

AD-A082 087

FOSTER-MILLER ASSOCIATES INC WALTHAM MASS
ANALYSIS OF TRUNK FLUTTER IN AN AIR CUSHION LANDING SYSTEM. USE--ETC(U)
NOV 79 R B FISH F33615-78-C-3412

F/6 1/3

UNCLASSIFIED

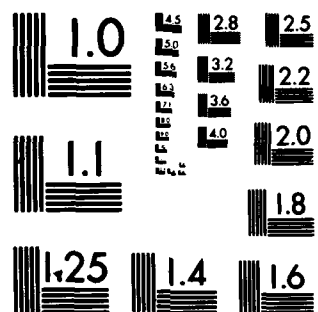
AFFDL-TR-79-3121

NL

1-2

1-2

1-2



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AFFDL-TR-79-3121

AD A082087

LEVEL

②
B.S.

**TRUNK FLUTTER ANALYSIS PROGRAM
USER'S MANUAL**

*FOSTER-MILLER ASSOCIATES, INC.
350 SECOND AVENUE
WALTHAM, MA 02154*

STIC
ELECTE
MAR 20 1980
S D C

NOVEMBER 1979

TECHNICAL REPORT AFFDL-TR-79-3121
USER'S MANUAL
Final Report for period May 1978 — July 1979

Approved for public release; distribution unlimited.

AIR FORCE FLIGHT DYNAMICS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

DDC FILE COPY

80 3 19 005

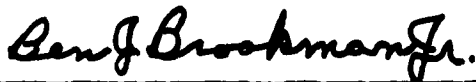
61102F

NOTI

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

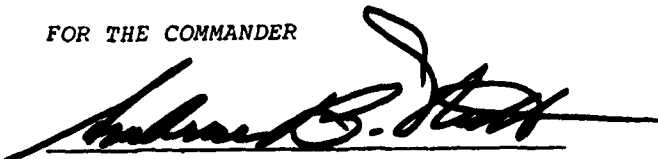


BEN J. BROOKMAN, JR.
Project Engineer



HOWELL K. BREWER
Chief, Mechanical Branch
Vehicle Equipment Division

FOR THE COMMANDER



AMBROSE B. NUTT
Director, Vehicle Equipment Division
AF Flight Dynamics Laboratory

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFEDL/FEMB, W-PAFB, OH 45433 to help us maintain a current mailing list".

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM	
1. REPORT NUMBER AFFDL-TR-79-3121	2. GOVT ACCESSION NO.	3. PERFORMING ORG. REPORT NUMBER Final report	
4. TITLE (and Subtitle) User's Manual for Analysis of Trunk Flutter in Air Cushion Landing System User's Manual		5. TYPE OF REPORT & PERIOD COVERED User's Manual (Computer Program) May 78 - Jul 79	
6. AUTHOR(s) Roger B. Fish		7. CONTRACT OR GRANT NUMBER(s) F33615-78-C-3412	
8. PERFORMING ORGANIZATION NAME AND ADDRESS Foster-Miller Associates, Inc. 350 Second Avenue Waltham, MA 02154		9. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 2307N204	
10. CONTROLLING OFFICE NAME AND ADDRESS		11. REPORT DATE November 1979	
		12. NUMBER OF PAGES 176	
13. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) United States Air Force AFSC Aeronautical Systems Division Wright-Patterson AFB, Ohio 45433		14. SECURITY CLASS. (of this report) UNCLASSIFIED	
15. DECLASSIFICATION/DOWNGRADING SCHEDULE			
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.			
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)			
18. SUPPLEMENTARY NOTES			
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Air Cushion Landing System Flutter Air Cushion Vehicles			
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This User's Manual (Computer Program) describes the computer programs for trunk flutter analysis. It includes descriptions, user instructions, and a sample run for the trunk flutter dynamic simulation program, and an eigenvalue calculation program for a linearized static trunk model. The results of the study under which this User's Manual was prepared are contained in AFFDL-TR-79-3102, "Analysis of Trunk Flutter in an Air Cushion System, dated August 1979.			

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

FOREWORD

The work in this document was performed under Contract No. F33615-78-C-3412, Work Unit No. 2307N204, "Trunk Flutter Analysis". The technical project officer of the project was Dr. Ben J. Brookman, Jr. The final report of the above contract is contained in AFFDL-TR-79-3102.

Accession For	
NTIS GMA&I	☒
DEC TAB	☐
Unannounced	☐
Justification	☐
By _____	
Distribution/ _____	
Availability Codes	
Dist	Avail and/or special
A	

TABLE OF CONTENTS

	<u>Page</u>
1. INTRODUCTION	1
2. PROGRAM ORGANIZATION	8
2.1 Program Execution Steps	8
3. PROGRAM USER INSTRUCTIONS	13
3.1 Program Input Data	13
3.1.1 Input Data Format	13
3.1.2 Internal Data	21
3.1.3 Special Notes on Input Data	21
3.1.4 Program Option Operation	24
3.2 Program Output	29
4. ILLUSTRATIVE SIMULATION	33
APPENDIX A - EQUATIONS USED IN THE DYNAMIC MODEL	44
APPENDIX B - THE EIGENVALUE PROGRAM	68
APPENDIX C - PRINCIPLE PROGRAM NOMENCLATURE	94
APPENDIX D - PROGRAM LISTINGS	99

LIST OF ILLUSTRATIONS

<u>Figure No.</u>		<u>Page</u>
1	ACLS trunk flow model.	2
2	Trunk representation for the dynamic simulation model.	3
3	Program structure.	4
4	Simulation flowchart.	9
5	Fan pressure versus flow polynomial.	19
6	Input data test case.	34
7	Program output.	36
A-1	Trunk representation for the dynamic simulation model.	45
A-2	The 2D coordinate system.	46
A-3	Trunk elasticity representation.	48
A-4	Bending force representation.	50
A-5	Bending force vectors.	52
A-6	Pressure force representation.	57
A-7	Damping representation.	58
A-8	Fluid flow representation.	60
A-9	Dynamic fluid model.	63
A-10	Fan pressure versus flow polynomial.	64
A-11	Trunk flow analysis.	66
B-1	Lateral vibration model.	70
B-2	Nodal dampers (lateral).	73
B-3	Longitudinal vibration model.	75
B-4	Nodal dampers (longitudinal).	78

LIST OF ILLUSTRATIONS (Continued)

<u>Figure No.</u>		<u>Page</u>
B-5	Element coordinate transformation.	80
B-6	Transverse node displacement.	82
B-7	Force transformation.	83

LIST OF TABLES

<u>Table No.</u>		<u>Page</u>
1	THE CAPABILITIES OF THE TRUNK FLUTTER SIMULATION PROGRAM	6
2	A SUMMARY OF SUBROUTINES	12
B-1	A SUMMARY OF EIGENVALUE PROGRAM SUBROUTINES	69

1. INTRODUCTION

This report describes a computer program developed as part of the contract F33615-78-C-3412, Trunk Flutter Analysis. The computer program simulates behavior of a two-dimensional trunk segment in presence of air flows existing in an air cushion landing system (ACLS). Through such simulations a greater understanding of trunk flutter mechanisms can be achieved and ways to eliminate flutter in future designs can be developed.

The trunk-fluid flow system addressed by the program is shown in Figure 1.* As shown in the figure the trunk is assumed to be fed by a fan with the user selected characteristics. The air supplied to the trunk by the fan flows out at two places.

- a. Through the trim valves to the cushion
- b. Through orifices at the side and the bottom of the trunk.

The cushion air flows to the atmosphere from the bottom of the trunk.

The trunk, assumed to be made of an elastic membrane with a finite mass and flexural rigidity is divided into a number of mass nodes, each connected to the other by springs as shown in Figure 2.

The separation point is assumed to be always at a particular slope of the trunk. However, in view of the additional work that needs to be done in identifying the location of the separation point, the computer program provides various options in defining the separation point location.

* See Final Report "Contract No." F33615-78-C-3412 for details of the model.

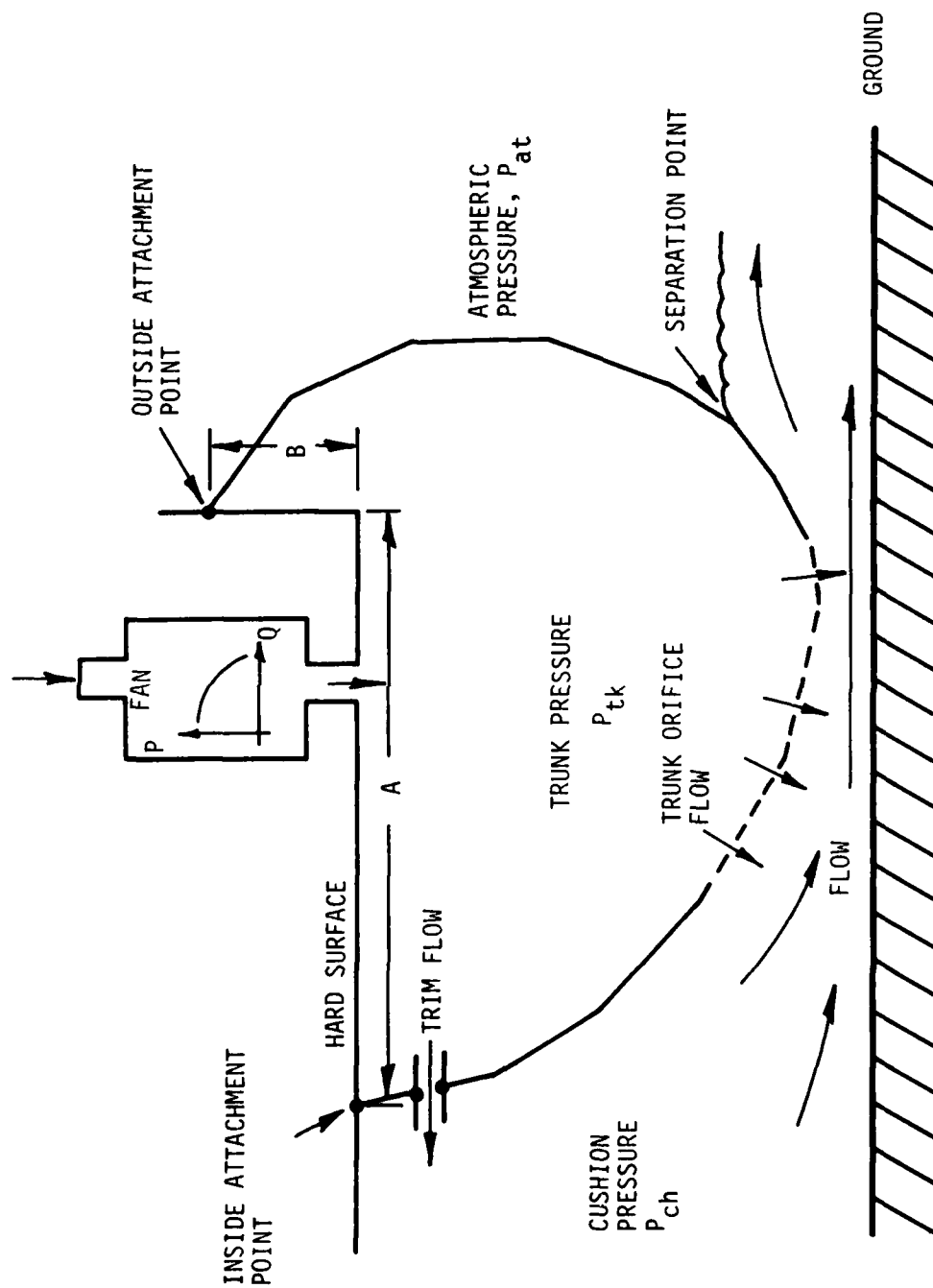


Figure 1. ACLS trunk flow model.

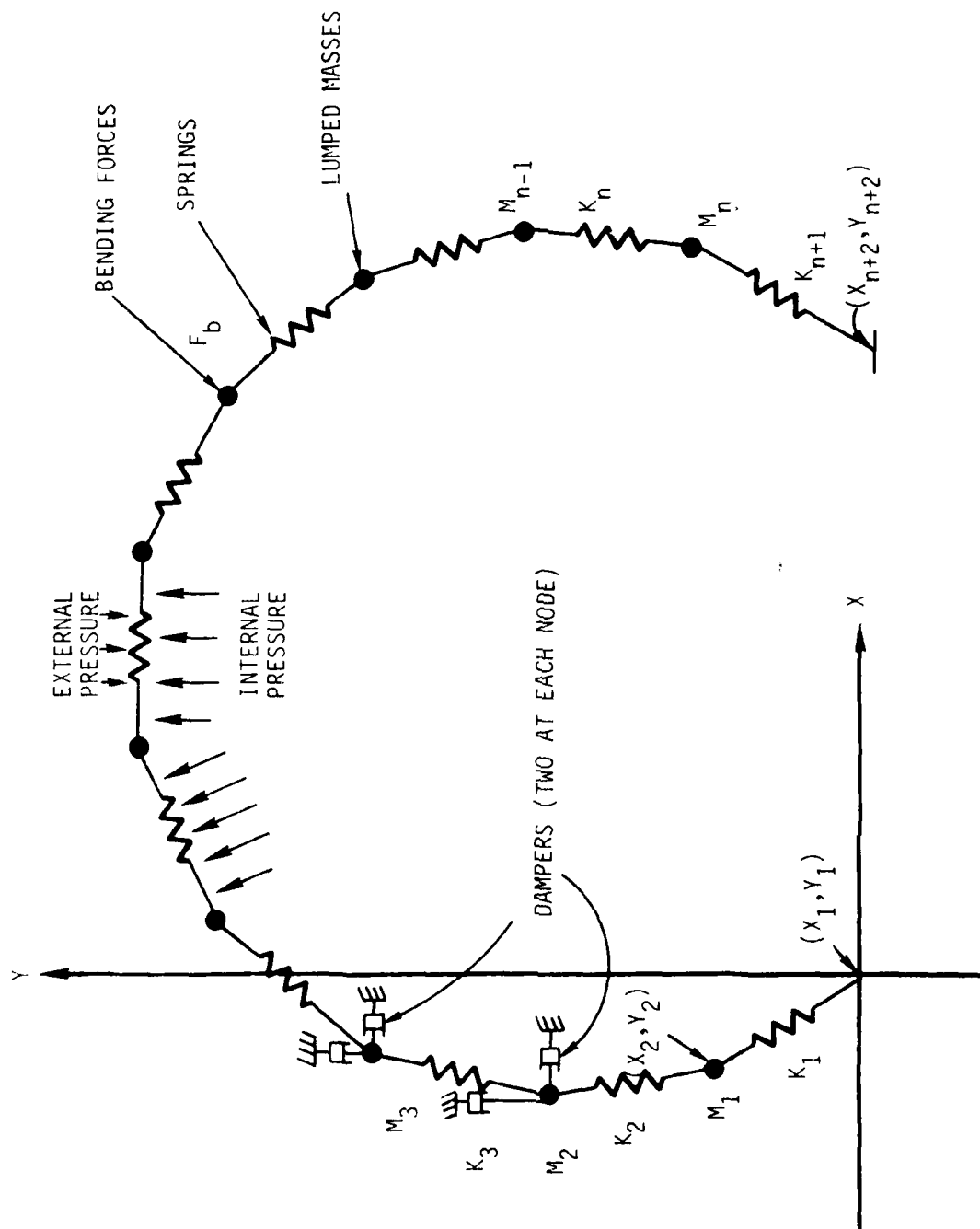


Figure 2. Trunk representation for the flutter models.

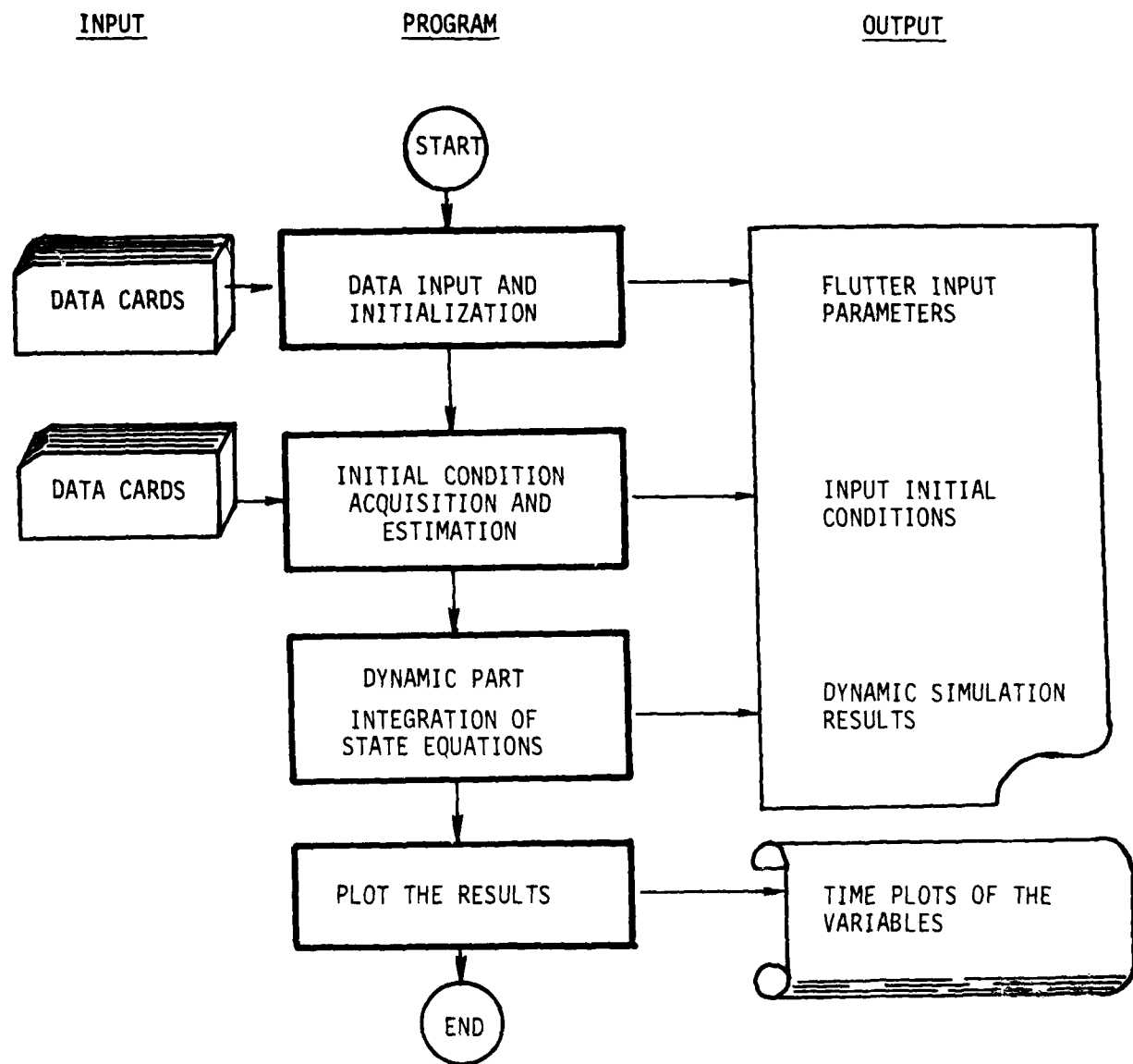


Figure 3. Program structure.

For this configuration of the trunk-airflow interaction the computer program simulates the trunk behavior for a variety of fluid flow and trunk parameters. A typical output of the simulation consists of the shape of the trunk defined by the location of the nodes as a function of time. In addition, the fluid parameters, such as the cushion and the trunk pressures, and various flows are also available in the output.

The computer program is designed to be flexible and versatile so that the trunk behavior can be studied for variations in a number of parameters. This way the program can be of assistance in developing flutter free configurations. A list of major parameters that can be varied is shown in Table 1. In order to ensure that the user can employ this flexibility easily, the program is organized in such a manner that:

- a. The major parameters are varied just through data input
- b. Any changes in the program made necessary due to additional experimental information on flutter characteristics could easily be performed.

The modular structure of the program which makes such changes easy to accomplish is described in Section 2 of the report. The instructions for using the program are in Section 3 whereas Section 4 illustrates the program capability through an example. Appendix A summarizes the equations incorporated in the program. These equations are based on the model described in the final report.*

*See Final Report of Contract No. F33615-78-C-3412 for details.

TABLE 1. THE CAPABILITIES OF THE TRUNK FLUTTER
SIMULATION PROGRAM

The program simulates the trunk behavior for variations in the following parameters.	
<u>ACLS Parameters</u>	
<u>Trunk Parameters</u>	
	attachment points
	cross section length
	elasticity variations along the length
	flexural stiffness variations along the length
	density variations along the length
	trim valve size
	trunk orifice size and location
<u>Fluid Parameters</u>	
	fan characteristics
	cushion volume
	trunk volume
	separation point
	global damping
<u>Operation Parameters</u>	
	hard surface clearance
<u>Flutter Suppression Parameters</u>	
	strake at any location
	minimum gap area
	external spring at any location
<u>Program Parameters</u>	
<u>Simulation Parameters</u>	
	time step
	time limit
	plotting options
<u>Trunk Model Parameters</u>	
	number of nodes
<u>Options</u>	
	separation point options*
	a. diffuser model, i.e. separation at a fixed slope
	b. fixed gap to separation point height
	c. trunk orifice flow induced separation i.e., separation occurs at the last orifice row if it is at a slope less than the diffuser model slope
	cushion - trunk pressures options
	a. fixed cushion and trunk pressures**
	b. variable cushion pressure fixed trunk pressure**
	c. variable cushion and trunk pressures with fan characteristics
	pressure profile on trunk bottom options
	a. nonvariable pressure profile**
	b. variable pressure profile without trunk orifice flow**
	c. variable pressure profile with trunk orifice flow
* For flutter studies conducted before further investigations in the separation point location are performed	
** Used for initial studies on the behavior of a particular trunk design	

Appendix B describes a program, which was also developed as a part of this contract, to calculate the eigenvalues of a linearized trunk model. This program can enhance the understanding of the dynamics of the trunk motion through prediction of the natural frequencies and the mode shape.

Appendix C summarizes Principal Program Nomenclature, and Appendix D has the listings of the computer programs.

2. PROGRAM ORGANIZATION

Overall structure of the computer program developed for simulating the dynamic behavior of the trunk of air cushion landing system (ACLS) is described in this section. Details of the eigenvalue computer program are, however, described in Appendix B.

The computer program has a modular structure, that is, there is a main program which coordinates operations of a number of subroutines, each of which perform a specific function. Such a structure makes the program efficient and easy to modify. As shown in Figure 3, there are four steps in the program execution:

- a. Data input and initialization
- b. Initial condition acquisition and estimation
- c. Dynamic part execution
- d. Plotting the results.

Details of each of these steps are described in the following subsection.

2.1 Program Execution Steps

The manner in which the main program executes each of the above four steps is shown in Figure 4. The main program, DYSYS, controls the dynamic simulation of the trunk model. DYSYS coordinates integration of the differential equations, printing of the state variables, and plotting the results. It initially calls subroutine EQSIM to initialize values of the derivatives of the state variables. DYSYS prints out the initial conditions and then enters the integration loop which calls subroutine RKDIF. DYSYS calls RKDIF at every time step until the time

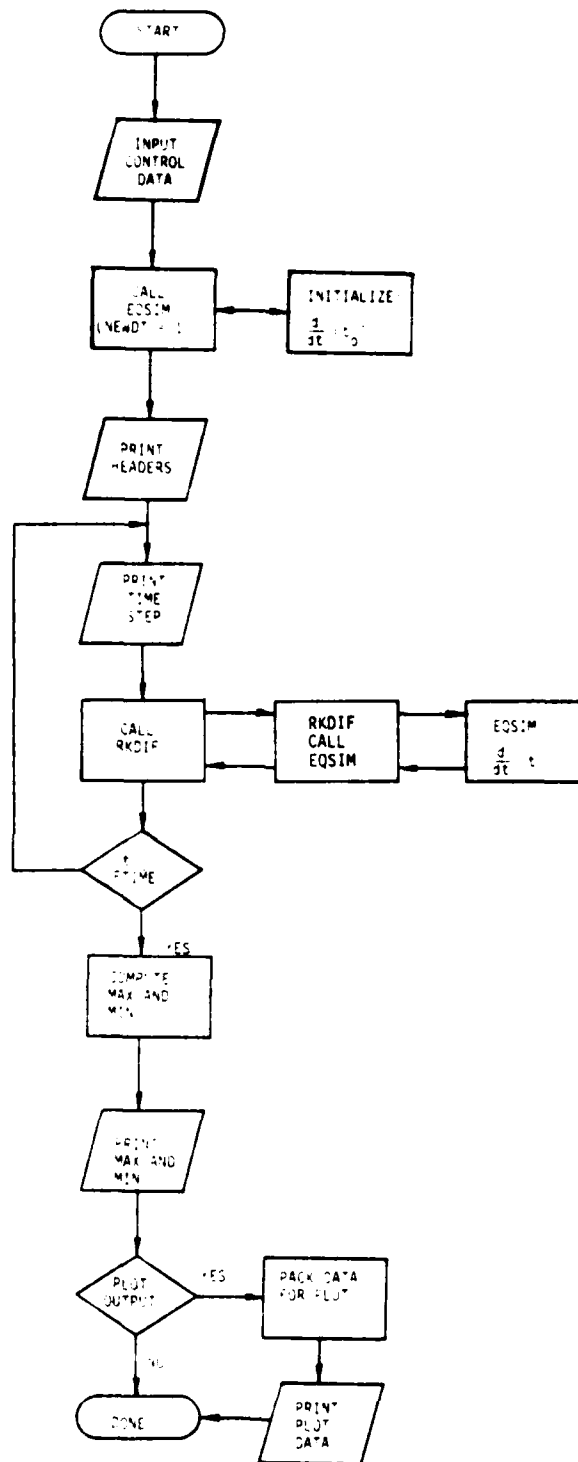


Figure 4. Simulation flowchart.

limit of the simulation is reached. RKDIF is the integration subroutine which incorporates a fourth order Runge-Kutta scheme. The integration scheme requires updating the differential values of the variables four times every time step, therefore, subroutine EQSIM is called four times by subroutine RKDIF.

RKDIF is the numerical integration subroutine which calculates the values of the state variables at time $t + dt$, given the values at time t , using a fourth order Runge-Kutta method. The integration scheme is summarized below:

- a. The iteration procedure starts with the values of the state variables y_1, y_2 etc., at time t .

$$y_i(t), \quad i = 1, n$$

- b. The slopes $Dy_i(t)$ are then determined from $y_i(t)$ by calling EQSIM

$$Dy_i(t) = dy_i(t)/dt$$

- c. The values y_{i1} at time $t + dt/2$ are then determined,

$$y_{i1} = y_i + Dy_i \cdot dt/2$$

- d. The slopes $Dy_{i1}(t + dt/2)$ are then determined by calling EQSIM and using the values of y_{i1} found in c.

- e. The values y_{i2} at time $t + dt/2$ are then determined

$$y_{i2} = y_i + Dy_{i1} \cdot dt/2$$

- f. The slopes Dy_{i2} ($t + dt/2$) are then determined from EQSIM using the values of y_{i2} found in e. above.

- g. The values y_{i3} at time $t + dt$ are then determined,

$$y_{i3} = y_i + Dy_{i2} \cdot dt$$

- h. The slopes Dy_{i3} at time $t + dt$ are then determined from EQSIM using the values of y_{i3} found in g. above.

- i. Finally, the values of the state variables at time $t + dt$ are found as follows:

$$y_i(t + dt) = y_i(t) + (Dy_i + 2Dy_{i1} + 2Dy_{i2} + Dy_{i3}) dt/6$$

During each integration step (that is, to advance from t to $t + dt$), EQSIM is called four times to determine the slopes (b, d, f, and h above).

Subroutine EQSIM, which calculates the various flows, pressures, motions, and forces, is the only model specific

subroutine in the simulator program and has all the parameters, variable initialization input and system equations contained in it. This subroutine calls subroutine TRUNK in order to calculate the initial trunk shape for the given trunk length and attachment points.

Once the simulation is completed the main program calls subroutines PLOTTER, PACKER, PRNTPT and PSTORE in order to produce time history plots of any of the system state variables on a printer plot. Table 2 summarizes the subroutines used in the program.

