This is to certify that the

thesis entitled

SIMULATION OF AN ELECTROMAGNETIC
ACTUATOR USING BOND GRAPHS

presented by

Nicholas John Hendriksma

has been accepted towards fulfillment
of the requirements for

Masters degree in Science

R. C. Rosenberg
Major professor

Date Nov. 6, 1984

SIMULATION OF AN ELECTROMAGNETIC ACTUATOR USING BOND GRAPHS

By

Nicholas John Hendriksma

A THESIS

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

MASTER OF SCIENCE

Department of Mechanical Engineering

1984

ABSTRACT

SIMULATION OF AN ELECTROMAGNETIC ACTUATOR USING BOND GRAPHS

BY

NICHOLAS JOHN HENDRIKSMA

Electromagnetic actuators are key components in numerous engineering systems. the typical approach in modeling such devices involves finite element methods. These techniques allow magnetic phenomena to be treated with a minimum of empiricism. In some cases, however, a simpler lumped-parameter approach may be more appropriate. Such situations arise when i) computer time is at a premium, ii) the dynamics of associated devices in a system must be included, or iii) the model is to be used for feedback control design. This thesis details the development and computer implementation of a lumped-parameter bond graph model for an electromagnetic actuator system. Comparisons of predicted and experimental results are included to verify the technique.

with love to my wife, Jane

ACKNOWLEDGMENTS

I would like to express my appreciation to my major professor, Dr. Ronald C. Rosenberg, for his guidance and assistance over the last year.

Special thanks to Mr. Rick Teerman for his interest and encouragement on this project.

Finally, I would like to thank Rochester Products for their generous support of my continued education.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

A = cross-sectional area

B = flux density

C = bond graph compliance

d = depth

Di = vector of inputs to nonlinear dissipation elements

Di_L = vector of inputs to linear dissipation elements

Do = vector of outputs from the nonlinear dissipation elements

Do_L = vector of outputs from the linear dissipation elements

E = energy

emf = electromotive force

F_{fl} = fluid damping force

Fm = magnetic force

g = gap length

GY = bond graph gyrator element

H = magnetic intensity

I = bond graph inertia element

l = length

M_f = fluid inertia

mmf = magnetomotive force

Pm = permeance

r = radius

R = bond graph dissipation element

Rm = reluctance

$S1$ = simple junction structure matrix

t = thickness

Ti = vector of inputs to modulated junction structure

To = vector of outputs from modulated junction structure

U = vector of outputs from the source elements

V = vector of inputs to the source elements

w = width

X = displacement

$\dot{X}$ = velocity

Y = state vector

$\dot{Y}$ = time derivative of the state vector

Z = vector of outputs from the energy storage elements

ρ = density

ϕ = magnetic flux

μ = viscosity

μ = permeability

μ_0 = permeability of free space

1 INTRODUCTION

Electromagnetic actuators are key components in numerous engineering systems. In recent years computer simulation programs have been developed to aid the design of these actuators(1,2,3,4). The nature of these devices has led to an extensive use of finite element methods which allow magnetic phenomena to be treated with a minimum of empiricism.

However, in some cases the needs of the designer may be met best by a simpler lumped-parameter model. The loss of theoretical rigor can be offset by the advantages offered by this approach.

First, the size and complexity of finite element models typically involves significantly more computational effort than the corresponding lumped-parameter representation. When several variables are to be optimized, the lumped-parameter model can be used interactively to focus the parameter search with considerable savings in computer time. The more expensive finite element models can then be used to detail a design.

Second, the dynamic behavior of the magnetic components may be coupled with that of the associated electrical and mechanical devices which comprise an actuator system. Modifications of finite element programs to include these effects can be time consuming and depend on the derivation of the coupled relationships. With lumped-parameters, the application of bond graphs(5,9) allows models representing the

electrical, mechanical, and magnetic components to be combined in a structured way. The differential equations implied by the bond graph properly account for the coupling between the various energy domains.

Finally, to achieve a desired system behavior, it may be necessary to implement feedback control. The finite element approach is not as useful in this regard due to the large number of equations involved in the model description. In contrast, only a small number of differential equations are required to describe the lumped-parameter bond graph model and the equations are normally cast in the preferred state-variable form.

Bond graphs are a powerful tool in the formation of lumped-parameter models. This method provides a unified basis for integrating the dynamics of multi-energy domain systems such as electromagnetic actuators. As a result, the relationships between energy domains can be symbolically and mathematically expressed by a bond graph model. The structure of these models is highly organized so that computer programs can be used to generate the required state equations. Potential difficulties in the equation formulation, such as nonlinear algebraic loops, can be diagnosed early in the analysis so that alternative models may be formed and compared on this basis.

This study details the development and computer implementation of a bond graph model of an electromagnetic actuator system. Comparisons to experimental data are provided to verify the techniques.

2 DEVELOPMENT OF THE MODEL

2.1 Description of the System

2.1.1 Physical Description

A system containing an electromagnetic actuator is shown in figure 1.

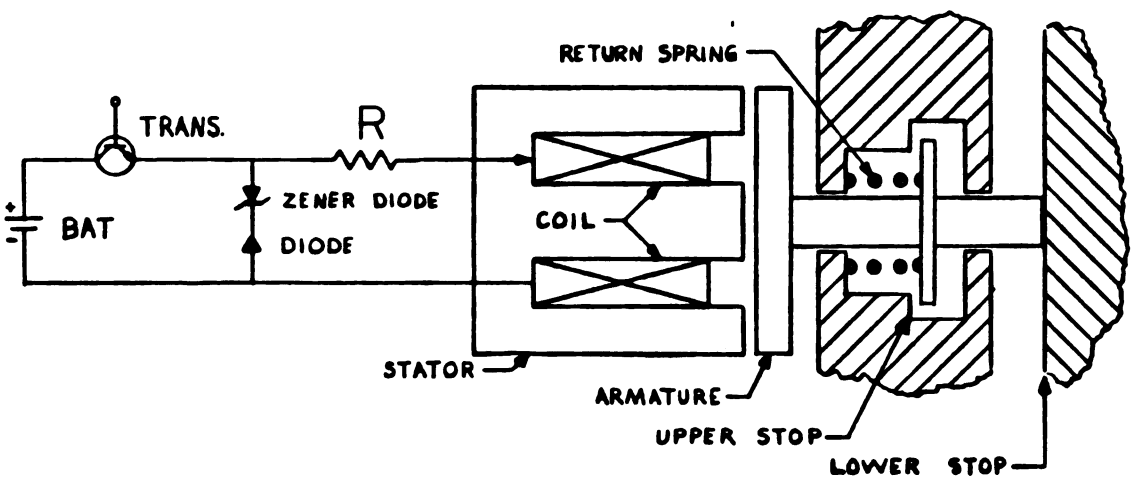


Figure 1: Electromagnetic Actuator System

The system consists of electrical, mechanical, and magnetic subsystems. The electrical and mechanical subsystems have been simplified in order to concentrate on the actuator model.

The electrical subsystem consists of a battery, transistor, zener diode, diode, resistor, and coil. The circuits which control the

transistor are neglected for simplicity. The resistor characterizes the resistances of the coil and wire leads.

The armature, guiding shaft, and return spring comprise the simplified mechanical subsystem. The motion of the armature is limited by two fixed stops. The return spring is preloaded to hold the armature at the lower stop. The armature and shaft move within a fluid medium which results in fluid damping between the stator/armature faces and guide shaft/lower stop surfaces.

The magnetic subsystem includes the stator, armature, leakage and the working air gaps. The flux paths through nonferrous materials are termed "air gaps". The leakage air gaps are flux paths which do not involve the moveable armature. The stator and armature materials are characterized by high magnetic permeability, while that of the air gaps is extremely low. The stator is fabricated from thin laminations. The armature is a continuous rectangular slab. Although this study focuses on this specific type of magnetic arrangement, the analysis may be applied to other configurations as well.

2.1.2 Operational Description

The formation of a model requires an understanding of the operation of the system. A study of the related energies within and between the three subsystems can help develop this understanding. The manner in which energy is supplied, stored, dissipated or converted to other domains must be identified.

The electrical subsystem is the source of energy for the system. The transistor controls the flow of energy from the battery by regulating the current. The ability to control the current is different, however, for the cases of current increase and current decrease. Current can be reduced to zero arbitrarily, but increases are subject to the transistor saturation voltage. When saturation occurs, the current flow depends on the magnetic subsystem. The zener diode protects the electronic components by limiting the negative voltage transients due to the inductive nature of the coil. Current flow through the resistor and zener diode results in energy dissipation.

Energy conversions between the electrical and magnetic domains take place in the coil. Current through the coil produces a magnetomotive force(mmf) in the magnetic domain, while the flux flow in the magnetic domain results in a voltage in the electrical domain.

Energy storage in the magnetic subsystem is characterized by magnetic flux. Increasing flux indicates conversion of electrical or mechanical energy to magnetic energy, while decreasing flux denotes the reverse conversions. The magnetic energy is stored in both the metallic and nonmetallic elements of the magnetic "circuit". While some of the energy resides in the stator and armature, the bulk is stored in the air gaps. High flux densities, however, can saturate certain metallic sections which leads to an increased rate of energy storage in these components. Thus, the energy stored in the metallic elements must be considered as well as energy in the air gaps.

Dissipation of magnetic energy occurs in the metallic elements in the form of eddy currents and hysteresis. Eddy currents are small current loops within the metal caused by voltages induced by time-varying magnetic flux. These currents and the electrical resistance of the material convert a portion of the magnetic energy to heat. The hysteresis in the DC magnetization curve results in additional energy losses in the form of heat. For an alternating magnetic flux the eddy current losses per cycle vary with frequency and amplitude, while the hysteresis losses per cycle depend only on the amplitude. The hysteresis losses were neglected in this study.

The energy stored in the working air gaps generates an attractive force on the armature. Armature motion indicates energy conversion between the magnetic and mechanical subsystems. Motion toward the stator transfers energy from magnetic to mechanical, while motion away from the stator results in the opposite conversion.

Energy storage in the mechanical subsystem is characterized by the armature displacement and momentum; the displacement is related to the potential energy of the return spring. The momentum is related to the kinetic energy of the moving mass. Energy is dissipated by mechanical friction and fluid damping. It is assumed that the fluid damping losses are dominant; the mechanical friction is neglected.

2.2 Bond Graph Representation

2.2.1 Definition of Key Variables

The flow of power within the physical system is the basis for the bond graph model. Since three energy domains are involved, a consistent set of power variables is required for each domain. The SI system of units is useful for these mixed-domain systems because power is measured in watts in each domain. The bond graph power variables are generalized "efforts" and "flows". Thus, the effort and flow variables for the electrical, mechanical, and magnetic energy domains must be defined. For electrical power, emf and current are the required effort and flow variables, while mechanical power is the product of force and velocity.

The magnetic variables are not as obvious. Many texts(6,7) dealing with magnetic devices draw an analogy between the following electrical and magnetic variables: emf-mmf, current-flux, and resistance-reluctance.

The analogy implies that mmf and flux may be used as the effort-flow power variables. However, this choice is not suitable for bond graphs because the product of mmf and flux is energy, not power. Also, the resistance-reluctance analogy is misleading because an electrical resistor dissipates energy while magnetic reluctance connotes energy storage. Thus, dynamic models based on this analogy have difficulty accounting for the energy dissipation in magnetic circuits(8).

These problems are eliminated when mmf and flux rate, $\dot{\Phi}$, are defined as the effort-flow variables(9). First, the product of mmf and flux rate is power. Second, it allows a more appropriate analogy between magnetic permeance (reciprocal of reluctance) and electrical capacitance since both relate to energy storage. Finally, it provides an analogous magnetic "resistance" element to model the energy dissipation in magnetic circuits.

Table 1 summarizes the variable definitions for the three subsystems. The bond graph R elements in each domain represent the corresponding energy dissipation modes.

Table 1: Summary of Key Bond Graph Variables

Generalized	Mechanical	Magnetic	Electrical
effort	force	mmf	emf
flow	velocity	flux rate	current
displacement	displacement	flux	charge
C-parameter	compliance	permeance	capacitance
momentum	momentum	--	flux linkage
I-parameter	mass	--	inductance

2.2.2 Structure

The bond graph structure is based on the energy and power relationships of the physical system. Elements which supply, store, dissipate or transform energy in the physical system are modeled by corresponding lumped-parameter bond graph elements. The relationships between these lumped-parameter elements are modeled by the junction structure of the bond graph. Definitions of the bond graph elements are provided in Appendix A. In this section bond graph models are formed

for the electrical, mechanical and magnetic subsystems. The use of the effort-flow variables defined in the previous section permits the combination of these submodels into an overall system model.

The bond graph model for the electrical subsystem is shown in Figure 2.

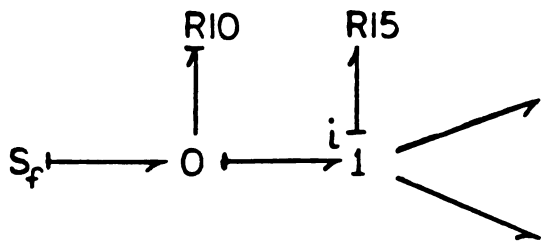


Figure 2: Bond Graph Model of the Electrical Subsystem

The 1-junction represents the current flowing through the coil. The combined resistance of the wire leads and coil is portrayed by the linear R15 element attached to this junction. The S and R10 elements model the behavior of the battery, transistor, and zener diode. It is assumed that the desired current is a known function so that the battery, transistor, and control circuitry can be modeled as a current source. However, the desired current cannot always be enforced due to transistor saturation and zener diode breakdown. The nonlinear R10 element models these effects by modifying the relationship between the source flow and the 1-junction flow based on the voltage at the 0-junction. Obviously, more sophisticated models for the battery, transistor, and zener diode could be formed, but the simplifications

above are sufficient for the purposes of this study. Techniques for modeling diodes and transistors can be found in the references(5,16). The dynamic effects of the coil are included in the magnetic subsystem.

The mechanical subsystem bond graph model is shown in Figure 3.

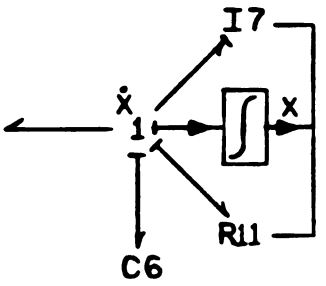


Figure 3: Bond Graph Model of the Mechanical Subsystem

The 1-junction represents the velocity of the armature which is common to the mass, I7, return spring, C6, and fluid damping, R11, lumped elements.

The integration block indicates that the I7 and R11 parameters are functions of the displacement. The activation of the bonds by the full arrows means that no effort is fed back to the 1-junction. The C6 element represents the action of the two stops in addition to the return spring. This subsystem is characterized by two state variables: the displacement, Y(6), and the momentum, Y(7).

The electrical-magnetic analogies developed previously can aid the formation of a bond graph model for the magnetic subsystem. Figure 4

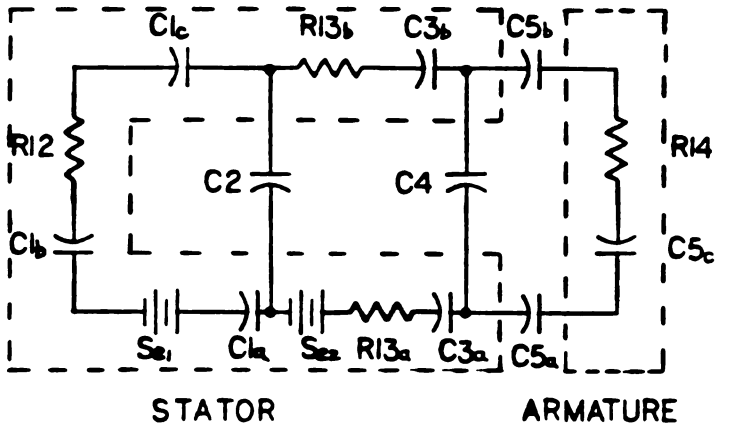


Figure 4. Electrical Analogy to the Magnetic Circuit.

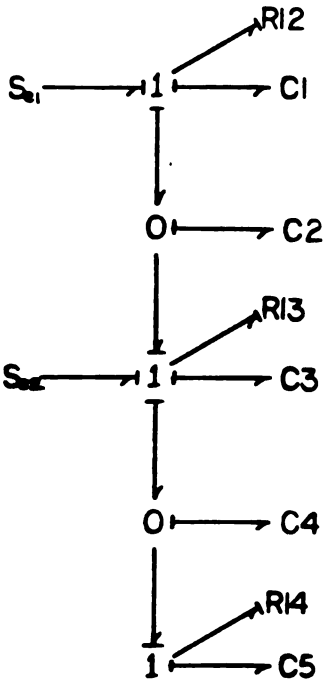


Figure 5. Bond Graph Model of the Magnetic Subsystem.

shows an electrical equivalent of the magnetic circuit. The symmetry of the stator allows a simplification to the "horseshoe" shape. Lumped-parameter elements are utilized to model the distributed-parameter characteristics of this subsystem.

Five discrete paths, each consisting of one or more elements, are employed in the approximation. The battery symbols indicate the mmf generated by the coil. The resistor symbols represent eddy current losses while the capacitor symbols illustrate the storage of magnetic energy. The coil is modeled as two discrete sources separated by a storage element; this approximates the flux linkage produced by the distributed leakage flux. Separate energy storage and dissipation elements are defined for the various cross-sectional areas and material properties which characterize the stator and armature. Standard bond graph techniques are used to convert the magnetic circuit description into the bond graph format. Elements which share a common flow are combined into single equivalent elements on the bond graph. The bond graph representation is shown in Figure 5. This subsystem is characterized by five state variables, Y(1)-Y(5), which represent the flux in each of the five paths defined in Figure 4.

The system model is formed by linking the three component models together as shown in Figure 6. The coupling elements enforce the relationships between the energy domains. The gyrator elements allow the current flow in the coil to generate mmfs (efforts) in the magnetic subsystem. These efforts represent the lumped effort sources in the magnetic circuit of Figure 5. Also, the flux flow in the magnetic

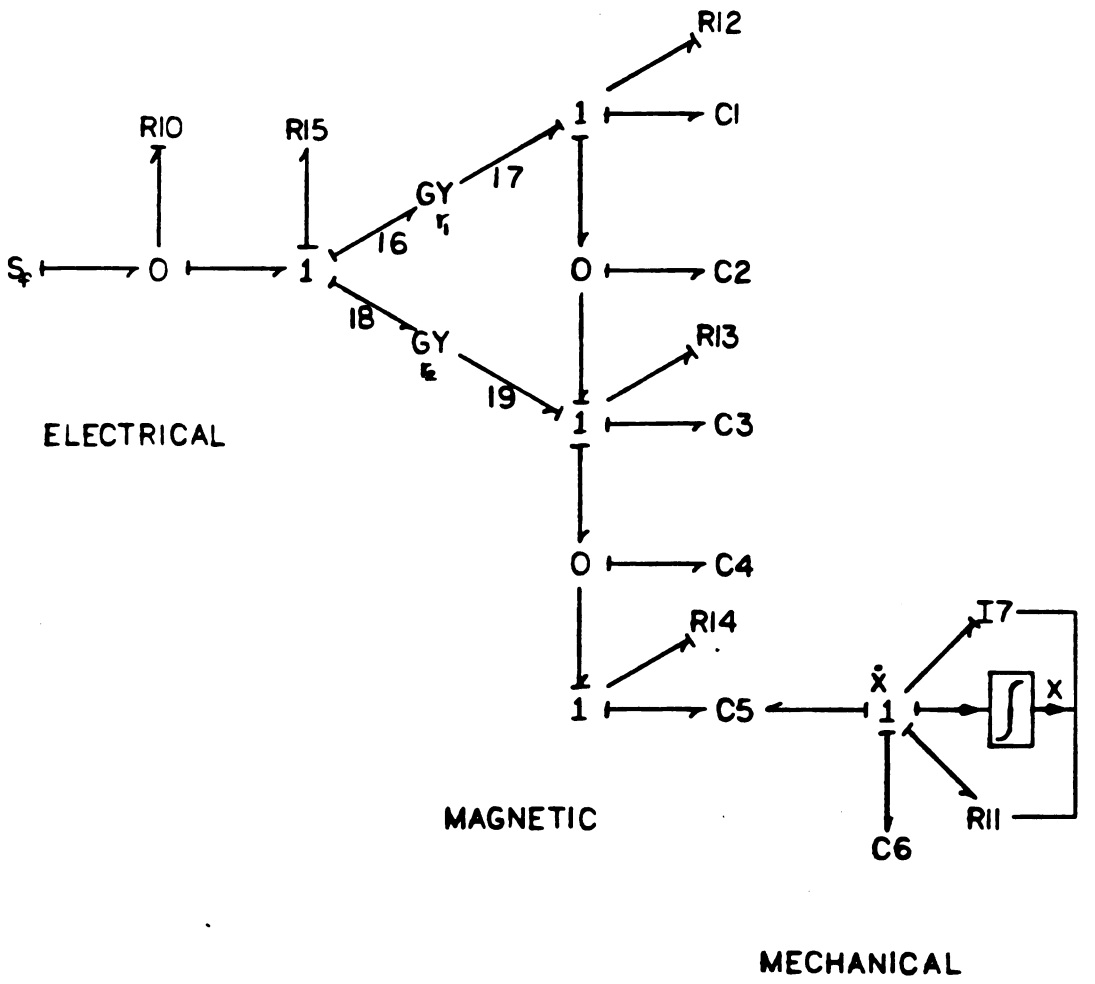


Figure 6. Bond Graph Model of the Actuator System.

circuit produces voltages (efforts) in the electrical circuit. The gyrator moduli are functions of the number of turns of the coil. The relationship between the mechanical and magnetic domains is represented by the two-port C element, C5. The energy stored in this field exerts efforts in both domains, as indicated by the causality. Variations of the displacement variables in either domain impact the magnitudes of the output efforts.

