TREE

Table C-1. The subprogram calling tree.

```
GFGNOC (ENPORT)
  MCKLOD
    MCKGFN
      MCKFRD
        MCKHDR
        MCKDMN
        MCKFRC
        MCKMAT
      MCKMCR
      MCKSHO
      MCKRST
      MCKBLD
        MCKMOD
        MCKEIG
          INVMAT
            LUDCMP
            LUBKSB
          MTXMUL
          BALAN2
          ELMHS2
          HQR3
          EVECTS
            MTXCML
            MTXADD
            DETCPY
            DETMTX
        MODDAT
          SETVAL
          MTXTRN
          MTXMUL
      MCKVIB
      INVMAT
        LUDCMP
        LUBKSB
      VIBDAT
        SETVAL
```

## APPENDIX C2

### SUBPROGRAM LIST

Table C-2. The subprogram list and associated source files.

BALAN2	(MCKMTH.FOR)
DETCPY	(MCKMTH.FOR)
DETMTH	(MCKMTH.FOR)
ELMHS2	(MCKMTH.FOR)
EQNINI	(MBMCKG.FOR)
EVECTS	(MCKMTH.FOR)
GETNOD	(MBMCKG.FOR)
HQR3	(MCKMTH.FOR)
INVMAT	(MCKMTH.FOR)
LUBKSB	(MCKMTH.FOR)
LUDCMP	(MCKMTH.FOR)
MCKBLD	(MBMCKG.FOR)
MCKDMN	(MBMCK.FOR)
MCKEIG	(MBMCKG.FOR)
MCKFRC	(MBMCK.FOR)
MCKFRD	(MBMCK.FOR)
MCKGFN	(MBMCK.FOR)
MCKHDR	(MBMCK.FOR)
MCKLOD	(MBMCK.FOR)
MCKMAT	(MBMCK.FOR)
MCKMCR	(MBMCK.FOR)
MCKMOD	(MBMCKG.FOR)
MCKRST	(MBMCK.FOR)
MCKSHO	(MBMCK.FOR)
MCKVIB	(MBMCKG.FOR)
MODDAT	(MBMCKG.FOR)
MTXADD	(MCKMTH.FOR)
MTXCML	(MCKMTH.FOR)
MTXMUL	(MCKMTH.FOR)
MTXTRN	(MCKMTH.FOR)
SETVAL	(MBMCKG.FOR)
VIBDAT	(MBMCKG.FOR)

## **APPENDIX C3**

### **FORTRAN SOURCE CODE LISTINGS**

**This appendix includes the FORTRAN source code for the software utility created for ENPORT.**

```
CCOMMONFILE:MCKSIZ----- Last Change: 09/19/91 RM
C
C---- MCK Header File
C
C---- Components
C      MCKMAX,   Maximum dimension of the MCK model.
C
C---- Declarations.
C
C      INTEGER   MCKMAX
C
C      PARAMETER (MCKMAX = 10)
C
CCOMMONEOF:MCKHDR -----*
```

```

CCOMMONFILE:MCKHDR----- Last Change: 09/09/91 RM
C
C---- MCK Header File
C
C---- Components
C      MCKNAM,      model_name
C      MCKFNM,      model_file_name
C      MCKTTL,      model_title
C      MCKDSC,      model_description.
C      MCKDIM,      Dimension of the MCK model.
C      MCKNLD,      Number of description lines.
C      MCKMCT,      Number of elements counted in the mass matrix.
C      MCKCCT,      Number of elements counted in the damping matrix.
C      MCKKCT,      Number of elements counted in the stiffness matrix.
C      NUMNOD,      Number of nodes required for the macro.
C      NUMCON,      Number of connectors required for the macro.
C      NUMMAC,      Number of macros added to the graph (for names only).
C      FRCTOK,      Token value for force (0 = ignore, 1 = include).
C      FRCVAL,      Force value at the node (constant) used if FRCTOK=1.
C      MASMAT,      Mass matrix
C      STFMAT,      Stiffness matrix
C      EIGMAT,      Matrix of eigenvalues for the system.
C      DMPMAT,      Damping matrix
C      MCKMAC,      Macro option - TRUE = modal, FALSE = vibrations.
C
C---- Include the size specifications.
C
C      INCLUDE 'MCKSIZ.CBK'
C
C---- Declarations
C      CHARACTER MCKNAM*8, MCKFNM*32, MCKTTL*72, MCKDSC(10)*72
C      INTEGER   MCKDIM, MCKNLD, MCKMCT, MCKCCT, MCKKCT
C      INTEGER   NUMNOD, NUMCON, NUMMAC, FRCTOK(MCKMAX)
C      REAL      MASMAT(MCKMAX,MCKMAX), STFMAT(MCKMAX,MCKMAX)
C      REAL      DMPMAT(MCKMAX,MCKMAX), EIGMAT(MCKMAX,MCKMAX)
C      REAL      FRCVAL(MCKMAX)
C      LOGICAL   MCKMAC
C
C      COMMON /MCKHD1/MCKNAM,MCKFNM,MCKTTL,MCKDSC
C      COMMON /MCKHD2/MCKDIM,MCKNLD,MCKMCT,MCKCCT,MCKKCT,NUMNOD,NUMCON
C      COMMON /MCKHD3/NUMMAC,FRCTOK
C      COMMON /MCKHD4/MASMAT,STFMAT,DMPMAT,EIGMAT,FRCVAL
C      COMMON /MCKHD5/MCKMAC
C
CCOMMONEOF:MCKHDR -----*
```

```
C FILE:MBMCK.FOR-----*
```

```
C
```

```
C---- MBMCK.FOR:   Model_build M,C,K file load/manipulation support.
```

```
C
```

```
C---- PURPOSE:      This file contains all the code required to load
```

```
C                   MCK matrix models from file.
```

```
C
```

```
C---- Contents:
```

```
C           MCKLOD,    Load M,C,K matrices from file.
```

```
C           MCKRST,    Restores the graph screen
```

```
C           MCKGFN,    Get file name from user by dialog.
```

```
C           MCKFRD,    Controls the MCK data file read.
```

```
C           MCKHDR,    Reads MCK header from file.
```

```
C           MCKDMN,    Reads MCK model dimension from file.
```

```
C           MCKFRC,    Reads MCK force info from file.
```

```
C           MCKMAT,    Reads MCK matrices from a file.
```

```
C           MCKMCR,    Allows the user to change macro types.
```

```
C           MCKSHO,    Determines number of nodes/connectors.
```

```
C
```

```
C---- Last Change: September 9, 1991. RM
```

```
C
```

```
CEOFH-----*
```

```
C
```

```
$MESSAGE:'MCKLOD'
```

```
CMCKLOD>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Last Change: 9/9/91 RM
```

```
C
```

```
        SUBROUTINE MCKLOD
```

```
C
```

```
C---- PURPOSE:   Loads a M,C,K set of matrices from file.
```

```
C
```

```
C---- Inputs:
```

```
C
```

```
C---- Outputs:
```

```
C
```

```
        CHARACTER  EVTP*4, FILNAM*12
```

```
        INTEGER    IR, IC, IPAG
```

```
        LOGICAL     TXTMOD
```

```
C
```

```
        EXTERNAL   MODSET, SPKEYL, MOVCUR, GETKEY, EVTYP, MCKRST
```

```
        EXTERNAL   MCKBLD
```

```
C
```

```
C***MCKLOD*****
```

```
C
```

```
C---- Set text mode
```

```
        IPAG= 0
```

```
        TXTMOD= .TRUE.
```

```
        CALL MODSET(TXTMOD)
```

```
        CALL SPKEYL
```

```
C
```

```
C---- Get the MCK data from file.
```

```
C
```

```
100 CONTINUE
```

```
        IR= 2
```

```
        IC= 6
```

```
        FILNAM = '*.MCK'
```

```
        CALL MCKGFN(IR,IC,FILNAM,EVTP)
```

```
C
```

```
C---- Get the macro option (MODAL or VIBRATIONS).
```

```
C
```

```
        CALL MCKMCR
```

```
C
```

```
C---- Calculate and display required nodes and connectors for macro.
```

```
C
```

```
        CALL MCKSHO
```