TABLE 2. A SUMMARY OF SUBROUTINES

No.	Subroutine	Primary Function	Group
1	DYSYS*	Main program; control integration of state equations and I/O	Main*
2	EQSIM	Compute state derivatives and system pressure-flow-geometry	Dynamic
3	RKDIF	Coordinates Runge-Kutta integration algorithm	Dynamic
4	PLOTTER	Controls data plotting	Plot
5	PACKER	Read plot data from simulation output file	Plot
6	PRNTPT	Plot data on printer	Plot
7	PSTORE	Write simulation output file	Plot
8	TRUNK	Compute TRUNK shape	Geometry
*DYSYS is the main calling program.			

3. PROGRAM USER INSTRUCTIONS

3.1 Program Input Data

The required program input data are supplied to the program in three ways:

- a. By data cards for parameters and design variables which are frequently changed.
- b. By data specifications included within the program, for example, physical constants.
- c. Through subroutine TRUNK which can be used to compute a trunk shape.

3.1.1 Input Data Format

The input data set format specification is designed to allow flexibility in the program parameter initialization and option specification. Program variables are input in a specified format and sequence using FORTRAN formatted I/O. Some input variables are not required for certain operating conditions. These special cases are noted in the input description (marked with an * after the card number). Variables which are vectors are read sequentially under their format specification, as noted by the index after the variable name. Special notes on some variables are included in subsection 3.1.3 where further explanation of the variable is required. The format used in the following input card description is:

CARD NO., NAME, VECTOR INDEX, FORMAT, DESCRIPTION.

1. FTIME, DTIME, STIME (I2, 8X, 2G10.5)

FTIME - Final time of simulation
DTIME - Time step for simulation
STIME - Starting time for simulation
(see note 1)

2. IPRNT(I); I = 1, 9 (9I2)

IPRNT - Print control vector, print state variable
IPRNT(I) every time step.

3. NPLTM (I2)

NPLTM - Plot control flag, NPLTM = number of plots
to be made.

4. XLAB(I); I = 1, 40 (40A2)

Lable card for simulation output

5. ICNTL(I); I = 1, 16 (16I1)

ICNTL(I) = Execution option selection

ICNTL(1) = 1 for dynamic cushion pressure
ICNTL(2) = 1 for dynamic trunk pressure and fan
characteristics
ICNTL(3) = 1 for trunk orifice flow
ICNTL(4) = 1 for separation point interpolation
ICNTL(5) = 1 for static trunk shape
ICNTL(6) = 1 for nonvariable pressure profile (see
card 25)
ICNTL(1) to ICNTL(3) have to be 1 for complete
simulation

6. NODES, ICFLAG, IFXN, IFSEP, ILENG, INSEP

(10I5)

NODES - Number of trunk nodes

ICFLAG - Trunk shape initial condition; 1 = compute,
0 = read initial condition in terms of X, Y
(see card No. 12, 13)

IFXN - Number of steps to be skipped in printout

IFSEP - Separation point model option
= 1 for fixed gap to separation point height
= 2 for diffuser model
= 3 for separation point fixed at node number
"INSEP"

ILENG - Initial length of segments option
= 0 compute length of node-node segments
= 1 read data cards for the initial length (see
card No. 11)

INSEP - Separation point node (IF IFSEP = 3)

7. A, B, L, HYI

(8G10.5)

A - Horizontal separation between the attachment points.

B - Vertical separation between the attachment points.

L - Trunk membrane length, between the attachment points.

HYI - Initial trunk height.

8. SSLENG, TPERIM

(8G10.5)

SSLENG - Length of trunk with significant gap area
for flow to atmosphere from cushion

TPERIM - Perimeter of trunk, around complete ACLS

9. ATC, ATRIM

(8G10.5)

ATC - Area of trunk to cushion flow, not including trim
valve (ATRIM) (ATC not used if option 3 used)

ATRIM - Area of trim valve for fixed flow area to cushion

10. ATCF(I); I = 1, nodes + 1

(8G10.5)

ATCF - Flow area of trunk orifice per unit width for
node to node link
(assume uniform density for each link)

11. *RLENGO(I); I = 1, nodes + 1

(8G10.5)

RLENGO - Element lengths at zero extension; needed if
ILENG = 1 (see card No. 6)

12. *X(I); I = 2, nodes + 1

(8G10.5)

X Node position values; needed if ICFLAG = 0
(see card No. 6)

13. *Y(I); I = 2, nodes + 1

(8G10.5)

Y Node position values; needed if ICFLAG = 0
(see card No. 6)

14. RMASS(I); I = 1, nodes

(8G10.5)

RMASS - Nodal mass values

15. RKVEC(I); I = 1, nodes (8G10.5)
RKVEC - Trunk elasticity of membrane links between nodes
16. RBVEC(I); I = 1, nodes (8G10.5)
RBVEC - Flexural stiffness per node
17. DAMP(I); I = 1, nodes (8G10.5)
DAMP - Nodal damping ratio (see note 2)
18. TREST, DAMPR (8G10.5)
TREST - Simulation time damping ratio change
DAMPR - Damping ratio change multiplication factor
(see note 3)
19. IEXT, RKEXTX, RKEXTY (I2, 2G10.5)
IEXT - Node of external spring attachment
RKEXTX - X direction spring constant
RKEXTY - Y direction spring constant (see note 4)
20. AIFAN, TKVOL, VCH (8G10.5)
AIFAN - Inertance of air in the fan (see note 5)
TKVOL - Trunk volume
VCH - Cushion volume

21. CQ0, CQ1, CQ2, CQ3, CQ4 (8G10.5)

CQ0 - Fan polynomial coefficient (see Figure 5)

CQ1 - Fan polynomial coefficient

CQ2 - Fan polynomial coefficient

CQ3 - Fan polynomial coefficient

CQ4 - Fan polynomial coefficient (see note 7)

22. CTC, CTRIM, CGAP, TSEP (8G10.5)

CTC - Discharge coefficient for the trunk orifices

CTRIM - Discharge coefficient for the trim valve

CGAP - Discharge coefficient for the gap

TSEP - Separation angle (in radians)

23. YGRNDS, SRATIO, YDMIN (8G10.5)

YGRNDS - Hard surface clearance

SRATIO - Constant gap to separation point height ratio

YDMIN - Maximum allowable trunk height for the minimum
gap area method of flutter suppression.
(strip or puck induced minimum gap area)

24. PTK, PCH, QGAP (8G10.5)

PTK - Trunk pressure initial condition (or constant)

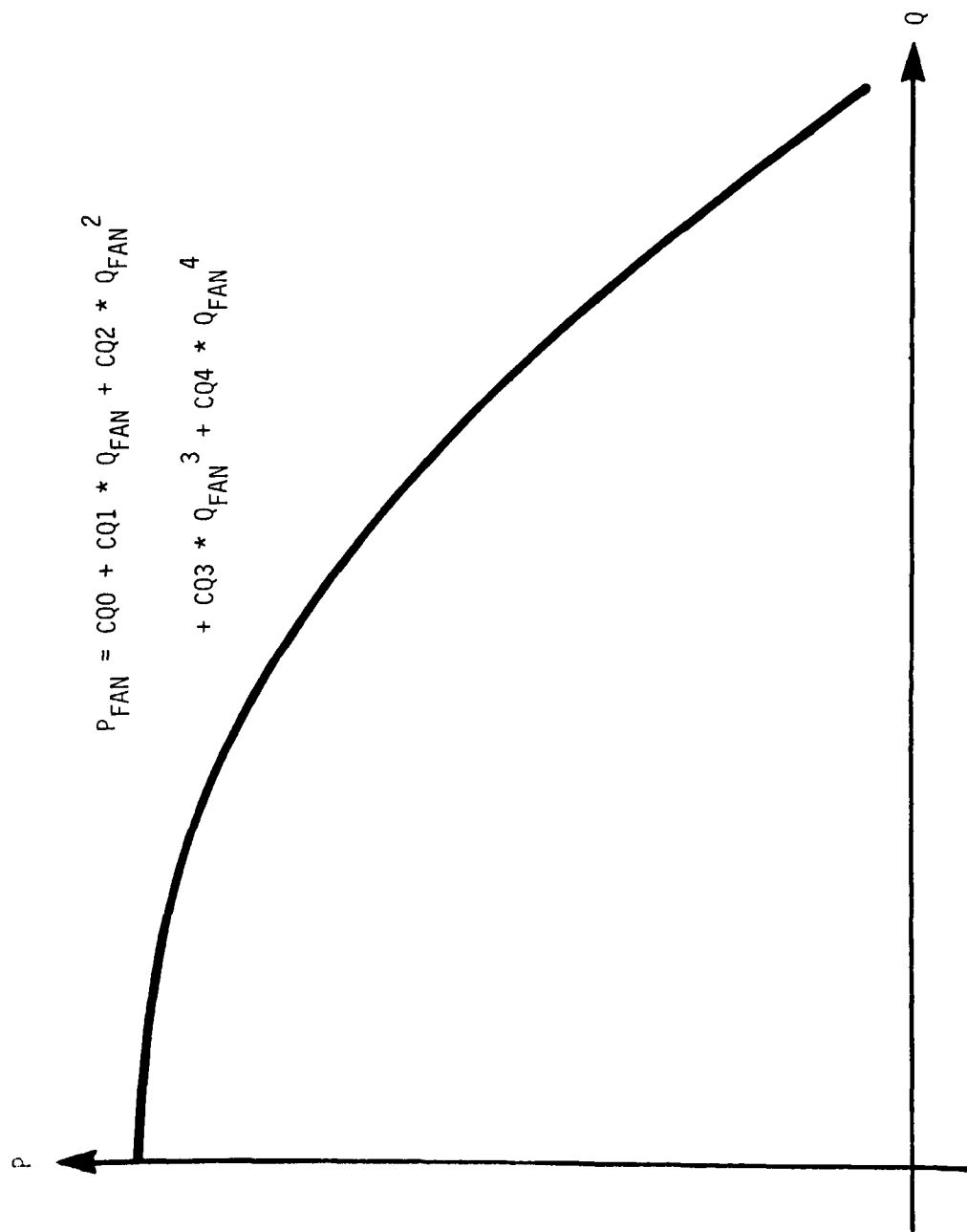


Figure 5. Fan pressure versus flow polynomial.

PCH - Cushion pressure initial condition (or constant)

QGAP - Exit flow, initial condition (not required
unless ICNTL(1) and ICNTL(2) are 1)

25.* PEXT(I); I = 1, nodes (8G10.5)

PEXT - External pressure at nodes, used for initial
shape computation (ICNTL(6) has to be = 1)

26.* NVPLOT, NPLT(I), I = 1,5 (11, 4X, 5I2)

NVPLOT - Number of variables on plot

NPLT - Variable to be plotted (state variable number
to be plotted = NPLT(I) * 2 + 4, see note 6)
(see note below and note 6)

27.* TPLSRT, TPLSTP, DTPLOT, XMIN, XMAX (6F12.5)

TPLSRT - Plot start time

TPLSTP - Plot end time

DTPLOT - Time increment between plot points

XMIN - Minimum value of plot axis (optional)

XMAX - Maximum value of plot axis (optional)

If XMIN = XMAX = 0.0, plots will be autoscaled by
program. (See note below)

*Note: Cards 26 and 27 are input only if NPLTM > 0

3.1.2 Internal Data

Internal constants used in the program include:

CKK - Polytropic expansion exponent, k , 1.4

G - Gravitational acceleration, g , 32.174 ft/sec²

PI - π , 3.1415926535

PAT - Atmospheric pressure (absolute), P_{at} ,
2116.8 lb/ft²

RHO - Air density, ρ , 0.00234151 slugs/ft³

3.1.3 Special Notes on Input Data

Note 1 - Time step for simulation has to be judiciously chosen by the user. Using too large a time step will cause numerical instability, using too small a time step will make the simulation uneconomical and inaccurate. In fact, the time step should be "small" compared to the smallest period of system vibration.

For the simulation results described in this manual, a time step of 0.001 sec was used. This may serve as a good starting point for initial calibration simulations for a different ACLS system. If the variables, particularly the higher frequency variables, such as the node displacements (X , Y) show a rapidly fluctuating characteristic, the time step should be reduced until such tendencies disappear. On the other hand, the time step may be increased if the time step of 0.001 sec is much smaller than the time step at which the fluctuations appear.

Note 2 - The global damping values (ratios) specified on input must be chosen by making an engineering estimate of the energy dissipated by trunk motion. Actual extension and bending motion damping values have been approximated by a global (X, Y) velocity damping force.

Note 3 - TREST, DAMPR - TREST is the simulation time that the program can change the global damping ratio values DAMP(I) that were given at input. At time TREST the values of DAMP(I) are multiplied by the constant DAMPR. This option allows the initial shape of the trunk and fluid flow variables to be computed for a highly damped system to get the required initial conditions and then to simulate the lightly damped system from time equals TREST onward.

Note 4 - IEXT, RKEXTX, RKEXTY - This option allows the attachment of a spring from node number IEXT to the ACLS frame. The X and Y direction stiffnesses are input as RKEXTX and RKEXTY. The initial trunk shape is assumed at the zero extension position.

Note 5 - Fan air inertance is inertance of the air residing in the fan at any instant. A good estimate of the inertance is obtained by:

$$I_f = \frac{\rho l}{A} = AIFAN$$

where

I_f = air inertance

l = average flow path length in the fan, which may be approximated as the length of the fan

ρ = average air density

A = Cross section area of flow.

Note 6 - The state variables for the simulation are stored in a vector. Each node requires four state variables for integration:

$I = (J + 1) * 4 + 1$ for node No. J

STATE(I) = X Velocity of node (J)

STATE(I + 1) = X Displacement of node (J)

STATE(I + 2) = Y Velocity of node (J)

STATE(I + 3) = Y Position of node (J)

If optional dynamic trunk or cushion pressure are used these variables are appended at the node state vector:

$I = (NODES + 2) * 4 + ICNTL(1) + ICNTL(2) * 2$

STATE(I) = Dynamic cushion pressure

STATE(I - 1) = Dynamic trunk pressure

STATE(I - 2) = Dynamic fan flow

STATE(I + 1) = Trunk membrane length

STATE(I + 2) = Flow, trunk to cushion

STATE(I + 3) = Flow, trim

STATE(I + 4) = Flow, cushion to atmosphere

STATE(I + 5) = Average cushion pressure P(t)

STATE(I + 6) = Flow, cushion and trunk orifices to
atmosphere

STATE(I + 7) = Flow, trunk to atmosphere.

Note 7 - CQ0, CQ1, CQ2, CQ3, CQ4 - These fan flow variables are coefficients for a fourth order polynomial. They are found by doing a linear polynomial regression on the fan data using a standard regression program. (See IBM-SSP manual.)

Note 8 - SRATIO - This ratio is an optional flow separation calculation technique which bases the height of the separation point gap as a ratio of minimum gap to separation gap equal to SRATIO.

The last seven variables are not state variables but auxiliary storage variables.

3.1.4 Program Option Operation

The dynamic flutter simulation program includes a number of model options which allow the user to simulate a variety of ACLS designs and to study a number of system features. The simulation options include:

STATIC PRESSURE LOAD PROFILE
FLOW INDUCED, DYNAMIC, PRESSURE PROFILE
VARIOUS FLOW SEPARATION POINT MODELS
EXTERNAL SPRING ATTACHMENT
DAMPING RATIO STEP CHANGE VERSUS TIME
TRUNK SHAPE INPUT OR CALCULATION

FLOW DYNAMICS INTEGRATION WITH STATIC TRUNK SHAPE
VARIOUS PRESSURE DYNAMIC MODELS

These features allow a number of types and phases of analysis to be performed with the program. A description of the options and how to select them is presented here.

Program options:

1. Static pressure load profile.

ICNTL(6) = 1

PEXT(I) = Node pressure (external) values. (Optional input)

This option is useful for measuring pressurized trunk shapes without pressure load dynamics on trunk surface, (for example: out-of-ground-effect).

2. Flow induced dynamic pressure profile.

ICNTL(6) = 0 (DEFAULT MODEL)

This option uses a Bernoulli flow equation for flow from the cushion area to atmosphere. This is the normal default flow model. When trunk flow orifices are included a modification to the flow equations is required:

ICNTL(3) = 1

This results in an interactive flow computation which includes cushion exit flow and trunk orifice flow which combine and flow to atmosphere.

3. Flow separation point models.

Several models exist for the flow separation point calculation.

a. Diffuser model.

IFSEP = 2

This model uses a diffuser slope value of TSEP radians to set the separation point at the node where the trunk surface slope is closest to TSEP. Typical values are in the range of 6 to 12 degrees.

b. Fixed gap ratio.

IFSEP = 1

This model sets the separation point at the node point where the gap height is closest to the ratio of minimum gap height divided by the ratio, SRATIO. (See note 8.)

c. Fixed separation node.

IFSEP = 3

INSEP = separation node number

This option allows the user to set the separation point at the specific node. This feature is useful for simulating strakes or other flow separation inducing devices.

d. Trunk orifice flow induced separation.

ICNTL(3) = 1

IFSEP = 2

For the diffuser model a special case can exist when the trunk flow orifices are utilized. If separation would occur after the trunk orifice area it will actually occur at the last trunk orifice row.

4. External spring attachment.

IEXT = node #
RKEXTX = X direction spring stiffness (linear constant value)
RKEXTY = Y direction spring stiffness (linear constant value)

This option is used to simulate the attachment of an external spring to the trunk membrane from the ACLS frame. If IEXT = 0 the option is overridden.

5. Damping ratio step change versus time.

TREST = time of damping change
DAMPR = damper ratio multiplication factor

This option allows the global damping ratio of the trunk nodes, DAMP(I), to be changed during the simulation. This feature can be used to integrate the trunk equations in an overly damped manner to reach an equilibrium value or to compute quasi-static trunk dynamics. When the initial conditions of the system are not well known the initial computation with a high damping ratio will allow them to stabilize before simulating a flutter situation.

6. Trunk shape selection.

ICFLAG = 0 Input node x, y values
 = 1 Compute node x, y values

ICFLAG = 0 Input node x, y values
 = 1 Compute node x, y values

The TRUNK subroutine can only calculate the positions of equispaced trunk nodes. The user inputted trunk node x, y values are the most useful and flexible because it allows variable node placement and spacing. RLENGO is also controlled in a similar manner.

ILENG = 0 Compute lengths
 = 1 Input lengths

The computed lengths are only good for equi-spaced nodes and user input is the recommended system.

7. Flow dynamics with static trunk profile.

ICNTL(5) = 1

This option allows flow and pressure data to be computed dynamically, but for the trunk shape to be held fixed. This feature is useful for checking out flow dynamics with a controlled static trunk system.

8. Various pressure dynamic models.

Three different fluid system models are included in the program:

ICNTL(1), ICNTL(2)

- a. The pressures can be fixed to the input values.

ICNTL(1) = ICNTL(2) = 0

- b. The cushion pressure can be a dynamic function of the flows and orifice areas in the ACLS system.

ICNTL(1) = 1, ICNTL(2) = 0

- c. The cushion and trunk pressures and fan flow are dynamic functions of the fan dynamics, orifice areas, and exit flows. If the dynamic trunk and fan option is used, the dynamic cushion pressure must be used also.

ICNTL(1) = ICNTL(2) = 1

3.2 Program Output

The printout includes the following data:

- a. Input Parameters - Trunk parameters, fan parameters, control parameters, simulation parameters, flow path parameters, structural parameters are printed out after input. The options activated by the ICNTL vector are printed out to indicate the features of the model being simulated.
- b. Dynamic Simulation Data - During the dynamic simulation the trunk shape, pressures and flow values, and other requested data are printed out every IFXN time steps.
- c. Time Response Plots - After the simulation reaches its final time the program can produce a printer plot of any of the dynamic variables such as trunk node position, dynamic pressures, or flow etc.

Listing of output data in sequential order.

1. Integration start time, STIME; integration final time, FTIME; integration time step, TSTEP.
2. System state variables, to be printed (see card No. 2).
3. Simulation label, XLAB(I).
4. Control vector, ICNTL(I).
5. Options in effect for simulation,
 - a. Dynamic cushion pressure
 - b. Dynamic trunk-fan pressure
 - c. Trunk orifice flow

- d. Separation point interpolation
 - e. No trunk motion test
 - f. Pressure profile.
-
- 6. Number of nodes, NODES; number of system state variables, NSTATE; X, Y node point selection option ICFLAG; print out skip number IFXN; separation point selection model, IFSEP; separation point, INSEP; element length option, ILENG.
 - 7. Horizontal attachment separation, A; vertical attachment separation, B; trunk length, L; trunk height, HYI.
 - 8. Trunk gap exit flow length, SSLENG; trunk perimeter, TPERIM.
 - 9. Trunk to cushion flow area, ATC; trim valve area ATRIM.
 - 10. Trunk orifice element flow areas, ATCF(I).
 - 11. Trunk element zero extension length, RLENGO(I) (option).
 - 12. X node position values, X(I).
 - 13. Y node position values, Y(I).
 - 14. Trunk nodal masses, RMASS(I).
 - 15. Trunk elastic stiffness, RKVEC(I).
 - 16. Trunk bending stiffness, RBVEC(I).
 - 17. Nodal damping ratios, DAMP(I).
 - 18. Damping reset time, TREST, reset factor DAMPR.

19. External spring attachment node, IEXT; X direction spring constant, RKEXTX; Y direction spring constant, RKEXTY.
20. Fan air inertance, AIFAN; trunk volume TKVOL; cushion volume VCH.
21. Fan polynomial coefficients, CQ0, CQ1, CQ2, CQ3, CQ4.
22. Trunk to cushion flow discharge coefficient, CTC; trim valve flow discharge coefficient, CTRIM; cushion flow area discharge coefficient, CGAP; separation angle, TSEP.
23. Hard surface clearance, YGRNDS; gap to separation point height ratio, SRATIO; maximum allowable trunk height, YDMIN.
24. Trunk pressure initial condition or constant, PTK; cushion pressure initial condition or constant, PCH; exit flow (fan) initial condition, QFAN.
25. External pressure profile, PEXT(I) (option).

NOTE

Outputs 26 to 30 are printed every IFXN time steps.

26. Simulation time, TIME.
27. Dynamic cushion pressure, STATE(N); trunk length, STATE(N+1); trunk to cushion flow, QTC; trim flow, QTRIM; cushion to atmosphere flow, QCA. (option, ICNTL(1) = 1.)
28. Y node values, Y(I).

29. Cushion separation node, ICS; exit separation node, ISEP; lowest node, INODE, separation point area, YASEP; minimum gap area, YGAPM; gap area at separation node, AGAP(ISEP); exit flow velocity, VEXIT; exit flow, QEXIT; cushion pressure, PCH; trunk pressure, PTK; fan flow, QFAN.
30. External pressure at nodes, PEXT(I).

NOTE

Outputs 31 and 32 are for plot outputs.

31. Plot curve state variable numbers, IDUM(I).
32. Print plot output (see sample output).
33. State variable maximum and minimum values, YMAX(I); YMIN(I); (option, if any IPRNT(I) $\neq$ 0).

4. ILLUSTRATIVE SIMULATION

The following describes the input data and the output print-out and print plot for a flutter simulation of a typical ACLS. The typical case simulated includes the effects of dynamic trunk and cushion pressure, fan dynamics, and trunk orifice flow. The separation point calculation is based on the diffuser model and no external spring is used. The input variables are shown in Figure 6 and the resulting printout is shown in Figure 7.

CARD NO.

```

1 — 1.0,0.001,0.0
2 — 0000000000000000
3 — 03
4 — DYNAMIC TEST 12 DEG Y=2.775 C-T-F WITH TRUNK FLOW DYN.
5 — 1100000000000000
6 — 16,0.50,2,1.0
7 — 4.725,1.272,0.75,2.5
8 — 18.50,59.60
9 — 2.569,2.355
10 — 0.0,0.0,0.0,0.0,0.00616,0.00616,0.00616,0.00616
    0.00616,0.00616,0.0,0.0,0.0,0.0,0.0
    0.0
11 — 0.88636,0.88636,0.44318,0.44318,0.44318,0.44318,0.44318,0.44318
    0.44318,0.44318,0.44318,0.44318,0.44318,0.44318,0.88636,0.88636
    0.88636
12 — 0.13573,0.68098,1.1196,1.5820,2.0711,2.5803,3.1072,3.6285
    4.1501,4.6451,5.0918,5.4731,5.7762,5.9915,6.1391,5.6370
13 — 0.96155,1.7605,2.0631,2.3131,2.5060,2.6377,2.7044,2.7023
    2.6299,2.4486,2.1678,1.8024,1.3692,0.88574,-0.082173,-0.92289
14 — 1.0135,1.0135,0.50675,0.50675,0.50675,0.50675,0.50675,0.50675
    0.50675,0.50675,0.50675,0.50675,0.50675,1.0135,1.0135

```

Figure 6. Input data test case.