The ability to link models in this way is very useful since different electrical or mechanical models can be implemented with the same actuator model. The unified bond graph approach provides a method of deriving the resulting coupled differential equations in a structured way.

2.2.3 Some Modeling Considerations Based on Causality

A number of possible bond graph representations were considered in the development of the actuator model. The objective was to keep the model as simple as possible while still characterizing the important behavior of the system. Certain computational aspects of these models can be evaluated at the onset based on causality. These causal implications influence modeling decisions.

Since the stator is laminated, it may seem reasonable to disregard eddy current losses in this component. However, the causality of these models results in dependent energy storage elements in the magnetic subsystem. While this presents no difficulty in the linear case, it can

lead to complexities when nonlinearities are involved. To avoid these complications, the eddy current losses were included using simple linear R-elements. The inverse causality of these elements means that these parameters cannot be set to zero. Setting these parameters too small also leads to integration difficulties. Thus, the proportion of stator losses to the total magnetic losses may not be accurately represented in the model.

Causality also indicates an algebraic loop involving the dissipation elements in the electrical and magnetic subsystems. In this study, the linear assumption for the magnetic dissipation elements allows the reduction of the loop to a single implicit equation which can be easily solved. The use of this model with other electrical subsystems or nonlinear magnetic dissipation elements may require special techniques to solve the loop(10).

2.2.4 Constitutive Equations

The constitutive equations are the input-output relationships of the individual lumped-parameter bond graph elements. The causal strokes identify the proper input and output variables. This section discusses these relationships for the nonlinear elements in the actuator model. The numbering of the elements corresponds to Figure 6.

In the electrical subsystem, the current output from element R10 is a nonlinear function of the input voltage as shown in Figure 7.

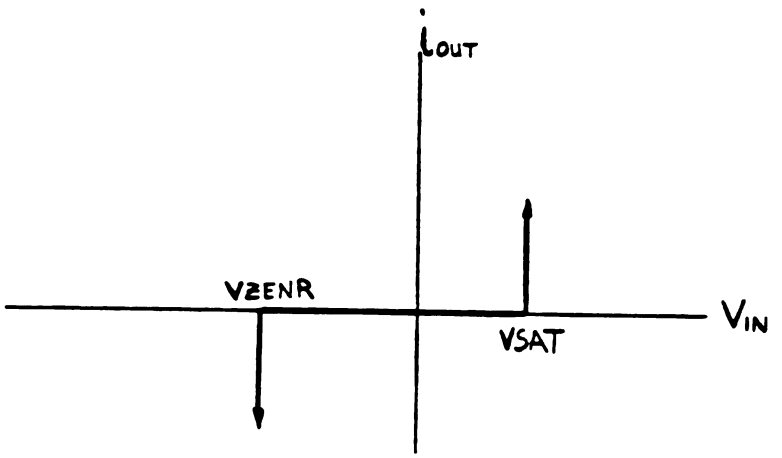


Figure 7: Constitutive Law for Element R10

For voltage inputs between VZENR and VSAT the output current is zero. This corresponds to transistor regulation and all the source current flows through the coil. Transistor saturation occurs when the source current cannot be delivered to the coil without exceeding the VSAT threshold on the input to R10. In this situation, a portion of the source current is shunted to R10 so that the voltage does not exceed VSAT. Zener diode breakdown occurs when delivery of the source current to the coil results in voltage inputs to R10 less than VZENR. The output current in this case is adjusted to limit the voltage input to VZENR. This type of constitutive law is possible because the equation for the input to this element is implicit.

In the mechanical subsystem the output force from element C6 is a piecewise linear function of the displacement state variable, Y(6). The stiffness of the return spring is used for displacements between the two stops while the impacts with the two stops are modeled using a much

stiffer linear spring rate.

It is assumed that the major damping forces on the armature are due to the two fluid films shown in Figure 8. The radius, r_a, is an effective radius to approximate the film above the rectangular armature while r_s is an effective radius representing the shaft diameter.

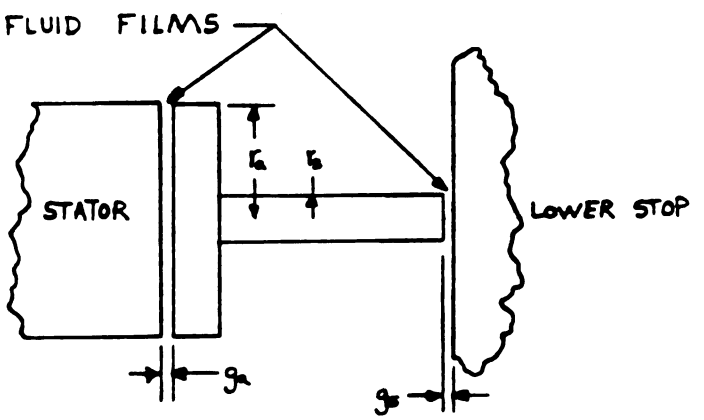


Figure 8: Fluid Damping Films

The equations for a disk approaching a flat wall in a fluid medium are used to obtain an approximation to the fluid damping forces(8).

$$F_{fl} = \frac{3\pi\mu r^4 \dot{x}}{2g^3} + \left(\frac{15}{56} - \frac{j}{8}\right)\frac{\pi\rho r^4 \dot{x}|\dot{x}|}{g^2} \qquad (1)$$

The j is used to account for the pressure loss when fluid enters the gaps: for increasing gaps j=1, for decreasing gaps, j=0. In the

actual implementation of the equation, a minimum gap was set to prevent unrealistic forces when g approached zero.

The disk-flat wall analysis is also used to obtain the inertial effects of these fluid films(8). The fluid equivalent mass is given by

$$m_f = \frac{3\pi\rho r^4}{20g} \qquad (2)$$

Therefore, the mass associated with the I7 bond graph element is the sum of the mechanical and fluid equivalent masses. In some cases it may be possible to neglect the fluid inertial effects, depending on the fluid density and gaps involved. The implementation of these nonlinearities is straightforward since both are functions of state variables.

The R12, R13, and R14 elements represent the eddy current losses in the stator and armature. These losses depend on the lamination thickness, material resistivity, flux density, and excitation frequency. Since these losses are difficult to express analytically, linear relationships are assumed. Experimental data aids the selection of these values within a desired operating range. It may also be possible to utilize finite element models in this regard.

The constitutive laws for the magnetic C elements are defined by reluctance or its inverse, permeance. The reluctance of various ferromagnetic sections is computed as

$$Rm = 1/Pm = l/A\mu \qquad (3)$$

and the output effort as

$$mmf = Rm \phi \; . \qquad (4)$$

The lengths and areas used in equation (3) must be "effective" values determined by estimating the flux distribution within the device. The permeability, μ, varies as a function of the flux density, B, in each section. The relationship can be derived from B-H curves provided by material suppliers. Magnetic saturation, which occurs at high flux densities, is characterized by a rapid decrease in permeability for further increases in flux.

In the model, cubic spline equations are used to describe the μ-B relationship for the various materials involved. Although a uniform flux density is assumed, the actual flux distribution is distorted due to the shape of the device and the eddy current effects. Consequently, this assumption leads to inaccurate permeability values when materials are in transition to magnetic saturation. The sharp "knee" in the B-H curves causes this problem. Again, these nonlinearities are easily implemented because the flux quantities involved are state variables.

Until saturation occurs, the dominant reluctances of the magnetic circuit are those of the air gaps. The reluctance parameters for the leakage air gaps are linear, while those of the working air gaps are nonlinear due to the armature motion. Special equations must be used to

calculate the reluctance of air gaps due to "fringing" effects. Further details of the magnetic reluctance calculations are included in Appendix B.

The constitutive equations for the two port C-field are obtained by differentiating the stored energy function for the field which guarantees conservation of energy. The energy stored in the working air gaps is given by equation (5) as a function of the gap length, X, and the flux, ϕ.

$$E(x,\phi) = \int_0^{\phi} Rm\ \phi d\phi = \frac{Rm\ \phi^2}{2} \qquad (5)$$

The mechanical displacement is held fixed so that the gap reluctance is constant with respect to $d\phi$ in the integration. The port constitutive equations are obtained by taking the partial derivative of equation (5) with respect to X and ϕ.

$$mmf = \frac{\partial E(x,\phi)}{\partial \phi} = Rm\ \phi \qquad (6)$$

$$Fm = \frac{\partial E(x,0)}{\partial x} = \frac{\phi^2 dRm}{2\ dx} \qquad (7)$$

Therefore, the energy stored in the gap generates an mmf in the magnetic domain and a force in the mechanical domain.

3 COMPUTER IMPLEMENTATION OF THE MODEL

Implementing the actuator model requires the formulation and integration of the system state equations. A powerful feature of the bond graph approach is the ability to perform these operations automatically using computer programs. ENPORT(12), for example, is a highly developed simulation package for linear systems based on bond graph representations. Since the actuator model contains significant nonlinearities, a computer program, BGSIM, was written to formulate the coupled state equations. The program is general so that additions or modifications to the bond graph presented in this paper can be handled as well. The user, however, must code the required constitutive equations in the appropriate subroutines. This chapter outlines the techniques used to implement the bond graph model. A users guide for BGSIM is included in Appendix C.

3.1 Organization

The program is organized around the input-output relationships of the lumped-parameter elements. These elements are categorized according to energy properties into one of five groups as shown in Figure 9. The bond graph elements corresponding to each group are indicated in the appropriate blocks.

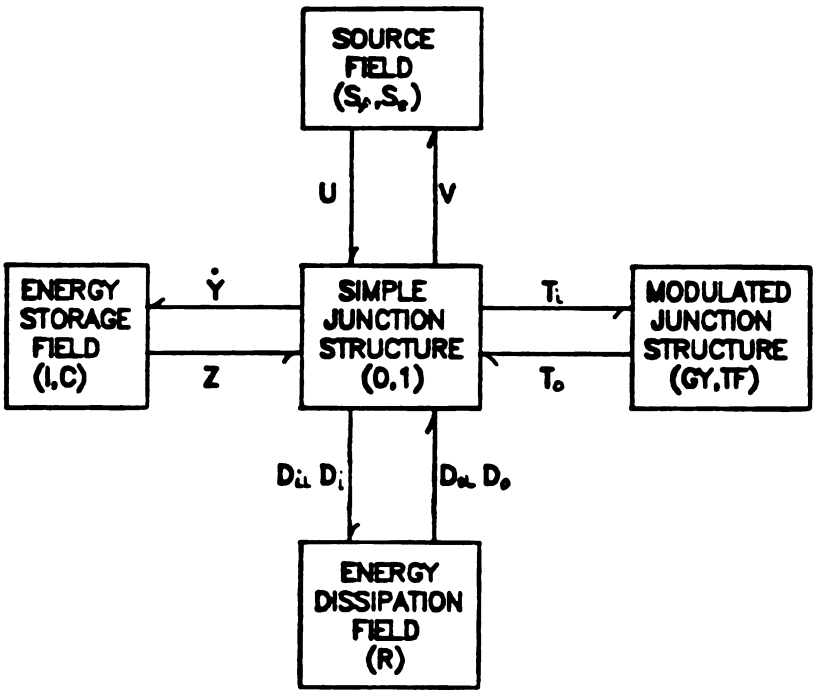


Figure 9: Symbolic Bond Graph Relationships

Each element in the peripheral blocks is defined by a constitutive equation which determines an output for each input. The input-output variables are efforts or flows as defined by causality. The center block represents the topology of the bond graph and enforces the relationships among the peripheral blocks. Since each block typically contains several elements, vector-matrix notation is used to aid the manipulation of the equations involved. The lines to and from each block represent vectors of complementary input and output variables related by the constitutive equations. The simple junction structure block can be expressed as a matrix which relates these vectors.

The constitutive equations for each of the peripheral blocks are contained in separate subroutines in BGSIM. Subroutines ZVECTR, UVECTR, DVECTR, and TVECTR implement the equations for the storage, source, dissipation, and modulated juncture structure blocks, respectively. The junction structure matrix is formed in subroutine JS.

3.2 State Equation Formulation

The relationships implied by Figure 9 are expressed in vector-matrix form by equation (8).

$$\begin{bmatrix} \dot{Y} \\ V \\ Di \\ Di_L \\ Ti \end{bmatrix} = \begin{bmatrix} & & \\ & S1 & \\ & & \end{bmatrix} \begin{bmatrix} Z \\ U \\ Do \\ Do_L \\ To \end{bmatrix} \qquad (8)$$

Equation (8) is simplified by eliminating vectors related by linear constitutive laws. The Ti,To vectors are eliminated by partitioning equation (8) and solving the simultaneous equations using the constitutive matrix. The same process is used for Di_L,Do_L and the ordering of the vectors in equation (8) allows the same subroutines to be used for both operations. Elimination of the linear vectors reduces the size of the S1 matrix by weighting the remaining entries of S1 to account for these elements. The reduced form of equation (8) is given by equation (9).

$$\begin{bmatrix} \dot{Y} \\ V \\ Di \end{bmatrix} = \begin{bmatrix} & S1'' & \end{bmatrix} \begin{bmatrix} Z \\ U \\ Do \end{bmatrix} \qquad (9)$$

Evaluation of equation (9) to obtain $\dot{Y}$ is straightforward when the nonlinear dissipation elements are not coupled algebraically. When coupling is present, special techniques are usually required to solve the implicit equations. Fortunately, the coupling in the actuator model involves only a single implicit equation which can be solved without iteration. The existence of such coupling is predicted when causality is assigned.

At each time step equation (9) is used to compute $\dot{Y}$ from updated values of the Z, U, and Do vectors. These vectors are computed in ZVECTR, UVECTR, and DVECTR respectively.

3.3 Integration of the State Equations

The differential equations which characterize the actuator system can become mathematically "stiff" depending on the specific configuration and operating conditions. Magnetic saturation, small eddy current R parameters, and the rigid armature stops all contribute to this problem. Integration of these equations can involve significant computer time because small time increments must be used. Several integration routines were tried in an effort to minimize the computer time required to obtain accurate solutions. The best results were obtained with an Adams-Moulton-Bashforth algorithm, ODERT(13), which

provides a variable time step. This routine can also be used in conjunction with a root finding function (subroutine RCHK) when discontinuities are present in the equations. The routine uses status flags to control the implementation of the discontinuities and adjusts the integration step size to coincide with the subsequent switch. This function was used in the actuator model to handle the return spring-rigid stop and transistor saturation-regulation discontinuities.

4 COMPARISONS TO EXPERIMENTAL RESULTS

The bond graph model was used to simulate the behavior of a physical system similar to that of Figure 1. Comparisons between predicted and experimental results are made for two modes of operation: i) the steady-state magnetic force for constant inputs and fixed armature positions, and ii) dynamic system response to a step input. A third mode, dynamic coil current in response to step inputs with fixed armature positions, was used to obtain eddy current R values for the model. These modes, representing increasing complexity of operation, allow a systematic verification of the model.

4.1 Steady State Magnetic Force

The steady-state magnetic force is obtained by fixing the armature position, supplying a constant current input and allowing the system to come to equilibrium. The reluctance calculations for the magnetic C elements can be verified in this mode because the eddy current R elements do not affect the magnitude of the steady-state force. A comparison of the predicted and experimental magnetic force for a range of input currents and fixed air gaps is shown in Figure 10.

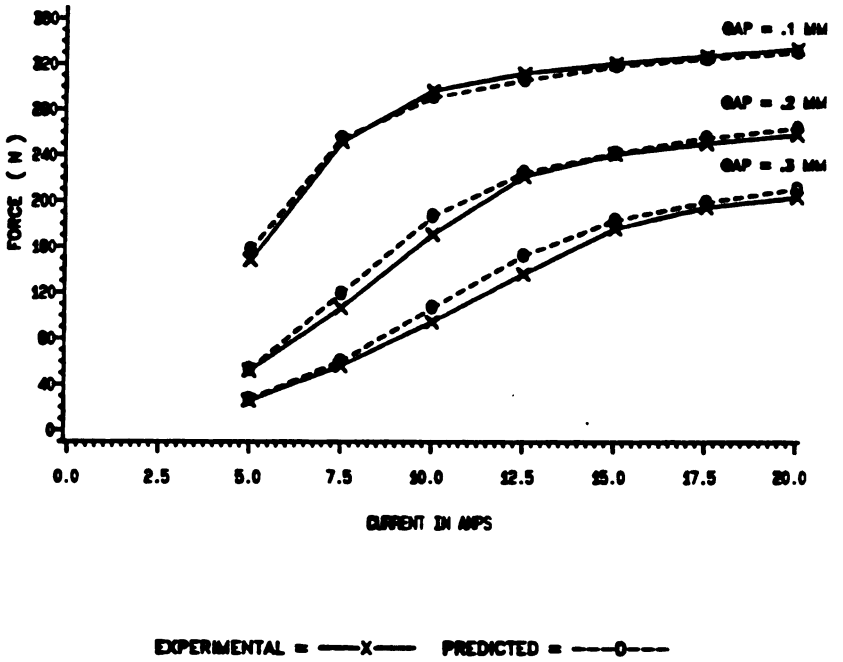


Figure 10: Steady State Magnetic Force

The largest discrepancies, approximately 13%, occur as the magnetic elements begin to saturate. This transition is difficult to model accurately due to the assumption of uniform flux density and the abrupt "knee" in the magnetization curves at saturation.

The magnitude of the magnetic force is quite sensitive to the air gap length. Since the stator and armature faces of the experimental device are not perfectly flat or parallel, the absolute value of the experimental air gap is difficult to quantify. To overcome this difficulty, the value of the air gap used in the simulation was adjusted

to match the experimental and predicted force at the longest air gap and lowest current. At these conditions the force generated is solely a function of the air gap length. The resulting simulation air gap was 30 microns longer than the measured value. However, a portion of this difference is due to the effects of a countersunk screw used to attach the armature and guide shaft. The relative changes in gap length are the same for the experimental and simulated devices and the resulting comparisons demonstrate good agreement.

The dynamic response of the magnetic subsystem is governed by the reluctance and eddy current dissipation of the magnetic circuit. Values for the magnetic R elements were obtained by comparing predicted and experimental coil current in response to step inputs and fixed armature positions. The application of the step input saturates the transistor, so that the rate of current rise in the coil is a function of the saturation voltage and the induced voltage in the coil. The causality of the system bond graph indicates that the induced voltage is due to the flow outputs from the magnetic R elements. Increasing the magnitude of these parameters leads to a steeper current rise, while decreasing the values slows the rate of current rise. Thus, these parameters can be adjusted to approximate the behavior of the actual system. The linearity assumption did limit the ability to precisely match the experimental behavior for all conditions. However, it was possible to approximate the response reasonably well over a significant range of operation.

4.2 Dynamic System Response to a Step Input

The coupling between the electrical, magnetic, and mechanical subsystems is evidenced when the armature is allowed to move in response to the magnetic force. The model was used to simulate the system in two cases for which experimental data was available. In each case the minimum air gap between the stator and armature faces was limited to 0.1080 millimeters(mm) by the upper stop. The maximum air gap was varied by adjusting the position of the lower stop to provide armature strokes of 0.10 mm and 0.15 mm. A step input of 10.7 amps was applied while the armature was at rest on the lower stop.

Figures 11 and 12 show comparisons of the predicted and experimental coil voltage and armature position versus time for the above conditions. The step input saturates the transistor, due to voltages induced in the coil, until approximately 0.6 milliseconds for both stroke values. The coil voltage decreases from 12.5 volts to nearly 10 volts over this interval due to the rising coil current and a resistor placed upstream from the measurement point. Consequently, the voltage in this region provides information on the coil current. When the coil current reaches the input level of 10.7 amperes, the voltage drops sharply, corresponding to transistor regulation.

The armature begins to move when the magnetic force overcomes the preload of the return spring. This motion reduces the reluctance of the air gap which leads to an increased flux rate in the magnetic circuit.

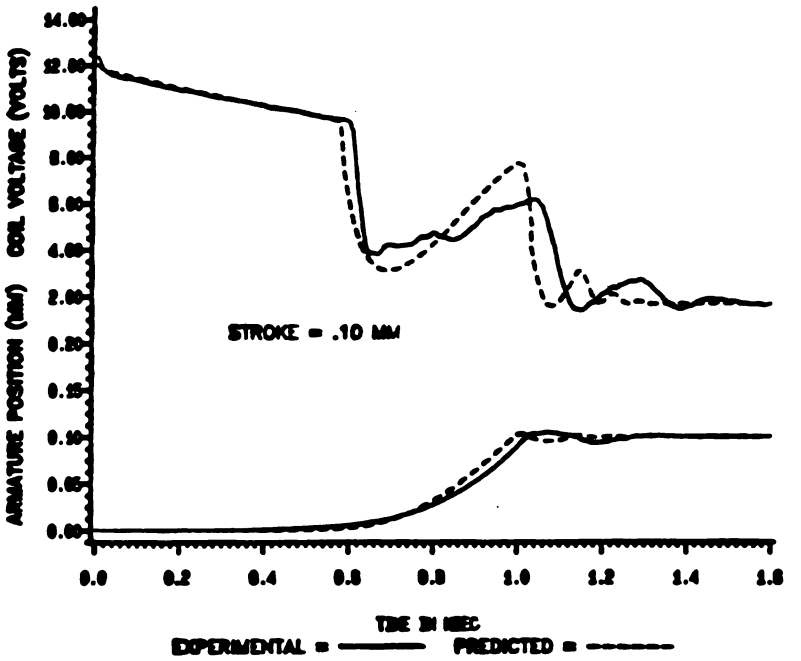


Figure 11. Dynamic Response to Step Input for Stroke = .10 mm

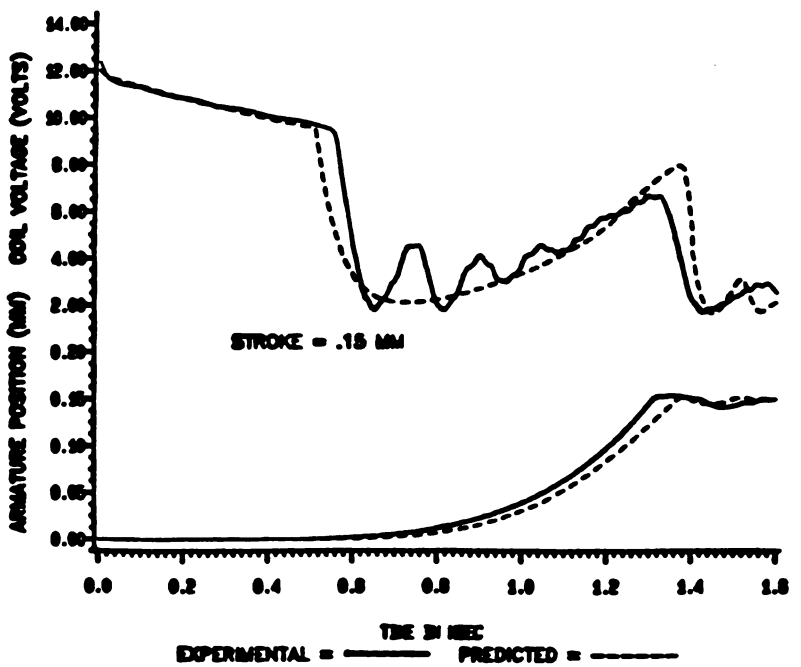


Figure 12. Dynamic Response to Step Input for Stroke = .15 mm

This is evidenced by the increase in coil voltage with armature motion.