[illegible]

```

      MESSAG= ' Processing file command: '
      CALL WRTSTG(0, LN, 7, ICLRM, MESSAG, 55)
      ENDIF

C
C----- Interpret the string as a command
20  CONTINUE
      IF (NXTCOM.EQ.' HEAD') THEN
        NXTCML=' HEADING'
      ELSEIF (NXTCOM.EQ.' DIME') THEN
        NXTCML=' DIMENSION'
      ELSEIF (NXTCOM.EQ.' FORC') THEN
        NXTCML=' FORCE INFORMATION'
      ELSEIF (NXTCOM.EQ.' MASS') THEN
        NXTCML=' MASS MATRIX'
      ELSEIF (NXTCOM.EQ.' STIF') THEN
        NXTCML=' STIFFNESS MATRIX'
      ELSEIF (NXTCOM.EQ.' DAMP') THEN
        NXTCML=' DAMPING MATRIX'
      ELSEIF (NXTCOM.EQ.' END-') THEN
        NXTCML=' END-FILE'
      ELSE
        MESSAG= ' *** Bad command in file: '//NXTCOM
        GOTO 900
      ENDIF

C----- Inform user of file read status
      IF (LN.LE.21) THEN
        LN= LN+1
        MESSAG= NXTCML
        CALL WRTSTG(0, LN, 7, ICLRM, MESSAG, 55)
      ENDIF

C
C----- All routines below return MESSAG filled in if OKAY=.FALSE.
C
      IF (NXTCOM.EQ.' HEAD') THEN
C----- Read the MCK file header.
        CALL MCKHDR( NXTCOM )
      ELSEIF (NXTCOM.EQ.' DIME') THEN
C----- Read the MCK matrix dimensions from file and post results.
        CALL MCKDMN( NXTCOM )
        WRITE(STRING,100) MCKDIM
        CALL WRTSTG(0, LN, 25, ICLRM, STRING, 4)
      ELSEIF (NXTCOM.EQ.' FORC') THEN
C----- Read the force info from file and post the results.
        CALL MCKFRC( NXTCOM )
        WRITE(STRING,100) FRCTOK(1)
        CALL WRTSTG(0, LN, 25, ICLRM, STRING, 4)
        WRITE(STRING,110) FRCVAL(1)
        CALL WRTSTG(0, LN, 30, ICLRM, STRING, 13)
      ELSEIF (NXTCOM.EQ.' MASS') THEN
C----- Read the MCK mass matrix from file and post results.
        CALL MCKMAT( 'MASS', NXTCOM )
        WRITE(STRING,100) MCKMCT
        CALL WRTSTG(0, LN, 25, ICLRM, STRING, 4)
        WRITE(STRING,110) MASMAT(1,1)
        CALL WRTSTG(0, LN, 30, ICLRM, STRING, 13)
        WRITE(STRING,110) MASMAT(MCKDIM, MCKDIM)
        CALL WRTSTG(0, LN, 45, ICLRM, STRING, 13)
      ELSEIF (NXTCOM.EQ.' STIF') THEN
C----- Read the MCK stiffness matrix from file and post results.
        CALL MCKMAT( 'STIF', NXTCOM )
        WRITE(STRING,100) MCKKCT
        CALL WRTSTG(0, LN, 25, ICLRM, STRING, 4)
        WRITE(STRING,110) STFMAT(1,1)
        CALL WRTSTG(0, LN, 30, ICLRM, STRING, 13)

```



```

C
    CALL NEWLF(STRING,LCS)
    IF(LCS.GE.5) GOTO 900
    NXTCOM = STRING(1:5)
    IF (NXTCOM.EQ.' FI') THEN
C----- Get the file name from the file.
    CALL NEWLF(STRING,LCS)
    IF (LCS.GE.8) GOTO 900
    MCKFNM = STRING(7:18)
    ELSE
    GOTO 900
    ENDIF

C
C----- Get the MCK macro name.
C
    CALL NEWLF(STRING,LCS)
    IF(LCS.GE.5) GOTO 900
    NXTCOM = STRING(1:5)
    IF (NXTCOM.EQ.' NA') THEN
C----- Get the macro name from the file.
    CALL NEWLF(STRING,LCS)
    IF (LCS.GE.8) GOTO 900
    MCKNAM = STRING(7:14)
    ELSE
    GOTO 900
    ENDIF

C
C----- Get the title.
C
    CALL NEWLF(STRING,LCS)
    IF(LCS.GE.5) GOTO 900
    NXTCOM = STRING(1:5)
    IF (NXTCOM.EQ.' TI') THEN
C----- Get the mck title from the file.
    CALL NEWLF(STRING,LCS)
    IF (LCS.LT.7) GOTO 900
    MCKTTL = ' '
    LEN = NCHARS(STRING)
    IF (LEN.GE.7) MCKTTL = STRING(7:LEN)
    ELSE
    GOTO 900
    ENDIF

C
C----- Get the description.
C
    MCKNLD = 0
    CALL NEWLF(STRING,LCS)
    IF(LCS.GE.5) GOTO 900
    NXTCOM = STRING(1:5)
    IF (NXTCOM.EQ.' DE') THEN
C----- Get the mck description from the file.
50    CALL NEWLF(STRING,LCS)
    IF (LCS.GE.7) THEN
        MCKNLD = MCKNLD+1
        MCKDSC(MCKNLD)=' '
        LEN = NCHARS(STRING)
        IF (LEN.GE.7) MCKDSC(MCKNLD) = STRING(7:LEN)
        GOTO 50
    ENDIF
    NXTCOM = STRING(1:5)
    ENDIF

C
C----- All done.
C