CARD NO.

```

15 --- 3926.0,3926.0,3926.0,3926.0,3926.0,3926.0,3926.0,3926.0,3926.0,3926.0
      3926.0,3926.0,3926.0,3926.0,3926.0,3926.0,3926.0,3926.0,3926.0,3926.0
      3926.0
16 --- 0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
      0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
17 --- 0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05
      0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05
18 --- 10.0,1.0
19 --- 0.0,0.0,0
20 --- 0.029,1041.0,245.0
21 --- -11224.0,61.4279,-.120792,1.05604E-4-3.524F-R
22 --- 1.0,0.50,1.0,-.2094395
23 --- 2.775,0.5,2.775
24 --- 330.0,135.0,1446.0
26 --- 3XXXX32364044A
27 --- 0.0,0.99,0.005,0.0,0.0
26 --- 3XXXX757480000
27 --- 0.0,0.99,0.005,0.0,0.0
26 --- 2XXXX718200000
27 --- 0.0,0.99,0.005,0.0,0.0

```

Figure 6. Input data test case. (Continued)


```

NODE MASSING
.31501E-01 .15750E-01 .15750E-01 .15750E-01 .15750E-01 .15750E-01
.15750E-01 .15750E-01 .15750E-01 .15750E-01 .15750E-01 .15750E-01

SPRING CONSTANTS
3926.0 3926.0 3926.0 3926.0 3926.0 3926.0
3926.0 3926.0 3926.0 3926.0 3926.0 3926.0

BENDING STIFFNESS
1. 0. 0. 0. 0. 0.
1. 0. 0. 0. 0. 0.

NODE DAMPING
.50000E-01 .50000E-01 .50000E-01 .50000E-01 .50000E-01 .50000E-01
.50000E-01 .50000E-01 .50000E-01 .50000E-01 .50000E-01 .50000E-01

TREST, NAMED
10.00000 1.00000
ATFAN, TAVOL, VCM 245.00000
CIR, COI, CO2, CO3, CO4 -1.00000
CIC, TRIM, CGAP, TSEP 1.00000
VGRNDS, SRATIO, YDMIN 2.77500
PTK, PCH 335.00000
TIME 0.00000

NODE Y VALUES
2.4436 2.1687 1.7605 2.0631 2.3131 2.5066 2.6377 2.7044 2.7023 2.6299
2.4436 2.1687 1.7605 2.0631 2.3131 2.5066 2.6377 2.7044 2.7023 2.6299

TIC, TSEP, INODE, VASEP, YCAPM, AGAP, ISEP, VEXIT, DEKIT, PCH, PTK, OFAN
4 10 8 1.510 1.510 339.57361 49.27213 135.00000 330.00000 1446.00000
KL, QTC, QTRIM, QCA, PCAVE, QTA, QTA 1.4510 339.57361 49.27213 135.00000 330.00000 1446.00000
Q.75000 0.00000 480.55719 397.19638 135.00000 20.58884 1064.08728
NODE EXT DEPRESSURE 0.00000 134.96 134.87 134.73 133.67 133.30 106.21 11.594 106.832 -98265E-11
0. 0. 0. 0. 0. 0. 0. 0. 0. 0.

TIME .05000
NODE Y VALUES
2.4436 2.0934 1.7736 2.0597 2.2884 2.4636 2.5914 2.6712 2.6958 2.6226
2.4436 2.0934 1.7736 2.0597 2.2884 2.4636 2.5914 2.6712 2.6958 2.6226

TIC, TSEP, INODE, VASEP, YCAPM, AGAP, ISEP, VEXIT, DEKIT, PCH, PTK, OFAN
3 10 9 1.243 1.243 370.26177 56.44465 100.51319 326.89146 1446.08367
VL, QTC, QTRIM, QCA, PCAVE, QTA, QTA 1.5243 370.26177 56.44465 100.51319 326.89146 1446.08367
11.34577 0.00000 443.90426 790.29516 153.68069 38.45111 1010.07118
NODE EXT DEPRESSURE 0.00000 159.97 159.46 157.19 150.26 125.30 27.173 13.811 27.173 13.811
160.50 150.34 159.97 159.46 157.19 150.26 125.30 27.173 13.811 27.173 13.811
0. 0. 0. 0. 0. 0. 0. 0. 0. 0.

TIME .10000
NODE Y VALUES
2.4436 2.0934 1.7736 2.0597 2.2884 2.4636 2.5914 2.6712 2.6958 2.6226
2.4436 2.0934 1.7736 2.0597 2.2884 2.4636 2.5914 2.6712 2.6958 2.6226

```

Figure 7. Program output. (Continued)

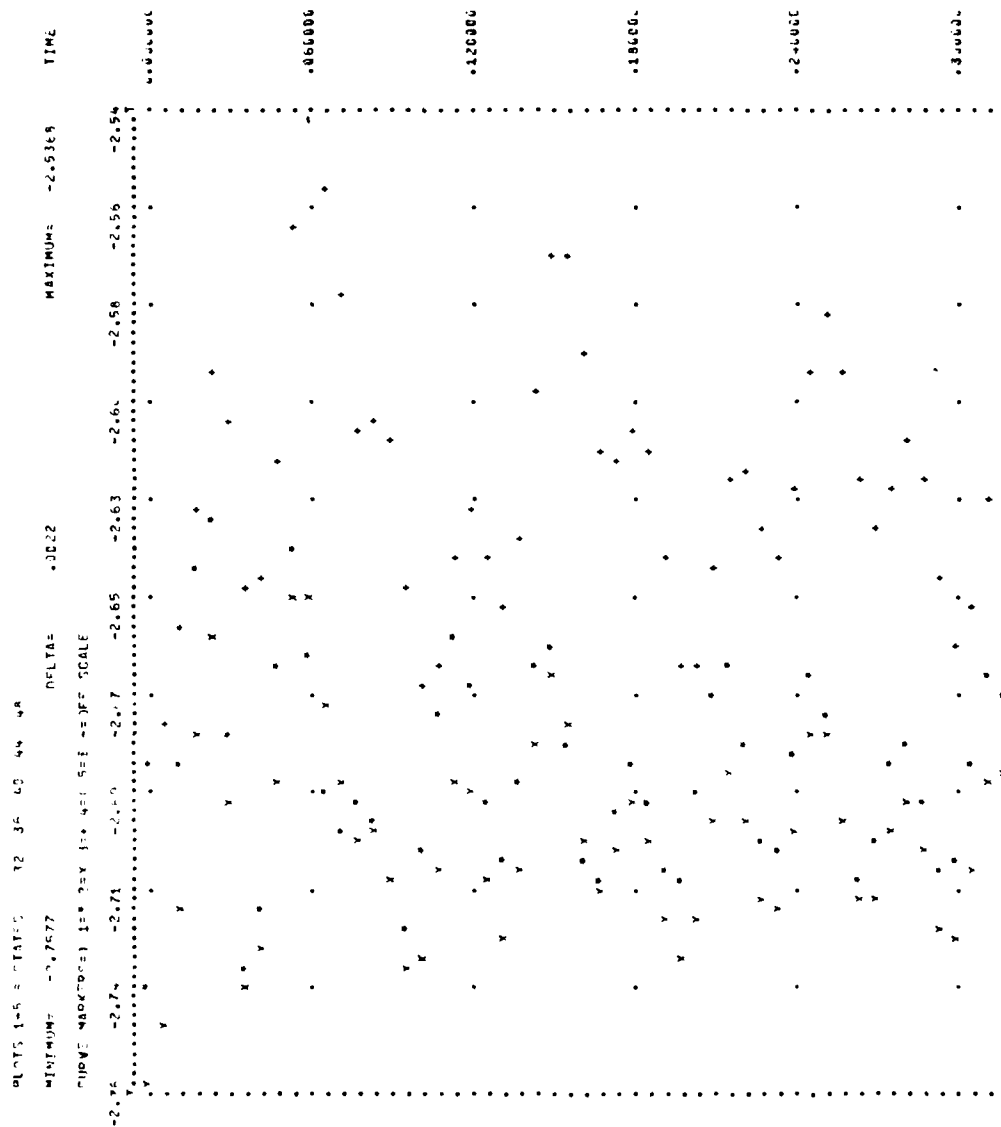


Figure 7. Program output. (Continued)

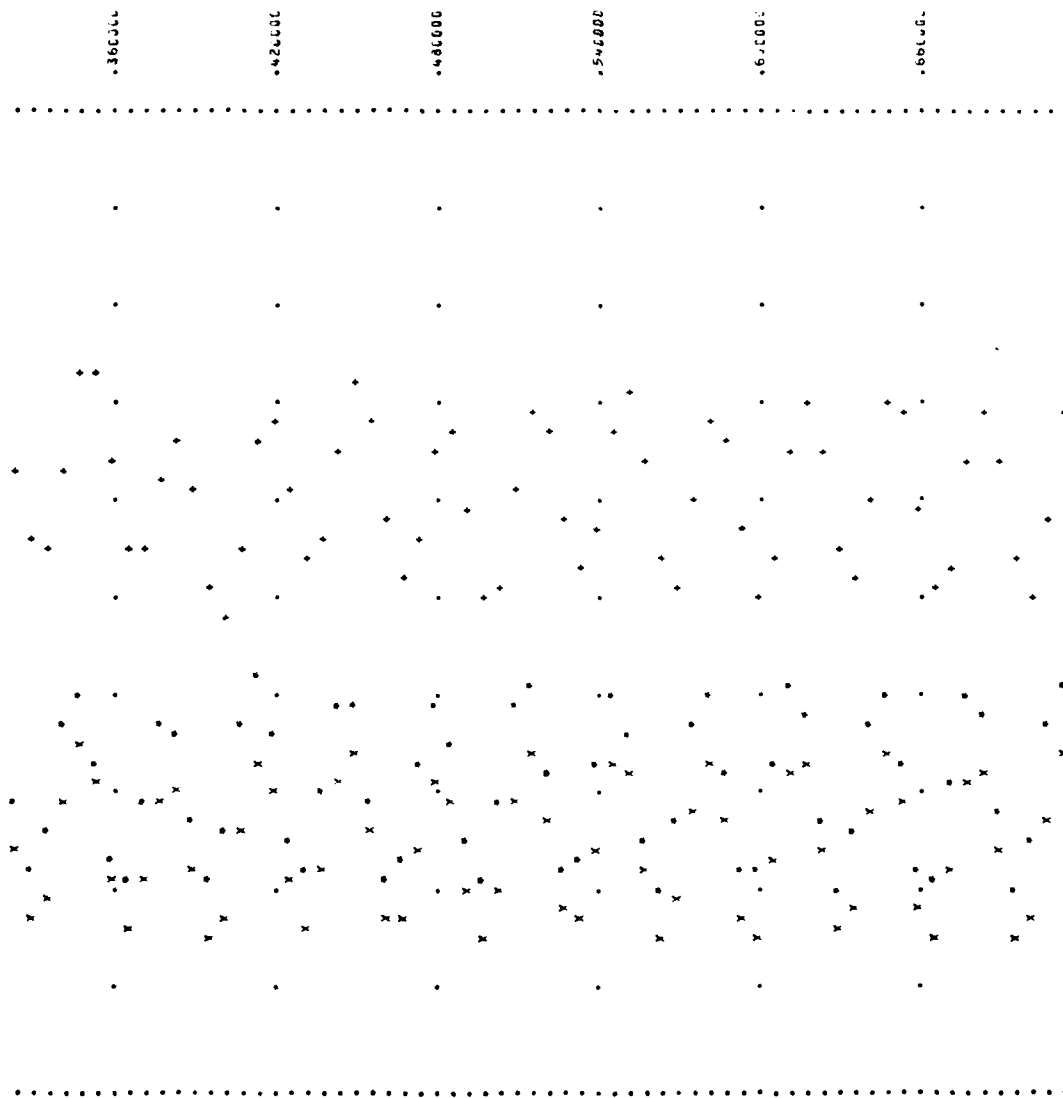


Figure 7. Program output. (Continued)

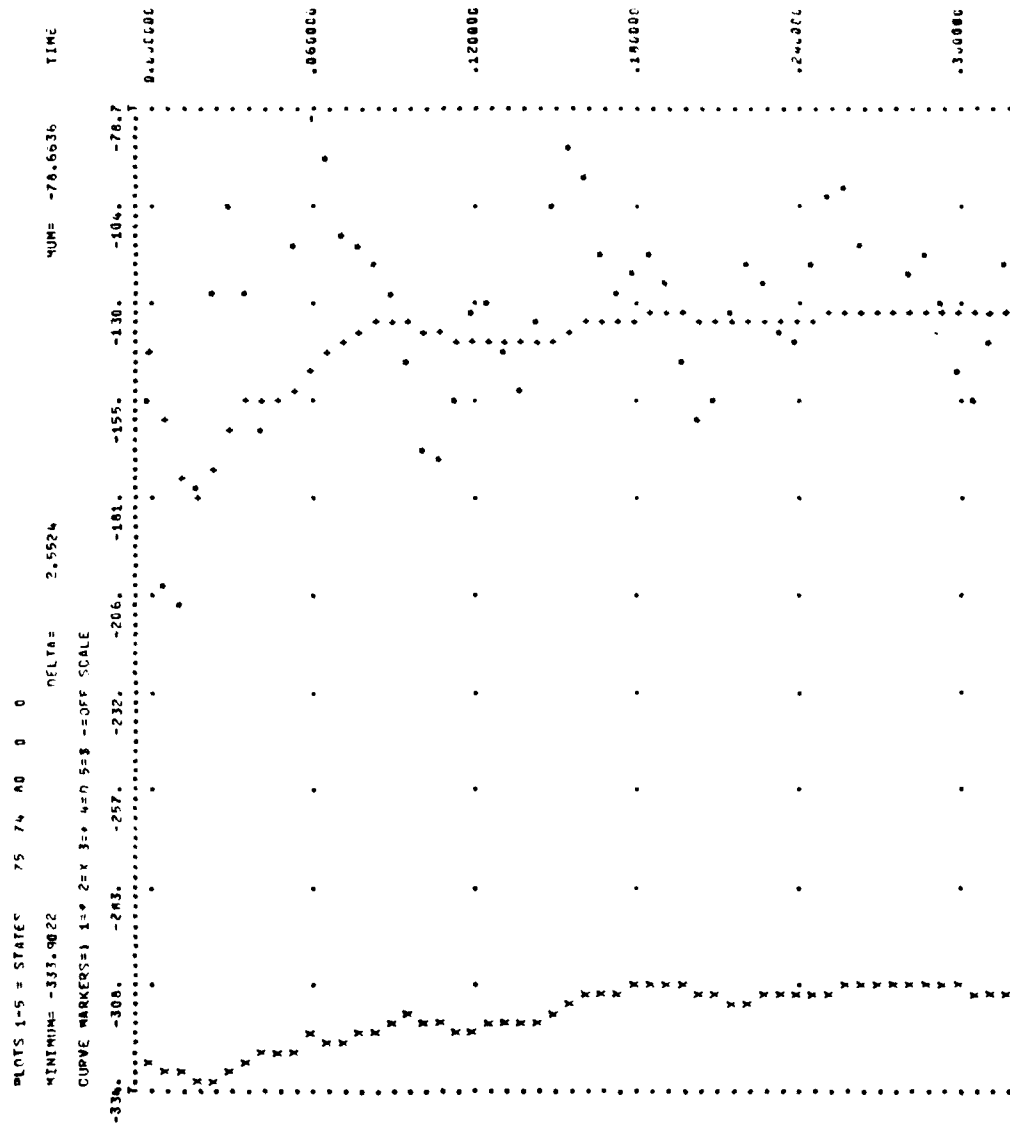


Figure 7. Program output. (Continued)

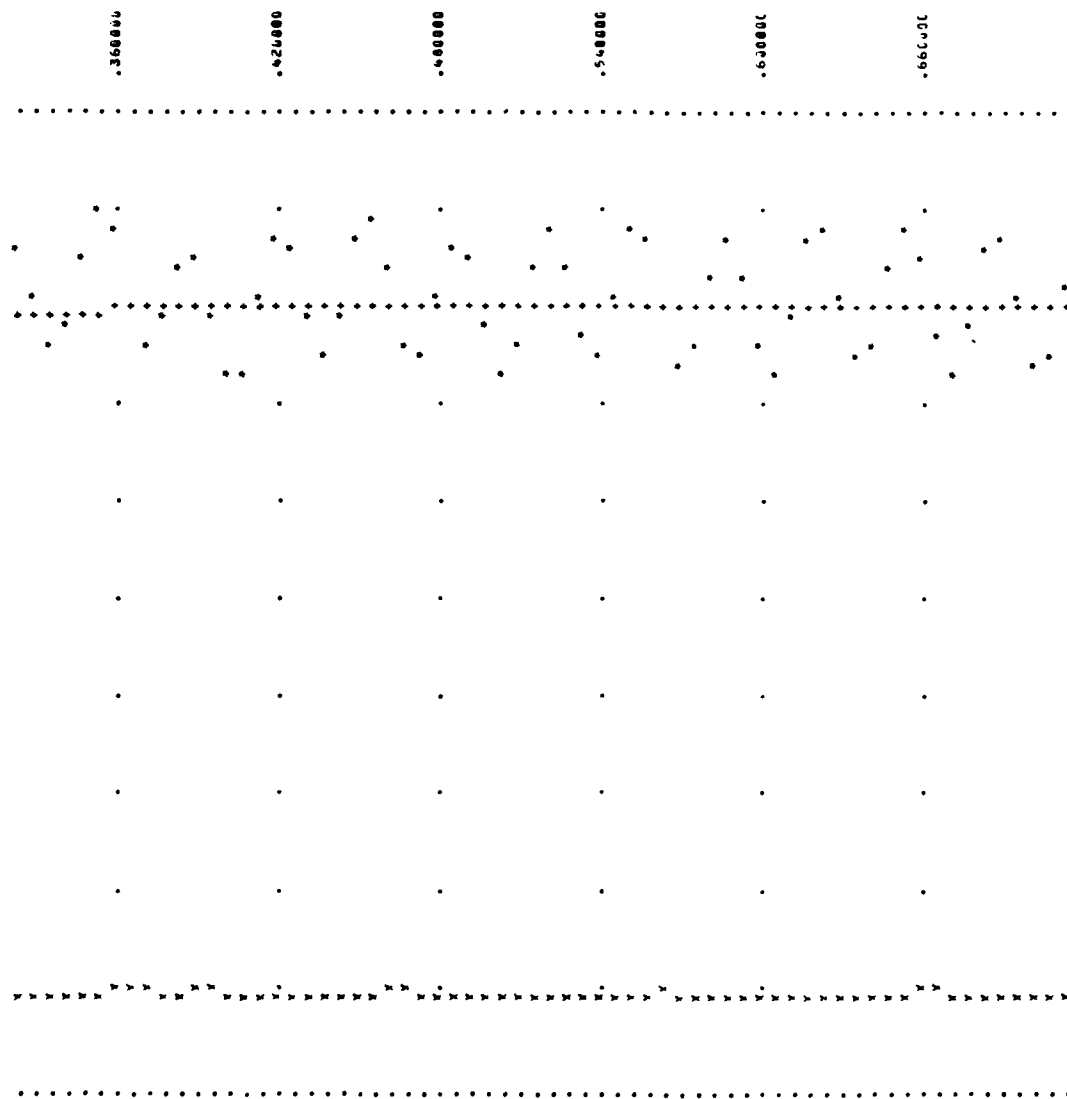


Figure 7. Program output. (Continued)

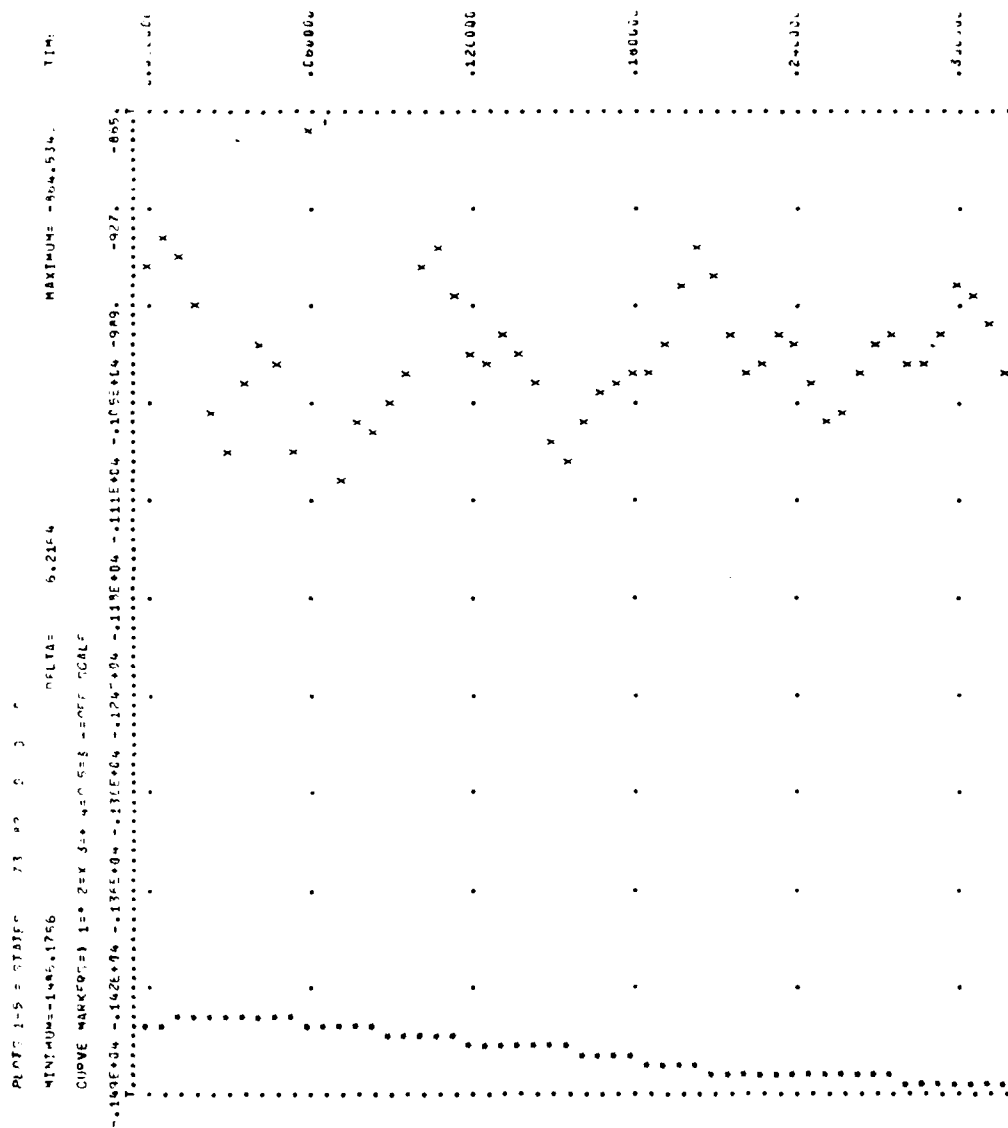


Figure 7. Program output. (Continued.)

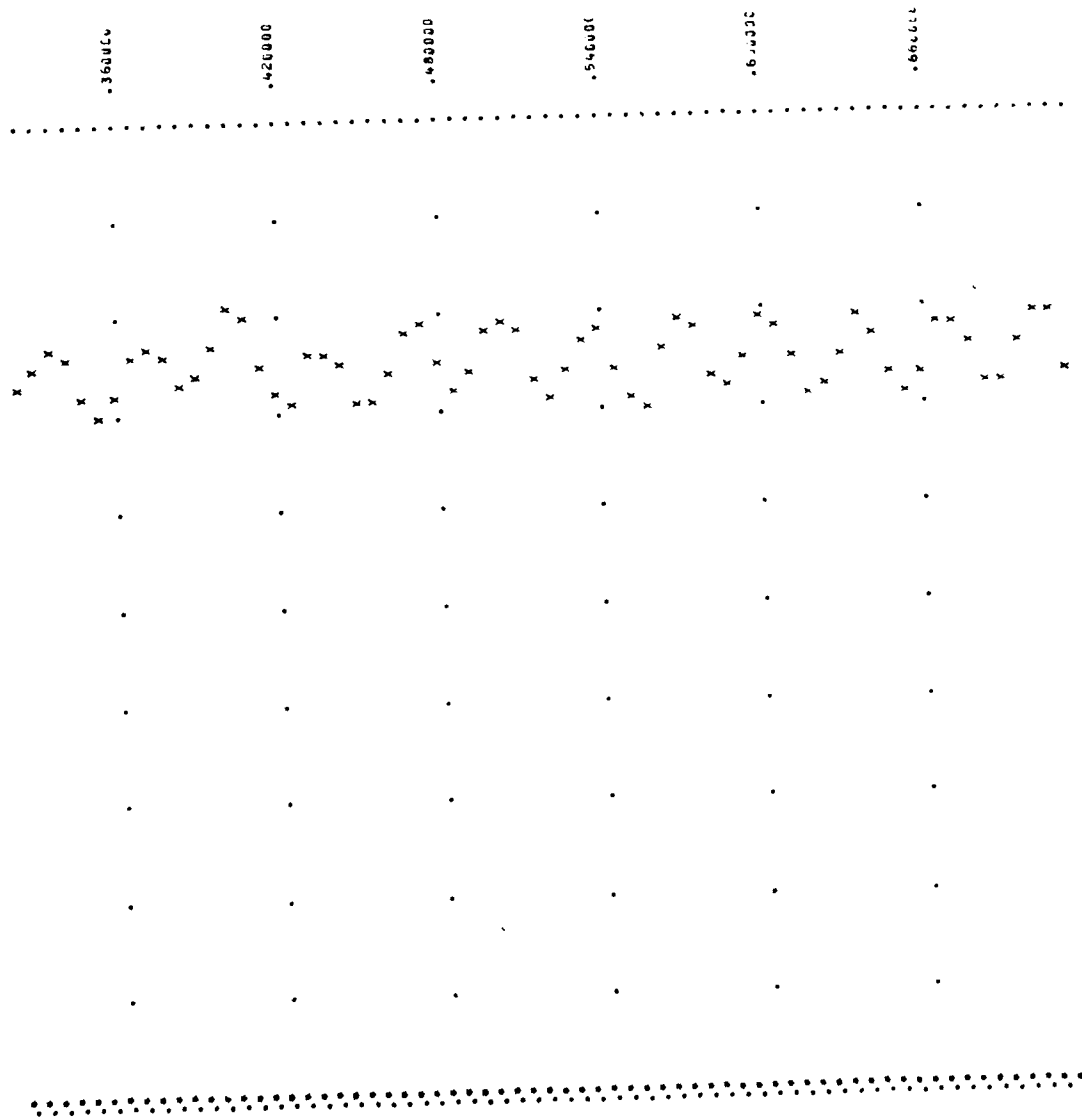


Figure 7. Program output. (Continued)

APPENDIX A
EQUATIONS USED IN THE MODEL

This program incorporates the nonlinear relationships between the motion of the lumped trunk masses and the system parameters, such as the trunk elasticity, damping and the pressure forces arising due to the fluid flow under the trunk, taking into account the geometry of the trunk. Figure A-1 shows a schematic diagram of the model. As shown in the model, the lumped masses are connected by springs representing elasticity of the trunk. Also included are the pressure forces acting between the lumped masses which are divided between the adjacent masses. The system damping is represented in this initial model by global dampers, which develop opposing forces proportional to the absolute velocities of the lumped masses.

The acceleration components of a trunk mass are calculated along the X and Y axes from summation of the stiffness, pressure and damping forces acting along the respective axes. Double integration of the accelerations gives positions which are then plotted to obtain instantaneous trunk shapes.

The equations used in the model are summarized in the following:

A.1 Geometry Relations (Figure A-2)

$$\theta_i = \tan^{-1} ((Y_{i+1} - Y_i) / (X_{i+1} - X_i)) \quad (A-1)$$

$$\phi_i = \tan^{-1} ((Y_i - Y_{i+1}) / (X_i - X_{i+1})) \quad (A-2)$$

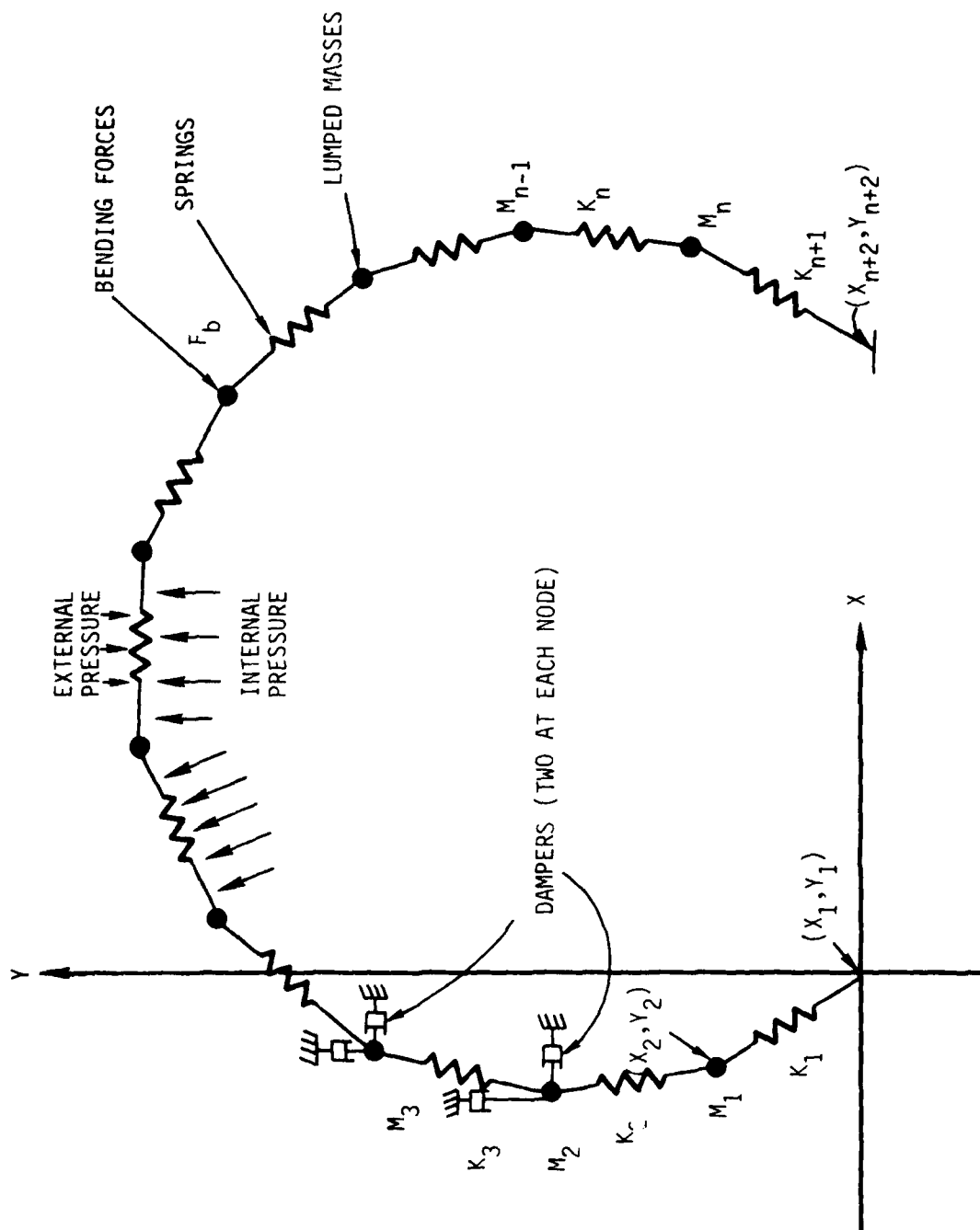


Figure A-1. Trunk representation for the dynamic simulation model.