The comparisons shown in Figures 11 and 12 demonstrate reasonable agreement between the predicted and experimental behavior. The simplified transistor model used in the simulation contributes to the discrepancies in the coil voltages. However, the general characteristics are predicted quite well. The coil voltage during armature motion is sensitive to the magnetic saturation of the stator and armature. Therefore, the effective lengths and areas defining the magnetic reluctances must be carefully estimated to obtain accurate results. The agreement between the predicted and experimental armature motion confirms the magnetic force and fluid damping equations. The predicted motion following the impact with the upper stop does not agree closely with experimental motion.

When the air gaps are increased beyond those of the typical operating range, the correlations are not as favorable. Although more sophisticated transistor and impact models could be developed, the results indicate sufficient agreement for most engineering purposes.

5 CONCLUSIONS

A lumped-parameter bond graph model for an electromagnetic actuator has been developed. In certain situations the lumped-parameter model may be preferable to the typical finite element approach. The lumped-parameter representation conserves computer resources, allows the dynamics of related devices to be included and is useful for control system design. However, more judgement is required to estimate values for certain of the lumped parameters.

The bond graph format aids the formation and implementation of the lumped-parameter model in several ways. First, bond graph models of the various subsystems involved can be linked together. Therefore, the same actuator model can be easily used with other electrical or mechanical models. Second, potential difficulties, such as nonlinear algebraic loops, are indicated during causal assignment. Finally, the coupled differential equations implied by the graph can be formulated automatically using computer algorithms.

A computer program, BGSIM, was written to implement the bond graph model. Modifications to the bond graph, such as alternative electrical or mechanical submodels, can also be implemented with this program. However, the user must code the required constitutive equations.

Comparisons of predicted and experimental results demonstrate good agreement over a significant range of operation.

LIST OF REFERENCES

REFERENCES

1. Colonias, J. S., Particle Acceleration Design: Computer Programs, Academic Press, New York, 1974.

2. Newman, M. J., Trowbridge, C. W., and Turner, L. R., "GFUN: An Interactive Program as an Aid to Magnet Design." Proc. Fourth International Conference on Magnet Technology, Brookhaven National Laboratory, 1972.

3. MacBain, J., "A Numerical Analysis of Time-Dependent Two-Dimensional Magnetic Fields", IEEE Transactions on Magnetics, Vol. MA6-17, No. 6, November, 1981.

4. Sabonnadiere, J., Meunier, G., and Murel, B., "FLUX: A General Interactive Finite Element Package for 2D Electromagnetic Fields", IEEE Transactions on Magnetics, Vol. MA6-18, No. 2, March, 1982.

5. Karnopp, D., and Rosenberg, R., Introduction to Physical System Dynamics, McGraw Hill, Inc., New York, 1983.

6. Nasar, S. A., Electromagnetic Energy Conversion Devices and Systems, Prentice Hall, New Jersey, 1970.

7. Matsch, L. W., Electromagnetic and Electromechanical Machines, Harper and Row Publishers, Inc., New York 1977.

8. Karidis, J. and Turns, S., "Fast-Acting Electromagnetic Actuators--Computer Model Development and Verification", SAE 820202, 1982.

9. Karnopp, D., and Rosenberg, R., System Dynamics: A Unified Approach, John Wiley and Sons, New York, 1975.

10. Zhou, T., "A Parallel Computation Method for Dynamic Systems with Coupled Nonlinear Dissipation", M.S. Thesis, Michigan State University, East Lansing, 1984.

11. Roters, H. C., Electromagnetic Devices, John Wiley and Sons, New York, 1941.

12. Rosenburg, R., ENPORT-5 User's Manual , Michigan State University, 1983.

13. Shampine, L. F., and Gordon, M. K., Computer Solution of Ordinary Differential Equations, the Initial Value Problem, W. H. Freeman and Co., San Francisco, 1975.

14. Conte, S., and DeBoor, C., Elementary Numerical Analysis, McGraw Hill, Inc., New York 1980.

15. The IMSL Library , International Mathematical & Statistical Libraries, Houston, Texas, 1982.

16. Millman, J., and Halkias, C., Integrated Electronics: Analog and Digital Circuits and Systems, McGraw Hill, Inc., New York, 1972.

APPENDIX A

R. C. ROSENBERG

Associate Professor, Department of Mechanical Engineering, Michigan State University, East Lansing, Mich.

D. C. KARNOPP

Professor, Department of Mechanical Engineering, University of California, Davis, Calif.

A Definition of the Bond Graph Language

Introduction

THE purpose of this paper is to present the basic definitions of the bond graph language in a compact but general form. The language presented herein is a formal mathematical system of definitions and symbolism. The descriptive names are stated in terms related to energy and power, because that is the historical basis of the multiport concept.

It is important that the fundamental definitions of the language be standardised because an increasing number of people around the world are using and developing the bond graph language as a modeling tool in relation to multiport systems. A common set of reference definitions will be an aid to all in promoting ease of communication.

Some care has been taken from the start to construct definitions and notation which are helpful in communicating with digital computers through special programs, such as ENPORT [5].[1] It is hoped that any subsequent modifications and extensions to the language will give due consideration to this goal.

Principal sources of extended descriptions of the language and physical applications and interpretations will be found in Paynter [1], Karnopp and Rosenberg [2, 3], and Takahashi, et al. [4]. This paper is the most highly codified version of language definition, drawing as it does upon all previous efforts.

Basic Definitions

Multiport Elements, Ports, and Bonds. *Multiport elements* are the nodes of the graph, and are designated by alpha-numeric characters. They are referred to as elements, for convenience. For example, in Fig. 1(*a*) two multiport elements, 1 and *R*, are shown. *Ports* of a multiport element are designated by line segments incident on the element at one end. Ports are places where the element can interact with its environment.

For example, in Fig. 1(*b*) the 1 element has three ports and the *R* element has one port. We say that the 1 element is a 3-port, and the *R* element is a 1-port.

Bonds are formed when pairs of ports are joined. Thus bonds are connections between pairs of multiport elements.

For example, in Fig. 1(*c*) two ports have been joined, forming a bond between the 1 and the *R*.

Bond Graphs. A *bond graph* is a collection of multiport elements bonded together. In the general sense it is a linear graph whose nodes are multiport elements and whose branches are bonds.

A bond graph may have one part or several parts, may have no loops or several loops, and in general has the characteristics of any linear graph.

An example of a bond graph is given in Fig. 2. In part (*a*) a bond graph with seven elements and six bonds is shown. In part (*b*) the same graph has had its powers directed and bonds labeled.

A *bond graph fragment* is a bond graph not all of whose ports have been paired as bonds.

An example of a bond graph fragment is given in Fig. 1(*c*), which has one bond and two open, or unconnected, ports.

Port Variables. Associated with a given port are three direct and three integral quantities.

Effort, $e(t)$, and *flow*, $f(t)$, are directly associated with a given port, and are called the port power variables. They are assumed to be scalar functions of an independent variable (t).

Power, $P(t)$, is found directly from the scalar product of effort and flow, as

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

The direction of positive power is indicated by a half-arrow on the bond.

Momentum, $p(t)$, and *displacement*, $q(t)$, are related to the effort and flow at a port by integral relations. That is,

[1] Numbers in brackets designate References at end of paper.

Contributed by the Automatic Control Division for publication (without presentation) in the JOURNAL OF DYNAMIC SYSTEMS, MEASUREMENT, AND CONTROL. Manuscript received at ASME Headquarters, May 9, 1972. Paper No. 72-Aut-T.

Copies will be available until September, 1973.

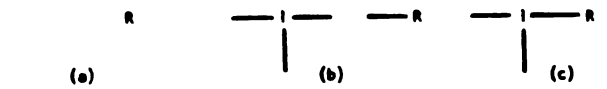


Fig. 1 Multiport elements, ports, and bonds: (a) two multiport elements; (b) the elements and their ports; (c) formation of a bond

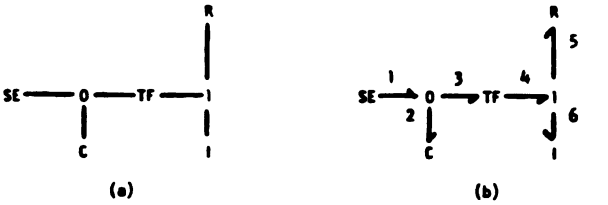


Fig. 2 An example of a bond graph: (a) a bond graph; (b) the bond graph with powers directed and bonds labeled

$$p(t) = p(t_0) + \int_{t_0}^{t} e(\lambda)d\lambda$$

and $q(t) = q(t_0) + \int_{t_0}^{t} f(\lambda)d\lambda$, respectively.

Momentum and displacement are sometimes referred to as energy variables.

Energy, $E(t)$, is related to the power at a port by

$$E(t) = E(t_0) + \int_{t_0}^{t} P(\lambda)d\lambda.$$

The quantity $E(t) - E(t_0)$ represents the net energy transferred through the port in the direction of the half-arrow (i.e., positive power) over the interval (t_0, t).

In common bond graph usage the effort and the flow are often shown explicitly next to the port (or bond). The power, displacement, momentum, and energy quantities are all implied.

Basic Multiport Elements. There are nine basic multiport elements, grouped into four categories according to their energy characteristics. These elements and their definitions are summarized in Fig. 3.

Sources.

Source of effort, written $SE\ e$, is defined by $e = e(t)$.

Source of flow, written $SF\ f$, is defined by $f = f(t)$.

Storages.

Capacitance, written $\frac{e}{f}\ C$, is defined by

$$e = \Phi(q) \text{ and } q(t) = q(t_0) + \int_{t_0}^{t} f(\lambda)d\lambda.$$

That is, the effort is a static function of the displacement and the displacement is the time integral of the flow.

Inertance, written $\frac{e}{f}\ I$, is defined by

$$f = \Phi(p) \text{ and } p(t) = p(t_0) + \int_{t_0}^{t} e(\lambda)d\lambda.$$

That is, the flow is a static function of the momentum and the momentum is the time integral of the effort.

Dissipation.

Resistance, written $\frac{e}{f}\ R$, is defined by

$$\Phi(e, f) = 0.$$

SYMBOL	DEFINITION	NAME
SE —e→	$e = e(t)$	source of effort
SF —f→	$f = f(t)$	source of flow
C (e/f)	$e = \Phi(q)$	capacitance
	$q(t) = q(t_0) + \int f \cdot dt$	
I (e/f)	$f = \Phi(p)$	inertance
	$p(t) = p(t_0) + \int e \cdot dt$	
R (e/f)	$\Phi(e, f) = 0$	resistance
1 —TF 1:m→ 2	$e_1 = m \cdot e_2$	transformer
	$m \cdot f_1 = f_2$	
1 —GY r→ 2	$e_1 = r \cdot f_2$	gyrator
	$e_2 = r \cdot f_1$	
1 — 0 — 3, 2	$e_1 = e_2 = e_3$	common effort junction
	$f_1 + f_2 - f_3 = 0$	
1 — 1 — 3, 2	$f_1 = f_2 = f_3$	common flow junction
	$e_1 + e_2 - e_3 = 0$	

Fig. 3 Definitions of the basic multiport elements

That is, a static relation exists between the effort and flow at the port.

Junctions: 2-Port.

Transformer, written $\frac{e_1}{f_1}\ TF\ \frac{e_2}{f_2}$, is a linear 2-port element defined by

$$e_1 = m \cdot e_2$$

and $$m \cdot f_1 = f_2,$$

where m is the modulus.

Gyrator, written $\frac{e_1}{f_1}\ GY\ \frac{e_2}{f_2}$, is a linear 2-port element defined by

$$e_1 = r \cdot f_2$$

and $$e_2 = r \cdot f_1,$$

where r is the modulus.

Both the transformer and gyrator preserve power (i.e., $P_1 = P_2$ in each case shown), and they must each have two ports, so they are called essential 2-port junctions.

Junctions: 3-Port.

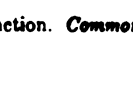
Common effort junction, written $\xrightarrow{1}\ 0\ \xrightarrow[2]{3}$ (1 at 2)

is a linear 3-port element defined by

$$e_1 = e_2 = e_3 \qquad \text{(common effort)}$$

and $$f_1 + f_2 - f_3 = 0. \qquad \text{(flow summation)}$$

Other names for this element are the *flow* junction and the

zero junction. *Common flow junction*, written  , is a linear 3-port element defined by

$$f_1 = f_2 = f_3 \qquad \text{(common flow)}$$

and

$$e_1 + e_2 - e_3 = 0. \qquad \text{(effort summation)}$$

Other names for this element are the *effort* junction and the *one* junction.

Both the common effort junction and the common flow junction preserve power (i.e., the *net* power in is zero at all times), so they are called junctions. If the reference power directions are changed the signs on the summation relation must change accordingly.

Extended Definitions

Multiport Fields.

Storage Fields. *Multiport capacitances*, or *C-fields*, are written , and characterised by

$$e_i = \Phi_i(q_1, q_2, \ldots q_n),\ i = 1 \text{ to } n,$$

and $q_i(t) = q_i(t_0) + \int_{t_0}^{t} f_i(\lambda)d\lambda,\ i = 1 \text{ to } n.$

Multiport inertances, or *I-fields*, are written ,

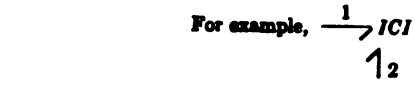
and characterized by

$$f_i = \Phi_i(p_1, p_2, \ldots p_n),\ i = 1 \text{ to } n,$$

and $p_i(t) = p_i(t_0) + \int_{t_0}^{t} e_i(\lambda)d\lambda,\ i = 1 \text{ to } n.$

If a *C*-field or *I*-field is to have an associated "energy" state function then certain integrability conditions must be met by the Φ_i functions. In multiport terms the relations given in the foregoing are sufficient to define a *C*-field and *I*-field, respectively.

Mixed multiport storage fields can arise when both *C* and *I*-type storage effects are present simultaneously. The symbol for such an element consists of a set of *C*'s and *I*'s with appropriate ports indicated.

For example,  indicates the existence of a set of relations

$$f_1 = \Phi_1(p_1, q_2, p_3),$$

$$e_2 = \Phi_2(p_1, q_2, p_3),$$

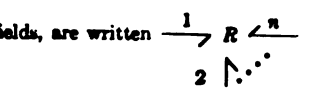
$$f_3 = \Phi_3(p_1, q_2, p_3),$$

and

$$p_1(t) = p_1(t_0) + \int_{t_0}^{t} e_1(\lambda)d\lambda,$$

$$q_2(t) = q_2(t_0) + \int_{t_0}^{t} f_2(\lambda)d\lambda,$$

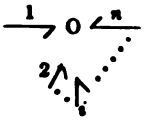
$$p_3(t) = p_3(t_0) + \int_{t_0}^{t} e_3(\lambda)d\lambda.$$

Multiport dissipators, or *R*-fields, are written  , and are characterised by

$$\Phi_i(e_1, f_1, e_2, f_2, \ldots e_n, f_n) = 0,\ i = 1 \text{ to } n.$$

If the *R*-field is to represent pure dissipation, then the power function associated with the *R*-field must be positive definite.

Multiport junctions include 0 junctions and 1 - junctions with n ports, $n \geq 2$. The general case for each junction is given in the following.



$$e_1 = e_2 = \ldots = e_n \qquad f_1 = f_2 = \ldots = f_n$$

$$\sum_{i=1}^{n} f_i = 0 \qquad \sum_{i=1}^{n} e_i = 0$$

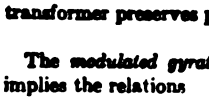
Modulated 2-Port Junctions. The *modulated transformer*, or *MTF* written  implies the realtions

$$e_1 = m(x) \cdot e_2$$

and

$$m(x) \cdot f_1 = f_2,$$

where $m(x)$ is a function of a set of variables, x. The modulated transformer preserves power; i.e., $P_1(t) = P_2(t)$.

The *modulated gyrator*, or *MGY*, written  implies the relations

$$e_1 = r(x) \cdot f_2$$

and

$$e_2 = r(x) \cdot f_1,$$

where $r(x)$ is a function of set of variables, x. The modulated gyrator preserves power; i.e., $P_1(t) = P_2(t)$.

Junction Structure. The *junction structure* of a bond graph is the set of all 0, 1, *GY*, and *TF* elements and their bonds and ports. The junction structure is an n-port that preserves power (i.e., the *net* power in is zero). The junction structure may be modulated (if it contains any *MGY*'s or *MTF*'s) or unmodulated.

For example, the junction structure of the graph in Fig. 2(*b*) is a 4-port element with ports 1, 2, 5, and 6 and bonds 3 and 4. It contains the elements 0, *TF*, and 1.

Physical Interpretations

The physical interpretations given in this section are very succinctly stated. References [1], [2], and [3] contain extensive descriptions of physical applications and the interested reader is encouraged to consult them.

Mechanical Translation. To represent mechanical translational phenomena we may make the following variable associations:

1 *effort*, e, is interpreted as *force;*
2 *flow*, f, is interpreted as *velocity;*
3 momentum, p, is interpreted as impulse-momentum;
4 displacement, q, is interpreted as mechanical displacement.

Then the basic bond graph elements have the following interpretations:

1 source of effort, *SE*, is a force source;
2 source of flow, *SF*, is a velocity source (or may be thought of as a geometric constraint);

3 resistance, R, represents friction and other mechanical loss mechanisms;

4 capacitance, C, represents potential or elastic energy storage effects (or spring-like behavior);

5 inertance, I, represents kinetic energy storage (or mass effects);

6 transformer, TF, represents linear lever or linkage action (motion restricted to small angles);

7 gyrator, GY, represents gyrational coupling or interaction between two ports;

8 0-junction represents a common force coupling among the several incident ports (or among the ports of the system bonded to the 0-junction); and

9 1-junction represents a common velocity constraint among the several incident ports (or among the ports of the system bonded to the 1-junction).

The extension of the interpretation to rotational mechanics is a natural one. It is based on the following associations:

1 effort, e, is associated with torque; and

2 flow, f, is associated with angular velocity.

Because the development is so similar to the one for translational mechanics it will not be repeated here.

Electrical Networks. In electrical networks the key step is to interpret a port as a terminal-pair. Then variable associations may be made as follows:

1 *effort*, e, is interpreted as *voltage*;

2 *flow*, f, is interpreted as *current*;

3 momentum, p, is interpreted as flux linkage;

4 displacement, q, is interpreted as charge.

The basic bond graph elements have the following interpretations:

1 source of effort, SE, is a voltage source;

2 source of flow, SF, is a current source;

3 resistance, R, represents electrical resistance;

4 capacitance, C, represents capacitance effect (stored electric energy);

5 inertance, I, represents inductance (stored magnetic energy);

6 transformer, TF, represents ideal transformer coupling;

7 gyrator, GY, represents gyrational coupling;

8 0-junction represents a parallel connection of ports (common voltage across the terminal pairs); and

9 1-junction represents a series connection of ports (common current through the terminal pairs).

Hydraulic Circuits. For fluid systems in which the significant fluid power is given as the product of pressure times volume flow, the following variable associations are useful:

1 *effort*, e, is interpreted as *pressure*;

2 *flow*, f, is interpreted as *volume flow*.

3 momentum, p, is interpreted as pressure-momentum;

4 displacement, q, is interpreted as volume.

The basic bond graph elements have the following interpretations:

1 source of effort, SE, is a pressure source;

2 source of flow, SF, is a volume flow source;

3 resistance, R, represents loss effects (e.g., due to leakage, valves, orifices, etc.);

4 capacitance, C, represents accumulation or tank-like effects (head storage);

5 inertance, I, represents slug-flow inertia effects;

6 0-junction represents a set of ports having a common pressure (e.g., a pipe tee);

7 1-junction represents a set of ports having a common volume flow (i.e., series).