```



```
C MESSAGE: 'MCKFRC'
CMCKFRC>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Last Change: 10/21/91 RM
C
      SUBROUTINE MCKFRC( NXTCOM )
C
C----- PURPOSE:    Reads force information from a file.
C
C----- Inputs:     None.
C
C----- Outputs:    OKAY,       logical flag for operation status.
                     MESSAGE,   status message.
                     NXTCOM,    next command in file.
C
      INCLUDE 'COMMBK.CBK'
      INCLUDE 'MCKHDR.CBK'
C
      CHARACTER STRING*80, NXTCOM*5, SFORCE*13, SFRCTK*1
      INTEGER LCS, IP, ICOUNT
C
      EXTERNAL NEWLF
C
C***MCKFRC*****
C
      OKAY=.FALSE.
C
C----- Read the { start specifier.
C
      CALL NEWLF(String,LCS)
      IF (String(3:3).NE.'{') GOTO 900
C
C----- Read the matrix data.
C
      ICOUNT = 0
100 CONTINUE
      CALL NEWLF(String,LCS)
      IF (String(3:3).EQ.')') THEN
        GOTO 210
      ELSEIF (LCS.LE.3) THEN
        MESSAG = String
        GOTO 910
      ENDIF
C----- Parse the string into numbers and tokens.
      IP = 7
      SFRCTK = String(IP:IP)
      IP = 18
      SFORCE = String(IP:IP+12)
      ICOUNT = ICOUNT + 1
C----- Store the number in the proper place.
      READ(SFRCTK,'(I4)') FRCTOK(ICOUNT)
      READ(SFORCE,'(1P,E13.4)') FRCVAL(ICOUNT)
      GOTO 100
C
C----- Advance to the next command.
C
210 CONTINUE
      CALL NEWLF(String,LCS)
      IF(LCS.GE.3) GOTO 900
      NXTCOM = String(1:5)
C
C----- All done.
C
      OKAY=.TRUE.
      RETURN
C
```

[illegible]





```

        LOGICAL    ANS1
C
        EXTERNAL  MODSET, WRTSTG, MOVCUR, GETCUR, EVTTYP, SPKEYL
        EXTERNAL  PCWIND, NEWPLT
        INTRINSIC MAX, MIN
C
C***MCKMCR*****
C
C----- Set up the screen label fields.
C
        IR  = 16
        ICS = 6
        STRING= ' Macro type?          Modal      Vibrations'
        CALL WRTSTG(IPAG,IR,ICS,ICLRL,STRING,42)
C
C----- Set up the data fields with the default values.
C
C        Fields 1 and 2: Line 20, cols ICS+18 and ICS+27
C
20      CONTINUE
        IF1= ICS+18
        IF2= ICS+27
        STRING= 'X'
        CALL WRTSTG(IPAG,IR,IF1,ICLRD,STRING,1)
        STRING= ' '
        CALL WRTSTG(IPAG,IR,IF2,ICLRD,STRING,1)
C
C----- Position the cursor initially.
C
        IC= IF1
C
C----- Read the selection here
100     CALL MOVCUR(IPAG,IR,IC)
        CALL GETKEY(I1,I2)
        EVTP= EVTTYP(I1,I2)
C
C----- Process the selection
        IF (EVTP.EQ.'CANC') THEN
            GOTO 800
        ELSEIF (EVTP.EQ.'DONE') THEN
            CALL MOVCUR(IPAG,IR,IF1)
            CALL GETCAT(IPAG,I,IATT)
            ANS1= CHAR(I).EQ.'X'
            MCKMAC= ANS1
            GOTO 800
        ELSEIF (EVTP.EQ.'DATA'.OR. EVTP.EQ.'CRET') THEN
C----- Blank the unmarked field and mark the selected field
            ICB= IF1+IF2 -IC
            STRING= ' '
            CALL WRTSTG(IPAG,IR,ICB,ICLRD,STRING,1)
            STRING= 'X'
            CALL WRTSTG(IPAG,IR,IC,ICLRD,STRING,1)
            ELSEIF (EVTP.EQ.'MOVE') THEN
                IF (I2.EQ.75) THEN                ! Left
                    IC= IF1
                ELSEIF (I2.EQ.77) THEN            ! Right
                    IC= IF2
                ENDIF
            ENDIF
            GOTO 100
C
C----- Return section
800     CONTINUE
C

```

[illegible]