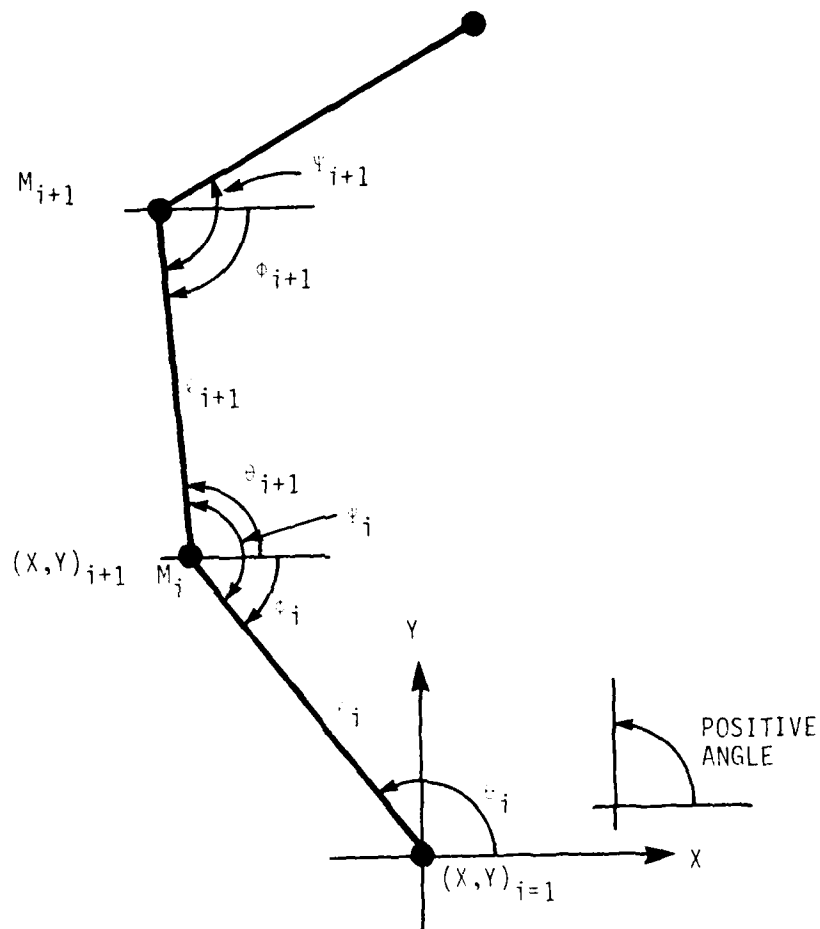


Figure A-2. The 2D coordinate system.

$$l_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \quad (A-3)$$

Note: $\tan^{-1}$ is a four quadrant function, therefore, if
 $\theta = \tan^{-1} (\Delta Y / \Delta X)$, $-\pi \leq \theta \leq \pi$.

A.2 Force Relations

Spring Force (Figure A-3)

$$\begin{aligned} F_{XK_i} &= \cos(\phi_i) * K_i * \Delta l_i \\ &+ \cos(\theta_{i+1}) * K_{i+1} * \Delta l_{i+1} \end{aligned} \quad (A-4)$$

$$\begin{aligned} F_{YK_i} &= \sin(\phi_i) * K_i * \Delta l_i \\ &+ \sin(\theta_{i+1}) * K_{i+1} * \Delta l_{i+1} \end{aligned} \quad (A-5)$$

where

$$\Delta l_i = l_i - l_{oi}$$

l_{oi} (l_{oi} is initial l_i)

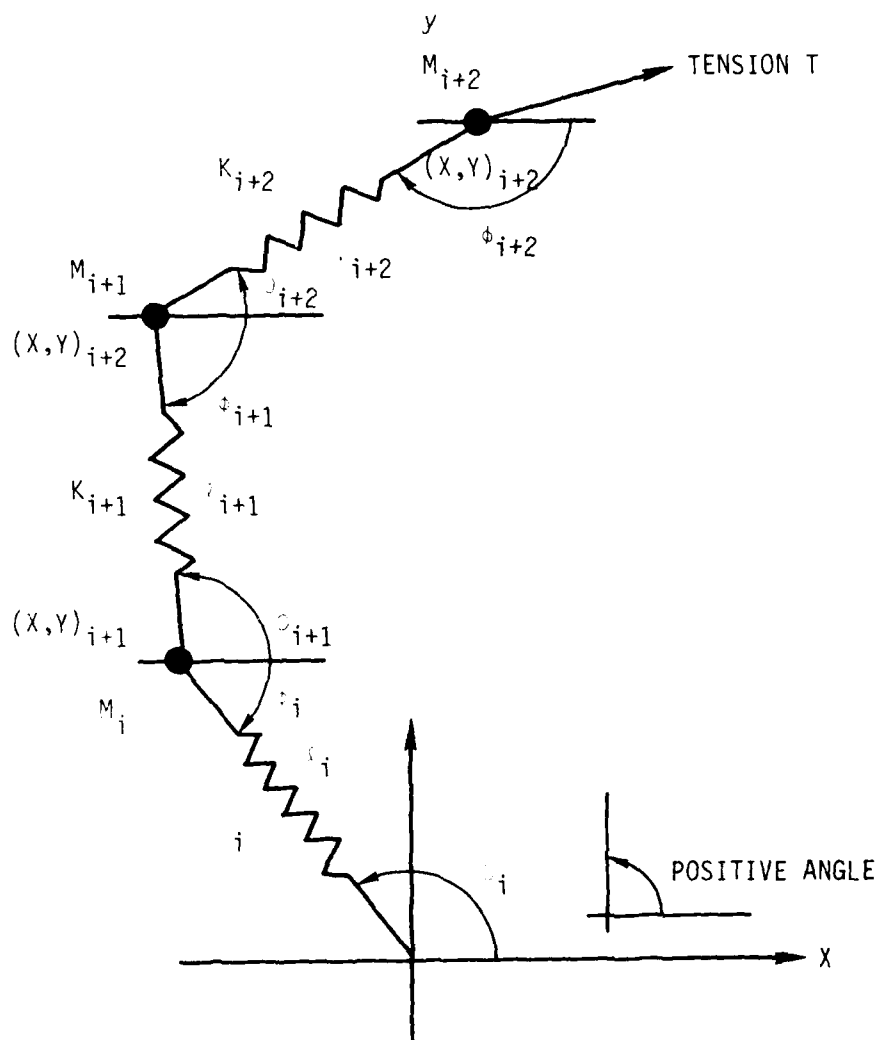


Figure A-3. Trunk elasticity representation.

Bending Forces (Figure A-4)

$$\text{TORQUE, } \tau = K_B * \Delta\psi$$

or

$$\tau = K_B * (\psi_O - \psi) \quad (\text{A-6})$$

since

$$\tau = F * \ell$$

$$F * \ell = K_B (\psi_O - \psi)$$

$$F = \frac{K_B (\psi_O - \psi)}{\ell} \quad (\text{A-7})$$

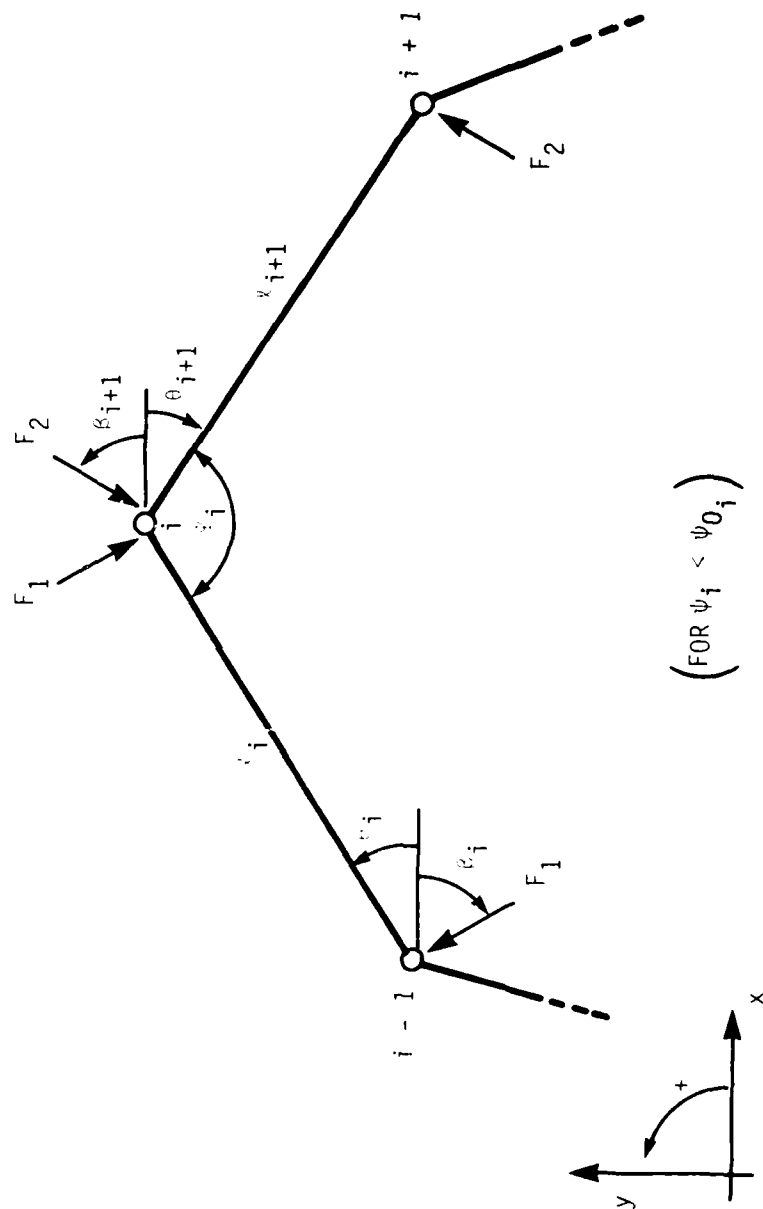
where

$$K_B = K_{\ell} * \frac{2}{\ell_1 + \ell_2}$$

so

$$F_1 = \frac{K_{\ell_i} (\psi_O - \psi)^2}{\ell_i (\ell_i + \ell_{i+1})} \quad (\text{A-8})$$

$$F_2 = \frac{K_{\ell_{i+1}} (\psi_O - \psi)^2}{\ell_{i+1} (\ell_i + \ell_{i+1})} \quad (\text{A-9})$$



F_1 ORTHOGONAL TO ψ_i

F_2 ORTHOGONAL TO ψ_{i+1}

ψ_{0_i} = INITIAL "NO BENDING" NODE ANGLE

(ALL ANGLES ARE (+) COUNTERCLOCKWISE ROTATION)

Figure A-4. Bending force representation.

Converting forces F_1 and F_2 into component F_x, F_y forces at each node (i) gives (see Figure A-5)

$$F_{X_i} = -F_1 \cos(\beta_i) \quad (A-10)$$

$$F_{Y_i} = -F_1 \sin(\beta_i) \quad (A-11)$$

$$F_{X_{i+1}} = -F_1 \cos(\beta_i + \pi) - F_2 \cos(\beta_{i+1}) \quad (A-12)$$

$$F_{Y_{i+1}} = -F_1 \sin(\beta_i + \pi) - F_2 \sin(\beta_{i+1}) \quad (A-13)$$

$$F_{X_{i+2}} = -F_2 \cos(\beta_{i+1} + \pi) \quad (A-14)$$

$$F_{Y_{i+2}} = -F_2 \sin(\beta_{i+1} + \pi) \quad (A-15)$$

Due to orthogonality:

$$\beta_i = \theta_i - \frac{\pi}{2} \quad (A-16)$$

$$\beta_{i+1} = \theta_{i+1} - \frac{\pi}{2} \quad (A-17)$$

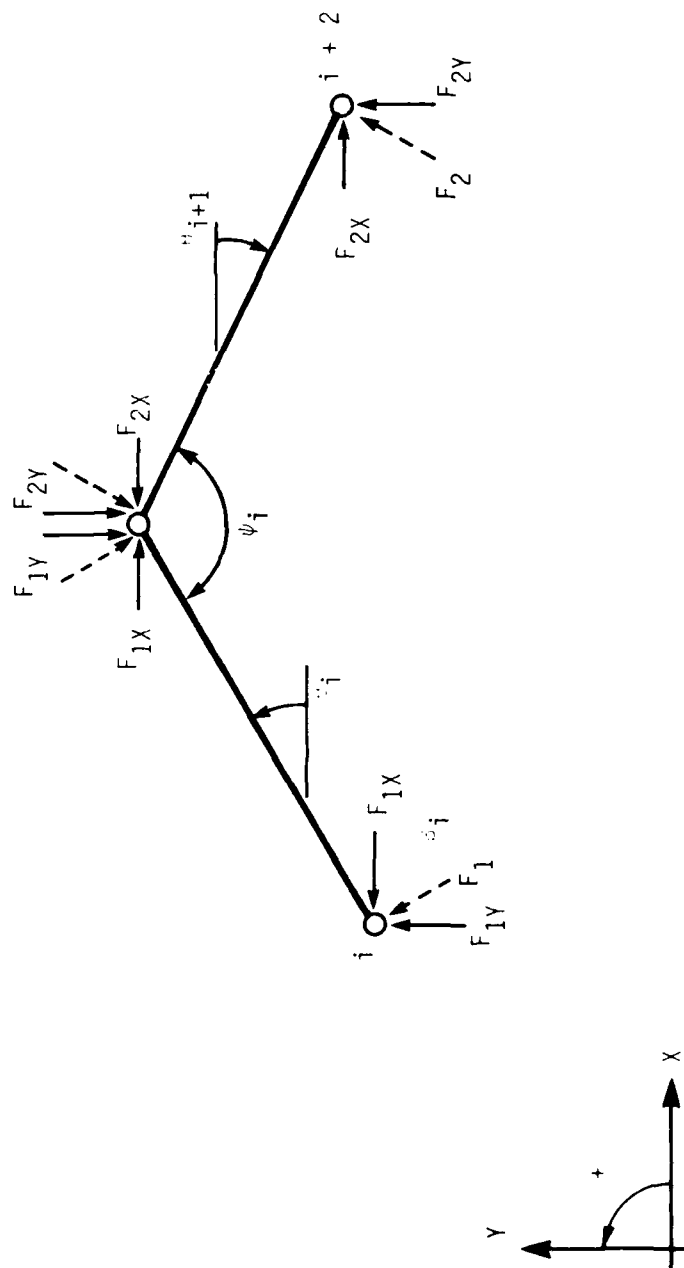


Figure A-5. Bending force vectors.

From trigonometric identities:

$$\cos(\theta_i - \frac{\pi}{2}) = \sin(\theta_i) \quad (\text{A-18})$$

$$\sin(\theta_i - \frac{\pi}{2}) = -\cos(\theta_i) \quad (\text{A-19})$$

$$\cos(\theta_i + \frac{\pi}{2}) = -\sin(\theta_i) \quad (\text{A-20})$$

$$\sin(\theta_i + \frac{\pi}{2}) = \cos(\theta_i) \quad (\text{A-21})$$

So replacement of β terms with equivalent θ terms gives:

$$F_{X_i} = -F_1 \sin(\theta_i) \quad (\text{A-22})$$

$$F_{Y_i} = -F_1 [-\cos(\theta_i)] \quad (\text{A-23})$$

$$F_{X_{i+1}} = -F_1 [-\sin(\theta_i)] - F_2 \sin(\theta_{i+1}) \quad (\text{A-24})$$

$$F_{Y_{i+1}} = -F_1 \cos(\theta_i) - F_2 [-\cos(\theta_{i+1})] \quad (\text{A-25})$$

$$F_{X_{i+2}} = -F_2 [-\sin(\theta_{i+1})] \quad (\text{A-26})$$

$$F_{Y_{i+2}} = -F_2 \cos(\theta_{i+1}) \quad (A-27)$$

Reducing the equations leads to:

$$F_{X_i} = -F_1 \sin(\theta_i) \quad (A-28)$$

$$F_{Y_i} = F_1 \cos(\theta_i) \quad (A-29)$$

$$F_{X_{i+1}} = F_1 \sin(\theta_i) - F_2 \sin(\theta_{i+1}) \quad (A-30)$$

$$F_{Y_{i+1}} = -F_1 \cos(\theta_i) + F_2 \cos(\theta_{i+1}) \quad (A-31)$$

$$F_{X_{i+2}} = F_2 \sin(\theta_{i+1}) \quad (A-32)$$

$$F_{Y_{i+2}} = -F_2 \cos(\theta_{i+1}) \quad (A-33)$$

Attached Spring Forces

If an external spring is attached to node I to suppress flutter, it creates a generalized force $F_G(I)$.

$$F_G = -K_{EXT} \Delta D \quad (A-34)$$

In X, Y components:

$$F_{XG_i} = -K_{EXT} * \Delta X_i \quad (A-35)$$

$$F_{YG_i} = -K_{EXT} * \Delta Y_i \quad (A-36)$$

where

$$-\Delta X_i = (X_{O_i} - X_i) \quad (A-37)$$

$$-\Delta Y_i = (Y_{O_i} - Y_i) \quad (A-38)$$

therefore

$$F_{XG_i} = K_{EXT} (X_{O_i} - X_i) \quad (A-39)$$

$$F_{YG_i} = K_{EXT} (Y_{O_i} - Y_i) \quad (A-40)$$

Pressure Force (Figure A-6)

Defining

$$P_i = P_{tk} - P_{ext_i} \quad (A-41)$$

where P_{tk} = Internal Pressure on Membrane
and P_{ext_i} = External Pressure on Membrane (static or dynamic)
leads to pressure forces at node points:

$$F_{XP_i} = \frac{-P_i}{2} \left[\ell_i \sin(\theta_i) + \ell_{i+1} \sin(\theta_{i+1}) \right] \quad (A-42)$$

$$F_{YP_i} = \frac{P_i}{2} \left[\ell_i \cos(\theta_i) + \ell_{i+1} \cos(\theta_{i+1}) \right] \quad (A-43)$$

Damper Forces (Figure A-7)

$$\text{By definition } \bar{F} = D \cdot \bar{V} \quad (A-44)$$

$$F_{XD_i} = V_{x_i} \cdot \xi_i \cdot 2 \sqrt{M_i \cdot K_i} \quad (A-45)$$

$$F_{YD_i} = V_{y_i} \cdot \xi_i \cdot 2 \sqrt{M_i \cdot K_i} \quad (A-46)$$

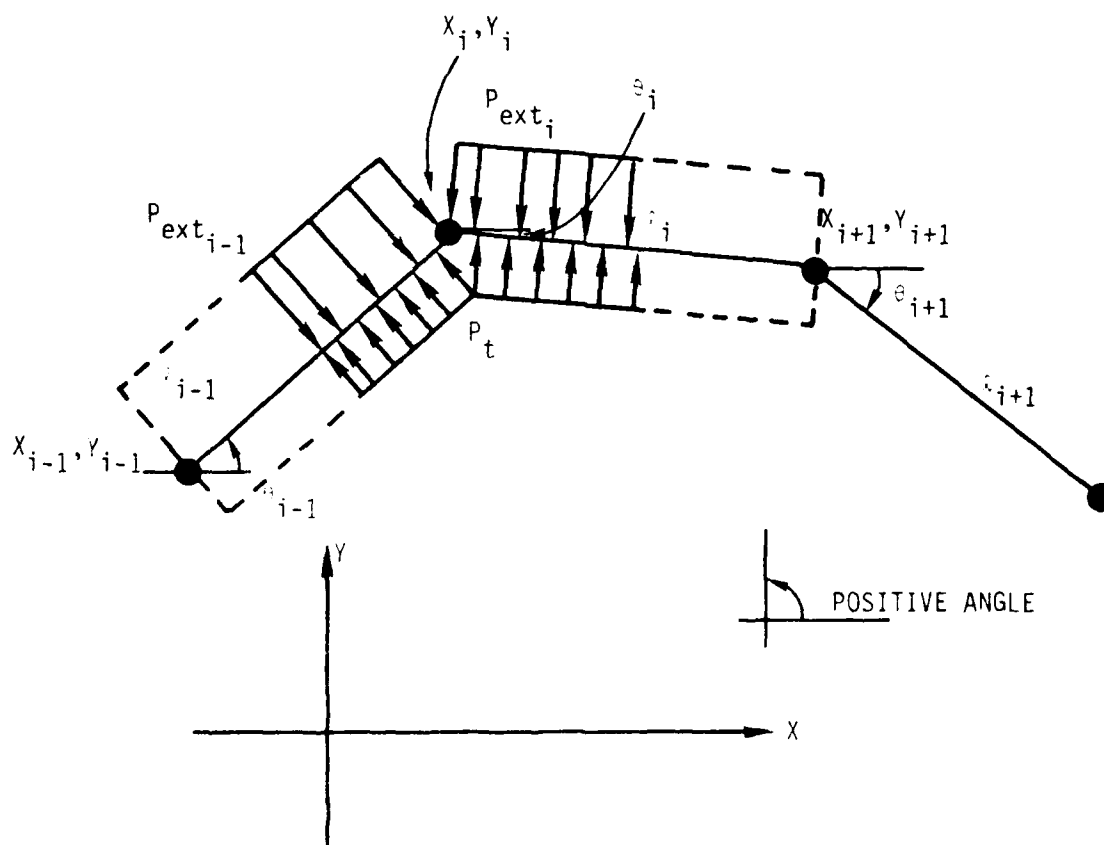


Figure A-6. Pressure force representation.

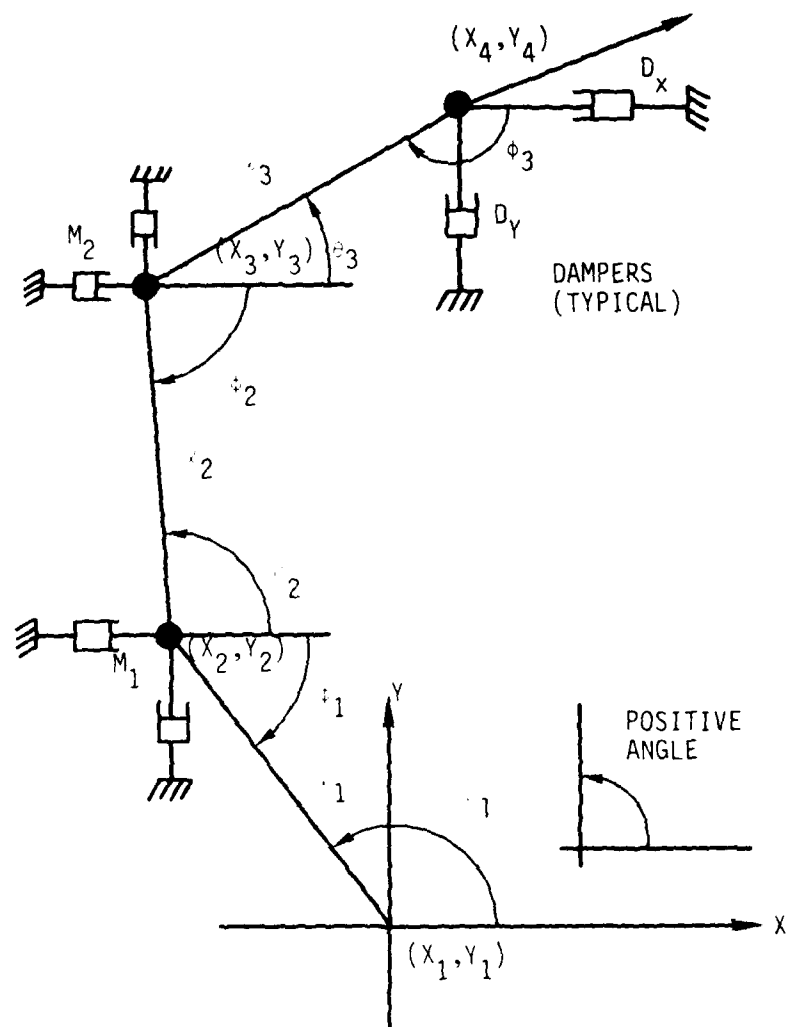


Figure A-7. Damping representation.

where

- ξ_i = damping ratio at node i
- M_i = nodal mass at node i
- K_i = average stiffness at node i

Differential Equations:

$$\frac{d^2x_i}{dt^2} = (F_{xd_i} + F_{xk_i} + F_{xp_i} + F_{xb_i} + F_{xG_i})/M_i \quad (A-47)$$

$$\frac{d^2y_i}{dt^2} = (F_{yd_i} + F_{yk_i} + F_{yp_i} + F_{yb_i} + F_{yG_i})/M_i \quad (A-48)$$

$$\frac{dx_i}{dt} = \dot{x}_i \quad (A-49)$$

$$\frac{dy_i}{dt} = \dot{y}_i$$

A.3 Flow Relations

Pressure - Flow Relationship (Figure A-8)

Dynamic pressure under membrane modelled by Bernoulli flow equation for ideal flow with no trunk flow.

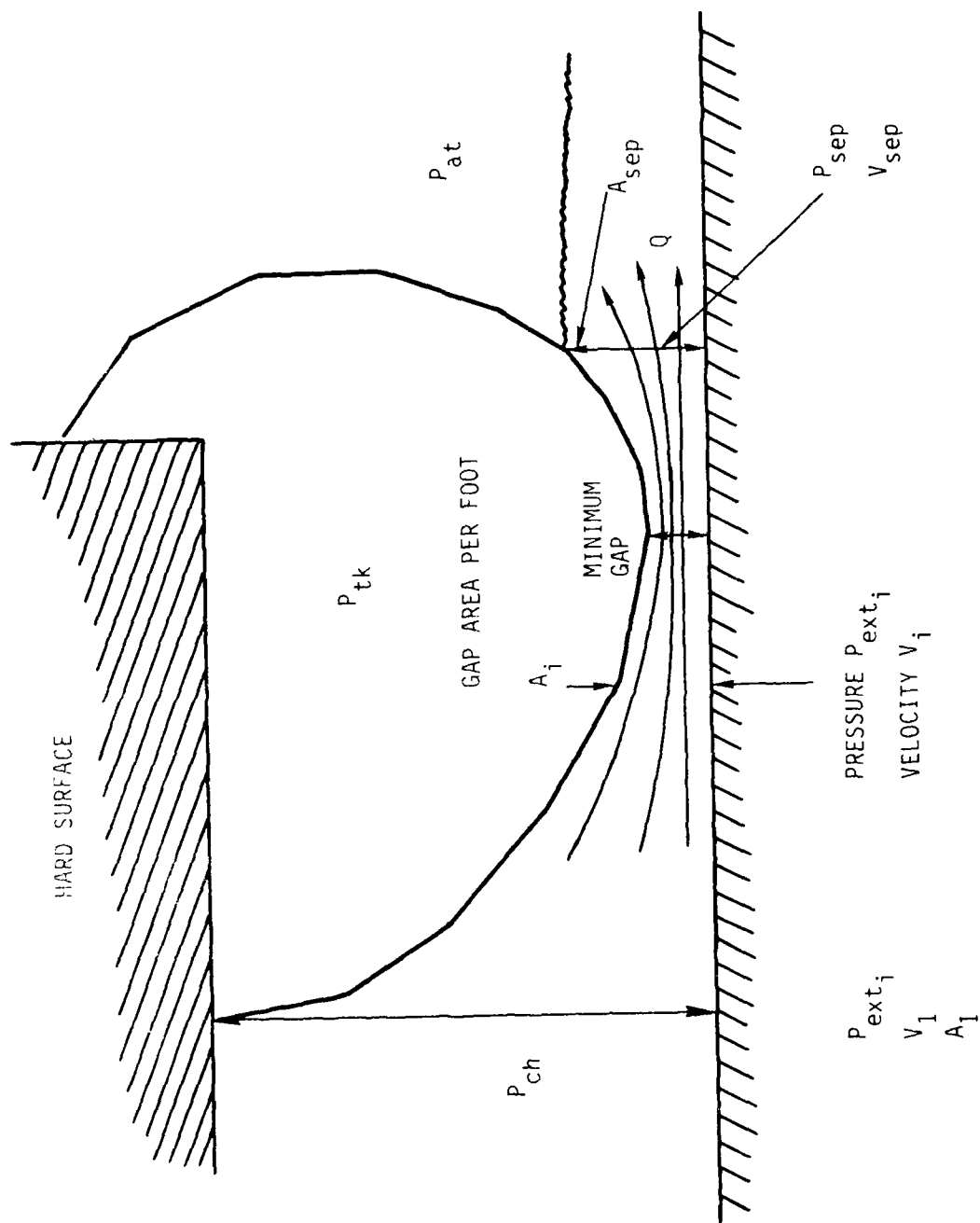


Figure A-8. Fluid flow representation.

$$\frac{P_1}{\rho} + \frac{V_1^2}{2} = \frac{P_2}{\rho} + \frac{V_2^2}{2} \quad (A-51)$$

where

$$Q = V_1 A_1 = V_2 A_2 = V_{sep} A_{sep} \quad (A-52)$$

therefore

$$P_{ext_i} - P_{sep} = \rho \frac{V_{sep}^2}{2} \left[1 - \left(\frac{A_{sep}}{A_i} \right)^2 \right] \quad (A-53)$$

at separation point

$$P_{sep} = P_{at} = 0$$

$$P_1 = P_{ch}, V_1 = 0$$

Therefore,

$$\frac{P_{ch}}{\rho} = \frac{V_{sep}^2}{2} \quad (A-54)$$

or

$$v_{sep} = \sqrt{\frac{2P_{ch}}{\rho}} \quad (A-55)$$

since

$$Q = A_{sep} \sqrt{\frac{2P_{ch}}{\rho}} = A_{sep} * v_{sep} \quad (A-56)$$

therefore

$$P_i = P_{ch} \left(1 - \left(\frac{A_{sep}}{A_i} \right)^2 \right) \quad (A-57)$$

Pressure Source Dynamics

The system model includes the capabilities of fan-trunk-cushion dynamics shown in Figure A-9. The fan includes a pressure versus flow polynomial curve fit and a fluid model of the trunk and cushion volumes and orifices. (See final report for derivation of equations A-58 through A-61).