Other Interpretations. This brief listing of physical interpretations of bond graph elements is restricted to the simplest, most direct, applications. Such applications came first by virtue of historical development, and they are a natural point of departure for most classically trained scientists and engineers. As references [1–4] and the special issue collection in the JOURNAL OF DYNAMIC SYSTEMS, MEASUREMENT, AND CONTROL, TRANS. ASME, Sept. 1972, indicate, bond graph elements can be used to describe an amazingly rich variety of complex dynamic systems. The limits of applicability are not bound by energy and power in the sense of physics; they include any areas in which there exist useful analogous quantities to energy.

Concluding Remarks

In this brief definition of the bond graph language two important concepts have been omitted. The first is the concept of *bond activation*, in which one of the two power variables is suppressed, producing a pure signal coupling in place of the bond. This is very useful modeling device in active systems. Further discussion of activation will be found in reference [3], section 2.4, as well as in references [1] and [2].

Another concept omitted from discussion in this definitional paper is that of *operational causality*. It is by means of causality operations applied to bond graphs that the algebraic and differential relations implied by the graph and its elements may be organized and reduced to state-space form in a systematic manner. Extensive discussion of causality will be found in reference [3], section 3.4 and chapter 5. Systematic formulation of relations is presented in reference [6].

References

1 Paynter, H. M., *Analysis and Design of Engineering Systems*, M.I.T. Press, 1961.

2 Karnopp, D. C., and Rosenberg, R. C., *Analysis and Simulation of Multiport Systems*, M.I.T. Press, 1968.

3 Karnopp, D. C., and Rosenberg, R. C., "System Dynamics: A Unified Approach," Division of Engineering Research, College of Engineering, Michigan State University, East Lansing, Mich., 1971.

4 Takahashi, Y., Rabins, M., and Auslander, D., *Control*, Addison-Wesley, Reading, Ma., 1970 (see chapter 6 in particular).

5 Rosenberg, R. C., "ENPORT User's Guide," Division of Engineering Research, College of Engineering, Michigan State University, East Lansing, Mich., 1972.

6 Rosenberg, R. C., "State-Space Formulation for Bond Graph Models of Multiport Systems," JOURNAL OF DYNAMIC SYSTEMS, MEASUREMENT, AND CONTROL, TRANS. ASME, Series G, Vol. 93, No. 1, Mar. 1971, pp. 35-40.

Reprinted from September 1972 Journal of Dynamic Systems, Measurement, and Control

Printed in U.S.A.

APPENDIX B

Appendix B: Magnetic Reluctance Calculations

The reluctance calculations for the magnetic subsystem involve the 10 sub-elements shown in Figure B1.

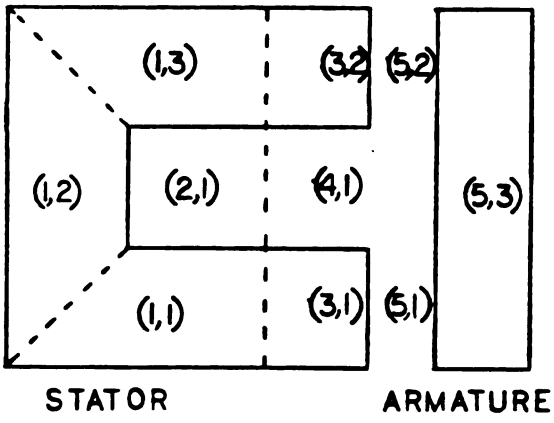


Figure B1: Magnetic Reluctance Sub-Elements

The first subscript identifies the flux path; the second subscript distinguishes the sub-elements with common flux paths.

Reluctance of Ferromagnetic Sub-Elements

Sub-elements (1,1), (1,2), (1,3), (3,1), (3,2), (5,3) represent ferromagnetic materials. An effective length, cross-sectional area, and flux is identified for each of these sections. The reluctance of each section is calculated from equation B1.

$$Rm = l/A\mu \qquad \text{(B1)}$$

The permeability, μ, is a nonlinear function of the flux density, B. The relationship between the permeability and flux density is obtained from B-H curves for each of the materials involved. The permeability can be found from these curves using equation B2.

$$\mu = B/H \qquad \text{(B2)}$$

Cubic spline equations are then used to provide expressions for μ in terms of B for each material specified by the user. The flux and area associated with each sub-element is used to compute the flux density from equation B3.

$$B = \phi/A \qquad \text{(B3)}$$

The permeability is found by evaluating the corresponding cubic spline equation at this flux density.

The flux used to calculate B for each section is found from the state vector, Y. The flux stored in the five paths of Figure B1 correspond to the first five components of this vector. Table B1 shows the flux quantity used for the permeability derivation for each sub-element.

The flux, Y(1), is not used for sections (1,1) and (1,3) because of the distributed nature of the leakage flux, Y(2). Roters(11) suggests the approximations shown in Table B1 for these situations. Sub-elements (3,1) and (3,2) are treated in a similar fashion.

Table B1 Flux Quantities Used for Permeability Derivations

Sub-Element	Flux Quantity
(1,1)	Y(3) + 2/3 Y(2)
(1,2)	Y(1)
(1,3)	Y(3) + 2/3 Y(2)
(3,1)	Y(5) + 2/3 Y(4)
(3,2)	Y(5) + 2/3 Y(4)
(5,3)	Y(5)

Reluctance of Air Gaps

Sub-elements (2,1), (4,1), (5,1) and (5,2) of Figure B1 represent air gaps. The reluctance calculations for the air gaps differ from the ferromagnetic components in three ways: i) the permeability is constant with respect to the flux density, ii) the physical dimensions change due to armature motion, and iii) the flux distributed around the edges of the air gaps, called fringing flux, must be taken into account. Since the fringing flux paths are in parallel, it is easier to work with permeances because the net permeance of the gap is the sum of the individual permeances. The fringing flux becomes increasingly significant as the air gap length increases.

The flux paths for the leakage air gaps are summarized in Figure B2.

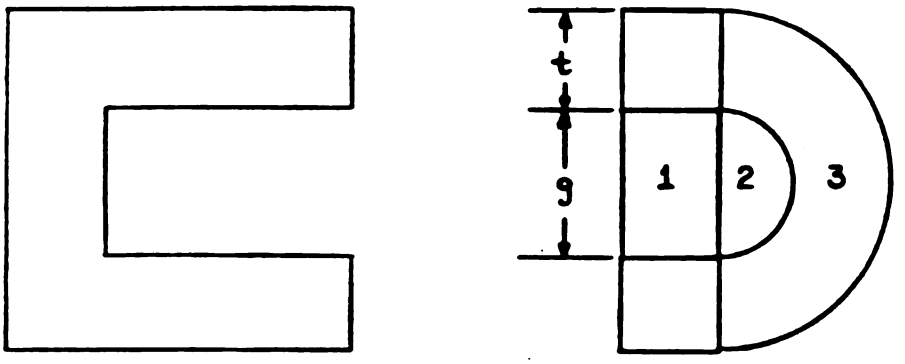


Figure B2: Leakage Flux Paths

The permeance of each path is given by the equations below. The permeance of paths 2 and 3 is doubled since the identical path exists on the opposite side.

$$Pm_1 = \frac{w d \mu_o}{g} \qquad \text{(B3a)}$$

$$Pm_2 = 2\ (.26\ \mu_o d) \qquad \text{(B3b)}$$

$$Pm_3 = 2\ \frac{\mu_o d}{\pi}\ \ln\left(\frac{2t + g}{g}\right) \qquad \text{(B3c)}$$

where d is the depth into the page.

The flux paths for the working air gaps are summarized in Figure B3. It is assumed that the length-width dimensions of the armature are less than or equal to those of the stator. The air gaps shown in Figure B3 are greatly exaggerated.

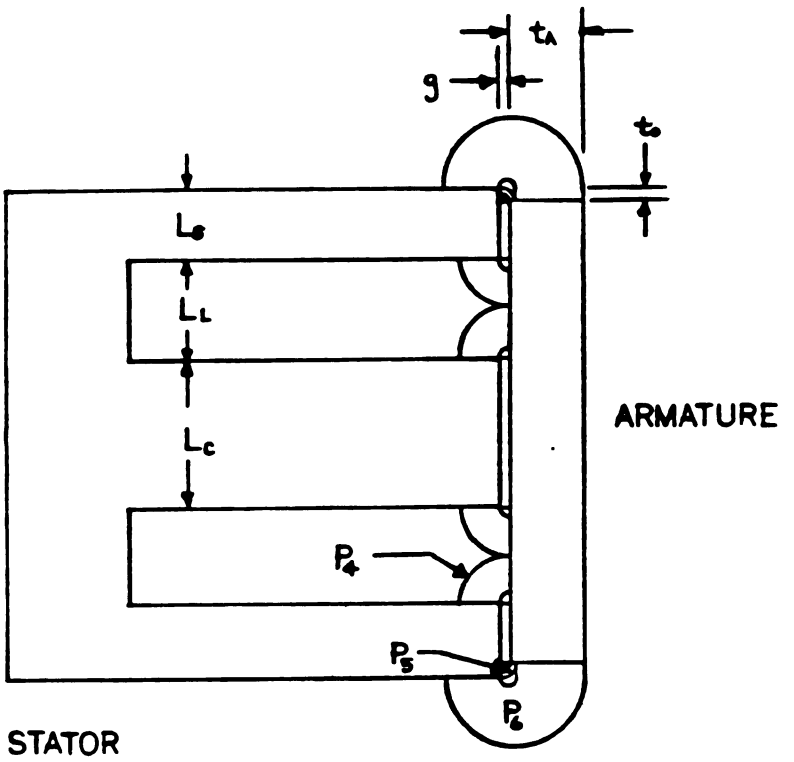


Figure B3: Working Air Gap Flux Paths

The main permeance of each gap is given by equation B4

$$Pm = A\mu_o/g \qquad (B4)$$

where A is the projected area of the stator leg on the armature. Rotors(11) derives the general permeance equation for the circular flux paths of Figure B3 as

$$Pm = \frac{\mu_o d}{\theta} \ln (r_o / r_i) \qquad (B5)$$

where θ is the angle thru which the flux travels, d is a length normal to the gap and r_i, r_o are the inner and outer radii, respectively. The permeance of flux paths P_4, P_5 and P_6 of Figure B3 are calculated using equation (B5).

$$Pm_4 = \frac{2\mu_o d}{\pi} \ln (1/2g) \qquad (B6)$$

$$Pm_5 = \frac{2\mu_o d}{\pi} \ln (t_o/g) \qquad (B7)$$

$$Pm_6 = \frac{\mu_o d}{\pi} \ln\left(\frac{2(t_a + g)}{t_o} - 1\right) \qquad (B8)$$

The length, d, in equations (B6), (B7) and (B8) is the depth into the page of Figure B3.

The sum of the main and fringing permeances is the total permeance for each gap. Note that the fringing permeances exist around all four edges of the legs. Since the working air gaps are usually very small, some of the less significant flux paths suggested by Rotors(11) are neglected. For larger gaps it may be necessary to include the permeances of these paths as well.

APPENDIX C

The program is written in Fortran IV for use on an IBM 3083 computer under a "VM" environment. Post processing of the results is carried out using separate routines which must be supplied by the user. The process can be automated using an "EXEC" file.

BGSIM Inputs

The main input file for BGSIM is shown in Figure C1. The file is organized according to the bond graph energy fields and contains the information necessary to initialize the parameters and carry out the integration. The subroutines which read each block of data are listed first in the major headings. Numerical inputs are placed directly below descriptive headings to aid subsequent parameter changes. The definitions for the input variables are given below.

TERMINAL	Defines the logical unit number for I/O to the terminal.
OUTPUT	Defines the logical unit number of the output file.
JS MATRIX	Defines the logical unit number of the junction structure matrix for input or output.
LLIMIT	Lower time limit of integration.
ULIMIT	Upper time limit of integration.
DELTA	Integration step size. The integration routine may reduce the step size depending on the local behavior of the system.
NY	The number of equations to be integrated.
NAUX	Number of auxiliary variables to be output.
NDPTS	Number of output intervals between LLIMIT, ULIMIT

```
                    DESCRIPTION LINE #1
                    DESCRIPTION LINE #2
                    DESCRIPTION LINE #3
                 LOGICAL UNIT DEFS FOR I/O
TERMINAL      OUTPUT      JS MATRIX
  10            6            7              /
               INTEGRATION PARAMETERS
 LLIMIT      ULIMIT      DELTA       NY      NAUX      NDPTS
  0.          2.25E-3     5.0E-6       7       10        300   /
                 SYSTEM SIZE PARAMETERS
  NZ     NU      NDNL      NDL      NT     NROOTS     JS
   8      1        2         4        2       4         1      /
                   INITIAL CONDITIONS
   Y(1)         Y(2)         Y(3)         Y(4)         Y(5)  Y(6)     Y(7)
  0.           0.           0.           0.           0.     0.       0. /
ZVECTR ---------- ENERGY STORAGE PARAMETERS ------------------------
NMATL    LUMATL     SEC(1,1)    SEC(1,2)     SEC(1,3)     SEC(5,3)
  4         1           1           2            1            3 /
                  STATOR DIMENSIONS #1 (MM)
  MS       ML       LCL        LSL        LSG     HLK
33.84    26.34    10.5        7.98      7.46     4.59  /
                  STATOR DIMENSIONS #2 (MM)
 TLAM         SF        NLAMS
 .3356       .92          80   /
                 ARMATURE DIMENSIONS (MM)
 LA       WA         TA
36.0     27.3       5.35       /
                    VALVE SUBSYSTEM          (N/M)      (KG)
AGMIN(MM)  STROKE(MM) PRELD(N)   KSTOPS     KSPRING      VMASS
.10          .10          47.      5.0E8        2.E4       .038  /
UVECTR ---------- ENERGY SOURCE PARAMETERS ---------------------------
CURRENT LEVEL (AMPS)      PULSEWIDTH(SEC)
  11.00                      1.25E-3                                  /
DVECTR ------------ DISSIPATION PARAMETERS ---------------------------
NONLINEAR DISSIPATION PARAMETERS
R10: VZENR        VSAT
       -45.        12.5                  /
R11: DENSITY  ARM-R     SHAFT-R     VISCOSITY  R-LSTOP    R-USTOP
    733.2     15.0E-3   2.4E-3      3.34E-3     1.3E0      4.0E3 /
LINEAR RESISTANCE PARAMETERS FOR EDDY CURRENT LOSSES
R12-R14:       R12                 R13                      R14
               35.0                25.0                     100.   /
R15: RCOIL  LEAD WIRE: GAGE  LENGTH(M)      RCIRC
       .09                18.     2.0          .1 /
TVECTR ---------- TF-GY ELEMENT DATA --------------------------------
GY14,15 MODULUS   GY16,17 MODULUS
    37.5                20.3        /
JS  ---------------- BOND GRAPH DESCRIPTION -------------------------
  1 12 0 2 12 -13 0 3 13 0 4 13 -14 0 /
  5 14 0 6 7 0 7 -6 -8 -11 0 /
  8 7 0 9 15 16 18 0 10 15 16 18 0 /
  11 7 0 12 17 -1 -2 0 13 19 2 -3 -4 0 /
  14 4 -5 0 15 9 -10 0 16 9 -10 0 17 12 0 18 9 -10 0 19 13 0 /
```

Figure C1. Main Input File

NZ — Number of I or C bond graph ports. Defines the length of the complimentary state vector, Z.

NU — Number of Sources. Defines the length of the U, V vectors.

NDNL — Number of nonlinear dissipation elements. Defines the length of the Di, Do vectors.

NDL — Number of linear dissipation elements. Defines the size of the associated constitutive matrix.

NT — Number of transformer-gyrator two-port elements.

NROOTS — Number of root functions to be evaluated in subroutine RCHK.

JS — Flag for the junction structure matrix
- 0: Compute the junction structure and continue into the integration.
- 1: Compute and output the matrix only.
- 2: Read an existing matrix and continue into the integration.

Y(1)-Y(7) — Initial conditions for the state vector.

NMATL — Number of B-H curves for which cubic splines are to be computed.

LUMATL — Defines the logical unit number corresponding to the B-H data for input. The order of the B-H curves in this file determines how the materials are identified. The first B-H curve is called material 1, the second B-H curve is called material 2, and so on. A sample of this input file is shown in Figure C2.

```
12
0. 350.
.2 .025
2.6 .1
4. .12
8. .18
10. .21
12. .28
14. .425
16. 1.0
 :   :
```

Figure C2: Sample B-H Input

	The first entry in this file is the number of B-H data pairs to be input. The maximum number is set at 20 data pairs. The next line is the relative initial permeability corresponding to zero flux density. Rotors(11) gives typical values for several materials. The remaining entries are the B-H data in units of Oersteds and Kiloguass.
SEC(1,1)-SEC(5,3)	Defines the materials of the stator and armature. The input numbers correspond to the order of the B-H curves in the material input file.
STATOR DIMENSIONS	Refer to Figure C3 for the definitions of these dimensions (MM).
SF	Stacking factor of the stator laminations.
NLAMS	Number of laminations in the stator.
ARMATURE DIMENSIONS	Refer to Figure C3 for the definitions of these dimensions (mm).

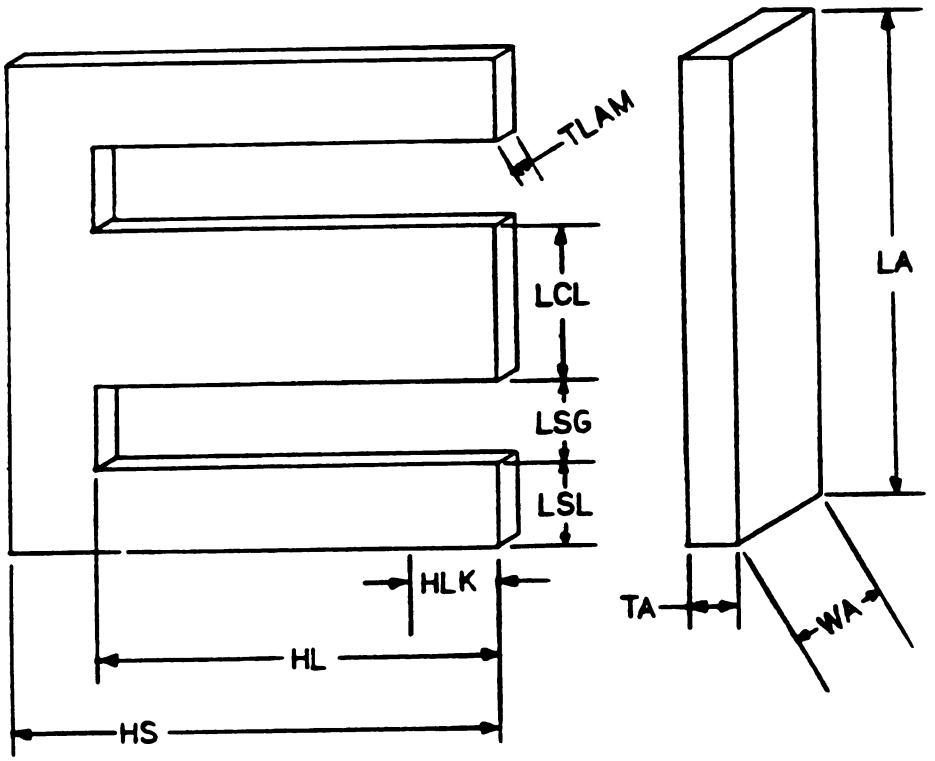


Figure C3. Stator-Armature Dimensions

AGMIN	Minimum working air gap (mm). Occurs when the armature is at the upper stop.
STROKE	The distance between the lower and upper stops (mm). Therefore, the maximum air gap is AGMIN + STROKE.
PRELD	Return spring preload, armature at the lower stop (newtons).
KSTOP	Linear spring rate for upper and lower stops (N/M)
KSPRING	Spring rate of the return spring (N/M).
VMASS	Mass of the armature and guide shaft (Kg).
CURRENT LEVEL	Defines the magnitude of the step current input (amps)
PULSEWIDTH	Defines the time length of the step input. The input current is set to zero after this period of time (sec).
VZENR	Breakdown voltage of the zener diode (volts).
VSAT	Saturation voltage of the transistor (volts).
DENSITY	Density of the fluid surrounding the armature (kg/m).
ARM-R	Effective radius of the armature (m).
SHAFT-R	Effective radius of the guide shaft (m).
VISCOSITY	Viscosity of the fluid surrounding the armature (kg/ms).
R-LSTOP, R-USTOP	Linear damping coefficients to model the impact of the guide shaft with the upper and lower stops (Ns/M).
R12-R14	Eddy current resistance coefficients for elements R12, R13 and R14 of the system bond graph.
RCOIL	Resistance of the wire coil in the stator (ohms).
GAGE, LENGTH	The gage and length of the wire connecting the electrical driver and the actuator.
RCIRC	Represents any other resistances in the electrical circuit (ohms).
GY14,15, GY16,17	Linear gyrator moduli for the system bond graph.

The final lines of input describe the bond graph structure so that the simple junction structure matrix can be formed. The rules organizing this input are given as follows: First, assign causality to the bond graph. Next, number all bonds attached to I, C, R, GY, TF, SF, and SE elements according to the priority of Table C1. The numbering sequence within each priority group is arbitrary. The particular sequency used, however, determines the ordering of the components within the key vectors. This order must be followed when coding the corresponding constitutive equations and matrices.

Table C1 Bond Numbering Priority, Key Vector Definitions

ELEMENTS	PRIORITY	KEY VECTORS	SUBROUTINE
I,C	1	Y,Z	ZVECTR
SE,SF	2	V,U	UVECTR
R (Nonlinear)	3	Di,Do	DVECTR
R (Linear)	4	Di ,Do	DVECTR
TF,GY	5	Ti,To	TVECTR

Third, the inputs to each of these elements must be expressed in terms of the outputs from the remaining elements based on causality, power directions and the 0,1 junction laws. Finally, the input bond numbers are listed in ascending order followed by a list of the output numbers which define them. The input-output groups are separated by zeros and a slash is used to indicate the end of the input line. For example, in the bond graph description of Figure C1, the input on bond 1 is defined in terms of the output from bond 12. A zero follows to indicate the end of the data for bond 1. The input on bond 2 is determined by the output from bond 12 minus the output from bond 13.