[illegible]

```
CFILE:MBMCKG.FOR-----*
```

```
C
```

```
C---- MBMCK.FOR:   M,C,K model builder support.
```

```
C
```

```
C---- PURPOSE:     This file contains all the code required to build
```

```
C                  MCK modal and vibrations models automatically.
```

```
C
```

```
C---- Contents:
```

```
C          MCKBLD,    Controls automatic building process.
```

```
C          MCKMOD,    Controls building of MODAL macro.
```

```
C          MCKVIB,    Controls building of VIBRATIONS macro.
```

```
C          GETNOD,    Finds node index from a grid location.
```

```
C          EQNINI,    Init. equations for automatic assignment.
```

```
C          VIBDAT,    Guides change in equations for nodes.
```

```
C          MCKEIG,    Controls eigen. eval. of M,K matrices.
```

```
C          MODDAT,    Guides change in equations for modal mac.
```

```
C          SETVAL,    Guides change in equations for nodes.
```

```
C
```

```
C---- Last Change: September 11, 1991. RM
```

```
C
```

```
CEOFH-----*
```

```
C
```

```
$MESSAGE:'MCKBLD'
```

```
CMCKBLD>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Last Change: 9/11/91 RM
```

```
C
```

```
      SUBROUTINE MCKBLD
```

```
C
```

```
C---- PURPOSE:     Controls the automatic building process.
```

```
C
```

```
C---- Inputs:
```

```
C
```

```
C---- Outputs:
```

```
C
```

```
      INCLUDE 'SIZEB.CBK'
```

```
      INCLUDE 'SIZESL.CBK'
```

```
      INCLUDE 'GREDBK.CBK'
```

```
      INCLUDE 'VIEWBK.CBK'
```

```
      INCLUDE 'COMMBK.CBK'
```

```
      INCLUDE 'GFSTBK.CBK'
```

```
      INCLUDE 'MCKHDR.CBK'
```

```
C
```

```
      CHARACTER STRING*80, EVTP*4
```

```
      INTEGER INOD, I, J, INDX(MCKMAX)
```

```
      REAL X, Y, ANS(MCKMAX,MCKMAX)
```

```
C
```

```
      EXTERNAL WRTSTG, LNMSGD, NODPIK
```

```
C
```

```
C***MCKBLD*****
```

```
C
```

```
C---- Clear the text field in the graph window.
```

```
C
```

```
      STRING=' '
```

```
      CALL WRTSTG(0,23,1,ICLRL,String,80)
```

```
C
```

```
C---- Check that there is enough room in the data base to hold the
```

```
macro.
```

```
C
```

```
      IF ( (INELS+NUMNOD).GE.MNEL) THEN
```

```
        MESSAGE=' EXCESSIVE NUMBER OF NODES...CANNOT BUILD MACRO!'
```

```
        CALL LNMSGD(24)
```

```
        GOTO 800
```

```
      ELSEIF ( (INBDS+NUMCON).GE.MNBD) THEN
```

```
        MESSAGE=' EXCESSIVE NUMBER OF CONNECTORS...CANNOT BUILD MACRO!'
```

```
        CALL LNMSGD(24)
```

```

      GOTO 800
    ENDIF
C
C----- Get the position for adding the macro.
C
      X=XG
      Y=YG
40    CALL NODPIK(24,'ADD',EVTP,X,Y,INOD)
      IF (EVTP.EQ.'DATA') THEN
        IF (INOD.NE.0) THEN
          CALL LNMSGD(24)
          GOTO 40
        ENDIF
      ELSEIF (EVTP.EQ.'CANC') THEN
        GOTO 800
      ENDIF
C
C----- Build the proper macro using X,Y as upper left corner.
C
      IF (MCKMAC) THEN
C
C----- Build the graph.
C
        CALL MCKMOD(X,Y)
C
C----- Perform the eigen analysis.
C
        CALL MCKEIG
C
C----- Set the flag to say, "There is now a system graph"
        E7SFLG(1)=.TRUE.
        E7CFLG(1)=.TRUE.
C----- Assign causality silently (sets E7SFLG(2), E7CFLG(2)).
        CALL ASSCAU(.TRUE.,.FALSE.)
C----- Initialize the equations for automatic assignment.
        CALL EQNINI
        E7CFLG(4)=.FALSE.
        E7CFLG(5)=.FALSE.
C
C----- Set the proper values.
C
        CALL MODDAT
C
        ELSE
C----- Set the flag to say, "There is now a system graph"
        E7SFLG(1)=.TRUE.
        E7CFLG(1)=.TRUE.
C----- Assign causality silently (sets E7SFLG(2), E7CFLG(2)).
        CCCCC CALL SSAVE
        CALL ASSCAU(.TRUE.,.FALSE.)
C----- Initialize the equations for automatic assignment.
        CALL EQNINI
        E7CFLG(4)=.FALSE.
        E7CFLG(5)=.FALSE.
C
C----- Get the inverse of the mass matrix.
C
        CALL INVMAT(MASMAT,MCKDIM,MCKMAX,INDX,ANS)
        DO 510 I=1,MCKDIM
          DO 500 J=1,MCKDIM
            MASMAT(I,J)=ANS(I,J)
500          CONTINUE
510          CONTINUE

```



```

      IF(FRCTOK(I).EQ.1) THEN
        WRITE(ST2,700) I
        IF (ST2(1:1).EQ.' ') ST2(1:1)='0'
        NAME = 'SM'//ST1(1:1)//ST2(1:2)
        XPOS = X+(MCKDIM+1)*(I-1)*GRDIST
        YPOS = Y
        CALL DRNOD(NAME,XPOS,YPOS,TYPE)
        CALL ADDNOD(NAME,TYPE,XPOS,YPOS)
      ENDIF
150    CONTINUE
C
C----- Build the 0 junctions (input ports).
C
      TYPE = 'MOG '
      DO 160 I=1,MCKDIM
        IF(FRCTOK(I).EQ.1) THEN
          WRITE(ST2,700) I
          IF (ST2(1:1).EQ.' ') ST2(1:1)='0'
          NAME = 'OM'//ST1(1:1)//ST2(1:2)
          XPOS = X+(MCKDIM+1)*(I-1)*GRDIST
          YPOS = Y-GRDIST
          CALL DRNOD(NAME,XPOS,YPOS,TYPE)
          CALL ADDNOD(NAME,TYPE,XPOS,YPOS)
        ENDIF
160    CONTINUE
C
C----- Build the transformers.
C
      TYPE = 'MTG '
      DO 180 I=1,MCKDIM
        IF(FRCTOK(I).EQ.1) THEN
          DO 170 J=1,MCKDIM
            WRITE(ST2,700) J + ( (I-1) * MCKDIM )
            IF (ST2(1:1).EQ.' ') ST2(1:1)='0'
            NAME = 'TF'//ST1(1:1)//ST2(1:2)
            XPOS = X+((J-1)+( (I-1)*(MCKDIM+1) ))*GRDIST
            YPOS = Y-2*GRDIST
            CALL DRNOD(NAME,XPOS,YPOS,TYPE)
            CALL ADDNOD(NAME,TYPE,XPOS,YPOS)
          170    CONTINUE
        ENDIF
180    CONTINUE
C
C----- Build the 1 junctions.
C
      TYPE = 'M1G '
      DO 190 I=1,MCKDIM
        WRITE(ST2,700) I
        IF (ST2(1:1).EQ.' ') ST2(1:1)='0'
        NAME = '1M'//ST1(1:1)//ST2(1:2)
        XPOS = X+(MCKDIM+1)*(I-1)*GRDIST
        YPOS = Y-4*GRDIST
        CALL DRNOD(NAME,XPOS,YPOS,TYPE)
        CALL ADDNOD(NAME,TYPE,XPOS,YPOS)
190    CONTINUE
C
C----- Build the I junctions.
C
      TYPE = 'MIG '
      DO 200 I=1,MCKDIM
        WRITE(ST2,700) I
        IF (ST2(1:1).EQ.' ') ST2(1:1)='0'
        NAME = 'IM'//ST1(1:1)//ST2(1:2)
        XPOS = X+(MCKDIM+1)*(I-1)*GRDIST