Differential Equations:

$$\frac{d}{dt} (Q_{FAN}) = \frac{P(Q_{FAN}) - P_{tk}}{I_{FAN}} \quad (A-58)$$

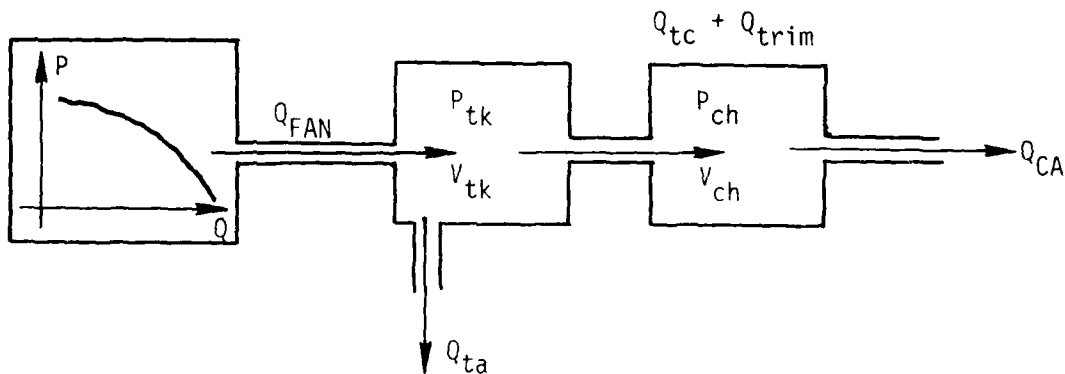


Figure A-9. Dynamic fluid model.

$$\frac{d}{dt} (P_{TK}) = \frac{C_{KK}}{V_{TK}} (P_{TK} + P_{AT}) * (Q_{FAN} - Q_{TC} - Q_{TA} - Q_{TRIM}) \quad (A-59)$$

$$\frac{d}{dt} (P_{CH}) = \frac{C_{KK}}{V_{CH}} (P_{CH} + P_{AT}) * (Q_{TC} - Q_{TRIM} - Q_{CA}) \quad (A-60)$$

where,

$$P(Q_{FAN}) = CQ_0 + CQ_1 * Q_{FAN} + CQ_2 * Q_{FAN}^2 + CQ_3 * Q_{FAN}^3 + CQ_4 * Q_{FAN}^4 \quad (A-61)$$

(See Figure A-10).

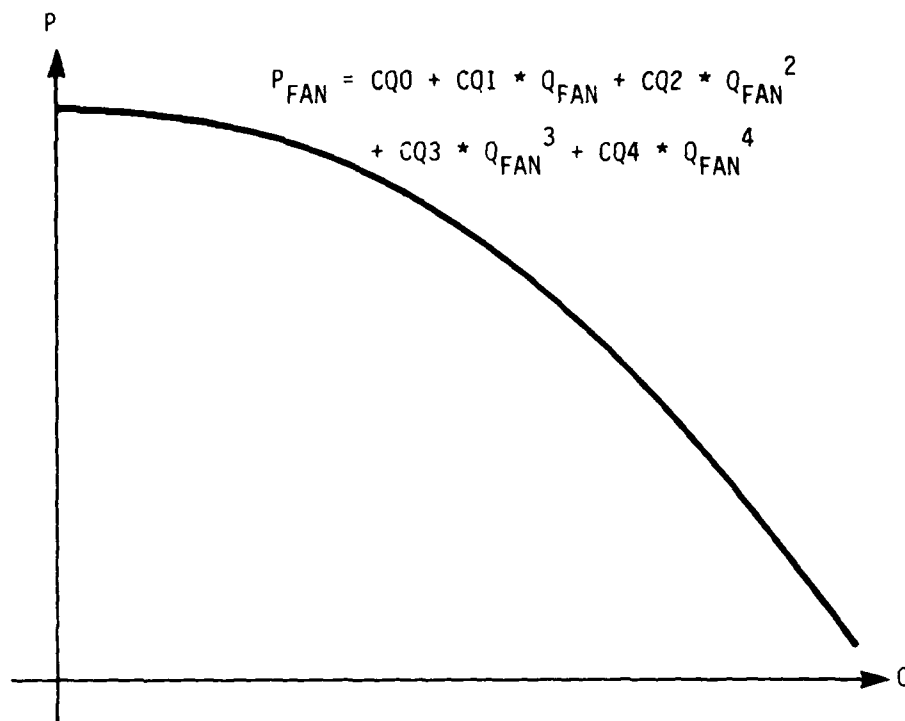


Figure A-10. Fan pressure versus flow polynomial.

The flow model determines the variations in pressures and flows as a function of time. There are two parts to the flow model: the fluid chambers (that is, cushion and trunk), and the fan. The principal assumptions of the flow model are as follows.

- a. The flow through all orifices is one-dimensional and quasi-static, that is, the pressure in the plane of the orifice is uniform, and the unsteady state terms in Bernoulli's equation are small compared to the change in velocity head.
- b. The flow through the orifices is incompressible, that is, the pressure drop is small compared to the total pressure, and the air density is constant.

- c. The pressure and volume changes of the air during expansion and compression in the various fluid chambers are governed by a polytropic relationship, that is,
 $p v^k = \text{const.}$

Trunk Orifice Flow Effects:

The addition of flow from the trunk orifice to the gap changes the pressure profile under the trunk. The flow/pressure relations are iteratively computed between the cushion and the separation point. The flow is computed by considering a number of control volumes between nodes. (see Figure A-11).

The flow into each control volume is computed as:

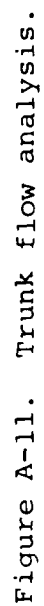
$$Q_{TC_i} = C_{TC} * A_{TC_i} * \sqrt{2 \left[P_{tk} - (P_i + P_{i+1})/2 \right]} \quad (A-62)$$

$$dW_i = C * Q_{TC_i} \quad (A-63)$$

$$W_{i+1} = W_i + dW \quad (A-64)$$

then a new P_{i+1} is computed

$$P_{i+1} = P_{ch} - \frac{C}{2} * V_i^2 - \int_0^x \frac{V_i}{A_i} dW \quad (A-65)$$



The flow-pressure pattern is computed under the trunk step-by-step out to the separation point. The flow computation is iterated by varying $W(1)$ until the exit pressure at the separation point is within bounds.

APPENDIX B

THE EIGENVALUE PROGRAM

This program was developed as a part of the contract. By generating eigenvalues and eigenvectors of the trunk for various models, this program can assist in understanding the trunk behavior.

B.1 Program Description

The eigenvalue program has three parts. The first part analyzes the two dimensional vibration of a membrane using coupled longitudinal and lateral motions including tension and elasticity effects. The second part analyzes vibration of a membrane which does not have longitudinal motion and vibrates only in the lateral direction similar to a stretched string. The third part of the program is relevant for an elastic membrane which can vibrate only in the longitudinal direction. This mode of vibration is similar to that of a bar. The program can work with damped or undamped systems. In addition, the program calculates the natural frequencies for an equivalent string. The output of the program consists of the eigenvalues and eigenvectors (optional).

A list of the subroutines used is presented in Table B-1. The input data description for the program and a sample output are also presented in this Appendix. However, first the various models used in the program are described in Table B-1.

TABLE B-1. A SUMMARY OF EIGENVALUE PROGRAM SUBROUTINES

No.	Subroutine	Primary Function	Group
1	FMAEVEC	Main program; I/O control, coordinate analysis; form matrices	*MAIN*
2	CLEAR	Clear matrix to zero	MATRIX
3	PUTMAT	Print matrix	I/O
4	TRUNK	Computer trunk shape	GEOMETRY
5	ELEMK	Form element stiffness matrix	GEOMETRY
6	MOVE	Copy matrix	MATRIX
7	EIGPAC	Eigenvalue/eigenvector computation coordination module	EIGEN
8	CMINV	Complex matrix inversion	MATRIX
9	PUTEIG	Print eigenvalues	I/O
10	HSBG	SSP eigenvalue routine	EIGEN
11	ATEIG	SSP eigenvalue routine	EIGEN
12	VECPAC	Eigenvector computation and output coordination module	EIGEN
13	EVECTR	Solve complex system of equations	EIGEN

B.2 Models Used in the Eigenvalue Program

B.2.1 Lateral Vibration Model

In Figure B-1, the force due to displacement y_p is:

$$F_p = -T \sin(\alpha_{p-1}) + T \sin(\alpha_p) \quad (B-1)$$

For a first order approximation,

$$\sin(\alpha_{p-1}) \approx \frac{y_p - y_{p-1}}{\ell_{p-1}}$$

$$\sin(\alpha_p) \approx \frac{y_{p+1} - y_p}{\ell_p}$$

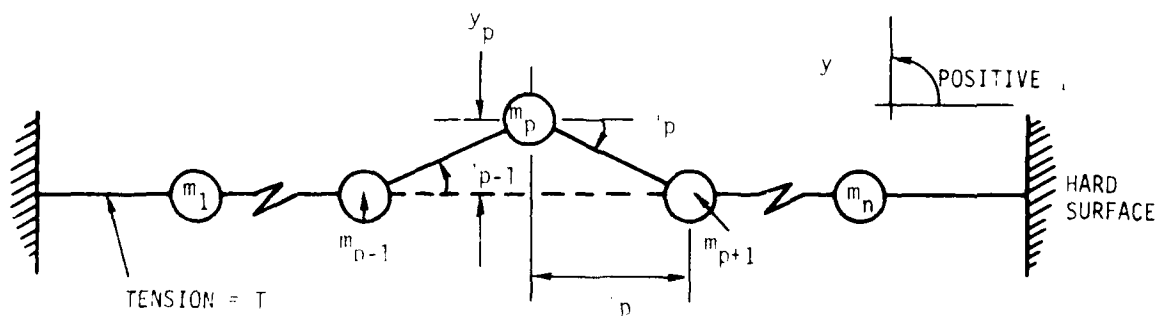


Figure B-1. Lateral vibration model.

therefore,

$$F_p = - \frac{T}{\ell_{p-1}} (y_p - y_{p-1}) + \frac{T}{\ell_p} (y_{p+1} - y_p). \quad (B-2)$$

From

$$\bar{F} = M\bar{A}$$

$$F_p = m_p \frac{d^2 y_p}{dt^2}$$

$$\frac{d^2 y_p}{dt^2} = \frac{-T}{m_p \ell_{p-1}} (y_p - y_{p-1}) + \frac{T}{m_p \ell_p} (y_{p+1} - y_p), \quad (B-3)$$

let

$$\lambda_p = \omega_p^2 = \frac{T}{m_p \ell} : \text{ for } \ell_p = \ell_{p-1} = \ell$$

Then,

$$\frac{d^2 y_p}{dt^2} + \omega_p^2 y_p - \omega_p^2 (y_{p+1} + y_{p-1}) = 0 \quad (B-4)$$

where the boundary conditions are:

$$y_0 = y_{n+1} = 0.$$

Writing equation (B-4) in a Matrix form:

$$\ddot{\underline{Y}} + \underline{\lambda} \underline{Y} = 0$$

$$\begin{bmatrix} \ddot{y}_1 \\ \ddot{y}_2 \\ \ddot{y}_3 \\ . \\ . \\ . \\ \ddot{y}_n \end{bmatrix} + \begin{bmatrix} 2\lambda_1, -\lambda_1, & 0 \dots 0 \\ -\lambda_2, 2\lambda_2, -\lambda_2, & 0 \dots 0 \\ 0, -\lambda_3, 2\lambda_3, -\lambda_3, & 0 \dots 0 \\ . & . \\ . & . \\ . & . \\ 0 \dots 0, & -\lambda_n, 2\lambda_n \end{bmatrix} * \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ . \\ . \\ y_n \end{bmatrix} = 0 \quad (B-5)$$

Solution $\det |\underline{\lambda}| = 0$ gives eigenvalues of the vibration modes.

$$\lambda_i = \omega_i^2$$

The addition of damping to the model requires the addition of a damper force at each node.

$$F_{DP} = -V_P * B_P \quad (\text{see Figure B-2}) \quad (\text{B-6})$$

where

B_P = Damping constant of point P

V_P = Velocity of point P

so equation (B-4) becomes:

$$\begin{aligned} \frac{d^2 y_p}{dt^2} = & \frac{-T}{m_p \ell_{p-1}} (y_p - y_{p-1}) \\ & + \frac{T}{m_p \ell_p} (y_{p+1} - y_p) - \frac{B \dot{y}_p}{m_p} \end{aligned} \quad (\text{B-7})$$

where

$$\frac{d}{dt} (y_p) = \dot{y}_p$$

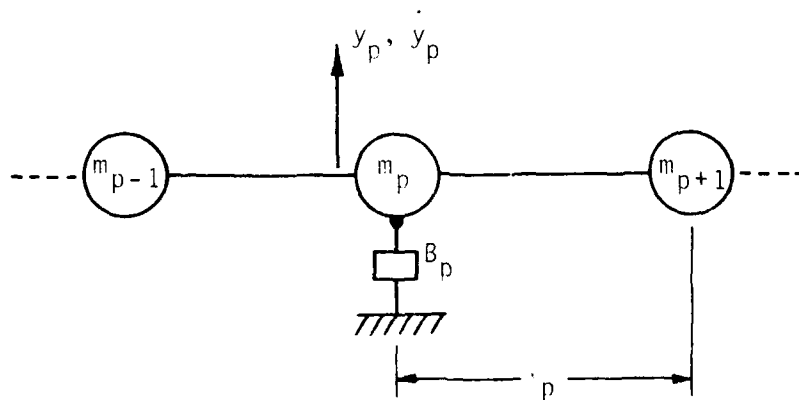


Figure B-2. Nodal dampers (lateral).

The matrix formulation (B-5) becomes (using $\ell_p = \ell_{p-1}$ assumption):

$$\begin{bmatrix} \ddot{Y}_1 \\ \dot{Y}_1 \\ \ddot{Y}_2 \\ \dot{Y}_2 \\ \ddot{Y}_3 \\ \dot{Y}_3 \\ . \\ . \\ . \\ \ddot{Y}_n \\ \dot{Y}_n \end{bmatrix} + \begin{bmatrix} \frac{B}{m_1}, & 2\lambda_1, & 0, & -\lambda_1, & 0 \dots \\ 1, & 0, & \dots \\ 0, & -\lambda_2, & \frac{B}{m_2}, & 2\lambda_2, & 0, & -\lambda_2, & 0 \dots \\ 0, & 0, & 1, & 0 \dots \\ 0, & 0, & 0, & -\lambda_3, & \frac{B}{m_3}, & 2\lambda_3, & 0, & -\lambda_3, & 0 \dots \\ 0, & 0, & 0, & 0, & 1, & 0, & \dots \\ . & . & . & . & . & . & . & . & . \\ 0 \dots & & & & 0, & -\lambda_n, & \frac{-B}{m_n}, & 2\lambda_n \\ 0 \dots & & & & 0, & 1, & 0 \end{bmatrix} \begin{bmatrix} \dot{Y}_1 \\ Y_1 \\ \dot{Y}_2 \\ Y_2 \\ \dot{Y}_3 \\ Y_3 \\ . \\ . \\ . \\ \dot{Y}_n \\ Y_n \end{bmatrix} \quad (B-8)$$

Solution of $\det [\underline{\lambda}] = 0$ gives the eigenvalues (natural frequencies) of the vibration modes.

B.2.2 Longitudinal Vibration Model

In Figure B-3 the force due to displacement X_i is:

$$F_i = -K_i X_i - K_{i+1} X_i + K_{i+1} X_{i+1} + K_i X_{i-1} \quad (B-9)$$

$$F_i = -(K_i + K_{i+1}) X_i + K_{i+1} X_{i+1} + K_i X_{i-1} \quad (B-10)$$

From,

$$\bar{F} = \bar{M}\bar{A}$$

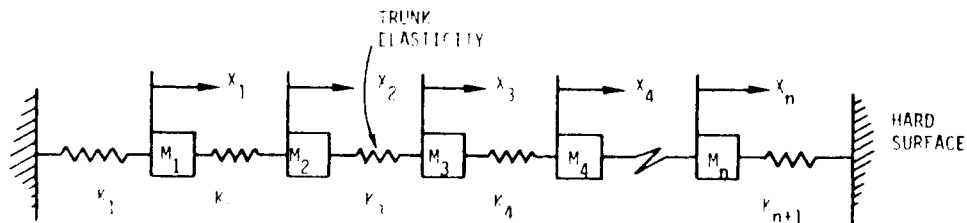


Figure B-3. Longitudinal vibration model.

$$F_i = m_i \frac{d^2 x_i}{dt^2} \quad (B-11)$$

therefore,

$$\begin{aligned} \frac{d^2 x_i}{dt^2} - \frac{K_i}{m_i} x_{i-1} - \frac{K_{i+1}}{m_i} x_{i+1} \\ + \left(\frac{K_i + K_{i+1}}{m_i} \right) x_i = 0 \end{aligned} \quad (B-12)$$

where boundary conditions are

$$x_0 = x_{n+1} = 0$$

In matrix form:

$$\ddot{\underline{X}} + \underline{\lambda} \underline{X} = 0$$

$$\underline{\lambda} = \underline{M}^{-1} \underline{K}$$

$$\begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \ddot{x}_3 \\ \vdots \\ \ddot{x}_n \end{bmatrix} + \begin{bmatrix} \frac{(K_1 + K_2)}{m_1}, \frac{-K_2}{m_1}, 0 \dots \\ \frac{-K_2}{m_2}, \frac{(K_2 + K_3)}{m_2}, \frac{-K_3}{m_3}, 0 \dots \\ 0, \frac{-K_3}{m_3}, \frac{(K_3 + K_4)}{m_3}, \frac{-K_4}{m_4} \dots \\ \vdots \\ 0 \dots \frac{-K_n}{m_n}, \frac{(K_n + K_{n+1})}{m_n} \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = 0 \quad (B-13)$$

Solution of $\det |\underline{\lambda}| = 0$ gives the eigenvalues of the vibration modes.

$$\lambda_i = \omega_i^2$$

Addition of damping to the model requires the addition of a damper force at each node.

$$F_{Di} = -V_i * B_i \quad (B-14)$$

(see Figure B-4).

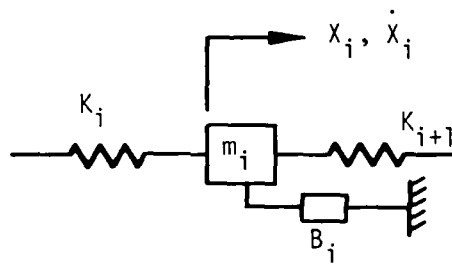


Figure B-4. Nodal dampers (longitudinal).

Equation (B-12) becomes

$$\begin{aligned} \frac{d^2 x_i}{dt^2} - \frac{K_i}{m_i} x_{i-1} - \frac{K_{i+1}}{m_i} x_{i+1} - \left(\frac{K_i + K_{i+1}}{m_i} \right) x_i \\ + \frac{B_i}{m_i} \dot{x}_i = 0 \end{aligned} \quad (B-15)$$

In matrix form:

$$\begin{bmatrix} \ddot{x}_1 \\ \dot{x}_1 \\ \ddot{x}_2 \\ \dot{x}_2 \\ \cdot \\ \cdot \\ \cdot \\ \ddot{x}_n \\ \dot{x}_n \end{bmatrix} + \begin{bmatrix} \frac{B_1}{m_1}, \frac{(K_1 + K_2)}{m_1}, 0, \frac{-K_2}{m_1}, 0, \dots \\ 1, 0, \dots \\ 0, \frac{-K_2}{m_2}, \frac{+B_2}{m_2}, \frac{(K_2 + K_3)}{m_2}, 0, \frac{-K_3}{m_2}, 0, \dots \\ 0, 0, 1, 0, \dots \\ \cdot \\ \cdot \\ \cdot \\ 0, \dots, 0, \frac{-K_n}{m_n}, \frac{B_n}{m_n}, \frac{(K_n + K_{n+1})}{m_n} \\ 0, \dots, 0, 1, 0 \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ x_1 \\ \dot{x}_2 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ \dot{x}_n \\ x_n \end{bmatrix} \quad (B-16)$$

B.2.3 Coupled Longitudinal and Lateral Motion

The two dimensional matrix formulation requires a two coordinate vector (x_n, y_n) at each node creating a $2 \times \text{NODES}$ state space. The matrix formulation of the model requires linearization of the equations about some configuration of the membrane. Each spring element has its linear stiffness matrix converted to the global coordinate frame as shown in Figure B-5.

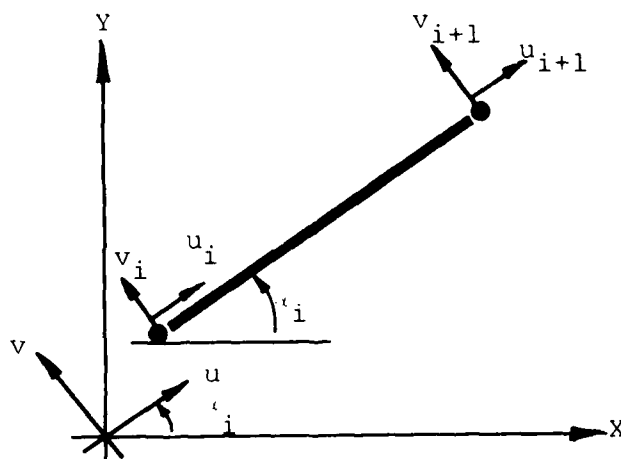


Figure B-5. Element coordinate transformation.

Stiffness of an element

$$\underline{K} = \int_{\ell} \underline{B}^T \underline{K}_B d\ell$$

$$\underline{K} = K_c \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \quad \text{in one dimension}$$

$$\bar{K}_i = \begin{bmatrix} \cos^2 \alpha & \sin \alpha \cos \alpha & -\cos^2 \alpha & -\sin \alpha \cos \alpha \\ \sin \alpha \cos \alpha & \sin^2 \alpha & -\sin \alpha \cos \alpha & -\sin^2 \alpha \\ -\cos^2 \alpha & -\sin \alpha \cos \alpha & \cos^2 \alpha & \sin \alpha \cos \alpha \\ -\sin \alpha \cos \alpha & -\sin^2 \alpha & \sin \alpha \cos \alpha & \sin^2 \alpha \end{bmatrix} \begin{Bmatrix} u_i \\ v_i \\ u_{i+1} \\ v_{i+1} \end{Bmatrix}$$

in two dimensions

$$\underline{R} = [K]_i \{\delta\}_i$$

$$\underline{U}^T = \begin{bmatrix} U_1 V_1 & U_2 V_2 \end{bmatrix}$$

$$\underline{K} = \sum_1^{N+1} \underline{K}_i ; \quad \underline{KU} = \underline{R} \quad (B-17)$$

$\underline{K}$ = global stiffness matrix

$\underline{U}$ = displacement vector

$\underline{R}$ = forcing load vector.

The membrane tension stiffness effects must be added to the element spring stiffness matrix. The computation of the two dimensional equivalent stiffness for tensile forces requires a linearization of transverse nodal motion into the second coordinate frame.

From lateral one dimensional development earlier, effective lateral stiffness due to tension:

$$\hat{k} = \frac{T*2}{(\ell_i + \ell_{i-1})} \quad \text{in Figure B-6} \quad (B-18)$$

since

$$\hat{k} * \cos(\hat{\alpha}) \hat{k}_y$$

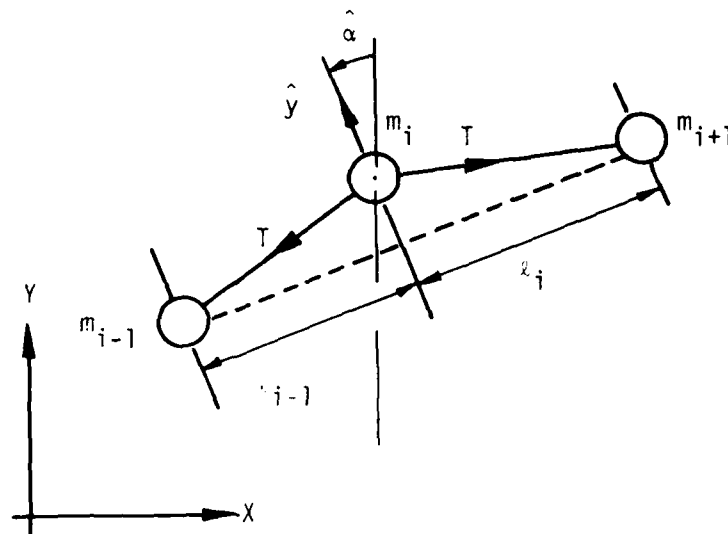


Figure B-6. Transverse node displacement.

$$\hat{k} * \sin(\hat{\alpha}) \approx k_x$$

with

$$\hat{\alpha}_i = \left(\frac{\alpha_i + \alpha_{i-1}}{2} \right); \quad \ell_a = \frac{\ell_i + \ell_{i-1}}{2}$$

then

$$F_T = \frac{-T}{\ell_a} (\hat{y}_i - \hat{y}_{i-1}) + \frac{T}{\ell_a} (\hat{y}_{i+1} - \hat{y}_i)$$

or

$$F_T = K(\hat{y}_{i+1} - 2\hat{y}_i + \hat{y}_{i-1}) \quad (B-19)$$

($\hat{y}$ is rotated relative to y by α .)

The lateral $\hat{y}$ displacement must be transformed into (X,Y) frame as follows

Transverse displacement D transformation into (X,Y) frame

$$\Delta y \cos(\hat{\alpha}_i) + \Delta x \sin(\hat{\alpha}_i) = D \quad (\text{Figure B-7}) \quad (B-20)$$

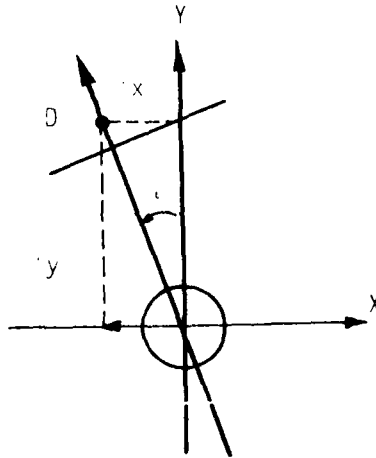


Figure B-7. Force transformation.

$$\hat{y}_i = y_i [\cos (\hat{\alpha}_i)] + x_i [\sin (\hat{\alpha}_i)]$$

$$F_{X_i} = \frac{T}{\ell_a} \sin \hat{\alpha}_i \left[\overbrace{\left[\sin \hat{\alpha}_{i-1} x_{i-1} + \cos \hat{\alpha}_{i-1} y_{i-1} \right]}^{\hat{y}_{i-1}} - 2(\hat{y}_i) + \hat{y}_{i+1} \right] \quad (B-21)$$

$$F_{Y_i} = \frac{T}{\ell_a} \cos \hat{\alpha}_i \left[\hat{y}_{i-1} - 2\hat{y}_i + \hat{y}_{i+1} \right] \quad (B-22)$$

B.3 Eigenvalue Input Data

1. ICNTL(I), I = 1, 10 IPRNT(I), I = 1, 10 (2011)

Program Control vectors

Values

ICNTL (N) = 1, Eigenvalues; 2, Eigenvectors; 0, skip
 N = (1) ; compute two-dimensional Eigenvalues
 N = (2) ; compute two-dimensional damped Eigenvalues
 N = (3) ; compute transverse string undamped Eigenvalues
 N = (4) ; compute transverse damped Eigenvalues
 N = (5) ; compute longitudinal bar undamped Eigenvalues
 N = (6) ; compute longitudinal bar damped Eigenvalues

IPRNT N = 1, print matrix as above; 0, No print

2. NODES (12)

NODES - Number of nodes

3. LS, RL, AX, BX, TENSX, HY (8G10.5)

LS - membrane length, stretched

RL - membrane length, unstretched

AX - horizontal distance between attachment point

BX - vertical distance between attachment point

TENSX - membrane preload tension

HY - trunk height

4. MASS(I), I = 1, nodes (8G10.5)

MASS - lumped parameter nodal mass

5. RKVEC(I), I = 1, nodes + 1 (8G10.5)

RKVEC - elastic stiffness of element

6. DAMP(I), I = 1, nodes (8G10.5)

DAMP - nodal damping in both x and y directions

7. IXF (20I1)

IXF - element length select flag

(0, default; 1, read card 8)

8. (Option) RLENGO(I), I = 1, nodes + 1 (8G10.5)

RLENGO - element unstretched length

9. IXY (2011)

IXY - coordinate point select flag
(0, read cards 10, 11; 1, compute)

10. (Option) X(I), I = 1, nodes (8G10.5)

X - X values of mass nodes

11. (Option) Y(I), I = 1, nodes (8G10.5)

Y - Y values of mass nodes.