These input-output groups are continued until the final input, bond 19, is defined in terms of the output from bond 13.

BGSIM OUTPUTS

At each output interval the state vector is written to the output file. The output vectors from the source, storage and nonlinear dissipation elements as well as a vector of auxiliary variables is also written to this file. Post-processing of this data for plotting and printing is performed with a separate software package. The output operation is performed in subroutine OUTPT.

Modifications

Modifications to the system bond graph require changes to the appropriate subroutines of BGSIM. Table C1 lists the subroutines corresponding to each of the bond graph elements. The constitutive equations are coded in these subroutines and the sequence used when the bonds are numbered determines the ordering of these equations. Changes can also be made to the input file to initialize the additional parameters.

APPENDIX D

APPENDIX D: Subroutine Calling Tree and Definitions

Subroutine Calling Tree

```
BGSIM
. ZVECTR
. . COEFF
. UVECTR
. DVECTR
. . ZMAT
. TVECTR
. . ZMAT
. JS
. . ZMAT
. . PRTITN
. . REDUCT
. . . VMULFF
. . . LINV2F
. RCHK
. . ZVECTR
. . . AIRREL
. . . METREL
. . UVECTR
. . VECMUL
. OUTPUT
. . FCT
. . . ZVECTR
. . . . AIRREL
. . . . METREL
. . . UVECTR
. . . DVECTR
. . . . VECMUL
. . . VECMUL
. ODERT
. . FCT
. . . ZVECTR
. . . . AIRREL
. . . . METREL
. . . UVECTR
. . . DVECTR
. . . . VECMUL
. . . VECMUL
. . RCHK
. . . ZVECTR
. . . . AIRREL
. . . . METREL
. . . UVECTR
. . . VECMUL
```

SUBROUTINE DEFINITIONS

AIRREL	Computes the reluctance and the rate of change of reluctance with gap length of the nonlinear air gaps.
BGSIM	Main calling program.
COEFF	Forms cubic splines of DC magnetization curves for up to 4 materials. Based on code in reference(14).
DVECTR	Inputs dissipation parameters and forms the constitutive matrix for the linear dissipation elements on initialization. Computes the input-output vectors from the nonlinear dissipation field during integration.
FCT	Computes the time derivative of the state vector, Y.
JS	Reads the bond graph structure and forms the junction structure matrix, S1. The linear key vectors are also eliminated.
LINVZF(15)	Matrix inversion.
METREL	Calculates the flux densities in ferromagnetic sections of the stator and armature and outputs corresponding reluctances using the cubic spline magnetization curves.
ODERT	Integration package.
OUTPT	Writes results to the output file.
PRTITN	Partitions the junction structure matrix for elimination of linear key vectors.
RCHK	Contains root functions which determine switching points for discontinuous parameters.
REDUCT	Uses the partitioned junction structure matrices with a constitutive matrix to eliminate linear key vectors.
TVECTR	Inputs gyrator moduli and forms the linear constitutive matrix on initialization.
VECMUL	Vector-matrix multiplication.
UVECTR	Inputs source parameters on initialization. Computes the source vector during integration.

VMULFF(15)	Matrix multiplication.
ZMAT	Initializes matrices.
ZVECTR	Inputs energy storage parameters on initialization. Computes the complementary state vector, Z, during integration.

APPENDIX E

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC                                                                      CC
CC-- NAME: BGSIM - MAIN CALLING PROGRAM                                 CC
CC                                                                      CC
CC-- FUNCTIONS:  1. INITIALIZES INTEGRATION PARAMETERS.                 CC
CC               2. PROVIDES INTEGRATION OF THE STATE EQUATIONS AND     CC
CC                  OUTPUT TO SPECIFIED FILE.                           CC
CC               3. HANDLES ERROR FLAGS FROM THE INTEGRATION ROUTINE.   CC
CC                                                                      CC
CC-- SUBROUTINES CALLED: ODERT, OUTPT, ZVECTR, UVECTR, DVECTR, TVECTR   CC
CC                       JS, RCHK                                       CC
CC                                                                      CC
CC-- PROGRAMMER: N. HENDRIKSMA   1984                                   CC
CC                                                                      CC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      COMMON /FLAG/ IBEGIN,JSOPT
      COMMON /LUDEF/ LUSN,LUIN,LUOT,LUJS
      COMMON /STATUS/ ISTAT(16),NROOTS
      COMMON /SYSTEM/ NZ,NU,NDNL,NDL,NT,NY,NZPNU,N1,N1MX,N2MX,
     +                ZOUT(15),UOUT(5),DOUT(10),AUX(10),
     +                S1(25,25),JS1D(25,10)
C
      DIMENSION Y(10),ABSERR(10),RELERR(10),WORK(310),IWORK(5),IMIN(16)
     +          ,GFT1(16),GFT2(16),IRT(16),JRT(16),TRT(16),IRFLAG(16),
     +          YPRIME(10)
C
C-- DIMENSION OF WORK(100 + 21*NY)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C------ INITIALIZE FOR THE INTEGRATION
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IFLAG = 1
      JCBN = 0
      ISTUCK = 1
      EPS = 1.0D0
      TMULT = 10.0D0
      IBEGIN = 1
      RERR = 1.D-9
      AERR = 1.D-10
      MAXNUM = 5000
C
C-- N1MX = NZ+NU+NDNL+NDL+NT*2 = MAX SIZE OF S1 BEFORE REDUCTION
C   N2MX = MAX OF (NDL OR NT*2) = MAX SIZE OF TM OR RL MATRICES
```

```
      N1MX = 25
      N2MX = 10
C
C-- READ THE LU DEFINITIONS FOR I/O     (LUIN IS FOR THE MAIN INPUT )
      LUIN = 5
      READ(LUIN,10)
      READ(LUIN,10)
      READ(LUIN,11)
 10   FORMAT(/I3)
 11   FORMAT(I3)
      READ(LUIN,*) LUSN,LUOT,LUJS
C
C-- READ INTEGRATION PARAMETERS
      READ(LUIN,10)
      READ(LUIN,*) T,ULIMIT,DELTA,NY,NAUX,NDPTS
C
C-- READ SYSTEM SIZE PARAMETERS
      READ(LUIN,10)
      READ(LUIN,*) NZ,NU,NDNL,NDL,NT,NROOTS,JSOPT
C
C-- READ I.C.'S
      READ(LUIN,10)
      READ(LUIN,*) (Y(I),I=1,NY)
C
      DO 20 I=1,NY
         RELERR(I) = RERR
         ABSERR(I) = AERR
 20   CONTINUE
C-- CHECK FOR SIZE ERRORS
      NMAX = NZ+NU+NDNL+NDL+NT*2
      IF (NMAX .GT. N1MX) WRITE(LUSN,22)
      IF (NDNL .GT. N2MX .OR. (NT*2) .GT. N2MX) WRITE(LUSN,23)
 22   FORMAT(/,' N1MX MUST BE INCREASED FOR THIS SYSTEM ',/)
 23   FORMAT(/,' N2MX MUST BE INCREASED FOR THIS SYSTEM ',/)
C
C-- SET IMSL TO WRITE ERRORS MSGS TO THE TERMINAL
      NIN = 0
      L = 3
      CALL UGETIO(L,NIN,LUSN)
C
C-- INITIALIZE ALL SUBROUTINES
      CALL ZVECTR(T,Y)
      CALL UVECTR(T,Y)
      CALL DVECTR(DIN,Y)
      IF (NT .GT. 0) CALL TVECTR
      CALL JS
C-- CHECK INITIAL GUESSES FOR STAUTUS FLAGS, SET INITIALIZE FLAG TO 0
      IF (NROOTS .NE. 0) CALL RCHK(T,Y,YPRIME,G,IGFLAG)
      IBEGIN = 0
C
C-- DETERMINE THE STORAGE INTERVAL FOR OUTPUT  ( SDELTA )
```

```
C   NOTE: DELTA IS THE INTERVAL AT WHICH 'ODERT' IS CALLED AND SDELTA
C   IS THE INTERVAL AT WHICH THE RESULTS ARE STORED.  THE ACTUAL TIME
C   STEP USED IS DETERMINED BY THE INTEGRATION ROUTINE.
C
       TINTVL = ULIMIT - T
       SDELTA = TINTVL / FLOAT(NDPTS)
       IRATIO = IDINT(SDELTA/DELTA + 1.0D0)
       IPRT = 1
C
C--------- WRITE DATA TO FILE AT SPECIFIED INTERVAL
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
  30   IF (IPRT .EQ. 0) GO TO 60
C
C-- CALL OUTPUT TO STORE RESULTS  ( FIRST CALL INITIALIZES SYSTEM )
       CALL OUTPT(T,Y,NAUX)
       IF(T .GE. ULIMIT) GO TO 1000
       ICNTR = 0
       IPRT = 0
C
C-- INCREMENT TIME STEP AND CALL INTEGRATION ROUTINE
 60    TOUT = T + DELTA
       ICNTR = ICNTR + 1
       IF (ICNTR .GE. IRATIO) IPRT = 1
C
 61    CALL ODERT(NY,Y,T,TOUT,RELERR,ABSERR,IFLAG,WORK,IWORK,
     + NROOTS,GFT1,GFT2,IRT,JRT,TRT,IRFLAG,TMULT,MAXNUM,NY,IMIN,EPS)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C--------- ERROR CHECK
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
       IAFLAG=IABS(IFLAG)
       GO TO (101,30,103,104,105,106,107,30,109,110),IAFLAG
  101  IF(IFLAG.EQ.1.AND.T.EQ.TOUT) GO TO 30
       WRITE(LUSN,1010)IFLAG,T
 1010  FORMAT(/' RETURN FROM ODERT WITH IFLAG =',I3,' AT T =',E12.4,
     + /' CHECK FOR LOGIC ERROR,  ONLY VALUES 2 - 9 SHOULD OCCUR.'/)
       GO TO 1000
C
  103  WRITE(LUSN,1030)T,EPS
 1030  FORMAT(/' IFLAG = 3 RETURN FROM ODERT AT T =',E12.4,
     + /' RELERR AND ABSERR INCREASED....',E16.8)
       GO TO 61
C
  104  WRITE(LUSN,1040)T,MAXNUM
 1040  FORMAT(/' IFLAG = 4 RETURN FROM ODERT AT T = ',E12.4,
     + /' MORE THAN',I5,' STEPS REQUIRED FOR INTEGRATION TO TOUT'/)
       ISTUCK=ISTUCK+1
```

```
      IF(ISTUCK.GE.6)CALL EXIT
      GO TO 61
C
  105 WRITE(LUSN,1050)T
 1050 FORMAT(/' IFLAG = 5 RETURN FROM ODERT AT T = ',E12.4,
     + /' EQUATIONS APPEAR TO BE STIFF'/)
      CALL OUTPT(T,Y,NAUX)
      GO TO 61
C
  106 WRITE(LUSN,1060)T
 1060 FORMAT(/' IFLAG = 6 RETURN FROM ODERT AT T = ',E12.4,
     + ' DEGREES BECAUSE A SOLUTION COMPONENT VANISHED'/' DO NOT
     +RUN WITH PURE RELATIVE ERROR.  RE-RUN WITH A NON-ZERO ABSERR.'/)
      GO TO 1000
C
  107 WRITE(LUSN,1070)T
 1070 FORMAT(/' IFLAG = 7 RETURN FROM ODERT AT T = ',E12.4,
     + ' INVALID INPUT PARAMETERS DETECTED. '/)
      GO TO 1000
C
  109 WRITE(LUSN,1090)T
 1090 FORMAT(/' IFLAG = 9 RETURN FROM ODERT AT T = ',E12.4,
     + /' ODD ORDER POLE OF A ROOT FUNCTION G HAS BEEN FOUND.'/)
      GO TO 1000
C
  110 WRITE(LUSN,1100)T
 1100 FORMAT(/' IFLAG = 10 RETURN FROM ODERT AT T = ',E12.4,
     + /' MORE THAN 500 EVALUATIONS OF A ROOT FUNCT
     2ION G WERE REQUIRED.'/)
C
C-- NOTE: IFLAG = 8 INDICATES A ROOT WAS FOUND AND INTEGRATION IS
C         CONTINUING NORMALLY
C
 1000  WRITE(LUSN,2000) T
 2000  FORMAT(1X,/' INTEGRATION COMPLETE AT T = ',E12.4,/)
       STOP
       END
C
       SUBROUTINE OUTPT(T,Y,NAUX)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC                                                                    CC
CC-- NAME: OUTPT                                                      CC
CC                                                                    CC
CC-- FUNCTIONS:  1. CALLS FCT TO COMPUTE VALUES FOR STORAGE.           CC
CC               2. WRITES RESULTS TO SPECIFIED FILE.                 CC
CC                                                                    CC
CC-- SUBROUTINES CALLED: FCT                                          CC
CC                                                                    CC
CC-- VARIABLE DEFINITIONS                                             CC
CC                                                                    CC
```

```
CC         T: THE INDEPENDENT VARIABLE OF INTEGRATION                    CC
CC         Y: THE STATE VECTOR                                           CC
CC      NAUX: THE NUMBER OF AUXILIARY VALUES TO BE OUTPUT                CC
CC                                                                       CC
CC-- PROGRAMMER: N. HENDRIKSMA   1984                                    CC
CC                                                                       CC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
       IMPLICIT REAL*8 (A-H,O-Z)
C
       DIMENSION Y(NY),YDOT(10)
       COMMON /LUDEF/ LUSN,LUIN,LUOT,LUJS
       COMMON /SYSTEM/ NZ,NU,NDNL,NDL,NT,NY,NZPNU,N1,N1MX,N2MX,
     +                 ZOUT(15),UOUT(5),DOUT(10),AUX(10),
     +                 S1(25,25),JS1D(25,10)
C
C-- CALL FCT TO GET VALUES FOR STORAGE ( DUE TO INTERPOLATION )
C   ( FIRST CALL INITIALIZES PARAMETERS
       CALL FCT(T,Y,YDOT)
       IF (NAUX .EQ. 0) GO TO 20
C
C-- WRITE DATA TO FILE
       WRITE(LUOT,10) T,(Y(K),K=1,NY)
       WRITE(LUOT,10) T,(AUX(J),J=1,NAUX)
       WRITE(LUOT,10) T,(ZOUT(K),K=1,NZ),(UOUT(J),J=1,NU),(DOUT(L),
     +     L=1,NDNL)
 10    FORMAT(1X,E11.3,10E12.4)
       WRITE(LUSN,*) ZOUT(8)
C
       GO TO 1000
C
 20    WRITE(LUOT,10) T,(Y(K),K=1,NY)
       WRITE(LUOT,10) T,(ZOUT(K),K=1,NZ),(UOUT(J),J=1,NU),(DOUT(L),
     +     L=1,NDNL)
C
 1000  RETURN
       END
C
       SUBROUTINE FCT(T,Y,YPRIME)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC                                                                       CC
CC-- NAME: FCT                                                           CC
CC                                                                       CC
CC-- FUNCTIONS:  1. COMPUTES THE TIME DERIVATIVE OF THE STATE VECTOR     CC
CC                  YPRIME                                               CC
CC                                                                       CC
CC-- SUBROUTINES CALLED: ZVECTR,UVECTR,DVECTR,JS                         CC
CC                                                                       CC
CC-- VARIABLE DEFINITIONS                                                CC
```

```
CC                                                                      CC
CC         T: THE INDEPENDENT VARIABLE OF INTEGRATION                   CC
CC         Y: THE STATE VECTOR                                          CC
CC     YPRIME: THE TIME DERIVATIVE OF THE STATE VECTOR                  CC
CC                                                                      CC
CC-- PROGRAMMER: N. HENDRIKSMA   1984                                   CC
CC                                                                      CC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
       IMPLICIT REAL*8 (A-H,O-Z)
C
       DIMENSION Y(NY),YPRIME(NY),DIN(10)
       COMMON /FLAG/ IBEGIN,JSOPT
       COMMON /LUDEF/ LUSN,LUIN,LUOT,LUJS
       COMMON /STATUS/ ISTAT(16),NROOTS
       COMMON /SYSTEM/ NZ,NU,NDNL,NDL,NT,NY,NZPNU,N1,N1MX,N2MX,
     +                 ZOUT(15),UOUT(5),DOUT(10),AUX(10),
     +                 S1(25,25),JS1D(25,10)
C
       DATA IZERO/0/
C
C-- COMPUTE THE Z-VECTOR
 10    CALL ZVECTR(T,Y)
C
C-- COMPUTE THE U-VECTOR
       CALL UVECTR(T,Y)
C
C-- COMPUTE THE DOUT-VECTOR  ( NONLINEAR )
       IF(NDNL .EQ. 0) GO TO 20
       CALL DVECTR(DIN,Y)
C
C-- COMPUTE YPRIME
 20    CALL VECMUL(IZERO,NY,N1,YPRIME)
C
       RETURN
       END
C
      SUBROUTINE JS
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC                                                                      CC
CC-- NAME: JS                                                           CC
CC                                                                      CC
CC-- FUNCTIONS:  1. COMPUTES OR READS THE SYSTEM MATRIX.  REDUCES THE CC
CC                  MATRIX AS MUCH AS POSSIBLE BY ELIMINATING THE     CC
CC                  LINEAR BONDS.                                     CC
CC                                                                      CC
CC-- SUBROUTINES CALLED: ZMAT,PRTITN,REDUCT                             CC
CC                                                                      CC
CC-- PROGRAMMER: N. HENDRIKSMA   1984                                   CC
```

```
CC                                                                      CC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      COMMON /FLAG/ IBEGIN,JSOPT
      COMMON /LUDEF/ LUSN,LUIN,LUOT,LUJS
      COMMON /SYSTEM/ NZ,NU,NDNL,NDL,NT,NY,NZPNU,N1,N1MX,N2MX,
     +                ZOUT(15),UOUT(5),DOUT(10),AUX(10),
     +                S1(25,25),JS1D(25,10)
      COMMON /LMATS/ RL(10,10),TM(10,10)
C
      DIMENSION S2(25,10),S3(10,25),S4(10,10),WK1(10,10),
     +      WK2(10,10),IBOND(30),IERR(7)

      DATA IZERO/0/,RJS/'JS'/
C
C-- THE DIMENSIONS FOR S2,S3,S4,TM,WK1,WK2 MUST BE N1MX AND N2MX
C
C-- NINMX IS THE MAX # OF ITEMS TO BE READ FROM AN INPUT LINE FOR THE
C   BOND GRAPH INPUT  ( THE DIMENSION FOR IBOND(NINMX) )
C   JS1DMX IS THE COLUMN DIMENSION OF JS1D(25,10)  THIS IS THE MAX # OF
C   ELEMENTS FOR THE MATRIX.  THIS IS LESS THAN N1MX TO SAVE STORAGE.
      NINMX = 30
      JS1DMX= 10
C
      NZPNU = NZ+NU
C
C-- OPTIONS
C    JSOPT = 0  -> COMPUTE JS  AND CONTINUE INTO INTEGRATION
C    JSOPT = 1  -> COMPUTE JS ONLY, PRINT OUT AND STOP
C    JSOPT = 2  -> READ EXISTING JS AND CONTINUE INTO INTEGRATION
C
C-- INITIALIZE THE ERROR FLAGS
      DO 5 I=1,7
      IERR(I) = 0
 5    CONTINUE
C
      IF (JSOPT .EQ. 2) GO TO 200
      NTB = NT*2
      N1 = NZ+NU+NDNL+NDL+NTB
C
C-- ZERO THE S1 MATRIX
      CALL ZMAT(N1MX,N1,N1,S1)
C
C-- READ IN THE BOND GRAPH STRUCTURE ONE LINE AT A TIME
      IROW = 1
      IFLG = 10000
      READ(LUIN,9) RCHCK
      IF (RCHCK .EQ. RJS) GO TO 13
```

```
      WRITE(LUSN,8) RCHCK
 8    FORMAT( ' ERROR ON INPUT TO SUBROUTINE JS   AT START OF BLOCK'
     + ,/ ,'     RCHCK = ', A4)
      CALL EXIT
 9    FORMAT(A2)
 10   FORMAT(/I3)
 11   FORMAT(I3)
C
 13   DO 15 M=1,NINMX
      IF (IBOND(M) .EQ. IFLG) GO TO 16
      IBOND(M) = IFLG
 15   CONTINUE
C
 16   READ(LUIN,*) (IBOND(M),M=1,NINMX)
      IF (IBOND(NINMX) .NE. IFLG) IERR(1) = 1
      IF (IBOND(NINMX) .NE. IFLG) GO TO 2000
C
C-- DETERMINE # OF DATA PTS READ, "NIN", FOR THE CURRENT LINE
      NIN = 0
      DO 17 M=1,NINMX
      IF (IBOND(M) .EQ. IFLG) GO TO 18
      NIN = NIN + 1
 17   CONTINUE
C
C-- ASSIGN THE BONDS TO THE S1 MATRIX FROM THE CURRENT LINE OF INPUT
 18   IF (NIN .LT. 3) IERR(2) = 1
      IF (NIN .LT. 3) GO TO 2000
C
      DO 90 JJ=1,NIN
C
C-- CHECK FOR INPUT ERRORS
      IF (JJ .EQ.1 .AND. IBOND(JJ) .NE. IROW) IERR(4) = IROW
      IF (JJ .EQ.1 .AND. IBOND(JJ) .NE. IROW) GO TO 2000
C
      IF (IBOND(JJ) .EQ. 0 .AND. IROW .EQ. N1) GO TO 110
C
      IF (IBOND(JJ) .EQ. 0 .AND. JJ .NE. NIN .AND.
     +            IBOND(JJ+1) .NE. (IROW+1)) IERR(5) = IROW
      IF (IERR(5) .NE. 0) GO TO 2000
C
C-- ASSIGN + OR - 1 TO APPROPRIATE LOCNS IN S1
       IF (IBOND(JJ) .EQ. IROW) GO TO 90
       SIGN = 1.0D0
       IF (IBOND(JJ)) 50,80,70
C
 50    SIGN = -1.0D0
 70    J = IABS(IBOND(JJ))
      IF (J .GT. N1) IERR(3) = IROW
      IF (J .GT. N1) GO TO 2000
       S1(IROW,J) = SIGN
       GO TO 90
```