```



```

      YPOS = Y-5*GRDIST
      CALL DRNOD(NAME,XPOS,YPOS,TYPE)
      CALL ADDNOD(NAME,TYPE,XPOS,YPOS)
200  CONTINUE
C
C---- Build the C junctions.
C
      TYPE = 'MCG '
      DO 210 I=1,MCKDIM
      WRITE(ST2,700) I
      IF (ST2(1:1).EQ.' ') ST2(1:1)='0'
      NAME = 'CM'//ST1(1:1)//ST2(1:2)
      XPOS = X+((MCKDIM+1)*(I-1)*GRDIST) + GRDIST
      YPOS = Y-5*GRDIST
      CALL DRNOD(NAME,XPOS,YPOS,TYPE)
      CALL ADDNOD(NAME,TYPE,XPOS,YPOS)
210  CONTINUE
C
C---- Build the R junctions.
C
      IF(MCKCCT.NE.0) THEN
      TYPE = 'MRG '
      DO 220 I=1,MCKDIM
      WRITE(ST2,700) I
      IF (ST2(1:1).EQ.' ') ST2(1:1)='0'
      NAME = 'RM'//ST1(1:1)//ST2(1:2)
      XPOS = X+((MCKDIM+1)*(I-1)*GRDIST) + GRDIST
      YPOS = Y-4*GRDIST
      CALL DRNOD(NAME,XPOS,YPOS,TYPE)
      CALL ADDNOD(NAME,TYPE,XPOS,YPOS)
220  CONTINUE
      ENDIF
C
C---- Add the connectors from effort sources to 0-junctions.
C
      INPTCT = 0
      TYPE = 'BG '
      DO 230 I=1,MCKDIM
      IF(FRCTOK(I).EQ.1) THEN
      WRITE(ST2,700) I
      IF (ST2(1:1).EQ.' ') ST2(1:1)='0'
      NAME = 'SB'//ST1(1:1)//ST2(1:2)
      XPOS = X+(MCKDIM+1)*(I-1)*GRDIST
      YPOS = Y
      CALL GETNOD(XPOS,YPOS,ISTART)
      XPOS = X+(MCKDIM+1)*(I-1)*GRDIST
      YPOS = Y-GRDIST
      CALL GETNOD(XPOS,YPOS,IEND)
      CALL ADDBND(NAME,ISTART,IEND,TYPE,POINTS,INPTCT)
      CALL DRBND(NAME,ISTART,IEND,TYPE(1:1),1)
      ENDIF
230  CONTINUE
C
C---- Add the connectors from 0-junctions to transformers.
C
      DO 250 I=1,MCKDIM
      IF(FRCTOK(I).EQ.1) THEN
      XPOS = X+(MCKDIM+1)*(I-1)*GRDIST
      YPOS = Y-GRDIST
      CALL GETNOD(XPOS,YPOS,ISTART)
      DO 240 J=1,MCKDIM
      WRITE(ST2,700) J + ( (I-1) * MCKDIM )
      IF (ST2(1:1).EQ.' ') ST2(1:1)='0'
      NAME = 'TB'//ST1(1:1)//ST2(1:2)

```

```

      XPOS = X+((J-1)+( I-1)*(MCKDIM+1) ))*GRDIST
      YPOS = Y-2*GRDIST
      CALL GETNOD(XPOS,YPOS,IEND)
      CALL ADDBND(NAME,ISTART,IEND,TYPE,POINTS,INPTCT)
      CALL DRBND(NAME,ISTART,IEND,TYPE(1:1),1)
240    CONTINUE
      ENDIF
250    CONTINUE
C
C----- Add the connectors from transformers to l-junctions.
C
      DO 270 I=1,MCKDIM
      IF(FRCTOK(I).EQ.1) THEN
        DO 260 J=1,MCKDIM
          WRITE(ST2,700) J + ( (I-1) * MCKDIM )
          IF (ST2(1:1).EQ.' ') ST2(1:1)='0'
          NAME = '1B'//ST1(1:1)//ST2(1:2)
          XPOS = X+(MCKDIM+1)*(J-1)*GRDIST
          YPOS = Y-4*GRDIST
          CALL GETNOD(XPOS,YPOS,IEND)
          XPOS = X+((J-1)+( I-1)*(MCKDIM+1) ))*GRDIST
          YPOS = Y-2*GRDIST
          CALL GETNOD(XPOS,YPOS,ISTART)
          CALL ADDBND(NAME,ISTART,IEND,TYPE,POINTS,INPTCT)
          CALL DRBND(NAME,ISTART,IEND,TYPE(1:1),1)
260        CONTINUE
          ENDIF
270      CONTINUE
C
C----- Add the connectors from l-junctions to I nodes.
C
      DO 280 I=1,MCKDIM
      WRITE(ST2,700) I
      IF (ST2(1:1).EQ.' ') ST2(1:1)='0'
      NAME = 'IB'//ST1(1:1)//ST2(1:2)
      XPOS = X+(MCKDIM+1)*(I-1)*GRDIST
      YPOS = Y-4*GRDIST
      CALL GETNOD(XPOS,YPOS,ISTART)
      XPOS = X+(MCKDIM+1)*(I-1)*GRDIST
      YPOS = Y-5*GRDIST
      CALL GETNOD(XPOS,YPOS,IEND)
      CALL ADDBND(NAME,ISTART,IEND,TYPE,POINTS,INPTCT)
      CALL DRBND(NAME,ISTART,IEND,TYPE(1:1),1)
280    CONTINUE
C
C----- Add the connectors from l-junctions to C nodes.
C
      DO 290 I=1,MCKDIM
      WRITE(ST2,700) I
      IF (ST2(1:1).EQ.' ') ST2(1:1)='0'
      NAME = 'CB'//ST1(1:1)//ST2(1:2)
      XPOS = X+(MCKDIM+1)*(I-1)*GRDIST
      YPOS = Y-4*GRDIST
      CALL GETNOD(XPOS,YPOS,ISTART)
      XPOS = X+(MCKDIM+1)*(I-1)*GRDIST + GRDIST
      YPOS = Y-5*GRDIST
      CALL GETNOD(XPOS,YPOS,IEND)
      CALL ADDBND(NAME,ISTART,IEND,TYPE,POINTS,INPTCT)
      CALL DRBND(NAME,ISTART,IEND,TYPE(1:1),1)
290    CONTINUE
C
C----- Add the connectors from l-junctions to R nodes.
C
      IF(MCKCCT.NE.0) THEN