B.4 Eigenvalue Program Output

The printout includes the following data:

- a. Input Parameters - Trunk parameters, structural parameters, and program control data are printed out after input. A sample input is shown in Figure B-8.
- b. Matrix Model Data (Optional) - The matrices generated for the models can be printed out. These matrices are the global stiffness matrices of the algebraic models for the trunk.
- c. Eigenvalues - The Eigenvalues and natural frequencies of the matrix models are printed out in radian and Hertz frequencies, respectively.
- d. Eigenvectors (Optional) - The Eigenvectors for each Eigenvalue of complex pair of Eigenvalues are printed and then the normalized displacement Eigenvectors (velocity terms for damped models are neglected) are printed.

[illegible]

Figure B-8. Sample input data.

A list of program output variables in sequential order is:

1. Number of nodes, NODES
2. Membrane length, LS; rest length, RL, X end point, AX; Y end point, BX; membrane tension, TENS
3. Mass Modes, MASS(I)
4. Elastic stiffness, RKVEC(I)
5. Damping ratio (global) at node, DAMP(I)
6. Membrane element unstretched length, RLENGO
7. Node coordinate positions, X(I), Y(I)
8. Continuous string frequencies, W(I)
9. 2D stiffness matrix, G(I,J) (Option)
10. Eigenvectors of matrix, Z, W(I), Y(I), Hertz, radians/sec, eigenvalue (real, imaginary)
11. Eigenvectors, X(I), Y(I) eigenvector (real, imaginary) (Option)
12. Eigenvectors, X(I), normalized displacement terms only (Option)
13. 2D damped stiffness matrix, A(I,J) (Option)
14. As 10 above, eigenvalue
15. As 11 above, eigenvector

AD-A082 007

FOSTER-MILLER ASSOCIATES INC WALTHAM MASS

F/8 1/3

ANALYSIS OF TRUNK FLUTTER IN AN AIR CUSHION LANDING SYSTEM. USE--ETC(U)

NOV 79 R B FISH

F33615-78-C-3412

UNCLASSIFIED

AFFDL-TR-79-3121

NL

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

2-2

END

DATE

FILED

4-80

DTIC

16. As 12 above, normalized eigenvector
17. 1D lateral stiffness matrix, GS(I,J) (Option)
18. As 10 above, eigenvalue
19. As 11 above, eigenvector
20. As 12 above, normalized eigenvector
21. 1D damped lateral stiffness matrix, XW(I,J) (Option)
22. As 10 above, eigenvalues
23. As 11 above, eigenvectors
24. As 12 above, normalized eigenvectors
25. 1D longitudinal stiffness matrix, GS(I,J) (Option)
26. As 10 above, eigenvalues
27. As 11 above, eigenvectors
28. As 12 above, normalized eigenvectors
29. 1D damped longitudinal stiffness matrix, XW(I,J) (Option)
30. As 10 above, eigenvalues
31. As 11 above, eigenvectors
32. As 12 above, normalized eigenvectors.

Figure B-9 shows a sample program output.

B.5 Principal Nomenclature

<u>Program Variable Name</u>	<u>Symbol</u>	<u>Explanation</u>
A, XW, XZ(I,J)		Matrix, global stiffness
AS, G, GS(I,J)		Matrix, global stiffness
AT, WORKM, XT(I,J)		Matrix, global stiffness
AX	a	Horizontal distance between attachment points
BX	b	Vertical distance between attachment points
COSALP(I)	$\cos(\theta_i)$	Cosine of ALPHA(I)
COSTHE(I)	$\cos(\theta_i)$	Cosine of THETA(I)
DAMP(I)	D_i	Damping ratio of node (I)
ESTIF(I,J)		Matrix, element stiffness
EVEC(I)		Work vector space
ICNTL(I)		Program control vector
IPRNT(I)		Printer control vector
IXF		Flag, element initial length selection
IXY		Flag, X, Y position selection
LS	ℓ_s	Trunk length
L1	ℓ_1	Trunk inner length
L2	ℓ_2	Trunk outer length
MASS(I)	m_i	Nodal mass
NODES		Number of nodes
PHI1	ϕ_1	Trunk angle, inner
PHI2	ϕ_2	Trunk angle, outer

<u>Program Variable Name</u>	<u>Symbol</u>	<u>Explanation</u>
R1	R_1	Trunk radius, inner
R2	R_2	Trunk radius, outer
RKVEC(I)	RK_i	Trunk elasticity, element (I)
RL		
RLENG(I)	RL_i	Trunk element length (I)
RLENGO(I)	RL_{0i}	Trunk element initial length (I)
SINALP(I)	$\sin(\theta_i)$	Sine of ALPHA(I)
SINTHE(I)	$\sin(\theta_i)$	Sine of THETA(I)
TENSN	T	Tension in trunk
W	ω	String frequency
WORK1(I)		Matrix, work space
WORK2(I)		Matrix, work space
X(I)	X_i	X position of node (I)
Y(I)	Y_i	Y position of node (I)

APPENDIX C
PRINCIPLE PROGRAM NOMENCLATURE

The variables used in the flutter simulation program are defined in this appendix. Also mentioned, corresponding to the appropriate computer program variables, are the symbols used in the analysis of the trunk model. All program variables are in ft-lb-sec units except where indicated to the contrary.

<u>Program Variable Name</u>	<u>Symbol</u>	<u>Explanation</u>
A	a	Horizontal distance between inner and outer trunk attachment point
AGAP(I)	A_i	Flow gap trunk to ground
AIFAN	I_f	Fan inertance
ATC	A_{tc}	Area trunk to cushion
ATCF(I)	A_{tcf_i}	Area trunk to cushion element
ATRIM	A_{tr}	Area trim valve
B	b	Vertical distance between inner and outer trunk attachment point
CGAP	C_g	Discharge coefficient gap flow
CKK		Polytropic expansion coefficient
COSPHI(I)	$\text{Cos}(\phi_i)$	Cosine of PHI (I)
COSTHE(I)	$\text{Cos}(\theta_i)$	Sine of THETA (I)
CQ0	α_0	Fan polynomial coefficient 1
CQ1	α_1	Fan polynomial coefficient 2
CQ2	α_2	Fan polynomial coefficient 3
CQ3	α_3	Fan polynomial coefficient 4
CQ4	α_4	Fan polynomial coefficient 5

<u>Program Variable Name</u>	<u>Symbol</u>	<u>Explanation</u>
CTC	C_{tc}	Discharge coefficient trunk to cushion, flow other than trim.
CTRIM	C_{tr}	Discharge coefficient trim value
DAMP(I)	D_i	Damping ratio X, Y at node (I)
DAMPR		Damper rest value
DERY(I)	$\frac{d}{dt} (S_i)$	Derivatives of state variables
DQ	dq	Incremental flow trunk-channel
DTIME	dt	Integration time step
DVCH	$\frac{d}{dt} (V_{ch})$	Cushion volume rate of change
DW(I)	dw_i	Incremental mass flow trunk - channel element (I)
FORCXB(I)	F_{xb_i}	Force, X direction, bending (I)
FORCXD(I)	F_{xd_i}	Force, X direction, damping (I)
FORCXB(I)	F_{xk_i}	Force, X direction, elasticity (I)
FORCXP(I)	F_{xp_i}	Force, X direction, pressure (I)
FORCYB(I)	F_{yb_i}	Force, Y direction, bending (I)
FORCYD(I)	F_{yd_i}	Force, Y direction, damping (I)
FORCYK(I)	F_{yk_i}	Force, Y direction, elasticity (I)
FORCYP(I)	F_{yp_i}	Force, Y direction, pressure (I)
GAPMIN		Minimum gap area
HY		Trunk height
ICFLAG		Flag, trunk shape selection

<u>Program Variable Name</u>	<u>Symbol</u>	<u>Explanation</u>
ICNTL(I)		Control vector for options
ICS		Cushion separation node
IFSEP		Flag, separation point selection
ILENG		Flag, segment length selection
INODE		Node number of lowest trunk point
INSEP		Stake location node
ISEP		Separation point node
L	ℓ	Trunk membrane length
L1	ℓ_1	Inner trunk length (inner attachment point to bottom)
L2	ℓ_2	Outer trunk length (outer attachment point to bottom) (ℓ_1 , ℓ_2 used only by subroutine TRUNK)
NODES		Number of mass nodes
PAT	P_a	Atmospheric pressure
PCH	P_{ch}	Cushion pressure
PCRIT		Critical flow pressure
PEXT(I)		External pressure on trunk
PHI(I)	ϕ_i	Angle PHI at node (I)
PLOST		Pressure loss due to momentum
PRESUR(I)		Pressure differential on trunk element
PTK	P_{tk}	Trunk pressure
QCA	Q_{CA}	Flow, cushion to atmosphere
QEXIT	Q_{EXIT}	Flow at exit (separation)
QFAN	Q_{FAN}	Fan flow
QGAP	Q_{GAP}	Gap flow
QIN	Q_{IN}	Flow to trunk

<u>Program Variable Name</u>	<u>Symbol</u>	<u>Explanation</u>
QINT		Momentum change integral
QOUT	Q_{out}	Flow, out of trunk
QTA	Q_{TA}	Flow, trunk to atmosphere
QTC	Q_{TC}	Flow, trunk to cushion
QTOT		Total gap flow
QTRIM	Q_{TR}	Flow, trim valve
RBVEC(I)	RB_i	Bending stiffness, node (I)
RHO	ρ	Air density
RITER		Iteration parameter
RKEXTX		External spring stiffness, X
RKEXTY		External spring stiffness, Y
RKVEC(I)	RK_i	Trunk elasticity
RLENG(I)	RL_i	Trunk element length
RLENG0(I)	RL_{0i}	Trunk element unstretched length
RMAS(I)	M_i	Nodal mass
SIE(I)	ψ_i	Angle SIE
SINPHI(I)	$SIN(\phi_i)$	Sine of PHI(I)
SINTHE(I)	$SIN(\theta_i)$	Sine of THETA(I)
SSLENG		Trunk gap length
STATE(I)	S_i	State variable (I)
TEMPAT		Atmospheric temperature
THETA(I)	θ_i	Angle THETA(I)
TIME	t	Simulation time
TKVOL	V_{tk}	Trunk volume
TREST		Damper reset time

<u>Program Variable Name</u>	<u>Symbol</u>	<u>Explanation</u>
TSEP		Separation point angle
VCH	V_{ch}	Volume of cushion
VEL(I)	V_i	Velocity of flow at node (I)
VEXIT	V_{exit}	Initial estimate of flow velocity at separation point
VSEP	V_s	Velocity at separation point
W(I)	W_i	Mass flow at node (I)
X(I)	X_i	X position of node (I)
XEXTO		X position of spring at rest
XZETA	ξ_x	X damping ratio
Y(I)	Y_i	Y position of node (I)
YASEP	Y_s	Flow separation point area
YCSEP		Cushion separation point area
YDMIN		Minimum trunk Y displacement
YEXTO		Y position of spring at rest
YGAPM		Minimum gap area
YGRNDS		Hard surface clearance
YSEPX		Separation point gap

APPENDIX D

PROGRAM LISTINGS

All programs and subroutines in this report have been designed to work under ANSI.66 FORTRAN IV and supersets of the former.

- Dynamic Program Listings (subsection D.1)
- Eigenvalue Program Listings (subsection D.2)

NOTE

Subroutines HSBG, and ATEIG have been omitted. Information on these routines can be found in the IBM SSP manual.

D.1 Dynamic Simulation Programs

The following programs and subroutines are included.

Programs - DYSYS

Subroutines - RKDIF
EQSIM
TRUNK
PUTVEC
ERROR
PLOTTER
PACKER
PRNTPLOT
PLOT
PSTORE

```

PROGRAM DYSYS(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE4)
C ****
C ***** FOSTER MILLER ASSOCIATES
C ***** ANALYSIS AND INSTRUMENTATION GROUP
C ***** 350 SECOND AVE
C ***** WALTHAM,MASS. 02154
C ***** (617) 890-3200
C ***** COPYRIGHT 1979
C *****
      REAL YMIN(100),YMAX(100)
      DIMENSION IPRMT(9),YPRMT(9)
      COMMON T,TSTEP,Y(100),F(100),STIME,ETIME,HEADT,IFWRT,N,
      1 IPR,ICD,ICN,INEXT,PHEXT,TRACK
      COMMON/PLT/NPLT(10),TRISPT,TRISPT,NPLTN,OTPLNT,XPLNTX(5,200)
      1 ,NNTM,NVPLNT,XMIN,XMAX,NMAX,NSTORE
      COMMON/IOLIST/NTYP,NINP,IPTAPE
C
C NTYP = OUTPUT DEVICE CHANNEL
C NINP = INPUT DEVICE CHANNEL
C
C HEADT IS NON-ZERO IF IT IS OK TO CHANGE OT (OR SCALING)
C
C FORMATS
      1 FORMAT(3G12.5)
      2 FORMAT(8G10.0)
      3 FORMAT(1P,10G12.4)
      4 FORMAT(1P,6X,*MAXIMUM ORDER OF SYSTEM=*,I3,6X,*INITIAL TIME=*,
      1G11.4, 3X,*FINAL TIME=*,G11.4,6X,*TIME STEP=*,G12.5)
      5 FORMAT (1H1 6X*SOLUTION OF STATE EQUATIONS USING DYSYS*//)
      6 FORMAT (9I2)
      7
      8 FORMAT(/15X,9(A1,*STATF*,6X))
      9 FORMAT(16X,9(A1,*(*,I2,*),*,7X))
      12 FORMAT(5X,*TIME=*,5X,9(A1,*VARIABLE*,3X))
      14 FORMAT(6X,*Y(*,I2,*),*,1P,2G18.4)
      15 FORMAT(/*NEXTPRME VALUES OF STATE AND AUXILIARY VARIABLES*/
      1 T2A,*MINIMUM*,T42,*MAXIMUM*)
C

```

```

C SFT UP DEVICE NUMBERS
  NTYP=6
  NINP=5
  IPTAPE=4
  ICD=NINP
  IPR=NTYP
  N=100

C INPUT INTEGRATION CONTROL PARAMETERS
100 READ(NINP,1) FTIME,TSSTEP,STIME
110 WRITE (IPR,6)
  WRITE (IPR,4) N,STIME,FTIME,TSSTEP
  IFWRT=0

C SFT STATES OUT OF BOUNDS
DO 120 I=1,N
  YMIN(I)=1.7E50
  YMAX(I)=-1.7E50
  F(I)=0.0
120 Y(I)=0.0

C
C FIND PRINTING VECTOR LENGTH.
  READ (ICD,7) IPRNT
  READ(NINP,7) NPLSW
  DO 130 I=1,9
    IF (IPRNT(I)) 140,140,130
130 CONTINUE
  I=10
  NPRNT=I-1
140
C
C ON FIRST CALL TO EOSIM, NEWDT IS NEGATIVE
  NEWDT=-1
  T=STIME

C
C CALL EOSIM FOR INITIALIZATION OF SYSTEM
  CALL EOSIM
  T=STIME
300 CONTINUE
  IF (N,LT,0) GO TO 100

```

```

C USE Y(N+1) THROUGH Y(NMAX) AS AUXILIARY STORAGE SPACE
IPR1=2
IF(NPRNT)410,410,310
310 IPR1=1
    WRITE (IPR,H) (IH,I=1,NPRNT)
    *WRITE (IPR,12) (IR,I=1,NPRNT)
    *WRITE (IPR,9) (IH,IPR,I(1),I=1,NPRNT)
    *WRITE (IPR,2)
C PRINT RESULTS
320 GO TO (340,410),IPR1
340 DO 350 I=1,NPRNT
    L=IPRNT(I)
350 YPRNT(I)=Y(L)
    *WRITE (IPR,3) T,(YPRNT(I),I=1,NPRNT)
C SAVE MAXIMUM AND MINIMUM
410 DO 450 I=1,N
    IF (Y(I)-YMIN(I)) 420,430,430
420 YMIN(I)=Y(I)
430 IF (Y(I)-YMAX(I)) 450,450,440
440 YMAX(I)=Y(I)
450 CONTINUE
C TEST FOR COMPLETE
C GET OUT IF T IS NOT BETWEEN STIME AND FTIME
    TX=T+.5*STSTEP
    IF((STIME-TX)*(FTIME-TX))460,480,480
C CALL INTEGRATION ROUTINE FOR TIME STEP
460 CALL RKDIF
    GO TO 320
480 IF(NPRNT.LE.0) GO TO 520
C
C *WRITE OUT MAXIMUM AND MINIMUM OF VARIABLE IF IT WAS USED
    WRITE(IPR,15)
    DO 510 I=1,N
    IF(YMIN(I))500,490,500
490 IF(YMAX(I))500,510,500
500 *WRITE(IPR,14)I,YMIN(I),YMAX(I)
510 CONTINUE
C

```

```
C IF PLOTTING OVER CALLS OUTPUT ROUTINES
520 IF (NPL14.GE.1) CALL PLOT1F
600 WRITE(N1YP,9010)
9010 FORMAT(5X,1H4 END OF OVSYS JOB '/')
      END
```

```

SUBROUTINE RKDIF
C RUNGE-KUTTA 4TH ORDER INTEGRATION ROUTINE
C
    REAL SY(100),Y0(100),Y1(100),Y2(100)
    COMMON T,TSTEP,Y(100),DY(100),STIME,FTIME,NEWDT,IFWRT,N,
    1 IPR,ICD,ICN,TNEXT,PNEXT,TBACK
    EQUIVALENCE (DT,TSTEP),(N,NSYS)

C NEWDT IS NON-ZERO IF IT IS OK TO CHANGE TIME STEPS
C SET NEWDT TO LOCK OUT CHANGES IN DT AND INPUTS
C
    NEWDT=0
    H=DT/2.0
    DO 10 I=1,NSYS
        SY(I)=Y(I)
        Y0(I)=DY(I)
        Y(I)=H*DY(I)+Y(I)
    10 C
        T=T+H
        CALL EOSIM
C
    DO 20 I=1,NSYS
        Y1(I)=DY(I)
        Y(I)=SY(I)+H*DY(I)
    20 C
        CALL EOSIM
C
    DO 30 I=1,NSYS
        Y2(I)=DY(I)
        Y(I)=SY(I)+DT*DY(I)
    30 C
        T=T+H
        CALL EOSIM
        H=H/3.0
C
    DO 40 I=1,NSYS
        PRT1=2.0*(Y1(I)+Y2(I))
        PRT2=Y0(I)+DY(I)

```

```

      Y(I)=SY(I)+H*PRT1+H*PH12
      CONTINUE
40    C SFT NE*DT = -1 FOR END OF STEP
      NE*DT=-1
      CALL EQSIM
      RETURN
      END

```

```

SURROUTINE EOSIM
C DYSYS DYNAMIC INTEGRATION STATE EQUATION SURROUTINE
C *****
C DYSYS STATE EQUATIONS FOR ACLS TRUNK
C LUMPED PARAMETER MEMBRANE MODEL
C
C UNITS ARE IN FT,SLUG,SECOND SYSTEM
C
C NODES 1 AND NUMBER+2 ARE FIXED BOUNDARY POINTS
C FOR I=5,NUMBER OF NODES+1,4
C STATE(I) = X(I) VELOCITY
C STATE(I+1) = X(I) POSITION
C STATE(I+2) = Y(I) VELOCITY
C STATE(I+3) = Y(I) POSITION
C
C FORCE COMPONENT VECTOR(X,Y)
C FORCX0,FORCY0 = DAMPER FORCE COMPONENTS
C FORCXK,FORCYK = SPRING FORCE COMPONENTS
C FORCXP,FORCYP = PRESSURE FORCE COMPONENTS
C FORCXB,FORCYB = BENDING FORCE COMPONENTS
C *****
C *****
C *****
C ***** FOSTER MILLER ASSOCIATES
C ***** ANALYSIS AND INSTRUMENTATION GROUP
C ***** 350 SECOND AVE
C ***** WALTHAM,MASS. 02154
C ***** (617) 890-3200
C ***** COPYRIGHT 1979
C *****
C REAL L,L1,L2
C INTEGER*2 XLAB
C LOGICAL LDATS(16)

```

```

C
DIMENSION RLENG(42),RLENG0(42)
DIMENSION RLENG0(42)
DIMENSION RMAS(40),RKVFC(42)
DIMENSION DAMP(42),X(48),Y(48)
DIMENSION PRESUR(42)
DIMENSION SINTE(42),COSTHE(42)
DIMENSION COSPHI(42),SINPHI(42)
DIMENSION FORCXP(42),FORCYP(42)
DIMENSION FORCXK(42),FORCYK(42)
DIMENSION FORCXD(42),FORCYD(42)
DIMENSION PEXT(42),AGAP(44)
DIMENSION RKFACT(5)
DIMENSION RKSAV(42),PLSAV(42)
DIMENSION THETA(42)
DIMENSION ICNTL(16)
DIMENSION PHI(42),SIE(42),RRVEC(42)
DIMENSION FORCXB(42),FORCYB(42)
DIMENSION XIAB(40),FORCXG(42),FORCYG(42)
DIMENSION W(42),VEL(42)
DIMENSION ATCF(42)
DIMENSION DW(42)

C
COMMON TIME,DTIME,STATE(100),DERY(100),STIME,FTIME,NEWDT,IFWRT,N,
1 IPR,ICN,TNEXT,PNEXT,TRACK
COMMON/GEOMFT/A,H,HY1,U,HY,PHI1,PHI2,R1,R2,I1,U2,TSHAPE
COMMON/IOI/IST/NTYP,NIMP,IPTAPE
COMMON/PLOT/NPLOT(10),TPISRT,TPLSTP,NPI,TM,DTPI,OT,XPLOTX(5,200)
1 ,NDIM,NVPLNT,XMIN,XMAX,NPMAX,NSTORE

C
EQUIVALENCE (TX,THX)
EQUIVALENCE (RLENG(1),RLENG0(1))
EQUIVALENCE(QPXTT,QOUT)
EQUIVALENCE(CDGAP,CGAP)

C
C ORIFICE FLOW EQUATION
QFLOW(X,Y)=X*SORT(ABS(Y))*SIGN(STIRHO,Y)
C

```

```
C FAN DYNAMICS PRESSURE VS. FLOW EQUATION
C   PFA''X(0)=C00+C01*Q+C02*Q**2+C03+Q**4
C *****
C EOSM ENTRY POINT
C *****
C *****
C NENDT = -1 ON FIRST CALL, =1 ON 4TH CALL PER TIME STEP ELSE 0
C   IF(NENDT)1,2,2
C *****
C INPUT INITIAL DATA
C CALLED FOR SIMULATION SET UP ONCE
C CONTINUE
C   NSMAX=100
C   NSTORF=100
C   NMW=40
C   NMW1=41
C   NMW2=NMW*2
C   PI=3.141592
C   YDMIN=.0
C   G=37.174
C   PCINT=.0
C   RMIN=.0
C   DVCH=.0
C   CKK=1.4
C   PAT=14.7*144.0
C   TEMPAT=70.0
C   RHG=1.241/(460.0+TEMPAT)
C   STHGH=SQRT(2.0/RHG)
C   QTC=.0
C   QTRIM=.0
C   QCA=.0
C   OFAN=.0
C CLEAR VECTORS TO ZERO FOR STATE AND DERIVATIVES
C   DERY(I)=0.0
C   DO 10 I=1,NSMAX
```

```

10 STATE(I)=0.0
   C CONTINUE
   DO 15 I=1,NMAX1
      PEXT(I)=0.0
      FORCND(I)=0.0
      FORCYD(I)=0.0
      FORCXK(I)=0.0
      FORCYK(I)=0.0
      FORCXP(I)=0.0
      FORCYP(I)=0.0
      FORCXH(I)=0.0
      FORCYH(I)=0.0
      FORCXG(I)=0.0
      FORCYG(I)=0.0
      PRESUP(I)=0.0
      RUENG(I)=0.0
15   C CONTINUE
   C
   C READ JOB LABEL CARD
      READ(NINP,9430)(XLAB(I),I=1,40)
9430  FORMAT(40A2)
      WRITE(NTYP,9431)(XLAB(I),I=1,40)
9431  FORMAT(/,5X,40A2,/)
   C
   C INPUT CONTROL VECTOR
      C ICNTL(1) ON = 1 FOR CUSHION PRESSURE DYNAMICS
      C ICNTL(2) ON = 1 FOR TRUNK-FAN PRESSURE DYNAMICS
      C ICNTL(3) ON = 1 FOR TRUNK FLOW INTO CHANNEL
      C ICNTL(4) ON = 1 FOR SEPARATION POINT INTERPOLATION
      C ICNTL(5) ON = 1 FOR SPECIAL NO TRUNK MOTION EXECUTION
      C ICNTL(6) ON = 1 FOR STATIC PRESSURE LOAD CASE
   C
      READ(NINP,9006)(ICNTL(I),I=1,16)
9006  FORMAT(80I1)
      WRITE(NTYP,9432)(ICNTL(I),I=1,16)
9432  FORMAT(5X,16I3,/)
      DO 12 I=1,16

```

```

LDATS(I)=.FALSE.
IF(ICNTL(I).GT.0) LDATS(I)=.TRUE.
12 CONTINUE
C LIST OPTIONS IN EFFECT
WRITE(NTYP,9440)
9440 FORMAT(5X,*OPTIONS IN EFFECT=*)
IF(LDATS(1)) WRITE(NTYP,9441)
9441 FORMAT(10X,*DYNAMIC CUSHION PRESSURE*)
IF(LDATS(2)) WRITE(NTYP,9442)
9442 FORMAT(10X,*DYNAMIC TRUNK-FAN PRESSURE*)
IF(LDATS(3)) WRITE(NTYP,9443)
9443 FORMAT(10X,*TRUNK TO CHANNEL FLOW*)
IF(LDATS(4)) WRITE(NTYP,9444)
9444 FORMAT(10X,*SEPARATION POINT INTERPOLATION*)
IF(LDATS(5)) WRITE(NTYP,9445)
9445 FORMAT(10X,*NO TRUNK MOTION TEST RUN*)
IF(LDATS(6)) WRITE(NTYP,9446)
9446 FORMAT(10X,*STATIC PRESSURE LOAD CASE*)
C
C INPUT NUMBER OF MASS NODES FOR ANALYSIS
C ICFLAG = 0 FOR READ X,Y ELSE COMPUTE
C IFXN = PRINT AND PLOTTER VS STEP NUMBER
C IFSEP = SEPARATION POINT SELECTION FLAG
C ILENG = ELEMENT LENGTH CONTROL FLAG
C INSEP = SEPARATION POINT NODE SET
C
READ(NTYP,9005) NODES,ICFLAG,IFXN,IFSEP
1,ILENG,INSEP
9005 FORMAT(10I5)
NODES1=NODES+1
NODES2=NODES+2
NSTATE=NODES2*4
N=NODES2*4+ICNTL(1)+ICNTL(2)*2
WRITE(NTYP,9402)
9402 FORMAT(/,5X,31H NODES,NSTATE,ICFLAG,IFXN,IFSEP,12H INSEP,ILENG)
WRITE(NTYP,9120) NODES,NSTATE,ICFLAG,IFXN,IFSEP
1,INSEP,ILENG
9120 FORMAT(10I10,/)