```
C
 80      IROW = IROW + 1
C
 90     CONTINUE
        GO TO 13
C
 110    IF (NT .EQ. 0) GO TO 115
C
C-- PARTITION THE S1 MATRIX FOR ELIMINATION OF THE LINEAR GYRATOR/
C   TRANSFORMER BONDS
        N1OLD = N1
        N1 = N1 - NTB
        CALL PRTITN(IZERO,N1,N1MX,N1,NTB,S2,N1MX,N1OLD,S1)
        CALL PRTITN(N1,IZERO,N2MX,NTB,N1,S3,N1MX,N1OLD,S1)
        CALL PRTITN(N1,N1,N2MX,NTB,NTB,S4,N1MX,N1OLD,S1)
C
C
C-- COMPUTE S1' BY ELIMINATION OF THE TRANSFORMER & GYRATOR BONDS
        CALL REDUCT(N1MX,N2MX,N1,NTB,S1,S2,S3,S4,TM,WK1,WK2)
C
 115    IF (NDL .EQ. 0) GO TO 175
C
C-- PARTITION THE RESULTING MATRIX FOR ELIMINATION OF THE LINEAR
C   DISSIPATION BONDS
        N1OLD = N1
        N1 = N1 - NDL
        CALL PRTITN(IZERO,N1,N1MX,N1,NDL,S2,N1MX,N1OLD,S1)
        CALL PRTITN(N1,IZERO,N2MX,NDL,N1,S3,N1MX,N1OLD,S1)
        CALL PRTITN(N1,N1,N2MX,NDL,NDL,S4,N1MX,N1OLD,S1)
C
C-- COMPUTE S1" BY ELIMINATION OF LINEAR RESISTANCE BONDS
        CALL REDUCT(N1MX,N2MX,N1,NDL,S1,S2,S3,S4,RL,WK1,WK2)
C
C-- FORM DIRECTORY MATRIX FOR MULTIPLICATION
 175    DO 170 I=1,N1
        LOCN = 0
        JS1D(I,1) = 0
        DO 160 J=1,N1
C
           IF (S1(I,J) .EQ. 0.) GO TO 160
           LOCN = LOCN+1
           IF (LOCN .LT. JS1DMX) GO TO 151
           IERR(6) = 1
           GO TO 2000
 151       JS1D(I,LOCN) = J
           IF (LOCN .EQ. JS1DMX) GO TO 160
           JS1D(I,LOCN+1) = 0
C
 160    CONTINUE
 170    CONTINUE
C
```

```
C-- WRITE OUT THE SYSTEM MATRIX AND DIRECTORY FOR LATER USE
      WRITE(LUJS,177) N1
 177  FORMAT(/,I4,/)
      DO 180 I=1,N1
      WRITE (LUJS,190) (S1(I,J),J=1,N1)
      WRITE (LUJS,190)
 180  CONTINUE
 190  FORMAT(8E14.5)
C
      DO 185 I=1,N1
      WRITE (LUJS,195) (JS1D(I,J),J=1,JS1DMX)
      WRITE (LUJS,190)
 185  CONTINUE
 195  FORMAT(30I4)
      WRITE(LUSN,196)
 196  FORMAT(/'JS MATRIX COMPLETE '/)
      IF (JSOPT .NE. 1) GO TO 1000
      CALL EXIT
C
C-- READ IN EXISTING SYSTEM MATRIX AND FORM DIRECTORY MATRIX
 200  N1 = NZ+NU+NDNL
      READ(LUJS,*) NCHECK
      IF (NCHECK .EQ. N1) GO TO 202
      IERR(7) = 1
      GO TO 2000
C
 202  DO 220 I=1,N1
      READ(LUJS,*) (S1(I,J),J=1,N1)
C
      LOCN = 0
      JS1D(I,1) = 0
      DO 210 J=1,N1
C
         IF (S1(I,J) .EQ. 0.) GO TO 210
         LOCN = LOCN+1
         JS1D(I,LOCN) = J
         IF (LOCN .EQ. JS1DMX) GO TO 210
         JS1D(I,LOCN+1) = 0
C
 210  CONTINUE
 220  CONTINUE
C
 1000 RETURN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C-- ERROR TABLE FOR JS SUBROUTINE
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
 2000 WRITE(LUSN,2010)
 2010 FORMAT(/,'ERROR CODE FROM SUBROUTINE JS - POSSIBLE CAUSES BELOW')
```

```
      WRITE(LUSN,2020)
 2020 FORMAT(//,'STATUS                          CONDITI0N',//)
      WRITE(LUSN,2030) (IERR(I),I=1,4)
 2030 FORMAT(I3,'  TOO MANY ENTRIES IN DATA LINE FOR THE BOND GRAPH',/
     +       ,I3,'  TOO FEW  ENTRIES IN DATA LINE FOR THE BOND GRAPH',/
     +       ,I3,'  ENCOUNTERED BOND NUMBER LARGER THAN N1          ',/
     +       ,I3,'  FIRST ELEMENT IN A LINE IS NOT IN ORDER         ')
      WRITE(LUSN,2040) (IERR(I),I=5,7)
 2040 FORMAT(I3,'  FIRST ELEMENT AFTER A ZERO NOT IN ORDER          ',/
     +       ,I3,'  COLUMN DIMENSION OF JS1D MUST BE INCREASED      ',/
     +       ,I3,'  SIZE OF EXISTING SYSTEM MATRIX IS INCORRECT     ',/)
      STOP
      END
C
      SUBROUTINE PRTITN(IS,JS,NRMX,NR,NC,SUBS,N1MX,N1,S)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC                                                                      CC
CC-- NAME: PRTITN                                                       CC
CC                                                                      CC
CC-- FUNCTIONS:  1. FORMS A SUBMATRIX OF 'S' OF APPROPRIATE DIMENSION CC
CC                  BASED ON THE CALLING LIST.  THE SUBMATRICES ARE   CC
CC                  USED IN THE REDUCTION OF THE LINEAR BONDS.        CC
CC                                                                      CC
CC-- SUBROUTINES CALLED: NONE                                           CC
CC                                                                      CC
CC-- VARIABLE DEFINITIONS                                               CC
CC                                                                      CC
CC        IS: IS+1 IS THE STARTING ROW OF 'S' FOR THE SUBMATRIX         CC
CC        JS: JS+1 IS THE STARTING COLUMN OF 'S' FOR THE SUBMATRIX      CC
CC      NRMX: THE ROW DIMENSION OF THE SUBMATRIX AS SPECIFIED IN THE    CC
CC            CALLING PROGRAM                                           CC
CC        NR: ACTUAL NUMBER OF ROWS IN THE SUBMATRIX                    CC
CC        NC: ACTUAL NUMBER OF COLUMNS IN THE SUBMATRIX                 CC
CC      SUBS: THE SUBMATRIX FORMED FROM 'S' -OUTPUT OF THE SUBROUTINE   CC
CC      N1MX: ROW DIMENSION OF 'S' AS SPECIFIED IN THE CALLING PROGRAMCC
CC        N1: ACTUAL NUMBER OF ROWS IN 'S'                              CC
CC         S: MATRIX WHICH IS TO BE PARTITIONED                         CC
CC                                                                      CC
CC-- PROGRAMMER: N. HENDRIKSMA  1984                                    CC
CC                                                                      CC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION SUBS(NRMX,NC),S(N1MX,N1)
C
      IE=IS+NR
      ISP1=IS+1
      JE=JS+NC
```

```
      JSP1=JS+1
      I=0
C
      DO 20 I1=ISP1,IE
         I = I+1
         J = 0
      DO 10 J1=JSP1,JE
         J = J+1
C
      SUBS(I,J) = S(I1,J1)
 10   CONTINUE
 20   CONTINUE
      RETURN
      END
C
      SUBROUTINE ZMAT(NRMX,NR,NC,A)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC                                                                      CC
CC-- NAME: ZMAT                                                         CC
CC                                                                      CC
CC-- FUNCTIONS:  1. USED TO SET ALL ELEMENTS OF A MATRIX TO ZERO        CC
CC                                                                      CC
CC-- SUBROUTINES CALLED: NONE                                           CC
CC                                                                      CC
CC-- VARIABLE DEFINITIONS                                               CC
CC                                                                      CC
CC      NRMX: ROW DIMENSION OF 'A' AS SPECIFIED IN THE CALLING PROGRAMCC
CC        NR: ACTUAL NUMBER OF ROWS IN 'A'                              CC
CC        NC: ACTUAL NUMBER OF COLUMNS OF 'A'                           CC
CC         A: MATRIX WHICH IS TO BE INITIALIZED TO ZERO                 CC
CC                                                                      CC
CC-- PROGRAMMER: N. HENDRIKSMA   1984                                   CC
CC                                                                      CC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION A(NRMX,NC)
C
      DO 20 I=1,NR
      DO 10 J=1,NC
         A(I,J) = 0.0D0
 10   CONTINUE
 20   CONTINUE
      RETURN
      END
C
     SUBROUTINE REDUCT(N1MX,N2MX,N1,N2,S1,S2,S3,S4,TM,WK1,WK2)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC                                                                      CC
CC-- NAME: REDUCT                                                       CC
CC                                                                      CC
CC-- FUNCTIONS:  1. PROVIDES FOR ELIMINATION OF THE LINEAR DISSIPATIONCC
CC                  OR TRANSFORMER/GYRATOR BONDS BY COMPUTING A NEW     CC
CC                  SYSTEM MATRIX USING THE LINEAR RELATIONSHIP.        CC
CC                  SUMMARY: S5  = ((I-S4*TM)**1)*S3                    CC
CC                           S6  = S2*TM*S5                             CC
CC                           S1' = S1 + S6                              CC
CC                                                                      CC
CC-- SUBROUTINES CALLED: VMULFF,LINV2F ( BOTH IMSL )                    CC
CC                                                                      CC
CC-- VARIABLE DEFINITIONS                                               CC
CC                                                                      CC
CC      N1MX: ROW DIMENSION OF S1,S2 AS SPECIFIED IN THE CALLING        CC
CC            PROGRAM                                                   CC
CC      N2MX: ROW DIMENSION OF S3,S4 AS SPECIFIED IN THE CALLING        CC
CC            PROGRAM                                                   CC
CC        N1: ACTUAL ROW DIMENSION OF S1,S2                             CC
CC        N2: ACTUAL ROW DIMENSION OF S3,S4                             CC
CC     S1-S4: SUBMATRICES OF THE SYSTEM MATRIX USED IN THE REDUCTION    CC
CC            S1 IS USED ON INPUT AND OUTPUT                            CC
CC        TM: MATRIX WHICH DEFINES THE LINEAR RELATIONSHIP              CC
CC   WK1,WK2: WORK SPACE MATRICES OF APPROPRIATE DIMENSIONS             CC
CC                                                                      CC
CC-- PROGRAMMER: N. HENDRIKSMA  1984                                    CC
CC                                                                      CC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /LUDEF/LUSN,LUIN,LUOT,LUJS,LUAB
C
     DIMENSION S1(N1MX,N1),S2(N1MX,N2),S3(N2MX,N1),S4(N2MX,N2),
    +  TM(N2MX,N2),
    +  WK1(N2,N2),WK2(N2,N2),WK3(130),S5(10,25),S6(25,25)
C
C-- NOTE DIMENSION ON WK3 SHOULD BE: WK3(N2MX**2  + 3*N2MX)
C                      S5             S5(N2MX,N1MX)
C                      S6             S5(N1MX,N1MX)
C
C-- MULTIPLY S4*TM AND STORE IN WK1
      CALL VMULFF(S4,TM,N2,N2,N2,N2MX,N2MX,WK1,N2,IER)
C
C-- FORM ( I - S4*TM )  CHECK IF S4 IS ZERO, SKIP INVERSION IF YES
      IFLAG = 0
      DO 20 I=1,N2
      DO 10 J=1,N2
        IF (WK1(I,J) .NE. 0.) IFLAG = 1
         WK1(I,J) = -1.0D0*WK1(I,J)
```

```
         IF (I .EQ. J) WK1(I,J) = WK1(I,J) + 1.0D0
         WK2(I,J) = WK1(I,J)
 10    CONTINUE
 20    CONTINUE
C
       IF (IFLAG .EQ. 0) GO TO 25
C
C-- INVERT WK1 AND STORE RESULT IN WK2
       IDGT = 4
       CALL LINV2F(WK1,N2,N2,WK2,IDGT,WK3,IER)
       IF(IER .NE. 0) WRITE(LUSN,22) N2,IER
 22    FORMAT('ERROR IN MATRIX INVERSION IN REDUCT - N2 = ',I3
     + ,' IER = ',I3)
       IF(IER .NE. 0) STOP
C
C-- MULTIPLY S3 BY THE RESULT AND STORE IN S5
 25    CALL VMULFF(WK2,S3,N2,N2,N1,N2,N2MX,S5,N2MX,IER)
C
C-- MULTIPLY BY TM AND STORE IN S3
       CALL VMULFF(TM,S5,N2,N2,N1,N2MX,N2MX,S3,N2MX,IER)
C
C-- MULTIPLY BY S2 AND STORE IN S6
       CALL VMULFF(S2,S3,N1,N2,N1,N1MX,N2MX,S6,N1MX,IER)
C
C-- ADD S1 AND S6 TO GET THE REDUCED SYSTEM MATRIX
       DO 40 I=1,N1
       DO 30 J=1,N1
C
          S1(I,J) = S1(I,J) + S6(I,J)
C
 30    CONTINUE
 40    CONTINUE
       RETURN
       END
C
       SUBROUTINE VECMUL(ISM1,NROWS,NVO,C)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC                                                                          CC
CC-- NAME: VECMUL                                                           CC
CC                                                                          CC
CC-- FUNCTIONS:  1. COMPUTES NEW INPUTS TO THE LUMPED PARAMETERS            CC
CC                  BY MULTIPLYING THE SYSTEM MATRIX BY THE OUTPUT          CC
CC                  VECTORS - ZOUT,UOUT AND DOUT.                           CC
CC                                                                          CC
CC-- SUBROUTINES CALLED: NONE                                               CC
CC                                                                          CC
CC-- VARIABLE DEFINITIONS                                                   CC
CC                                                                          CC
CC      ISM1: ISM1+1 IS THE STARTING ROW OF 'S1' USED IN THE                CC
CC            MULTIPLICATION                                                CC
```

```
CC                                                                      CC
CC     NROWS: NUMBER OF ROWS FOLLOWING ISM1+1 TO BE USED.               CC
CC       NVO: NUMBER OF ROWS OF THE OUTPUT VECTORS TO BE USED           CC
CC            ( NORMALLY THIS WILL BE 'N1' )                            CC
CC         C: VECTOR OF LENGTH 'NROWS' WHICH IS THE RESULT OF THE       CC
CC            CALCULATION                                               CC
CC                                                                      CC
CC-- PROGRAMMER: N. HENDRIKSMA   1984                                   CC
CC                                                                      CC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
       IMPLICIT REAL*8 (A-H,O-Z)
C
       DIMENSION C(NROWS)
       COMMON /SYSTEM/ NZ,NU,NDNL,NDL,NT,NY,NZPNU,N1,N1MX,N2MX,
     +                 ZOUT(15),UOUT(5),DOUT(10),AUX(10),
     +                 S1(25,25),JS1D(25,10)
C
C-- NOTE: NVO IS THE NUMBER OF ELEMENTS IN THE VECTOR TO BE USED IN THE
C         MULTIPLICATION.  USSUALLY THIS WILL BE N1
C-- COMPUTE THE START AND END ROW OF S1
       IE = ISM1 + NROWS
       IS = ISM1 + 1
       K = 0
       DO 500 I =IS,IE
       K = K + 1
       C(K) = 0.0D0
       DO 400 J =1,N1
C
       JD = JS1D(I,J)
       IF (JD .GT. NVO) GO TO 400
C
       IF (JD .EQ. 0) GO TO 500
       IF (JD .LE. NZ) VALUE=ZOUT(JD)
       IF (JD .GT. NZ .AND. JD .LE. NZPNU) VALUE=UOUT(JD-NZ)
       IF (JD .GT. NZPNU) VALUE=DOUT(JD-NZPNU)
C
       C(K) = C(K) + S1(I,JD)*VALUE
C
 400   CONTINUE
 500   CONTINUE
       RETURN
       END
       SUBROUTINE ZVECTR(TIME,Y)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC                                                                      CC
CC-- NAME: ZVECTR                                                       CC
CC                                                                      CC
CC-- FUNCTIONS:  1. INITIALIZES ENERGY STORAGE PARAMETERS               CC
```

```
CC              2. COMPUTES THE COMPLEMENTARY STATE VECTOR          CC
CC                                                                  CC
CC-- SUBROUTINES CALLED: COEFF,AIRREL,METREL                        CC
CC                                                                  CC
CC-- VARIABLE DEFINITIONS                                           CC
CC                                                                  CC
CC      TIME: THE INDEPENDENT VARIABLE OF INTEGRATION               CC
CC         Y: THE STATE VECTOR                                      CC
CC                                                                  CC
CC-- PROGRAMMER: N. HENDRIKSMA  1984                                CC
CC                                                                  CC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
       IMPLICIT REAL*8 (A-H,O-Z)
C
       DIMENSION Y(NY),FPL(5,3),REL(5,3),DRDX(2)
       REAL*8 LCL,LSL,LSG,LA,KSTOP,KSPRNG,MU0
       COMMON /FLAG/ IBEGIN,JSOPT
       COMMON /LUDEF/ LUSN,LUIN,LUOT,LUJS
       COMMON /SYSTEM/ NZ,NU,NDNL,NDL,NT,NY,NZPNU,N1,N1MX,N2MX,
     +                 ZOUT(15),UOUT(5),DOUT(10),AUX(10),
     +                 S1(25,25),JS1D(25,10)
       COMMON /ZVEC/ AG,STROKE
       COMMON /DVEC/ CFMAS1,CFMAS2,VBAT,VZENR
       COMMON /STATUS/ ISTAT(16),NROOTS
       COMMON /METAL/ A(5,3),FPLA(5,3),MATL(5,3)
       COMMON /AGPAR/ TA,T12,T18W,T18L,
     1                P1C(2),P12BC(2),P18BCW(2),P18BCL(2),P8BCW(2),
     2                P8BCL(2)
       DATA PIE/3.141592654D0/,MU0/1.2567D-6/,ZVEC/'ZVEC'/
C
       IF (IBEGIN .EQ. 0) GO TO 500
C
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C------------ INITIALIZATION
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C
C-- READ MATERIAL SPCIFICATIONS
       READ(LUIN,9) RCHCK
       IF (RCHCK .EQ. ZVEC) GO TO 13
       WRITE(LUSN,8) RCHCK
 8     FORMAT( ' ERROR ON INPUT TO SUBROUTINE ZVECTR AT START OF BLOCK'
     + ,/ ,'     RCHCK = ', A4)
       CALL EXIT
 9     FORMAT(A4)
 13    READ(LUIN,11)
 10    FORMAT(/I3)
 11    FORMAT(I3)
       READ(LUIN,*) NMATL,LUMATL,MATL(1,1),MATL(1,2),MATL(1,3),
     +              MATL(5,3)
```

```
      MATL(3,1)=MATL(1,1)
      MATL(3,2)=MATL(1,3)
C-- COMPUTE U-B CURVES FROM B-H DATA
      CALL COEFF(NMATL,LUMATL)
C-- READ STATOR DIMENSIONS  (MM)  AND CONVERT TO METERS
      READ(LUIN,10)
      READ(LUIN,*) HS,HL,LCL,LSL,LSG,HLK
      READ(LUIN,10)
      READ(LUIN,*) TLAM,SF,NLAMS
C
      HS=HS/1000.0D0
      HLK=HLK/1000.0D0
      LCL=LCL/1000.0D0
      TLAM=TLAM/1000.0D0
      HL=HL/1000.0D0
      LSL=LSL/1000.0D0
      LSG=LSG/1000.0D0
      WS=FLOAT(NLAMS)*TLAM/SF
C-- READ ARMATURE DIMENSIONS
      READ(LUIN,10)
      READ(LUIN,*) LA,WA,TA
      LA=LA/1000.0D0
      WA=WA/1000.0D0
      TA=TA/1000.0D0
C
C-- DEFINE AREAS, FLUX PATH LENGTHS, MATERIAL AND NUMBER OF METALLIC
C    ELEMENTS IN EACH FLUX PATH.   SEE SKETCH FOR PATH DEFINITIONS
C
C-- PATH 1   ( CENTER LEG, TOP YOKE AND OUTER LEGS )
C     ELEMENT #1 IS THE CENTER LEG
C
      A(1,1) = WS*SF*LCL
      FPL(1,1) = HL - HLK + (HS-HL)/2.0D0
      FPLA(1,1) = FPL(1,1)/A(1,1)
C
C-- ELEMENT #2 IS THE TOP YOKE.  ( THE AREA IS DOUBLED DUE TO SYMMETRY )
      A(1,2) = 2.0D0*WS*SF*(HS-HL)
      FPL(1,2) = .5D0*LSL+LSG+.5D0*LCL
      FPLA(1,2) = FPL(1,2)/A(1,2)
C
C-- ELEMENT #3 IS THE SIDE LEGS IN PARALLEL  (AREA OF ONE LEG IS DOUBLED
      A(1,3) = 2.0D0*LSL*WS*SF
      FPL(1,3) = HL - HLK + (HS-HL)/2.0D0
      FPLA(1,3) = FPL(1,3)/A(1,3)
C
C-- PATH 2 IS THE FIRST LEAKAGE PATH BETWEEN THE CENTER AND OUTER LEGS.
C   THIS RELUCTANCE IS CONSTANT AND SO IS DIRECTLY CALCULATED HERE.
C   SEE ALSO PAGE 97 OF ROTOR'S BOOK FOR REFERENCE.
      P1L = WS/LSG
      P7L = .52D0
      P8BL = (2.0D0/PIE)*DLOG(1.0D0 +2.0D0*LSL/LSG)
```