```



```

C---- Build the name constant string
C
    WRITE(ST1,690) NUMMAC
C
C---- Build the source nodes.
C
    TYPE = 'MEG '
    DO 150 I=1,MCKDIM
    IF(FRCTOK(I).EQ.1) THEN
        WRITE(ST2,700) I
        IF (ST2(1:1).EQ.' ') ST2(1:1)='0'
        NAME = 'SV'//ST1(1:1)//ST2(1:2)
        CALL DRNOD(NAME,X+(I-1)*GRDIST,Y,TYPE)
        CALL ADDNOD(NAME,TYPE,X+(I-1)*GRDIST,Y)
    ENDIF
150 CONTINUE
C
C---- Build the 1 junctions (input ports).
C
    TYPE = 'M1G '
    DO 160 I=1,MCKDIM
    WRITE(ST2,700) I
    IF (ST2(1:1).EQ.' ') ST2(1:1)='0'
    NAME = '1V'//ST1(1:1)//ST2(1:2)
    CALL DRNOD(NAME,X+(I-1)*GRDIST,Y-GRDIST,TYPE)
    CALL ADDNOD(NAME,TYPE,X+(I-1)*GRDIST,Y-GRDIST)
160 CONTINUE
C
C---- Build the "I" multi-port.
C
    NAME = 'I'//ST1(1:1)
    TYPE = 'MIG '
    CALL DRNOD(NAME,X,Y-3*GRDIST,TYPE)
    CALL ADDNOD(NAME,TYPE,X,Y-3*GRDIST)
C
C---- Build the "C" multi-port.
C
    NAME = 'C'//ST1(1:1)
    TYPE = 'MCG '
    CALL DRNOD(NAME,X+2*GRDIST,Y-3*GRDIST,TYPE)
    CALL ADDNOD(NAME,TYPE,X+2*GRDIST,Y-3*GRDIST)
C
C---- Build the "R" multi-port.
C
    IF (MCKCCT.NE.0) THEN
        NAME = 'R'//ST1(1:2)
        TYPE = 'MRG '
        CALL DRNOD(NAME,X+4*GRDIST,Y-3*GRDIST,TYPE)
        CALL ADDNOD(NAME,TYPE,X+4*GRDIST,Y-3*GRDIST)
    ENDIF
C
C---- Add the connectors from effort sources to 1-junctions.
C
    INPTCT = 0
    TYPE = 'BG '
    DO 170 I=1,MCKDIM
    IF(FRCTOK(I).EQ.1) THEN
        WRITE(ST2,700) I
        IF (ST2(1:1).EQ.' ') ST2(1:1)='0'
        NAME = 'SB'//ST1(1:1)//ST2(1:2)
        CALL GETNOD(X+(I-1)*GRDIST,Y,ISTART)
        CALL GETNOD(X+(I-1)*GRDIST,Y-GRDIST,IEND)
        CALL ADDBND(NAME,ISTART,IEND,TYPE,POINTS,INPTCT)
        CALL DRBND(NAME,ISTART,IEND,TYPE(1:1),1)
    ENDIF

```



[illegible]

[illegible]

```

      INCLUDE 'VIEWBK.CBK'
      INCLUDE 'EQVUBK.CBK'
      INCLUDE 'SIZEMB.CBK'
      INCLUDE 'SIZESL.CBK'
      INCLUDE 'GREDBK.CBK'
      INCLUDE 'MCKHDR.CBK'
C
      CHARACTER MATRX*4, ELSEL*8, ST1*10, ST2*10
      CHARACTER NVOTL(MXVOTQ)*12, NVINL(MXVINQ)*12, EVTTYP*4
      CHARACTER ST11*1, ST22*2, NAME*8
      INTEGER    NODE, IQ, IQ1, IQ2, I, NUMOUT, NUMIN, J
      INTEGER    NPR, FNC2I, NCHARS, IDX
      REAL       VAL
C
      EXTERNAL  LNMSGD, MOVCUR, WRTCAT, WRTSTG, CLEARS, DISPAG, SPKEYL
      EXTERNAL  LSTLST, EQNLST, TRFQDT, GETSTR, GETINT, TBLDSP, TB2DSP
      EXTERNAL  FNDISP, EDT1LQ, FTPLST, FNC2I, MTXDSP, PLYDSP
      EXTERNAL  NCHARS, GETRL, GETKEY, EVTTYP, SETVAL
      INTRINSIC MAX, MIN, INT, REAL
C
C***VIBDAT*****
C
C----- Get the node name.
C
      WRITE(ST1,10) NUMMAC
10    FORMAT(I1)
      IF (MATRX.EQ.'MASS') THEN
      ELSEL = 'I'//ST1(1:1)
      ELSEIF (MATRX.EQ.'STIF') THEN
      ELSEL = 'C'//ST1(1:1)
      ELSEIF (MATRX.EQ.'DAMP') THEN
      ELSEL = 'R'//ST1(1:1)
      ELSEIF (MATRX.EQ.'SRCE') THEN
      WRITE(ST11,690) NUMMAC
      DO 15 I=1,MCKDIM
        IF(FRCTOK(I).EQ.1) THEN
          WRITE(ST22,700) I
          IF (ST22(1:1).EQ.' ') ST22(1:1)='0'
          NAME = 'SV'//ST11(1:1)//ST22(1:2)
          VAL = FRCVAL(I)
          CALL SETVAL('CON',NAME,VAL)
        ENDIF
15      CONTINUE
      GOTO 800
      ENDIF
C
C----- Get the node index.
C
      DO 20 NODE= 1,INELS
20    IF (ELNAM(NODE).EQ.ELSEL) GOTO 30
      NODE=INELS
30    CONTINUE
C
C----- Check on validity of node request
C
      CALL EQNLST(NODE,'U',IQ1,IQ2,NUMOUT,NVOTL,NUMIN,NVINL)
      IF (.NOT.OKAY) THEN
      CALL LNMSGD(24)
      GOTO 800
      ENDIF
      DO 40 I= 1,NUMOUT
40    YL(I)=NVOTL(I)
      CONTINUE
C