```

```

C C GEOMETRY CONSTANTS
  READ(NINP,9000)A,B,L,HYI
  HY=HYI
  WRITE(NTYP,9401)
  FORMAT(5X,11H A,B,L,HYI )
  WRITE(NTYP,9110)A,B,L,HY
C
  READ(NINP,9000)SSLENG,TPERIM
  WRITE(NTYP,9421)
  FORMAT(5X,* SSLENG,TPERIM *)
  WRITE(NTYP,9110)SSLENG,TPERIM
  READ(NINP,9000) ATC,ATRIM
  WRITE(NTYP,942R)
  WRITE(NTYP,9110) ATC,ATRIM
  FORMAT(5X,* ATC,ATRIM *)
  READ(NINP,9000)(ATCF(I),I=1,NODES1)
  WRITE(NTYP,9425)
  FORMAT(/,5X,* AREA TRUNK-GAP EACH ELEMENT *)
  CALL PUTVFC(ATCF,NODES1)
C
C ILENG , 1=READ DATA CARDS , 0= COMPUTE I./NODES
  IF(ILENG)7,7,8
  READ(NINP,9000)(RLENGO(I),I=1,NODES1)
  GO TO 9
  DO 6 I=1,NODES1
  RLENGO(I)=L/FLOAT(NODES1)
  CONTINUE
  CONTINUE
  WRITE(NTYP,940R)
  FORMAT(5X,11H SPRING L 0 )
  CALL PUTVFC(RLENGO,NODES1)
C
C SFT BOUNDARY NODES
  X(1)=0.0
  Y(1)=0.0
  Y(NODES2)=-R
  X(NODES2)=A

```

```

C OPTION TO USE TRUNK MODEL
  IF(ICFLAG)16,16,25
C
C INPUT NODE COORDINATES
16  READ(NINP,9000)(X(I),I=2,NODES1)
    READ(NINP,9000)(Y(I),I=2,NODES1)
    GOON F7R4T(RG10,4)
    GO TO 45
C
C OPTIONAL TRUNK SHAPE INITIAL CONDITIONS
C ONLY GOOD FOR EQUISPACED NODES
25  CALL TRUNK
    NX=IFIX(FLOAT(NODES)*(PHI2*R2/L))
    NZ=NODES-NX
    WRITE(NTYP,9120)NX,NZ
C
C COMPUTE RIGHT SECTOR POINTS
    XCNT=R2*SIN(PHI2)
    YCNT=HY-R1
    TX=2.0*ASIN(R1/ENG0(1))*0.5/R1
    ANG=ATAN2((X(NODES2)-XCNT),(Y(NODES2)-YCNT))
    DO 40 I=1,NZ
      ANG=ANG-IX
      J=NODES2-I
      X(J)=XCNT+R1*SIN(ANG)
      Y(J)=YCNT+R1*COS(ANG)
    40 CONTINUE
C
C COMPUTE LEFT SECTOR POINTS
    YCNT=YCNT+R1-R2
    ANG=1.570796-PHI2
    TX=2.0*ASIN(R1/ENG0(1))*0.5/R2
    DO 43 I=1,NX
      ANG=ANG+TX
      J=I+1
      X(J)=XCNT-R2*COS(ANG)
      Y(J)=YCNT+R2*SIN(ANG)

```

```

43      CONTINUE
C
C FORCE NODES ABOVE GROUND LEVEL
45      DO 46 I=1,NODES2
          Y(I)=AMIN1(YOWIN,Y(I))
46      CONTINUE
          WRITE(NTYP,9010)
9010     FORMAT(//,5X,15H NODE POSITIONS,/,2X,2H X,10X,2H Y,/)
C LOAD STATE VECTOR X,Y WITH INPUT DATA
      DO 20 I=1,NODES2
          J=(I-1)*4+2
          STATE(J)=X(I)
          STATE(J+2)=Y(I)
          WRITE(NTYP,9020) X(I),Y(I)
9020     FORMAT(2(2X,F8.4))
20      CONTINUE
          WRITE(NTYP,9414)
9414     FORMAT(/)
C
C COMPUTE THETA NODE ANGLES
      DO 21 I=1,NODES1
          THETA(I)=ATAN2((Y(I+1)-Y(I)),(X(I+1)-X(I)))
          IF(LAG=IFXM)
21      CONTINUE
          WRITE(NTYP,9409)
9409     FORMAT(5X,12H NODE ANGLES )
          CALL PUTVEC(THETA,NODES1)
C
C INPUT MASS IN POUNDS
          READ(NTIP,9000)(RMASS(I),I=1,NODES)
C CHANGE MASS TO SLUGS
      DO 5 I=1,NODES
          RMASS(I)=RMASS(I)/G
5      CONTINUE
          WRITE(NTYP,9405)
22      CONTINUE
          WRITE(NTYP,9405)
9405     FORMAT(5X,12H NODE "ASSES ")
          CALL PUTVEC(RMASS,NODES)
C
C INPUT MEMBRANE STIFFNESS L/H/FT

```

```

      READ(NINP,9000)(RKVEC(I),I=1,NODES)
C SAVE SPRING DATA
      DO 4 J=1,NODES1
        RKSAB(J)=RKVEC(J)
      CONTINUE
      WRITE(NTYP,9406)
9406  FORMAT(5X,17H SPRING CONSTANTS )
      CALL PUTVEC(RKVEC,NODES1)
C
C
C INPUT MEMBRANE BENDING STIFFNESS LR/RAD
      READ(NINP,9000)(RRVEC(I),I=1,NODES)
      WRITE(NTYP,9415)
9415  FORMAT(5X,18H BENDING STIFFNESS )
      CALL PUTVEC(RRVEC,NODES)
C
C INPUT MEMBRANE DAMPING LR-SEC/FT
      READ(NINP,9000)(DAMP(I),I=1,NODES)
      WRITE(NTYP,9407)
9407  FORMAT(5X,13H NODE DAMPING )
      CALL PUTVEC(DAMP,NODES)
C
C DAMPER RESET TIME AND FACTOR
      ITR=0
      READ(NINP,9000)TRFST,DAMPR
      WRITE(NTYP,9429)
9429  FORMAT(5X,* TRFST,DAMPR *)
      WRITE(NTYP,9110) TRFST,DAMPR
C
C INPUT EXTERNAL STIFFNESS COEFFICIENTS
      READ(NINP,9001) IEXT,PKFXTX,RKFXTY
9001  FORMAT(I2,2G10.5)
C
C SAVE I.C. FOR SPRING ATTACHMENT
      XEXT0=X(IEXT)
      YEXT0=Y(IFXT)
      IF(IEXT.NE.0) WRITE(NTYP,9002) IEXT,RKEXTX,RKFXTY
9002  FORMAT(5X,25H EXTERNAL SPRING AT NODE ,I2,5H RKX=,

```

```

1 F12.5,5H PKY=F12.5,/)
C INPUT FLUID CONTROL PARAMETERS
  READ (NINP,9000) AIFAN,TKVOL,VCH
  WRITE(NTYP,9426)
9426  FORMAT(5X,* AIFAN,TKVOL,VCH *)
  WRITE(NTYP,9110) AIFAN,TKVOL,VCH
C FAN POLYNOMIAL COEFFICIENTS
  READ(NINP,9001) C00,C01,C02,C03,C04
  WRITE(NTYP,9427)
9427  FORMAT(5X,*C00,C01,C02,C03,C04 *)
  WRITE(NTYP,9002) C00,C01,C02,C03,C04
9001  FORMAT(5F15.5)
9002  FORMAT(5X,6(G13.6,2X))
  READ(NINP,9000) CTC,CTRM,CGAP,TSEP
  WRITE(NTYP,9420)
9420  FORMAT(5X,* CTC,CTRM,CGAP,TSEP *)
  WRITE(NTYP,9110) CTC,CTRM,CGAP,TSEP
  READ(NINP,9000) YGRNDS,SRATIO,YDMIN
  WRITE(NTYP,9403)
9403  FORMAT(5X,* YGRNDS,SRATIO,YDMIN*)
  WRITE(NTYP,9110) YGRNDS,SRATIO,YDMIN
C
  READ(NINP,9000) PTK,PCH,OGAP
C IF DYNAMIC FAN SET INITIAL CONDITION
  IF(LDATS(1)) STATE(N)=PCH
  IF(LDATS(2)) STATE(N-1)=PTK
  IF(LDATS(2)) STATE(N-2)=OGAP
  IF(LDATS(2)) OFAN=OGAP
C SET EQUILIBRIUM FLOW INPUT AS FUNCTION OF PCH AND GAP
  QIN=0.0
  QEXIT=0.0
C IF DYNAMIC CUSHION PRESSURE SET I.C.
  IF(LDATS(1)) STATE(N)=PCH
  WRITE(NTYP,9404)
9404  FORMAT(5X,* PTK,PCH,OGAP *)
  WRITE(NTYP,9110) PTK,PCH,OGAP
9110  FORMAT(5X,10F12.5,/)

```



```

C GAP AT LOWEST POINT
  YGAPM=AGAP(INODE)
C IF IDATS(6) TRUE STATIC PRESSURE INPUT, SKIP DYNAMICS
  IF(IDATS(6)) GO TO 5081
C
C IF NO GAP JUMP TO SPECIAL NO FLOW CASE
  IF(YGAPM.EQ.0.0) GO TO 5054
  YC=YGAPM*10.0
C LOOK BACK TO FIND CUSHION SEPARATION POINT
  DO 5010 I=1,INODE
    IF(AGAP(I).LT.YC) GO TO 5020
C REMEMBER POINT
    ICS=I
    YCSPP=AGAP(I)
5010 CONTINUE
C *****
C SEPARATION POINT SELECTION ROUTINE
5020 ISEP=INODE
    YASEP=AGAP(ISEP)
C IFSEP .EQ. 1 FOR FIXED GAP CASE
C IFSEP .EQ. 2 FOR DIFFUSER CASE
C IFSEP .EQ. 3 FOR SET TO NODE = INSEP
C IF TRUNK TO GAP FLOW SET SEPARATION POINT
    GO TO (5021,5031,5035),IFSEP
C
C *****
C SET SEPARATION GAP BY INPUT VALUE
C PICK FIRST NODE WITH GAP EQUAL TO YSEPX
5021 YSEPX=YGAPM/SRATIO
    DO 5030 I=INODE,NODES2
      YF(AGAP(I),LT.YSEPX) GO TO 5030
    ISEP=I
C Y AT SEPARATION POINT = YSEPX VALUE = YGAPM/SRATIO
    YASEP=AMIN1(YSEPX,AGAP(I))
    GO TO 5051
5030 CONTINUE
    CALL FRPQR(1)

```

```

C *****
C SEPARATION POINT FROM DIFFUSE MODEL
C 6 DEGREE SLOPE OR MORE FOR SEPARATION
C LOOK FROM LOWEST NODE OUTWARD
5031 ISEP=INDEF
      VASEP=AGAP(ISEP)
      DO 5040 I=ISEP,NODES2
      IF(THETA(I).GT.TSEP) GO TO 5040
      GO TO 5041
5040 CONTINUE
      CALL FROR(7)
      VASEP=AGAP(I)
      GO TO 5050
C *****
C FORCE SEPARATION POINT TO BE AT NODE ISEP
5035 ISEP=INSEP
      VASEP=AGAP(ISEP)
      GO TO 5051
C *****
C INTERPOLATION SCHEME FOR SEPARATION POINT GAP
5050 IF(.NOT.LOATS(4)) GO TO 6200
      TEMP1=TSEP-THETA(ISEP)
      TEMP2=THETA(ISEP)-THETA(ISEP+1)
      TEMP1=(AGAP(ISEP+1)-AGAP(ISEP))*(TEMP1/TEMP2)
      VASEP=AGAP(ISEP)+TEMP1
      GO TO 5051
C *****
C TRUNK FLOW SET SEPARATION POINT
6200 IF(.NOT.LOATS(3)) GO TO 5051
C LOOK FOR SEPARATION POINT
      DO 6210 I=1,NODES1
      J=NODES2-I
      ISX=J
      IF(ATCF(J).GT.0.0) GO TO 6250

```

```

6210 CONTINUE
      CALL PRPDR(3)
6250 IF(IX.GE.ISEP) G) TO 5051
      ISEP=IX
      YASEP=AGAP(ISEP)
C
C *****
C PRESSURE = PCH FOR NODES 1 TO ICS
C PRESSURE = PAT FOR NODES ISEP+1 TO NMAX
C PRESSURE = F(VFL) FOR NODES ICS+1 TO ISEP
C MINIMUM GAP FLOW AREA AT NODE = INODE
C
5051 ICS1=ICS+1
      ISEP1=ISEP+1
      ICS1=MIN0(ICS1,NODES2)
      ISEP1=MIN0(ISEP1,NODES2)
C
C COMPUTE VELOCITY AND FLOW AT EXIT POINT
C USE SEPARATION GAP TO CONTROL TOTAL FLOW Q=CA**F(P)
      VEXIT=SQRT(2.0*PCH/RHO)
      QEXIT=VEXIT*YASEP
      G) TO 5055
C
C *****
C SPECIAL NO FLOW CASE
5054 CONTINUE
      QEXIT=0.0
      ICS=INODE
      ISEP1=INODE+1
C
C *****
C LOAD OUTER PRESSURE DATA ARRAY
C LOAD CUSHION PRESSURE FLOW INNER EDGE TO CUSHION SEPARATION
5055 DO 5060 I=1,ICS
      PRXT(I)=PCH
5060 CONTINUE
C
C LOAD ATMOSPHERE PRESSURE FROM SEPARATION POINT TO END

```

```

07 5070 I=ISEP1, NODES2
PEXT(1)=0.0
5070 CONTINUE
C IF NO FLOW PATH VECTOR OUT
IF(PEXT.FO.0.0) GO TO 50R1
C
C ****
C FROM VELOCITY AT EACH POINT DETERMINE PRESSURE
C USE BERNOULLI IDEAL FLOW RELATION EQUATION
C COMPUTE MINIMUM PRESSURE DUE TO FLOW
PCH=(PCH+2116.8)*0.528-2116.8
07 50R0 I=ICSI, ISEP
PEXT(1)=PCH*(1.0-(VASEP/AGAP(1))**2)
PEXT(1)=AMAX1(PEXT(1), PCRT)
50R0 CONTINUE
C
IF(.NOT.LOATS(3)) GO TO 50R1
C ****
C TRUNK TO GAP FLOW EFFECT COMPUTATION
C
C TRUNK TO CHANNEL COMPUTATION ROUTINE
C MASS FLOW AT CUSHION SEPARATION POINT
07 6010 I=1, NODES2
VEL(1)=0.0
W(1)=0.0
DA(1)=0.0
6010 CONTINUE
C INITIAL MASS FLOW
ATMP=0.0
ITCNT=0
07 6100 I=1, NODES1
ATMP=ATMP+ATCF(I)
6100 CONTINUE
C ESTIMATE TRUNK FLOW INTO CHANNEL
QEST=ATMP*CTC*SQRT(2.0/RHO*(PTK-PCH*0.5))*0.90
VSEP=QEST/VASEP
C PRESSURE DROP ESTIMATE DUE TO MOMENTUM
PLDST=VSEP*RHO/VASEP*QEST

```

```

C INITIAL MASS OUT OF CUSHION
  W(1)=RHO*CTC*YASFP*SORT(2.0/RHO*AMAX1(0.0,(PCU-PLNST)))
  RITER=0.10

C
C USING ESTIMATE OF CUSHION PRESSURE FLOW ITERATE PROFILE
6015 QINT=0.0
  QTOT=W(1)/RHO
  DO 6050 I=2,ISEP
    DZ=0.0
    DW(I)=0.0
    IF(ATCF(I).EQ.0.0) GO TO 6040
    C USE PRESSURE AT I-1 AND ESTIMATE OF P(I)
    PAVE=(PEXT(I)+PEXT(I-1))*0.5
    C COMPUTE FLOW INTO CONTROL VOLUME FROM TRUNK
    DZ=QFLOW(CTC,(PTK-PAVE))*ATCF(I)
    DW(I)=DZ*RHO
    C COMPUTE MASS FLOW AT EXIT OF CONTROL VOLUME
  6040 QTOT=QTOT+DZ
    VEL(I)=QTOT/AGAP(I)
    W(I)=DW(I)+W(I-1)
    C COMPUTE MOMENTUM CHANGE PRESSURE DROP
    QINT=1.0/AGAP(I)*VEL(I)*DW(I)+QINT
    C COMPUTE NEW ESTIMATE OF NODE PRESSURE
    PEXT(I)=PCU-0.5*RHO*VEL(I)**2-QINT
    PEXT(I)=AMAX1(PEXT(I),PCRT)
  6050 CONTINUE
C
C TEST IF EXIT POINT PRESSURE IS WITHIN ROUNDS SET
  IF((PEXT(ISEP).GT.-1.0).AND.(PEXT(ISEP).LT.1.0)) GO TO 6075
C ERROR ON EXIT PRESSURE , ITERATE
  ITCNT=ITCNT+1
C ALLOW ONLY TEN ITERATIONS ON PRESSURE
  IF(ITCNT.GE.11) GO TO 6075
  IF(ITCNT.GE.5) RITER=RITER*0.90
C CORRECT INITIAL FLOW FROM CUSHION TO ZERO P EXIT
  W(1)=W(1)+CTC*RITER*RHO*QFLOW(YASFP,PEXT(ITER))
  GO TO 6015
6075 CONTINUE

```

```

C *****
C IF NOOF CONTACT SET TO TRUNK PRESSURE
5081 DO 5082 I=2,NODES1
      IF(AGAP(I),LE,0.0) PEXT(I)=PTK
5082 CONTINUE
5090 CONTINUE
C NOW HAVE PRESSURE PROFILE FOR TRUNK
DO 75 I=1,NODES1
  PRESUR(I)=PTK-PEXT(I)
75 CONTINUE
C *****
C DATA SWITCH 2 ON FOR TRUNK-FAN DYNAMICS
  IF(.NOT.LOATS(2)) GO TO 1480
  QFAN=STATE(N-2)
  OFAN=AMIN1(OFAN,3000.0)
  PTK=STATE(N-1)
  PTK=AMAX1(PTK,0.0)
C *****
C COMPUTE DYNAMIC CUSHION PRESSURE RELATIONS
C DATA SWITCH 1 MUST BE .TRUE. FOR DYNAMIC CUSHION PRESSURE
1480 IF(.NOT.LOATS(1)) GO TO 1490
C COMPUTE TOTAL FLOW FROM CUSHION
  AGAPX=YASEP*2.0*SSLENG
  PCH=STATE(N)
  PCH=AMAX1(PCH,0.0)
  QTA=0.0
  QCA=AGAPX*QFLOW(CGAP,PCH)
  QTC=ATC*QFLOW(CTC,(PTK-PCH))
  QTRIM=ATRIM*QFLOW(CTRIM,(PTK-PCH))
C TRUNK TO CHANNEL FLOW
  IF(.NOT.LOATS(3)) GO TO 1490
  QTC=0.0
  QTA=(QTC+QTRIM)/PHN*2.0*SSLENG
  QCA=QTA
C

```

```

C *****
C SAVE AUXILIARY VARIABLES IF ANY
C SAVE TOTAL LENGTH OF TRUNK SEGMENTS
1490 XL=0.0
      DO 1500 I=1,NODFS1
        XL=XL+RLENG(I)
1500 CONTINUE
      IF(NEWDT.EQ.-1) XL=L
      IF(.NOT. LDATS(1)) GO TO 1510
C IF DYNAMIC CUSHION PRESSURE SAVE FLOW DATA
      STATE(N+1)=XL
      STATE(N+2)=QTC
      STATE(N+3)=QTRIM
      STATE(N+4)=OCA
      PCTOT=PCTOT+PCH
      RNUM=RNUM+1.0
      PCAVE=PCTOT/RNUM
      STATE(N+5)=PCAVE
      STATE(N+6)=QTOT
      STATE(N+7)=QTA
      ISEP1=ISEP+1
C TEST PRINT OUT FLAG
1510 IF(IFXN.NE.IFLAG) GO TO 1650
      WRITE(NTYP,9056)TIME
      WRITE(NTYP,9412)
      FORMAT(5X,14H NODE Y VALUES )
      CALL PUTVEC(Y,NODES2)
      WRITE(NTYP,9413)
19413 FORMAT(5X,* ICS,ISEP,INODE,YASEP,YGAPM,AGAP(ISEP),VEXIT*,
1 * ,QEXIT,PCH,PTK,QFAN*)
      WRITE(NTYP,9055)ICS,ISEP,INODE,YASEP,YGAPM,AGAP(ISEP),
1 VEXIT,QEXIT,PCH,PTK,QFAN
19055 FORMAT(5X,3I5,8F12.5)
      NP1=NP+1
      NP7=NP+7
      IF(LDATS(1).OR.LDATS(2)) WRITE(NTYP,9050) (STATE(JK),JK=NP1,NP7)
19050 FORMAT(5X,* XL,QTC,QTRIM,OCA,PCAVE,QTOT,QTA*,/,8F15.5)

```

```

C
      WRITE(NTYP,9410)
      CALL PUTVEC(PEXT,NODES2)
9056  FORMAT(SX,*TIME*,F12.5)
      IFLAG=0
1650  IFLAG=IFLAG+1
C *****
C PLOTTING ROUTINE
      IF(NPLTM.GE.1) CALL PSTORE(NODES)
C
C RESET DAMPING RATION IF AT EQUILIBRIUM TIME
C FIRST CALL TEST TIME FOR RESET/STEP
      IF(TIME.LT.TREST) GO TO 1700
      IF(ITP)1710,1710,1700
1710  DO 1720 I=1,NODES
      DAMP(I)=DAMP(I)*DAMPR
1720  CONTINUE
      ITR=1
1700  CONTINUE
C
C COMPUTE SPRING CONSTANTS
      DO 1800 I=1,NODES1
      RKVEC(I)=RKSAV(I)*(RLENG(I)/RLENGO(I))
1800  CONTINUE
C *****
C DIFFERENTIAL EQUATIONS
C CALLED FOUR TIMES PER STEP
C *****
3     CONTINUE
C
C LOAD X,Y VECTORS FROM STATE ARRAY
      DO 150 I=2,NODES1
      J=(I-1)*4+2
      X(I)=STATE(J)

```

```

Y(I)=STATE(J+2)
150 CONTINUE
C SPECIAL NO MOTION EXECUTION
D7 160 I=1,N
160 DFRY(I)=0.0
IF(I.DATS(5)) GO TO 1013
C
C *****
D7 50 I=1,NODES1
C COMPUTE SPRING LENGTHS
RLENG(I)=SQRT((X(I+1)-X(I))**2+(Y(I+1)-Y(I))**2)
C
C COMPUTE PHI ANGLES
TX=ATAN2((Y(I)-Y(I+1)),(X(I)-X(I+1)))
PHI(I)=TX
COSPHI(I)=COS(TX)
SINPHI(I)=SIN(TX)
C
C COMPUTE THETA ANGLES
TX=ATAN2((Y(I+1)-Y(I)),(X(I+1)-X(I)))
THETA(I)=TX
SINTHF(I)=SIN(TX)
COSTHF(I)=COS(TX)
50 CONTINUE
C
C IF FIRST CALL COMPUTE BENDING STIFFNESS INITIAL CONDITION
IF(NEWDT)51,53,53
51 D7 52 I=1,NODES
SIF(I)=THETA(I+1)-PHI(I)
52 CONTINUE
C
C *****
C COMPUTE FORCES
53 D7 60 I=1,NODES
C FIND SPRING FORCES AT NODE
C RK1,RK2 ARE SPRING FORCE MAGNITUDES FOR 2 ATTACHED SPRINGS, +=TENSILE
C FORCE = SPRING CONSTANT * DELTA LENGTH/LENGTH
RK1=(PI/RLENG(I)-RLENG(I))*RKVFC(I)

```

```

RK1=RK1/RIENG(I)
RK2=(RIENG(I+1)-RIENG(I))*RKVEC(I+1)
RK2=RK2/RIENG(I)
FRCXK(I)=RK1*COSPHI(I)+RK2*COSTHE(I+1)
FRCYK(I)=RK1*SINPHI(I)+RK2*SINTHE(I+1)
C
C FIND DAMPER FORCES AT NODE
J=I+4+1
C FORCE = 2.0 * ZETA * OMEGA N * VELOCITY
XZETA=2.0*SORT(RMASS(I)*RKVEC(I)/RIENG(I))
FRCXD(I)=-STATE(J)*DAMP(I)*XZETA
FRCYD(I)=-STATE(J+2)*DAMP(I)*XZETA
60 CONTINUE
C
C COMPUTE PRESSURE FORCES
C PRESSURE X,Y COMPONENTS, FORCE = P*A ACTING ON LENGTH/2 ON SIDES OF NODE
DO 125 I=1,NODES
FRCXP(I)=-PRESUR(I)*0.5*(RIENG(I)*SINTHE(I)+RIENG(I+1)*SINTHE
1(I+1))
FRCYP(I)=PRESUR(I)*0.5*(RIENG(I)*COSTHE(I)+RIENG(I+1)*COSTHE
1(I+1))
125 CONTINUE
C
C ZERO BENDING FORCES
DO 129 I=1,NODES
FRCXB(I)=0.0
FRCYB(I)=0.0
129 CONTINUE
C
C COMPUTE BENDING FORCES
DO 130 I=1,NODES
IF(RBVEC(I).EQ.0.0) GO TO 130
C COMPUTE ANGULAR DISPLACEMENT FROM EQUILIBRIUM
TX=SI(I)-(THETA(I+1)-PHI(I))
C IF SIE(T) ANGLE INCREASE TORQUE IS NEGATIVE
TX=TX*RBVEC(I)
C COMPUTE LENGTH NORMALIZATION FACTOR
RX1=2.0/(RIENG(I)*(RIENG(I)+RIENG(I+1)))

```

```

      RX2=2.0/(RLENG(I+1))*(RLENG(I)+RLENG(I+1))
C IF FIRST NODE SKIP FORCE AT ATTACHMENT POINT
      IF(I.EQ.1) GO TO 132
C COMPUTE REMAINING EQUIVALENT FORCES IN X,Y TERMS
      FORCXR(I-1)=FORCXR(I-1)+RX1*(-TX)*SINTHE(I)
      FORCYR(I-1)=FORCYR(I-1)+RX1*TX*COSTHE(I)
132   FORCXR(I)=FORCXR(I)+RX1*TX*SINTHE(I)+RX2*TX*SINTHE(I+1)
      FORCYR(I)=FORCYR(I)+RX1*(-TX)*COSTHE(I)+RX2*(-TX)*COSTHE(I+1)
C IF LAST NODE SKIP FORCE AT ATTACHMENT POINT
      IF(I.EQ.NNODES) GO TO 130
      FORCXR(I+1)=FORCXR(I+1)+RX2*(-TX)*SINTHE(I+1)
      FORCYR(I+1)=FORCYR(I+1)+RX2*TX*COSTHE(I+1)
130   CONTINUE
C
C COMPUTE FORCES DUE TO EXTERNAL STIFFNESS TERMS
      IF(TEXT.EQ.0) GO TO 140
      FORCXG(TEXT)=-RKFXTX*(X(IFXT)-XFXT0)
      FORCYG(IFXT)=-RKFXTY*(Y(IFXT)-YFXT0)
C
C *****
C
C FORM DIFFERENTIAL VECTOR, 4 ELEMENTS PER NODE
C NODES 1 AND N+2 ARE FIXED
C
140   DO 1000 I=1,NNODES
1001   CONTINUE
      J=I+4+1
C X VELOCITY
      DERY(J)=(FORCXD(I)+FORCXK(I)+FORCXP(I)+FORCXA(I)+FORCXG(I))/RMASS(
      1I)
C X POSITION
      DERY(J+1)=STATE(J)
C Y VELOCITY
      DERY(J+2)=(FORCYD(I)+FORCYK(I)+FORCYP(I)+FORCYA(I)+FORCYG(I))/
      1RMASS(I)
C Y POSITION
      DERY(J+3)=STATE(J+2)
C

```

```

1000 CONTINUE
C
C CUSHION PRESSURE STATE VARIABLE
1013 IF(.NOT.LOATS(1)) GO TO 1050
PCH=STATE(N)
DERV(N)=CKK*(PCH+PAT)/VCH*(OTC+OTPI*-004)
C
IF(.NOT.LOATS(2)) GO TO 1050
PTR=STATE(N-1)
DFAN=STATE(N-2)
C TRUNK PRESSURE EQUATION
DERV(N-1)=CKK/TKVOL*(PTR+PAT)*(DFAN-OTC-OTA-JTRIM)
C
C FAN EQUATION
DERV(N-2)=(PEANX(DFAN*0.5)-PTR)/AIFAN
C
C ****
C CONSTRAINTS ON MEMBRANE
C CHECK FOR GROUND CONTACT OF TRUNK
1050 DO 2000 I=1,NODES
J=I+2
C TEST FOR CONTACT
IF(STATE(J+2).LT.YDMIN) GO TO 2000
C FORCE MAXIMUM ON Y DISPLACEMENT
STATE(J+2)=AMIN1(STATE(J+2),YDMIN)
C GROUND CONTACT ENERGY ABSORPTION
C FORCE ONLY REBOUND
DERV(J+1)=AMIN1(0.0,DERV(J+1))
STATE(J+1)=AMIN1(0.0,STATE(J+1))
2000 CONTINUE
C
RETURN
END