```
C
C-- THE PERMEANCE IS DOUBLED DUE TO THE SYMMETRY.
      PLTOT = MU0*(HL-HLK)*(P1L + P7L + P8BL)
      REL(2,1) = 1.0D0/PLTOT
C
C-- PATH 3 IS THE LOWER SEGMENTS OF THE CENTER AND OUTER LEGS.
C   THIS LENGTH IS DEFINED BY 'HLK'
C     ELEMENT #1 IS THE CENTER LEG
C
      A(3,1) = A(1,1)
      FPL(3,1) =  HLK
      FPLA(3,1) = FPL(3,1)/A(3,1)
C
C-- ELEMENT #2 IS THE OUTER LEGS IN PARALLEL (DOUBLED DUE TO SYMMETRY )
      A(3,2) = A(1,3)
      FPL(3,2) = HLK
      FPLA(3,2) = FPL(3,2)/A(3,2)
C
C-- PATH 4 IS THE SECOND LEAKAGE PATH BETWEEN THE CENTER AND OUTER LEGS.
C   THIS RELUCTANCE IS CONSTANT AND SO IS DIRECTLY CALCULATED HERE.
C   SEE ALSO PAGE 97 OF ROTOR'S BOOK FOR REFERENCE.
      PLTOT = 2.0D0*MU0*HLK*(P1L + P7L + P8BL)
      REL(4,1) = 1.0D0/PLTOT
C
C-- PATH 5 CONSISTS OF THE 2 AIR GAPS AND THE ARMATURE.
C   ELEMENT #3 IS THE ARMATURE.  ( AREA IS DOUBLED FOR SYMMETRY )
      A(5,3) = 2.0D0*WA*TA
      FPL(5,3) = .5D0*LSL+LSG+.5D0*LCL
      FPLA(5,3) = FPL(5,3)/A(5,3)
C
C-- ELEMENTS #1 AND #2 ARE AIR GAPS.  CONSTANTS ARE COMPUTED FOR THE
C   PERMEANCE CALCULATIONS BASED ON CHAPTER 5 OF ROTOR'S BOOK.
C
C-- COMPUTE AUXILIARY QUANTITIES
      T12 = LSG/2.0D0
      T18W = (WS - WA)/2.0D0
      SL = 2.0D0*(LSL+LSG) + LCL
      T18L = (SL - LA)/2.0D0
C
C-- IT IS ASSUMED THAT THE STATOR IS LARGER THAN THE ARMATURE
C   IF NOT, T18W AND/OR 18L ARE SET TO 0
      IF (T18W .LT. 0.0D0) T18W = 0.0D0
      IF (T18L .LT. 0.0D0) T18L = 0.0D0
      IF (T18W .GT. TA) T18W = TA
      IF (T18L .GT. TA) T18L = TA
C
C-- ELEMENT #1 IS THE AIR GAP OF THE CENTER LEG
      A(5,1) = WA*LCL
      P1C(1) = MU0*A(5,1)
      P12BC(1) = 4.0D0*MU0*WA/PIE
      P18BCW(1) = 4.0D0*MU0*LCL/PIE
```

```
       P18BCL(1) = 0.0D0
       P8BCW(1) = 2.0D0*MU0*LCL/PIE
       P8BCL(1) = 0.0D0
C
C-- ELEMENT #2 IS THE AIR GAP FOR THE OUTER LEG.  THESE CONSTANTS ARE
C   DOUBLED TO COMPENSATE FOR TWO LEGS.
       A(5,2) = 2.0D0*WA*(LSL - T18L)
       P1C(2) = MU0*A(5,2)
       P12BC(2) = 4.0D0*MU0*WA/PIE
       P18BCW(2) = 8.0D0*MU0*(LSL - T18L)/PIE
       P18BCL(2) = 4.0D0*MU0*WA/PIE
       P8BCW(2) = 4.0D0*MU0*(LSL - T18L)/PIE
       P8BCL(2) = 2.0D0*MU0*WA/PIE
C
C-- READ VALVE SUBSYSTEM QUANTITIES
       READ(LUIN,10)
       READ(LUIN,*) AGCLSD,STROKE,PRELD,KSTOP,KSPRNG,VMASS
       AGCLSD=AGCLSD/1000.0D0
       STROKE=STROKE/1000.0D0
       AGOPEN=AGCLSD+STROKE
C
C-- SET INITIAL CONDITION FOR Y(6) IF INPUT IS ZERO
       IF(Y(6) .EQ. 0.0D0) Y(6)=-1.0D0*PRELD/(KSTOP-KSPRNG)
C
C-- INITIALIZE THE ISTAT FLAGS
       ISTAT(1) = 0
       ISTAT(2) = 0
       IF (Y(6) .GT. 0.0D0) ISTAT(1) = 1
       IF (Y(6) .LT. STROKE) ISTAT(2) = 1
C
       GO TO 1000
C
 500   CONTINUE
C
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C-------------- COMPUTE THE Z-VECTOR
C$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
C
C-- COMPUTE RELUCTANCE OF PATH 1
       IPATH=1
       IELEM=1
       FLUX = Y(3) + .6667D0*Y(2)
       CALL METREL(IPATH,IELEM,FLUX,REL(IPATH,IELEM))
       IELEM=2
       CALL METREL(IPATH,IELEM,Y(1),REL(IPATH,IELEM))
       IELEM=3
       CALL METREL(IPATH,IELEM,FLUX,REL(IPATH,IELEM))
C
C-- PATH 2 HAS CONSTANT RELUCTANCE
C
C-- COMPUTE RELUCTANCE OF PATH 3
```

```
       IPATH=3
       DO 520 IELEM = 1,2
       CALL METREL(IPATH,IELEM,Y(IPATH),REL(IPATH,IELEM))
 520   CONTINUE
C
C-- PATH 4 HAS CONSTANT RELUCTANCE
C
C-- COMPUTE RELUCTANCE OF PATH 5     ( 1 IS THE CENTER LEG )
       AG=AGOPEN-Y(6)
       NGAP=1
       CALL AIRREL(NGAP,AG,REL(5,1),DRDX(1))
       NGAP=2
       CALL AIRREL(NGAP,AG,REL(5,2),DRDX(2))
       IPATH=5
       IELEM=3
       CALL METREL(IPATH,IELEM,Y(5),REL(5,3))
C
C-- Z(1),Z(2),Z(3),Z(4),Z(5) ARE MMF VALUES
       ZOUT(1)=(REL(1,1)+REL(1,3))*FLUX+REL(1,2)*Y(1)
       ZOUT(2)=REL(2,1)*Y(2)
       ZOUT(3)=(REL(3,1)+REL(3,2))*Y(3)
       ZOUT(4)=REL(4,1)*Y(4)
       ZOUT(5)=(REL(5,1)+REL(5,2)+REL(5,3))*Y(5)
C
C-- Z(6) IS THE FORCE DUE TO THE VALVE SPRING/SEATING CONDITIONS
C   THE SPRING RATE IS PIECEWISE LINEAR AS A FUNCTION OF VALVE POSITION.
C   Y(6) IS DEFINED AS 0.0D0 WHEN THE VALVE IS OPEN, RESTING ON THE STOP
C   POSITIVE DISPLACEMENT IS DEFINED AS DECREASING AIR GAP.
       IF (ISTAT(1) .EQ. 1 .AND. ISTAT(2) .EQ. 1) GO TO 610
       IF (ISTAT(2) .EQ. 0) GO TO 620
C
 600   ZOUT(6) = KSTOP*Y(6) - KSPRNG*Y(6) + PRELD
       GO TO 700
C
 610   ZOUT(6) = KSPRNG*Y(6) + PRELD
       GO TO 700
C
 620   ZOUT(6) = KSTOP*(Y(6)-STROKE) + KSPRNG*Y(6) + PRELD
 700   CONTINUE
C
C-- Z(7) IS THE VELOCITY OF THE VALVE
C   NOTE: THE INERTIA OF THE FLUID IS COMBINED WITH THAT OF THE VALVE TO
C         AVOID A DEPENDENT MASS CONDITION.
       IF (Y(6) .LT. 0.0D0) FMAS2 = CFMAS2/5.0D-6
       IF (Y(6) .GT. 0.0D0) FMAS2 = CFMAS2/(Y(6) + 5.0D-6)
       FMAS1 = CFMAS1/AG
       ZOUT(7) = Y(7)/(VMASS+FMAS1+FMAS2)
C
C-- Z(8)  IS THE MAGNETIC FORCE AT THE AIR GAPS.
       FLUX2=-.50D0*Y(5)*Y(5)
       ZOUT(8)=FLUX2*(DRDX(1)+DRDX(2))
```

```
C
      AUX(1)= Y(1)/A(1,1)
      AUX(2)= Y(1)/A(1,2)
      AUX(3)= Y(1)/A(1,3)
      AUX(5)= UOUT(1)-DOUT(1)
      AUX(6)= Y(3)/A(3,2)
      AUX(7)= Y(5)/A(5,3)
      AUX(8)= REL(5,1)+REL(5,2)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 1000 RETURN
      END
C
      SUBROUTINE RCHK(T,Y,YDOT,G,IGFLAG)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC                                                                      CC
CC-- NAME: RCHK                                                         CC
CC                                                                      CC
CC-- FUNCTIONS:  1. COMPUTES A FUNCTION 'G' SO THAT PARAMETER VALUES    CC
CC                  CAN BE SWITCHED WHEN 'G' CHANGES SIGN
CC               2. ISTAT(*) VALUES ARE SWITCHED AT THIS TIME           CC
CC                                                                      CC
CC-- SUBROUTINES CALLED: ZVECTR,UVECTR,VECMUL                           CC
CC                                                                      CC
CC-- VARIABLE DEFINITIONS                                               CC
CC                                                                      CC
CC         T: THE INDEPENDENT VARIABLE OF INTEGRATION                   CC
CC         Y: THE STATE VECTOR                                          CC
CC      YDOT: THE TIME DERIVATIVE OF THE STATE VECTOR                   CC
CC         G: ROOT FUNCTION VALUE                                       CC
CC     IGFLAG: IDENTIFIES WHICH ROOT FUNCTION IS TO BE EVALUATED OR     CC
CC             CHANGED                                                  CC
CC                                                                      CC
CC-- PROGRAMMER: N. HENDRIKSMA  1984                                    CC
CC                                                                      CC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION Y(NY),YDOT(NY)
      COMMON /LUDEF/ LUSN,LUIN,LUOT,LUJS
      COMMON /FLAG/ IBEGIN,JSOPT
      COMMON /SYSTEM/ NZ,NU,NDNL,NDL,NT,NY,NZPNU,N1,N1MX,N2MX,
     +                ZOUT(15),UOUT(5),DOUT(10),AUX(10),
     +                S1(25,25),JS1D(25,10)
      COMMON /ZVEC/ AG,STROKE
      COMMON /DVEC/ CFMAS1,CFMAS2,VBAT,VZENR
      COMMON /STATUS/ ISTAT(16),NROOTS
      DATA IONE/1/
```

```
C
C---- ROOT CHECKING ROUTINE FOR SANDIA PACKAGE
C
C-- THERE CAN BE A MAXIMUM OF 16 STATUS FLAGS IN ALL, DEPENDING ON THE
C   TYPE OF OPERATING MECHANISM UNDER STUDY.
C
C   IGFLAG = 1  -  SEPARATION BETWEEN VALVE AND VALVE STOP
C
C                  ISTAT(1) = 0 -  IN CONTACT
C                           = 1 -  SEPARATED
C
C   IGFLAG = 2  -  SEPARATION BETWEEN VALVE AND VALVE SEAT
C
C                  ISTAT(2) = 0 -  IN CONTACT
C                           = 1 -  SEPARATED
C
C   IGFLAG = 3  -  STATE OF CURRENT CONTROL TRANSISTOR
C
C                  ISTAT(3) = 0 -  REGULATING
C                           = 1 -  SATURATED
C
C   IGFLAG = 4  -  STATE OF ZENER DIODE
C
C                  ISTAT(4) = 0 -  OFF
C                           = 1 -  ON
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C-- FIRST TIME THRU CHECK VALIDITY OF STATUS FLAGS
      IF (IBEGIN .EQ. 0) GO TO 100
      IBEGIN = 0
C-- CHECK VALVE/STOP SEPARATION
      G = Y(6) + (2*ISTAT(1) -1)*1.0D-8
      IF (G .LT. 0.0D0) ISTAT(1) = 0
      IF (G .GE. 0.0D0) ISTAT(1) = 1
C
C-- CHECK VALVE/SEAT SEPARATION
      G = STROKE - Y(6) + (2*ISTAT(2) -1)*1.0D-8
      IF (G .LT. 0.0D0) ISTAT(2) = 0
      IF (G .GE. 0.0D0) ISTAT(2) = 1
C
C-- CHECK TRANSITOR STATE
      CALL ZVECTR(T,Y)
      CALL UVECTR(T,Y)
      CALL VECMUL(NZPNU,IONE,NZPNU,DIN)
      G = DIN - VBAT + (2*ISTAT(3) -1)*1.0D-8
      IF (G .LT. 0.0D0) ISTAT(3) = 0
      IF (G .GE. 0.0D0) ISTAT(3) = 1
C
C-- CHECK ZENER STATE
      G = DIN - VZENR - (2*ISTAT(4) -1)*1.0D-8
```

```
      IF (G .LT. 0.0D0) ISTAT(4) = 1
      IF (G .GE. 0.0D0) ISTAT(4) = 0
C
      WRITE(LUSN,60) (ISTAT(I),I=1,NROOTS)
 60   FORMAT(//,' STATUS FLAGS AT INITIAL CONDITIONS: ',5I3)
      GO TO 1000
C
C-- END OF INITIALIZATION
C
C-- IF IGFLAG IS NEGATIVE, THE CORRESPONDING ISTAT IS TO BE CHANGED
 100  IF(IGFLAG .LT. 0) GO TO 500
C
C-- IF IGFLAG IS POSITIVE, THE CORRESPONDING ROOT FUNCTION IS TO BE
C   EVALUATED
      GO TO (110,120,130,140,498), IGFLAG
C
C-- CHECK VALVE/STOP SEPARATION
 110  G = Y(6) + (2*ISTAT(1) -1)*1.0D-8
      GO TO 1000
C
C-- CHECK VALVE/SEAT SEPARATION
 120  G = STROKE - Y(6) + (2*ISTAT(2) -1)*1.0D-8
      GO TO 1000
C
C-- CHECK TRANSITOR STATE
 130  CALL ZVECTR(T,Y)
      CALL UVECTR(T,Y)
      CALL VECMUL(NZPNU,IONE,NZPNU,DIN)
      G = DIN - VBAT + (2*ISTAT(3) -1)*1.0D-8
      GO TO 1000
C
C-- CHECK ZENER STATE
 140  CALL ZVECTR(T,Y)
      CALL UVECTR(T,Y)
      CALL VECMUL(NZPNU,IONE,NZPNU,DIN)
      G = DIN - VZENR - (2*ISTAT(4) -1)*1.0D-8
      GO TO 1000
C
C-- ERROR
 498  WRITE(LUSN,499) T,IGFLAG
 499  FORMAT(' ERROR IN RCHK AT T = ',E12.4,' IGFLAG = ',I6)
      STOP
C
C
C---- STATUS CHANGE
 500  IGFLAG = -1*IGFLAG
      I = IGFLAG
C
      IF (ISTAT(I) .EQ. 0) GO TO 510
      ISTAT(I) = 0
      GO TO 1000
```

```
 510   ISTAT(I) = 1
C
 1000  RETURN
       END
       SUBROUTINE COEFF(NMATLS,LU)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC                                                                      CC
CC-- NAME: COEFF                                                        CC
CC                                                                      CC
CC-- FUNCTIONS:  1. COMPUTES COEFFICIENTS FOR CUBIC SPLINES TO THE      CC
CC                  USER SUPPLIED B-H DATA.  THE EQUATIONS ARE FOR      CC
CC                  PERMEABILITY, MU, AS A FUNCTION OF FLUX DENSITY,    CC
CC                  B.   CURRENTLY SET UP FOR UP TO 4 DIFFERENT         CC
CC                  MATERIALS WITH UP TO 20 DATA POINTS EACH.           CC
CC                  THE B-H DATA IS INPUT WITH UNITS OF KILOGAUSS AND   CC
CC                  OESTEDS AND CONVERTED TO MKS UNITS.                 CC
CC                                                                      CC
CC-- SUBROUTINES CALLED: NONE                                           CC
CC                                                                      CC
CC-- VARIABLE DEFINITIONS                                               CC
CC                                                                      CC
CC     NMATLS: THE NUMBER OF CURVES TO WHICH SPLINE EQUATIONS ARE TO    CC
CC             BE FITTED.                                               CC
CC         LU: THE LOGICAL UNIT NUMBER FROM WHICH THE B-H DATA IS TO    CC
CC             READ.                                                    CC
CC                                                                      CC
CC-- REFERENCES:  "ELEMENTARY NUMERICAL ANALYSIS" BY CONTE & DE BOOR    CC
CC                 MCGRAW-HILL, 1980   PAGE 290                         CC
CC                                                                      CC
CC-- PROGRAMMER: N. HENDRIKSMA  1984                                    CC
CC                                                                      CC
CC                                                                      CC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
       IMPLICIT REAL*8 (A-H,O-Z)
       DIMENSION  D(20),DIAG(20)
       REAL*8 MU
       COMMON /FIT/ C(16,20),BI(4,20),NPTS(4)
       COMMON /LUDEF/ LUSN,LUIN,LUOT,LUJS
C
       DO 1000 JJ = 1, NMATLS
       J = 1 + 4 * (JJ - 1)
C
C-- READ # POINTS PER CURVE AND INPUT THE B-H DATA
       READ(LU,*) NPTS(JJ)
       NUMPTS =NPTS(JJ)
C
C-- IF NUMPTS IS 1  IT INDICATES A LINEAR MATERIAL AND MU IS READ
       IF (NUMPTS .EQ. 1) GO TO 70
```

```
      IF (NUMPTS .LT. 21) GO TO 3
C
      WRITE(1,1)
 1    FORMAT(//1X,'TOO MANY DATA POINTS.  ONLY FIRST 20 USED',//)
      NUMPTS = 20
C
C-- DATA IS CONVERTED TO MKS UNITS FROM KILOGAUSS AND OERSTEDS
C   THE CURVE IS FIT AS MU VS B
 3    DO 9 I = 1, NUMPTS
      READ(LU,*) BB,HH
      BB = BB * .1D0
      HH = HH * 79.527D0
C
      BI(JJ,I)=BB
      C(J,I) = BB/HH
C
C-- IF B IS ZERO THEN THE FUNCTION VALUE IS THE RELATIVE INITIAL
C   PERMEABILITY.   THIS IS CONVERTED TO MKS UNITS.
      IF (BB .NE. 0.0D0) GO TO 9
      C(J,I) = (HH/79.527D0)*1.2567D-6
      C(J+1,I) = 0.0D0
      C(J+2,I) = 0.0D0
      C(J+3,I) = 0.0D0
C
 9    CONTINUE
C
      N = NPTS(JJ) - 1
      DIAG(1) = 1.0D0
      D(1) = 0.0D0
C
C-- CALCULATE THE BOUNDRY SLOPE A THE END POINTS BY USING THE SLOPE OVER
C   THE FIRST AND LAST INTERVAL
      C(J+1,1)=(C(J,2)-C(J,1))/(BI(JJ,2)-BI(JJ,1))
      C(J+1,N+1)=(C(J,N+1)-C(J,N))/(BI(JJ,N+1)-BI(JJ,N))
C
C-- THIS CODE BASED ON ALGORITHM FOR CUBIC SPLINE COEFFICIENTS IN
C   "ELEMENTARY NUMERICAL ANALYSIS"    MCGRAW-HILL 1980   N.Y.
C   BY CONTE AND DEBOER  (MODIFIED FOR FORTRAN IV)   PAGE 290,291
      DO 10 M = 2, NUMPTS
      D(M) = BI(JJ,M) - BI(JJ,M-1)
      DIAG(M) = (C(J,M) - C(J,M-1))/D(M)
 10   CONTINUE
C
      DO 20 M = 2,N
      C(J+1,M)=3.0D0*(D(M)*DIAG(M+1) + D(M+1)*DIAG(M))
      DIAG(M) =2.0D0*(D(M)+D(M+1))
 20   CONTINUE
C
      DO 30 M = 2,N
      G = -1.0D0*D(M+1)/DIAG(M-1)
      DIAG(M) = DIAG(M)+G*D(M-1)
```