```



```

C----- Do each eqn of this node -----
C
      IQ= IQ1
200  CONTINUE
C
C----- Fill in the equation_view data base from the model db.
C
      CALL TRFQDT('GET',IQ)
C
C----- Get the (new) function type.
C
300  CONTINUE
      FNAM= 'MATRIX'
      IF (FNAM.EQ.'MATRIX'.AND.NUMOUT.EQ.1) THEN
        MESSAG= ' *** Use SUM, not MATRIX, for single outputs.'
        CALL LNMSGD(24)
        GOTO 800
      ENDIF
C
C----- (Re-)set the number of outputs
C
      NYL= NUMOUT
C
      DO 310 I= 1,NUMOUT
        WRITE(ST2,315) I
315  FORMAT(I2)
        IF (ST2(1:1).EQ.' ') ST2(1:1)='0'
        IF (MATRX.EQ.'MASS') THEN
          XL(I) = 'PIB'//ST1(1:1)//ST2(1:2)
        ELSEIF (MATRX.EQ.'STIF') THEN
          XL(I) = 'QCB'//ST1(1:1)//ST2(1:2)
        ELSEIF (MATRX.EQ.'DAMP') THEN
          XL(I) = 'FRB'//ST1(1:1)//ST2(1:2)
        ENDIF
310  CONTINUE
C
C----- Set the number of inputs.
C
      NXL= MCKDIM
C
C----- Set the number of parameters.
C
      NPR= NYL*NXL
      NPL=NPR
C
C----- Set parameter values.
C
      DO 332 I = 1,NYL
        DO 331 J=1,NXL
          IDX = J + (MCKDIM*(I-1))
          IF (MATRX.EQ.'MASS') THEN
            WRITE(PL(IDX),333) MASMAT(I,J)
          ELSEIF (MATRX.EQ.'STIF') THEN
            WRITE(PL(IDX),333) STFMAT(I,J)
          ELSEIF (MATRX.EQ.'DAMP') THEN
            IF (MCKCCT.EQ.0) THEN
              PL(IDX) = ' 0.0000E+00'
            ELSE
              WRITE(PL(IDX),333) DMPMAT(I,J)
            ENDIF
          ENDIF
331  CONTINUE
332  CONTINUE
333  FORMAT(1P,E12.4)

```

[illegible]

[illegible]

```

C---- INPUTS:
C
C---- OUTPUTS:  none
C
      INCLUDE 'COMMBK.CBK'
      INCLUDE 'VIEWBK.CBK'
      INCLUDE 'EQVUBK.CBK'
      INCLUDE 'SIZEBK.CBK'
      INCLUDE 'SIZESL.CBK'
      INCLUDE 'GREDBK.CBK'
      INCLUDE 'MCKHDR.CBK'
C
      CHARACTER  ST1*1, ST2*2, NAME*8
      INTEGER    I, J
      REAL       EIGTRN(MCKMAX,MCKMAX), TEMP(MCKMAX,MCKMAX), VAL
C
      EXTERNAL   MTXTRN, SETVAL
C
C***MODDAT*****
C
C---- Build the name constant string
C
      WRITE(ST1,690) NUMMAC
C
C---- Set the source node values.
C
      DO 150 I=1,MCKDIM
      IF(FRCTOK(I).EQ.1) THEN
        WRITE(ST2,700) I
        IF (ST2(1:1).EQ.' ') ST2(1:1)='0'
        NAME = 'SM'//ST1(1:1)//ST2(1:2)
        VAL = FRCVAL(I)
        CALL SETVAL('CON      ',NAME,VAL)
      ENDIF
150    CONTINUE
C
C---- Set the transformer node values.
C
      DO 180 I=1,MCKDIM
      IF(FRCTOK(I).EQ.1) THEN
        DO 170 J=1,MCKDIM
          WRITE(ST2,700) J + ( (I-1) * MCKDIM )
          IF (ST2(1:1).EQ.' ') ST2(1:1)='0'
          NAME = 'TF'//ST1(1:1)//ST2(1:2)
          VAL = EIGMAT(I,J)
          CALL SETVAL('CON      ',NAME,VAL)
170        CONTINUE
      ENDIF
180    CONTINUE
C
C---- Set the inertia elements (get R'*M*R).
C
      CALL MTXTRN(EIGMAT,MCKDIM,MCKMAX,EIGTRN)
      CALL MTXMUL(EIGTRN,MCKDIM,MCKMAX,MASMAT,MCKDIM,MCKMAX,TEMP)
      CALL MTXMUL(TEMP,MCKDIM,MCKMAX,EIGMAT,MCKDIM,MCKMAX,EIGTRN)
C
      DO 200 I=1,MCKDIM
      WRITE(ST2,700) I
      IF (ST2(1:1).EQ.' ') ST2(1:1)='0'
      NAME = 'IM'//ST1(1:1)//ST2(1:2)
      VAL = EIGTRN(I,I)
      CALL SETVAL('ATT      ',NAME,VAL)
200    CONTINUE

```

C

```

      CHARACTER NVOTL(MXVOTQ)*12, NVINL(MXVINQ)*12, EVTTYP*4
      INTEGER  NODE, IQ, IQ1, IQ2, I, NUMOUT, NUMIN, NIV
      INTEGER  NPR, FNC2I, NCHARS, NYLOLD, NXLOLD, NPLOLD
      LOGICAL  SGLOLD, NEWFCN

C
      EXTERNAL LNMSGD, MOVCUR, WRTCAT, WRTSTG, CLEARS, DISPAG, SPKEYL
      EXTERNAL LSTLST, EQNLST, TRFQDT, GETSTR, GETINT, TBLDSP, TB2DSP
      EXTERNAL FNDISP, EDT1LQ, FTPLST, FNC2I, MTXDSP, PLYDSP
      EXTERNAL NCHARS, GETRL, GETKEY, EVTTYP
      INTRINSIC MAX, MIN, INT, REAL

C
C***SETVAL*****
C
C---- Get the node index.
C
      DO 20 NODE= 1,INELS
20    IF (ELNAM(NODE).EQ.ELSEL) GOTO 30
      NODE=INELS
30    CONTINUE
C
C---- Check on validity of node request.
C
      CALL EQNLST(NODE,'U',IQ1,IQ2,NUMOUT,NVOTL,NUMIN,NVINL)
      IF (.NOT.OKAY) THEN
      CALL LNMSGD(24)
      GOTO 800
      ENDIF
      NUMIN= MIN(NUMIN,NUMOUT)

C
C---- Do each eqn of this node -----
C
      IQ= IQ1
200  CONTINUE
C
C---- Fill in the equation_view data base from the model db.
C
      CALL TRFQDT('GET',IQ)
C
C---- Mark state of model eqn_data_base for later use
      SGLOLD= NYL.EQ.1
      FNMOLD= FNAM
      NYLOLD= NYL
      NXLOLD= NXL
      NPLOLD= NPL
C---- Get the (new) function type
300  CONTINUE
      FNAM = FTYPE
C
      FNAM = 'CON'
      IF (FNAM.EQ.' ') GOTO 800
      IF (FNAM.NE.FNMOLD) THEN
C----- Check the changed function type
      CALL LJSTR(FNAM,STRING,I)
      FNAM= STRING
      CALL UPCASE(FNAM)
      IF (FNC2I(FNAM).EQ.0) THEN
      MESSAG= ' *** Invalid function type: '//FNAM
      CALL LNMSGD(24)
      GOTO 800
      ELSE
      IF (FNAM.EQ.'MATRIX'.AND.NUMOUT.EQ.1) THEN
      MESSAG= ' *** Use SUM, not MATRIX, for single outputs.'
      CALL LNMSGD(24)
      GOTO 300
      ENDIF

```

[illegible]

```
*
C----- MCKMTH.FOR: Math support for building M,C,K macros. *
C *
C----- PURPOSE: This file contains math functions required to build *
C MCK macros. *
C *
C----- Contents: *
C INVMAT, Finds inverse of a matrix. *
C LUDCMP, Performs a LU decomposition of a matrix. *
C LUBKSB, Performs back substitution on LU matrix. *
C MTXMUL, Multiplies two square matrices. *
C MTXCML, Multiplies a matrix by a constant. *
C MTXTRN, Finds transpose of matrix (square only). *
C MTXADD, Adds two matrices. *
C DETCPY, Copies reduced matrix for finding det. *
C DETMTX, Finds the determinant of a matrix. *
C BALAN2, Balances a square matrix. *
C ELMHS2, Puts square matrix in Hessenberg form. *
C HQR3, Finds eigenvalues of matrix in Hess. form *
C *
C----- Last Change: September 19, 1991. RM *
C *
CEOFH----- *
C
$MESSAGE: 'INVMAT'
CINVMAT>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Last Change: 9/19/91 RM
C
SUBROUTINE INVMAT(A,N,NP,INDX,ANS)
C
C----- PURPOSE: Inverts a matrix.
C
C----- Inputs: A - The (NxN) matrix to be inverted.
C N - The dimension of the matrix to be inverted.
C NP - Physical size of the incoming matrix.
C INDX - Index matrix (NPx1).
C
C----- Outputs:
C ANS - The inverted matrix.
C
C----- Note: The A matrix is destroyed in this routine!
C
DIMENSION A(NP,NP), ANS(NP, NP), INDX(N)

INTEGER N, NP
REAL D
INTEGER I, J

EXTERNAL LUDCMP, LUBKSB

C***INVMAT*****
C
C----- Set up the identity matrix.
C
DO 12 I=1,N
DO 11 J=1,N
ANS(I,J)=0.0
CONTINUE
ANS(I,I) = 1.0
CONTINUE
11
12
C
C----- Perform a LU - decomposition of the marix.
C
CALL LUDCMP(A,N,NP,INDX,D)
```



```
C----- Invert by back substitution.  
C  
      DO 13 J=1,N  
      CALL LUBKSB(A,N,NP,INDX,ANS(1,J))  
13    CONTINUE  
C  
      RETURN  
      END  
  
C  
CEND:INVMAT<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<  
C  
$MESSAGE:'LUDCMP'  
CLUDCMP>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Last Change: 9/19/91 RM  
C  
      SUBROUTINE LUDCMP(A,N,NP,INDX,D)  
C  
C----- PURPOSE:   Performs a LU decomposition of a matrix.  
C  
C----- Inputs:     A - The (NxN) matrix to be decomposed.  
C                     N - The dimension of the matrix to be inverted.  
C                     NP - Physical size of the incoming matrix.  
C  
C----- Outputs:    ANS - The inverted matrix.  
C  
C----- Note:       The A matrix is destroyed in this routine!  
C  
C----- Note:       The source code has been omitted from this listing.  
C                     See reference [8] for details.  
C  
      INTEGER    NMAX  
      REAL VV  
      PARAMETER (NMAX=100,TINY=1.OE-20)  
      DIMENSION A(NP,NP),INDX(N),VV(NMAX)  
  
      REAL        D, AAMAX, SUM, DUM  
      INTEGER      I, J, K, IMAX  
  
C***LUDCMP*****  
C  
  
      See reference [8]  
  
      RETURN  
      END  
  
C  
CEND:LUDCMP<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<  
C  
$MESSAGE:'LUBKSB'  
CLUBKSB>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Last Change: 9/19/91 RM  
C  
      SUBROUTINE LUBKSB(A,N,NP,INDX,B)  
C  
C----- PURPOSE:   Performs back substitution on LU matrix.  
C  
C----- Inputs:     A - The (NxN) matrix to be decomposed.  
C                     N - The dimension of the matrix to be inverted.  
C                     NP - Physical size of the incoming matrix.  
C                     INDX - Permutation list from LUDCMP.  
C                     B - + or - for odd or even number of exchanges.  
C  
C----- Outputs:  
C
```

[illegible]

17  
18  
C

1

[illegible]



[illegible]

```

C      SUBROUTINE HQR3(A,N,NP,WR,WI)
C
C----- PURPOSE: Finds the eigenvalues of A matrix (square only).
C
C----- Inputs:  A,      - The matrix to be balanced.
C                  N      - The working dimension of the matrix (square).
C                  NP     - Physical size of the incoming matrices.
C
C----- Outputs:
C                  WR     - Real parts of the eigenvalues.
C                  WI     - Imaginary parts of eigenvalues.
C
C----- Note:
C          WR AND WI CONTAIN THE REAL AND IMAGINARY PARTS,
C          RESPECTIVELY, OF THE EIGENVALUES.  COMPLEX CONJUGATE
C          PAIRS OF EIGENVALUES APPEAR CONSECUTIVELY WITH THE
C          EIGENVALUE HAVING THE POSITIVE IMAGINARY PART LAST.
C
C----- Note:      The source code has been omitted from this listing.
C                  See reference [8] for details.
C
C          INTEGER      N, NP
C          REAL          A, WR, WI
C          DIMENSION     A(NP,NP),WR(NP),WI(NP)
C
C          INTEGER      I, J, M, NN, ITS, L, K
C          REAL          ANORM, X, Y, T, S, W, P, Q, Z, R, U, V
C
C***HQR3*****
C

```

**See reference [8]**

RETURN  
END

[illegible]

```

C      SUBROUTINE ETECTS(WI,NP)
C
C----- PURPOSE: Finds the eigenvectors of M,K matrices.
C
C----- Inputs:
C          WI      - Eigenvalue corresponding to the eigenvector.
C          NP      - Physical size of the incoming matrices.
C          MASMAT  - Mass matrix.
C          STFMAT  - Stiffness matrix.
C
C----- Outputs:
C          EIGMAT  - Matrix of eigenvectors.
C
C----- Note:
C          WI  CONTAINS THE REAL AND IMAGINARY PARTS,
C          RESPECTIVELY, OF THE EIGENVALUES.  COMPLEX CONJUGATE
C          PAIRS OF EIGENVALUES APPEAR CONSECUTIVELY WITH THE
C          EIGENVALUE HAVING THE POSITIVE IMAGINARY PART LAST.
C
C      INCLUDE 'MCKHDR.CBK'
C

```





MICHIGAN STATE UNIV. LIBRARIES



31293009010640