```
      C(J+1,M)=C(J+1,M)+G*C(J+1,M-1)
 30   CONTINUE
C
      M = NPTS(JJ)
 40   M = M -1
      C(J+1,M)=(C(J+1,M)-D(M)*C(J+1,M+1))/DIAG(M)
      IF (M .EQ. 2) GO TO 50
      GO TO 40
C
C-- THIS IS SUBROUTINE "CALCCF"   PAGE 287    CONTE & DEBOER
 50   DO 60 I = 1,N
      DX=BI(JJ,I+1)-BI(JJ,I)
      DIVDF1=(C(J,I+1)-C(J,I))/DX
      DIVDF3=C(J+1,I)+C(J+1,I+1)-2.0D0*DIVDF1
      C(J+2,I)=(DIVDF1-C(J+1,I)-DIVDF3)/DX
      C(J+3,I)=DIVDF3/(DX*DX)
 60   CONTINUE
      GO TO 1000
C
C-- THIS READS IN THE SLOPE FOR A LINEAR MAGNETIC MATERIAL
 70   READ(LU,*) MU
      C(J+1,1) = MU
C
 1000 CONTINUE
C
      RETURN
      END
C
      SUBROUTINE METREL(IPATH,M,FLUX,REL)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC                                                                      CC
CC-- NAME: METREL                                                       CC
CC                                                                      CC
CC-- FUNCTIONS:  1. COMPUTES THE RELUCTANCE OF METALLIC ELEMENTS        CC
CC                  AS A FUNCTION OF THE FLUX THRU THE ELEMENT          CC
CC                  USING THE CUBIC SPLINE EQUATIONS FROM THE COEFF     CC
CC                  SUBROUTINE.                                         CC
CC                                                                      CC
CC-- SUBROUTINES CALLED: NONE                                           CC
CC                                                                      CC
CC-- VARIABLE DEFINITIONS                                               CC
CC                                                                      CC
CC     IPATH: DENOTES THE FLUX PATH TO BE EVALUATED                     CC
CC         M: DENOTES THE ELEMENT IN THE FLUX PATH                      CC
CC      FLUX: THE FLUX LEVEL IN THE FLUX PATH                           CC
CC       REL: THE RELUCTANCE OF THE ELEMENT...OUTPUT OF THE SUBROUTINECC
CC                                                                      CC
CC-- PROGRAMMER: N. HENDRIKSMA 1984                                     CC
CC                                                                      CC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 MU
      COMMON /LUDEF/ LUSN,LUIN,LUOT,LUJS
      COMMON /FIT/ C(16,20),BI(4,20),NPTS(4)
      COMMON /METAL/ A(5,3),FPLA(5,3),MATL(5,3)
C
      ML = MATL(IPATH,M)
      K = 1 + 4 * (ML-1)
      NUMPTS = NPTS(ML)
C
C-- FIRST COMPUTE B FOR THE ELEMENT FROM THE GIVEN FLUX
      B = FLUX / A(IPATH,M)
      BABS = DABS(B)
C
C-- CHECK FOR LINEAR MATERIAL    ( NUMPTS = 1  IS FLAG )
      IF (NUMPTS .EQ. 1) GO TO 300
C
C-- IF BABS IS GREATER THAN MAX TABLE VALUE NOTIFY USER AND STOP.
      IF (BABS .LE. BI(ML,NUMPTS)) GO TO 5
C
      WRITE(LUSN,2) IPATH,M
 2    FORMAT(/,'EXCEEDED MAX VALUE IN B-H TABLE FOR PATH',I3,' ELEMENT
     + ',I2,//,'  EXECUTION HALTED',/)
      STOP
C
C-- FIND THE INTERVAL CONTAINING BABS
 5    I =INT(NUMPTS/2.)
      IF (BABS .GE. BI(ML,I)) GO TO 50
      J=I
 10   J=J-1
      IF (BABS .GE. BI(ML,J)) GO TO 100
      GO TO 10
C
 50   DO 20 J = I,NUMPTS
      IF (J .EQ. NUMPTS) GO TO 100
      IF (BABS .LT. BI(ML,J+1)) GO TO 100
 20   CONTINUE
C
 100  DX = BABS - BI(ML,J)
      MU =C(K,J)+DX*(C(K+1,J)+DX*(C(K+2,J)+DX*C(K+3,J)))
C     DMUDB =C(K+1,J)+DX*(2.0D0*C(K+2,J)+DX*3.0D0*C(K+3,J))
      IF (MU .EQ. 0.0D0) GO TO 203
      REL =FPLA(IPATH,M)/MU
      GO TO  1000
C
 203  WRITE(LUSN,201) ML
 201  FORMAT(//,'ERROR IN SUBROUTINE METREL.  MU = 0. FOR MATL ',I2)
      WRITE(LUSN,202) BABS
 202  FORMAT(//,'FLUX DENSITY WAS ',E15.7,/
```

```
     1 /,'PROGRAM EXECUTION HALTED.')
       STOP
C
C-- THIS HANDLES THE LINEAR MATERIALS
 300   MU = C(K+1,1)
C      DMUDB = 0.0D0
       IF (MU .EQ. 0.0D0) GO TO 203
       REL =FPLA(IPATH,M)/MU
C
 1000  RETURN
C
       END
C
       SUBROUTINE AIRREL(NAG,G,REL,DRDX)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC                                                                          CC
CC-- NAME: AIRREL                                                           CC
CC                                                                          CC
CC-- FUNCTIONS:  1. COMPUTES THE RELUCTANCE OF AIR GAPS AS A FUNCTION       CC
CC                  OF THE GAP LENGTH.                                      CC
CC               2. COMPUTES THE RATE OF CHANGE OF THE RELUCTANCE AS        CC
CC                  FUCTION OF THE GAP LENGTH, DR/DX                        CC
CC                                                                          CC
CC-- SUBROUTINES CALLED: NONE                                               CC
CC                                                                          CC
CC-- VARIABLE DEFINITIONS                                                   CC
CC                                                                          CC
CC        NAG: IDENTIFIES THE AIR GAP TO BE EVALUATED                       CC
CC         AG: THE AIR GAP LENGTH IN METERS                                 CC
CC        REL: THE RELUCTANCE OF THE AIR GAP -OUTPUT OF SUBROUTINE.         CC
CC       DRDX: THE RATE OF CHANGE OF "REL" AS A FUNCTION OF "AG"            CC
CC             AND IS AN OUTPUT OF THE SUBROUTINE.                          CC
CC                                                                          CC
CC-- REFERENCES: "ELECTROMAGNETIC DEVICES"  BY H. C. ROTORS                 CC
CC                JOHN WILEY AND SONS, NEW YORK  1941                       CC
CC                                                                          CC
CC-- PROGRAMMER: N. HENDRIKSMA  1984                                        CC
CC                                                                          CC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
       IMPLICIT REAL*8 (A-H,O-Z)
       COMMON /AGPAR/ TA,T12,T18W,T18L,
     1                P1C(2),P12BC(2),P18BCW(2),P18BCL(2),P8BCW(2),
     2                P8BCL(2)
C
C--  NOTE:  THE PERMEANCE OF EACH PATH AND DPDX IS CALCULATED AND
C           SUMMED TO COMPUTE THE RELUCTANCE AND DRDX.
C
       I = NAG
```

```
C
C-- COMPUTE MAIN PERMEANCE AND DERIVATIVE       PATH P1
      PMAIN = P1C(I)/G
      DPMAIN = -1.0D0*PMAIN/G
C
C-- COMPUTE 'INNER' FRINGING PERMEANCE AND DERIVATIVE       PATH P12B
      T12G = T12/G
      IF (T12G .LT. 1.0D0) T12G = 1.0D0
      P12B = P12BC(I)*DLOG(T12G)
      DP12B = -1.0D0*P12BC(I)/G
C
C-- COMPUTE PERMEANCE AND DERIVATIVE THRU PATH P18B
      T18WG = T18W/G
      IF (T18WG .LT. 1.0D0) T18WG = 1.0D0
      P18BW = P18BCW(I)*DLOG(T18WG)
      DP18BW = -1.0D0*P18BCW(I)/G
C
      T18LG = T18L/G
      IF (T18LG .LT. 1.0D0) T18LG = 1.0D0
      P18BL = P18BCL(I)*DLOG(T18LG)
      DP18BL = -1.0D0*P18BCL(I)/G
C
C-- COMPUTE PERMEANCE AND DERIVATIVE THRU PATH P8B
      IF (G .GT. T18W) GO TO 100
      TEMPW = (2.0D0*(TA+G)-T18W)/T18W
      P8BW = P8BCW(I)*DLOG(TEMPW)
      DP8BW = P8BCW(I)*2.0D0/(TEMPW*T18W)
      GO TO 200
C
 100  TEMPW = 2.0D0*TA/G
      P8BW = P8BCW(I)*DLOG(TEMPW + 1.0D0)
      DTEMPW = -1.0D0*TEMPW/G
      DP8BW = P8BCW(I)*DTEMPW/(TEMPW + 1.0D0)
C
 200  IF (G .GT. T18L) GO TO 300
      TEMPL = (2.0D0*(TA+G)-T18L)/T18L
      P8BL = P8BCL(I)*DLOG(TEMPL)
      DP8BL = P8BCL(I)*2./(TEMPL*T18L)
      GO TO 400
C
 300  TEMPL = 2.0D0*TA/G
      P8BL = P8BCL(I)*DLOG(TEMPL + 1.0D0)
      DTEMPL = -1.0D0*TEMPL/G
      DP8BL = P8BCL(I)*DTEMPL/(TEMPL+ 1.0D0)
C
C-- SUM TOTAL PERMEANCE AND DERIVATIVE  ( PARALLEL PERMEANCE ADD )
 400  PTOT = PMAIN+P12B+P18BW+P18BL+P8BW+P8BL
      DPTOT = DPMAIN+DP12B+DP18BW+DP18BL+DP8BW+DP8BL
C
C
C-- COMPUTE RELUCTANCE AND DERIVATIVE DRDX
```

```
      REL = 1.0D0/PTOT
      DRDX = -1.0D0*DPTOT/(PTOT*PTOT)
C
      RETURN
      END
      SUBROUTINE UVECTR(TIME,Y)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC                                                                              CC
CC-- NAME: UVECTR                                                               CC
CC                                                                              CC
CC-- FUNCTIONS:  1. INITIALIZES SOURCE PARAMETERS.                              CC
CC               2. COMPUTES THE SOURCE VALUES                                  CC
CC                                                                              CC
CC-- SUBROUTINES CALLED: NONE                                                   CC
CC                                                                              CC
CC-- VARIABLE DEFINITIONS                                                       CC
CC                                                                              CC
CC      TIME: THE INDEPENDENT VARIABLE OF INTEGRATION                           CC
CC         Y: THE STATE VECTOR                                                  CC
CC                                                                              CC
CC-- PROGRAMMER: N. HENDRIKSMA  1984                                            CC
CC                                                                              CC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      COMMON /FLAG/ IBEGIN,JSOPT
      COMMON /LUDEF/ LUSN,LUIN,LUOT,LUJS
      COMMON /SYSTEM/ NZ,NU,NDNL,NDL,NT,NY,NZPNU,N1,N1MX,N2MX,
     +                ZOUT(15),UOUT(5),DOUT(10),AUX(10),
     +                S1(25,25),JS1D(25,10)
      DIMENSION Y(NY)
      DATA UVEC/'UVEC'/
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C-------- INITIALIZATION
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IF (IBEGIN .EQ. 0) GO TO 500
      READ(LUIN,9) RCHCK
      IF (RCHCK .EQ.  JS ) GO TO 13
      WRITE(LUSN,8) RCHCK
 8    FORMAT( ' ERROR ON INPUT TO SUBROUTINE UVECTR AT START OF BLOCK'
     + ,/ ,'     RCHCK = ', A4)
      CALL EXIT
 9    FORMAT(A4)
 13   READ(LUIN,11)
 11   FORMAT(I3)
      READ(LUIN,*) CURR,PW
```

```
      GO TO 1000
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C----------- COMPUTE THE SOURCES
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
 500   UOUT(1)=CURR
       IF(TIME .GE. PW) UOUT(1) = 0.0D0
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 1000  RETURN
       END
       SUBROUTINE DVECTR(DIN,Y)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC                                                                    CC
CC-- NAME: DVECTR                                                     CC
CC                                                                    CC
CC-- FUNCTIONS:  1. INITIALIZES DISSIPATION PARAMETERS.               CC
CC               2. COMPUTES THE OUTPUTS OF THE NONLINEAR DISSIPATION CC
CC                  FIELD AS A FUNCTION OF THE INPUTS                 CC
CC                                                                    CC
CC-- SUBROUTINES CALLED: VECMUL                                       CC
CC                                                                    CC
CC-- VARIABLE DEFINITIONS                                             CC
CC                                                                    CC
CC        DIN:  VECTOR OF INPUTS TO THE NONLINEAR DISSIPATION FIELD   CC
CC          Y:  STATE VECTOR                                          CC
CC                                                                    CC
CC-- PROGRAMMER: N. HENDRIKSMA 1984                                    C
CC                                                                    CC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
       IMPLICIT REAL*8 (A-H,O-Z)
C
       COMMON /FLAG/ IBEGIN,JSOPT
       COMMON /LUDEF/ LUSN,LUIN,LUOT,LUJS
       COMMON /SYSTEM/ NZ,NU,NDNL,NDL,NT,NY,NZPNU,N1,N1MX,N2MX,
     +                 ZOUT(15),UOUT(5),DOUT(10),AUX(10),
     +                 S1(25,25),JS1D(25,10)
       COMMON /LMATS/ RL(10,10),TM(10,10)
       COMMON /ZVEC/ AG,STROKE
       COMMON /DVEC/ CFMAS1,CFMAS2,VBAT,VZENR
       COMMON /STATUS/ ISTAT(16),NROOTS
       DIMENSION Y(NY),DIN(NDNL)
       REAL*8 LWIRE
C
       DATA PIE/3.141592654D0/,IONE/1/,DVEC/'DVEC'/
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C------------- INITIALIZATION
```

```
C
C------ NONLINEAR DISSIPATION PARAMETERS
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
       IF (IBEGIN .EQ. 0) GO TO 500
C
       READ(LUIN,9) RCHCK
       IF (RCHCK .EQ. DVEC) GO TO 13
       WRITE(LUSN,8) RCHCK
 8     FORMAT( ' ERROR ON INPUT TO SUBROUTINE DVECTR AT START OF BLOCK'
     + ,/ ,'     RCHCK = ', A4)
       CALL EXIT
  9    FORMAT(A4)
 13    READ(LUIN,10)
 10    FORMAT(/I3)
 11    FORMAT(I3)
C
C-- PARAMETERS FOR BOND 10
       READ(LUIN,*) VZENR,VBAT
C
C-- PARAMETERS FOR BOND 11    SET UP COEFFICIENTS FOR FLUID DAMPING
       READ(LUIN,11)
       READ(LUIN,*) DENSTY,RARM,SRAD,VISCOS,RLSTOP,RUSTOP
       PIER4 = PIE*RARM**4
       PIER4S= PIE*SRAD**4
       PAR21 = 3.0D0*VISCOS*PIER4/2.0D0
       PAR21S= 3.0D0*VISCOS*PIER4S/2.0D0
       PAR22 = DENSTY*PIER4
       PAR22S= DENSTY*PIER4S
       CFMAS1= 3.0D0*DENSTY*PIER4/20.0D0
       CFMAS2= 3.0D0*DENSTY*PIER4S/20.0D0
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C---------- LINEAR RESISTANCES    SET UP THE RL MATRIX
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
       CALL ZMAT(N2MX,NDL,NDL,RL)
C
C-- LUMPED EDDY CURRENT RESISTANCES BONDS 11,12 ( STORE AS CONDUCTANCE )
       READ(LUIN,10)
       READ(LUIN,*) RL(1,1),RL(2,2),RL(3,3)
       RL(1,1)=1.0D0/RL(1,1)
       RL(2,2)=1.0D0/RL(2,2)
       RL(3,3)=1.0D0/RL(3,3)
C
C-- RESISTANCE FOR BOND 13     ( CURVE FIT EQN USED TO COMPUTE LEAD
C   WIRE RESISTANCE,    LENGTH IS ONE WAY LENGTH )
       READ(LUIN,11)
       READ(LUIN,*) RCOIL,GWIRE,LWIRE,RCIRC
C
       RLEAD=2.0D0*LWIRE/(-955.4D0+32.5D0*GWIRE-.3802D0*GWIRE*GWIRE
     + +9737.0D0/GWIRE)
```

```
      RL(4,4) = RCOIL+RLEAD+RCIRC
C
C-- INITIALIZE STATUS OF 'TRANSISTOR/ZENER ELEMENT
C   SET FOR SATURATED TRANSISTOR AND ZENER DIODE OFF
      ISTAT(3) = 1
      ISTAT(4) = 0
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C---------- END OF INITIALIZATION
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      GO TO 1000
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C-------------- COMPUTE THE DOUT-VECTOR
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
 500  CONTINUE
C
C-- COMPUTE DOUT(1) BASED ON STATUS FLAGS
C    NOTE: ISTAT(3) = 0 -->  REGULATION
C          ISTAT(4) = 0 -->  ZENR DIODE OFF
C
      IF (ISTAT(3) .EQ. 1 .AND. ISTAT(4) .EQ. 1)
     + WRITE(LUSN,502)
 502  FORMAT('ERROR IN DVECTR.   ISTAT(3)=ISTAT(4)=1 ')
      IF (ISTAT(3) .EQ. 0 .AND. ISTAT(4) .EQ. 0) GO TO 600
C
C-- COMPUTE DIN(1) BASED ON DOUT(1)  = 0.
      CALL VECMUL(NZPNU,IONE,NZPNU,DIN)
C
      IF (ISTAT(3) .EQ. 1 .AND. ISTAT(4) .EQ. 0) GO TO 550
C-- ZENER DIODE TURNED ON, TRANSISTOR REGULATING
      DOUT(1) = (VZENR-DIN(1))/S1(10,10)
      GO TO 610
C
C-- ZENER DIODE OFF, TRANSISTOR SATURATED
 550  DOUT(1) = (VBAT-DIN(1))/S1(10,10)
      GO TO 610
C
C-- ZENER DIODE OFF, TRANSISTOR REGULATING
 600  DOUT(1) = 0.0D0
 610  CONTINUE
C
C-- COMPUTE NEW DIN BASED ON COMPUTED DOUT  DIN(2) CALCULATED HERE AS
C   WELL
      CALL VECMUL(NZPNU,NDNL,N1,DIN)
      AUX(4) = DIN(1)-RCIRC*(UOUT(1)-DOUT(1))
C
C-- OUTPUT ON BOND 11  ( FORCE )
C-- NOTE: ISTAT(*) = 0 INDICATES CONTACT AT STOP OR SEAT
```

```
      IF(ISTAT(1) .EQ. 1 .AND. ISTAT(2) .EQ. 1) GO TO 200
      IF(ISTAT(2) .EQ. 0) GO TO 300
C
C-- CONTACT AT THE LOWER STOP
 100  DOUT(2) = RLSTOP*DIN(2)
      GO TO 400
C
C-- VALVE IS BETWEEN STOPS
 200  AG2=AG*AG
C-- A SMALL GAP IS ADDED TO PREVENT A ZERO DIVIDE FOR Y(6) = 0.
      STRK = Y(6) + 5.0D-6
      STRK2 = STRK*STRK
      AG3=AG2*AG
      STRK3 = STRK2*STRK
      C1=PAR21*DIN(2)
      C1S=PAR21S*DIN(2)
      C2=PAR22*DIN(2)*DIN(2)
      C2S=PAR22S*DIN(2)*DIN(2)
C
      IF (DIN(2) .GT. 0.0D0) GO TO 220
C-- NEGATIVE VELOCITIES  I.E.  AIR GAP INCREASING
      DOUT(2) = C1/AG3 - (C2/(7.0D0*AG2))
     1        + C1S/STRK3 - (C2S/(3.75D0*STRK2))
      GO TO 400
C-- POSITIVE VELOCITIES  I.E.  AIR GAP DECREASING
 220  DOUT(2) = C1/AG3 + (C2/(3.75D0*AG2))
     1        + C1S/STRK3 + (C2S/(7.0D0*STRK2))
      GO TO 400
C
C-- CONTACT AT UPPER STOP
 300  DOUT(2) = RUSTOP*DIN(2)
C
 400  CONTINUE
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C-------------- DONE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
 1000 RETURN
      END
      SUBROUTINE TVECTR
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC                                                                          CC
CC-- NAME: TVECTR                                                           CC
CC                                                                          CC
CC-- FUNCTIONS:  1. INITIALIZES THE GYRATOR MODULII                         CC
CC               2. SETS UP THE CONSTITUTIVE MATRIX, TM                     CC
CC                                                                          CC
CC-- SUBROUTINES CALLED: ZMAT                                               CC
CC                                                                          CC
```

```
CC-- PROGRAMMER: N. HENDRIKSMA  1984                                    CC
CC                                                                      CC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      COMMON /FLAG/ IBEGIN,JSOPT
      COMMON /LUDEF/ LUSN,LUIN,LUOT,LUJS
      COMMON /SYSTEM/ NZ,NU,NDNL,NDL,NT,NY,NZPNU,N1,N1MX,N2MX,
     +                ZOUT(15),UOUT(5),DOUT(10),AUX(10),
     +                S1(25,25),JS1D(25,10)
      COMMON /LMATS/ RL(10,10),TM(10,10)
      DATA TVEC/'TVEC'/
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C-------- INITIALIZATION
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C-- READ THE GYRATOR/TRANSFORMER MODULI AND FORM THE "TM" MATRIX
      CALL ZMAT(N2MX,NTB,NTB,TM)
C
      READ(LUIN,9) RCHCK
      IF (RCHCK .EQ. TVEC) GO TO 13
      WRITE(LUSN,8) RCHCK
 8    FORMAT( ' ERROR ON INPUT TO SUBROUTINE TVECTR AT START OF BLOCK'
     + ,/ ,'     RCHCK = ', A4)
      CALL EXIT
 9    FORMAT(A4)
 13   READ(LUIN,11)
 11   FORMAT(I3)
      READ(LUIN,*) R1,R2
      TM(1,2)=R1
      TM(2,1)=R1
      TM(3,4)=R2
      TM(4,3)=R2
C
      RETURN
      END
```