```

```

SURROUTINE TRUNK
C TRUNK SHAPE ITERATION
REAL L,L1,L2
COMMON/GEOMET/A,R,HY1,L,HY,PHI1,PHI2,R1,R2,L1,L2,ISHAPE
DATA RTOL,PI2/0.01,6.28318/

C
1 IF(HY)11,11,1
  R2=SQRT(A*A*0.25+HY*HY)
  RHY=R+HY
  NMAX=100
  DO 10 I=1,NMAX
    PHI2=ARS(ACOS(AMAX1(-1.0,AMIN1(1.0,((R2-HY)/R2))))))
    SINPH2=SIN(PHI2)
    R1=((A-R2*SINPH2)**2+(BHY*RHY))/(RHY+BHY)
    PHI1=ARS(ACOS(AMAX1(-1.0,AMIN1(1.0,((R1-HY-B)/R1))))))
    XS=A-R2*SINPH2
    IF(XS)5.5,6
    PHI1=PI2-PHI1
    L2=L-PHI1*R1
    IF(ARS(PHI2).LT.1.0E-2) PHI2=1.0F-2
    R2S=L2/PHI2
    IF(ARS(R2-R2S).LE.RTOL) GO TO 50
    R2=(R2+R2S)*0.5
  10 CONTINUE
C
C ERROR RETURN
11 ISHAPE=0
C
C GOOD RETURN
50 L1=L-L2
  ISHAPE=1
  RETURN
  END

```

```

SUBROUTINE PHIVC(A,N)
C SUBROUTINE FOR VECTOR OUTPUT
  DIMENSION A(N)
  COMMON/IOLIST/NTYP,NTAP,IPTAPE
  WRITE(NTYP,9010)(A(I),I=1,N)
9010  FORMAT(5X,10G12.5)
  WRITE(NTYP,9000)
9000  FORMAT(/)
  RETURN
END

```

```

      SUBROUTINE ERROR (Y)
      C SUBROUTINE FOR ERROR HANDLING IN EOSIM
      COMMON/IOLIST/NTYP,NINP,IPTAPE
      C ERROR HANDLING ROUTINE FOR EOSIM
      WRITE(NTYP,9000) I
9000  FORMAT(/,5X,20H ***** ERROR CODE = ,I5,/)
      CALL EXIT
      RETURN
      END

```

```

SUBROUTINE PLOTTER
C PLOTTING CONTROL SUBROUTINE
  DIMENSION IDUM(5)
  COMMON/PLT/NPLT(10),TPLSRT,TPLSTP,NPLTM,DTPLOT,XPLOTX(5,200)
  1  ,NDIM,NVPLUT,XMIN,XMAX,NPMAX,NSTORE
  COMMON/IOLIST/NTYP,NINP,IPTAPE

C MARK END OF PLOT DATA FILE
  ENDFILE IPTAPE
  DO 1000 IJK=1,NPLTM
    WRITE(NTYP,9100)
  9100  FORMAT(14I)
C READ PLOTTER CONTROL CARD
  READ(NINP,9010) NVPLUT,(NPLT(I),I=1,5)
  READ(NINP,9060) TPLSRT,TPLSTP,DTPLOT,XMIN,XMAX
  9060  FORMAT(6F12.5)
  DO 1 I=1,5
    IDUM(I)=NPLT(I)
  1  CONTINUE
  WRITE(NTYP,1020)(IDUM(I),I=1,5)
  1020  FORMAT(5X,21H PLOTS 1-5 = STATES= ,514,/)
  9010  FORMAT(11,4X,512)
  IF(NVPLUT.EQ.0) GO TO 1000
  IF(NVPLUT.GT.5) NVPLUT=5
C CALL PACKER TO READ REQUESTEE VARIABLES INTO XPLOTX
  CALL PACKER(I)
C
  IF((XMIN.NE.0.0).AND.(XMAX.NE.0.0)) GO TO 100
  XMAX=XPLOTX(1,1)
  XMIN=XPLOTX(1,1)
  DO 10 K=1,NVPLUT
    DO 10 J=1,I
      IF((K.EQ.1).AND.(J.EQ.1)) GO TO 10
C FIND MINIMUM AND MAXIMUM OF DATA
      XMAX=AMAX1(XMAX,XPLOTX(K,J))
      XMIN=AMIN1(XMIN,XPLOTX(K,J))
  10  IF(XMAX.EQ.XMIN) XMAX=XMAX+1.0
      XMAX0=XMAX

```

```

XMAX=XMAX+0.01*(XMAX-XMIN)
XMIN=XMIN-0.01*(XMAX-XMIN)
C CALL PRNPT TO PLOT DATA ON PRINTER.
100 CALL PRNPT(I)
XD=(XMAX-XMIN)/100.0
1000 CONTINUE
2000 CONTINUE
      RETURN
      END

```

```

SUBROUTINE PACKER(1)
C SUBROUTINE TO PICK DATA POINTS FOR PLOT FROM TAPE
C DTPLT MUST BE INTEGRAL, WITH MAX DT
  DIMENSION X(200)
  COMMON/PLT/NPLT(10),TPLUS1,TPLSTP,NPLT*,DTPLT,DTPLT*(5,200)
  1  ,NDIM,NVPLT,XMIN,XMAX,NPMAX,NSSTORE
  COMMON/IOLIST/NIYP,NIIP,IPTAPE
C REWIND TAPE 4 TO START OF SIMULATION
  REWIND IPTAPE
  TIMOLD=0.0
  ISTOP=0
  I=0
  NPMAX=200
C READ TIME STEP VARIABLES AND TIME
  10  READ(IPTAPE)(X(IJK),IJK=1,NSSTORE),XTIME
C IF END OF FILE STOP READING
  20  CONTINUE
C IF TIME .LT. PLOT TIME READ NEXT
      IF((XTIME.LT.TPLSTP)) GO TO 10
C IF TIME .GT. PLOT TIME RETURN
  60  IF(XTIME.GT.TPLUS1) RETURN
C IF TIME SINCE LAST POINT EQUAL DTPLT SAVE
      IF(XTIME.GE.(TIMOLD+DTPLT)) GO TO 30
      GO TO 10
  100 ISTOP=1
  30  TIMOLD=XTIME
      I=I+1
  50  J=1,NVPLT
C LOAD APPROPRIATE CURVE WITH DATA
C NPLT(J) CONTAINS STATE NUMBER TO BE PLOTTED
      NJ=NPLT(J)
  50  XPLT(X(J,I))=X(NJ)
C IF ARRAY FULL RETURN
      IF((I.LT.NPMAX).AND.(ISTOP.EQ.0)) GO TO 10
      RETURN
  END

```

```

SUBROUTINE PRNTPT(NPNTS)
C PRINTER PLOT PROGRAM
C
INTEGER LINE,XFIG,HLANK,DOT,T,CHAR
INTEGER SOFF
DIMENSION LINE(120),XFIG(5)
C
DIMENSION XPLAB(11)
COMMON/PLT/NPLT(10),IPLST,IPLSTP,NPLTM,DTPLOT,XPLOTX(5,200)
1 .NDIM,NCURVE,XMIN,XMAX,NP*MAX,NSTORE
COMMON/IOLIST/NTYP,NIMP,IPTAPE
C
C SET PLOT CURVE SYMBOLS
DATA HLANK,DOT/4H ,4H. /
DATA XFIG/4H* ,4HX ,4H+ ,4HD ,4HS /
DATA T/4HT /
DATA SOFF/4H- /
C
C ROUTINE WILL PLOT 1 TO 5 CURVES PER CALL
DX=(XMAX-XMIN)/100.0
C QUANTIZATION LEVEL OF DEPENDENT AXIS IS RANGE/100
IF(NCURVE.GT.5) NCURVE=5
IF(NPNTS.GT.NP*MAX) NPNTS=NP*MAX
IF(NPNTS.LE.0) GO TO 300
IF(NCURVE.LE.0) GO TO 300
WRITE(NTYP,1005) XMIN,DX,XMAX
FORMAT(5X,9H MINIMUM= ,F10.4,22X,8H DELTA= ,F10.4,22X,
1 9H MAXIMUM= ,F10.4,5X,5H TIME,/)
WRITE(NTYP,1010)
1010 FORMAT(5X,36H CURVE MARKERS=) 1=* 2=X 3=+ 4=O 5=S ,
DO 5 I=1,11
XPLAB(I)=XMIN+DX*FLOAT(I-1)*10.0
WRITE(NTYP,9000)(XPLAB(I),I=1,11)
9000 FORMAT(1X,11G10.3)
C LOAD LINE WITH HLANKS AND SET MARKERS
DO 10 I=2,101
10 LINE(I)=DOT

```

```

LINE(1)=F
LINE(102)=F
C PRINT FIRST MARKER LINE
*RITE(4TYP,1001)(LINE(1),I=1,102)
1001  FORMAT(5X,102A1,F10.6)
      XTIME=TPUSRT
      DTIME=(TPUSRT-IPUSRT)/FLUAT(NPNTS)*10.0
      IF=9
C PLOT ARRAY OF CURVES
DO 100 IJK=1,NPNTS
DO 20 I=2,119
20  LINE(I)=BLANK
C CLEAR LINE TO BLANKS
C SET BORDER FOR PLOT
LINE(1)=DOT
LINE(102)=DOT
C FOR EACH CURVE SET UP LINE
DO 200 J=1,NCURVE
VAL=XPL0TX(J,IJK)
CHAR=XFIG(J)
KDEX=IFIX((VAL-XMIN+0.5*DX)/DX)+2
IF(KDEX.GE.2) GO TO 25
KDEX=2
CHAR=SOFF
25  IF(KDEX.LE.101)GO TO 26
KDEX=101
CHAR=SOFF
C DETERMINE POINTS POSITION IN LINE
26  LINE(KDEX)=CHAR
200  CONTINUE
C PRINT LINE OF PLOT
IF=IT+1
IF(IT-10)30,40,30
40  IT=0
C PUT IN ROWS OF DOTS EVERY TEN STEPS
DO 45 I=12,92,10
45  IF(LINE(I).EQ.BLANK) LINE(I)=DOT
      CONTINUE

```

```

WRITE(NTYP,1001)(LINE(K),K=1,102),XTIME
XTIME=XTIME+DTIME
GO TO 100
CONTINUE
WRITE(NTYP,1001)(LINE(K),K=1,102)
CONTINUE
DO 33 IJK=2,101
LINE(IJK)=DOT
WRITE(NTYP,1001)(LINE(K),K=1,102),TPLSTP
RETURN
WRITE(NTYP,1002)
FORMAT(5X,45H ERROR CONDITION ON PLOT CONTROL NUMBERS )
RETURN
END
30
100
33
300
1002

```

```

SUBROUTINE PSTORE(NODES)
COMMON/IDLIST/NTYP,NIMP,IPTAPE
COMMON/PLT/NPLT(10),TPLSRT,TPLSTP,NPLTM,DTPLT,XPLOIX(5,200)
1  ,NDIM,NVPLT,XMIG,XMAX,NPMAX,NSTORE
COMMON TIME,DTIME,STATE(100),DERV(100),STIME,ETIME,NEWT,IFWRT,N,
1  IPR,ICD,ICN,TNEXT,PNEXT,THACK
C
C PLOT FILE STORAGE ROUTINE WRITE IPTAPE OF DATA FOR PLOT
WRITE(IPTAPE)(STATE(I),I=1,NSTORE),TIME
C
RETURN
END

```

D.2 Eigenvalue Analysis Program

The following programs and subroutines are included.

Programs - FMAEVEC

Subroutines - TRUNK

MOVE

CMINV (complex version of IBM-SSP MINV)

ELEMK

CLEAR

PUTMAT

PUTEIG

EIGPAC

EVECTR

VECPAC


```

DIMENSION SINALP(40),CONSALP(40)
DIMENSION A(40,40)
DIMENSION AS(40,40),AT(40,40)
DIMENSION G(40,40)
DIMENSION WORKM(40,40)
DIMENSION XW(40,40)
DIMENSION XZ(40,40),XT(40,40)
DIMENSION GS(40,40)
DIMENSION IPRNT(10)

```

```

COMMON/GEOMET/AX,BX,RY,RL,U1,U2,R1,R2,PHI1,PHI2

```

```

EQUIVALENCE (RIENGO(1),RIENGO(1))
EQUIVALENCE (G(1,1),AS(1,1))
EQUIVALENCE (G(1,1),GS(1,1))
EQUIVALENCE (WORKM(1,1),AT(1,1))
EQUIVALENCE (AT(1,1),XT(1,1))
EQUIVALENCE (XZ(1,1),A(1,1))
EQUIVALENCE (XW(1,1),A(1,1))

```

```

C STORAGE EQUIVALENCY MAPPING

```

```

C AT = WORKM = XT
C A = XW = XZ
C AS = G = GS

```

```

C *****

```

```

C INPUT INITIAL DATA

```

```

1 CONTINUE
  NTYP=6
  NINP=5

```

```

C NMAT IS STORAGE ARRAY LIMIT

```

```

  NMAT=40
  NMATV=21
  NUMAX=NMAT/2
  NDMAX=NMAT/4
  PI=3.141592
  PI02=1.570796

```

```

GD=32.174
WRITE(NTYP,9003)
9003 FORMAT(1H1)
9002 FORMAT(///)
9050 FORMAT(5X,10G12.5)
9001 FORMAT(/)
C
C CLEAR MATRICES
CALL CLEAR(MASS,NMAX,1)
CALL CLEAR(RKVEC,NMAX,1)
CALL CLEAR(DAMP,NMAX,1)
CALL CLEAR(RUENG,NMAX,1)
C
C INPUT CONTROL INFORMATION
READ(NINP,9015)(ICNTL(I),I=1,10),(IPRNT(J),J=1,10)
9015 FORMAT(20I1)
C
C ICNTL N = 1,EIGENVALUES 2,EIGENVECTORS
C ICNTL 1 = COMPUTE 2 DIMENSIONAL EIGENVALUES
C ICNTL 2 = COMPUTE 2 DIMENSIONAL DAMPED EIGENVALUES
C ICNTL 3 = COMPUTE TRANSVERSE STRING UNDAMPED EIGENVALUES
C ICNTL 4 = COMPUTE TRANSVERSE DAMPED EIGENVALUES
C ICNTL 5 = COMPUTE LONGITUDINAL BAR UNDAMPED EIGENVALUES
C ICNTL 6 = COMPUTE LONGITUDINAL BAR DAMPED EIGENVALUES
C
C IPRNT = 1 TO PRINT MATRIX AS IN ICNTL
C
C INPUT NUMBER OF MASS NODES FOR ANALYSIS
READ(NINP,9005) NODES
9005 FORMAT(I7)
NODES1=NODFS+1
NODES2=NODFS+2
NMAX=NODFS*2
NMAX2=NMAX*2
WRITE(NTYP,9051) NODES
9051 FORMAT(5X,8H NODES =,I3,/)
READ (NINP,9000)I,S,RI,AX,BX,TENSN
1,HY

```

```

WRITE(NTYP,9056)
9056 FORMAT(5X,12H LENGTH 0 ,12H LENGTH 1C ,12H A POINT ,12H R PD
11NT ,12H TENSION ,3H HY )
WRITE(NTYP,9050)(S,RI,AX,RX,TENSN
1,HY
WRITE(NTYP,9001)
C
C INPUT MASS IN POUNDS
READ(NINP,9000)(MASS(I),I=1,NODES)
C INPUT MEMBRANE STIFFNESS LR/FT
READ(NINP,9000)(RKVEC(I),I=1,NODES1)
C INPUT MEMBRANE DAMPING LR-SEC/FT
READ(NINP,9000)(DAMP(I),I=1,NODES)
C
C CHANGE MASS TO SLUGS
DO 5 I=1,NODES
MASS(I)=MASS(I)/GO
5 CONTINUE
C
C SET EQU1 SPACED ELEMENT LENGTHS
DO 6 I=1,NODES1
RLENGO(I)=LS/FLOAT(NODES1)
6 CONTINUE
C RLENGO SETTING OPTION
READ(NTNP,9015) IXF
C INPUT RLENGO IF IXF = 1
IF(IXF.EQ.1) READ(NINP,9000)(RLENGO(I),I=1,NODES1)
C
C NORMALIZE SPRING CONSTANTS
DO 7 I=1,NODES1
RKVEC(I)=RKVEC(I)/RLENGO(I)
7 CONTINUE
C
C TRANSFORM DAMPING RATION INTO DAMPER VALUE
DO 8 I=1,NODES
DAMP(I)=2.0*SORT(MASS(I)*RKVEC(I))*DAMP(I)
8 CONTINUE
C

```

```

C PRINT INITIAL VECTORS
  WRITE(NTYP,9053)
9053 FORMAT(5X,11H MASS NODES)
  CALL PUTMAT(MASS,NODES,NMATV,1)
  WRITE(NTYP,9054)
9054 FORMAT(5X,20H SPRING COEFFICIENTS )
  CALL PUTMAT(PKVEC,NODES1,NMATV,1)
  WRITE(NTYP,9055)
9055 FORMAT(5X,13H NODE DAMPING)
  CALL PUTMAT(DAMP,NODES,NMATV,1)
  WRITE(NTYP,9057)
9057 FORMAT(5X,9H LENGTH 0 )
  CALL PUTMAT(PLFNGD,NODES1,NMATV,1)
C *****
C
C SET BOUNDARY NODES
  X(1)=0.0
  Y(1)=0.0
  Y(NODES2)=-BX
  X(NODES2)=AX
C
C X,Y POINT OPTION
  READ(NINP,9015)IXY
  IF(IXY.EQ.1) GO TO 17
C
C INPUT NODE COORDINATES
C FOR INITIAL STIFFNESS CALCULATION
16 READ(NINP,9000)(X(I),I=2,NODES1)
  READ(NINP,9000)(Y(I),I=2,NODES1)
9000 FORMAT(8F10.5)
  GO TO 45
C
C COMPUTE TRUNK SHAPE
17 CALL TRUNK(ISHAPE)
  NX=IFIX(PLMAT(NODES)*(PHI2*R2/RL))
  NZ=NODES-NX
C

```

```

C COMPUTE RIGHT SECTOR POINTS
  XCNTR=R2*SIN(PHI2)
  YCNTR=HY-R1
  TX=2.0*ASIN(RLFNGD(1)*0.5/R1)
  ANG=ATAN2((X(NODES2)-XCNTR),(Y(NODES2)-YCNTR))
  DO 18 I=1,NZ
    ANG=ANG-TX
  J=NODES2-1
  X(J)=XCNTR+R1*SIN(ANG)
  Y(J)=YCNTR+R1*COS(ANG)
18 CONTINUE
C
C COMPUTE LEFT SECTOR POINTS
  YCNTR=YCNTR+R1-R2
  ANG=1.570796-PI2
  TX=2.0*ASIN(RLFNGD(1)*0.5/R2)
  DO 19 I=1,NX
    ANG=ANG+TX
  J=I+1
  X(J)=XCNTR-R2*COS(ANG)
  Y(J)=YCNTR+R2*SIN(ANG)
19 CONTINUE
C
C PRINT X,Y POSITIONS
45 WRITE(NTYP,9010)
9010 FORMAT(/,5X,15H NODE POSITIONS,/,2X,2H X,10X,2H Y ,/)
  DO 20 I=1,NODES2
    WRITE(NTYP,9020) X(I),Y(I)
9020 FORMAT(2(2X,F8.4))
20 CONTINUE
  WRITE(NTYP,9001)
C
  DO 50 I=1,NODES1
C COMPUTE SPRING LENGTHS
  RLFNG(I)=SORT((X(I+1)-X(I))**2+(Y(I+1)-Y(I))**2)
C COMPUTE THETA ANGLES
  TX=ATAN2((Y(I+1)-Y(I)), (X(I+1)-X(I)))
  THETA(I)=TX

```

```

50 COSINE(I)=COS(TX)
   SINTE(I)=SIN(TX)
   CONTINUE
C
C COMPUTE LOCAL SLOPE ANGLE, ALPHA
   DO 55 I=1,NODES
      TX=ATAN2((Y(I+2)-Y(I)),(Y(I+2)-X(I)))
      SINAP(I)=SIN(TX)
      COSAP(I)=COS(TX)
55 CONTINUE
C
C *****
C
C CONTINUOUS STRING FREQUENCIES
C NOTE * ONLY GOOD FOR EQUAL SPACED NODES, PLEASE USE STRING LATERAL, FREQUE
C
      RMASST=0.0
      DO 70 I=1,NODES
         RMASST=RMASST+MASS(I)
70 CONTINUE
      WRITE(6,9052)
9052 FORMAT(/,5X,30H CONTINUOUS STRING FREQUENCIES ,3H HZ,/)
      DO 80 I=1,NODES
         W=FLQAT(I)*SORT(TENSN*RL/RMASST)/(RL*2.0)
         WRITE(6,9050)W
80 CONTINUE
C
C *****
C
C 2 DIMENSIONAL UNDAMPED MATRIX MODEL
C
      IF(ICNTL(1).LE.0) GO TO 202
C
      IF(NODES.GT.NUMAX) GO TO 615
      CALL CLEAR(WORKM,NMAT,NMAT)
      CALL CLEAR(G,NMAT,NMAT)
      CALL CLEAR(XW,NMAT,NMAT)
      K=0

```

```

C
C
C      NJ 2000 IJK=1,NODES1
C      ADD DIRECT STIFFNESS SUBMATRIX INTO GLOBAL MATRIX G
C
C      CALL ELEMK(ESTIF(1,1),THETA(IJK),-RKVEC(IJK))
C
C      DO 100 I=1,4
C      DO 100 J=1,4
C      IK=I+K-2
C      JF(IK,LT,1).OR.(IK,GT,NMAX)) GO TO 100
C      JK=J+K-2
C      JF(JK,LT,1).OR.(JK,GT,NMAX)) GO TO 100
C      WORKM(IK,JK)=WORKM(IK,JK)+ESTIF(I,J)
C      CONTINUE
C      K=K+2
C      100 CONTINUE
C
C      2000 CONTINUE
C
C      ADD TENSION SPRING EFFECT MODEL INTO STIFFNESS MATRIX
C
C      DO 450 I=1,NODES
C      J=(I-1)*2+1
C      TX=TENSN*2.0/(RLENG(I)+RLENG(I+1))*SINALP(I)
C      TY=TENSN*2.0/(RLENG(I)+RLENG(I+1))*CSALP(I)
C
C      IF(I.EQ.1) GO TO 440
C      G(J,J-2)=G(J,J-2)+TX*SINALP(I-1)
C      G(J,J-1)=G(J,J-1)+TX*CSALP(I-1)
C      G(J+1,J-2)=G(J+1,J-2)+TX*SINALP(I-1)
C      G(J+1,J-1)=G(J+1,J-1)+TX*CSALP(I-1)
C
C      G(J,J)=G(J,J)+TX*SINALP(I)*(-2.0)
C      G(J,J+1)=G(J,J+1)+TX*CSALP(I)*(-2.0)
C      G(J+1,J)=G(J+1,J)+TX*SINALP(I)*(-2.0)
C      G(J+1,J+1)=G(J+1,J+1)+TX*CSALP(I)*(-2.0)
C
C      IF(I.EQ,NODES) GO TO 450
C      G(J,J+2)=G(J,J+2)+TX*SINALP(I+1)
C      G(J,J+3)=G(J,J+3)+TX*CSALP(I+1)
C      G(J+1,J+2)=G(J+1,J+2)+TX*SINALP(I+1)

```

```

450      G(J+1,J+3)=G(J+1,J+3)+TY*CONSALP(I+1)
C      CONTINUE
C  SUM LONGITUDINAL AND LATERAL FORCE COMPONENTS
DO 420 I=1,NMAX
  K=(I-1)/2+1
DO 420 J=1,NMAX
  G(I,J)=(G(I,J)+WORKM(I,J))/MASS(K)
C      CONTINUE
C      CALL MOVE(G,WORKM,NMAX,NMAT)
C
C      WRITE(6,9101)
9101  FORMAT(/,20H 2D UNDAMPED MATRIX ,/)
      IF(IPRNT(1).EQ.1) CALL PUTMAT(G,NMAX,NMAT,NMAX)
C
C  COMPUTE EIGENVALUES/EIGENVECTORS OF 2D MATRIX
C      CALL EIGPAC(NMAX,NMAT,G,WORK1,WORK2,FVEC,+2,ICNTL(1),WORKM,XW)
C
C *****
C  2 DIMENSIONAL DAMPED MEMBRANE MATRIX MODEL
C
202  IF(ICNTL(2).LE.0) GO TO 615
C
C  A IS DAMPED 2*N STATE COEFFICIENT MATRIX
      IF(NODES.GT.NDMAX) GO TO 615
      CALL CLEAR(A,NMAT,NMAT)
      CALL CLEAR(AS,NMAT,NMAT)
      CALL CLEAR(AT,NMAT,NMAT)
      IO=1
      JO=1
C
C  SET DAMPING AND PURE INTEGRATOR COEFFICIENTS
DO 1010 I=1,NODES
  A(IO,JO)=-DAMP(I)
  A(IO+1,JO)=1.0
  A(IO+3,JO+2)=1.0
  A(IO+2,JO+2)=-DAMP(I)

```

```

10=JD+4
10=IN+4
1010 CONTINUE
C
C ADD STIFFNESS TERMS FOR ALL SPRINGS
DO 1020 I=1,NODES1
10=I-1)*4-5
C
CALL ELEMK(ESTIF(1,1),THETA(I),-RKVEC(I))
C
DO 1030 J=1,4
DO 1030 K=1,4
L=(J*2)+I0
IF((L.LT.1).OR.(L.GT.NMAX2)) GO TO 1030
M=(K*2)+I0+1
IF((M.LT.1).OR.(M.GT.NMAX2)) GO TO 1030
A(L,M)=A(L,M)+ESTIF(J,K)
1030 CONTINUE
1020 CONTINUE
C
C ADD TENSION SPRING EFFECT MODEL INTO STIFFNESS MATRIX
DO 1025 I=1,NODES
J=(I-1)*4+1
TX=TENSN*2.0/(RLFNG(I)+RLENG(I+1))*STNALP(I)
TY=TENSN*2.0/(RLFNG(I)+RLENG(I+1))*CSALP(I)
C
IF(I.EQ.1) GO TO 1022
A(J,J-1)=A(J,J-1)+COSALP(I-1)*TY
A(J,J-3)=A(J,J-3)+TX*STNALP(I-1)
A(J+2,J-3)=A(J+2,J-3)+TX*STNALP(I-1)
A(J+2,J-1)=A(J+2,J-1)+TY*CSALP(I-1)
A(J,J+1)=A(J,J+1)+TX*STNALP(I)*(-2.0)
A(J,J+3)=A(J,J+3)+TY*CSALP(I)*(-2.0)
A(J+2,J+1)=A(J+2,J+1)+TX*STNALP(I)*(-2.0)
A(J+2,J+3)=A(J+2,J+3)+TY*CSALP(I)*(-2.0)
1022
C
IF(I.EQ.NODES) GO TO 1025
A(J,J+5)=A(J,J+5)+TX*STNALP(I+1)

```

```

A(I,J+7)=A(J,J+7)+TY*CSALP(I+1)
A(I+2,J+5)=A(J+2,J+5)+TX*SINALP(I+1)
A(J+2,J+7)=A(J+2,J+7)+TY*CSALP(I+1)
CONTINUE
1025 DO 1040 I=1,NMAX2,2
      K=I/4+1
      DO 1040 J=1,NMAX2
        A(I,J)=A(I,J)/MASS(K)
      CONTINUE
      CALL MOVE(A,AS,NMAX2,NMAT)
C
      WRITE(6,9102)
9102  FORMAT(/,17H 2D DAMPED MATRIX ,/)
      IF(IPRNT(2).EQ.1) CALL PUTMAT(A,NMAX2,NMAT,NMAX2)
C
C COMPUTE EIGENVALUES/EIGENVECTORS OF 2D MATRIX
      CALL EIGPAC(NMAX2,NMAT,A,WORK1,WORK2,EVEC,-2,ICNTL(2),AS,AT)
C *****
C
C UNDAMPED LATERAL STRING MODEL
C
615  IF(ICNTL(3).LE.0) GO TO 700
C
C GS HOLDS STRING SYSTEM MATRIX
      IF(NODES.GT.(2*NUMAX)) GO TO 800
      CALL CLEAR(GS,NMAT,NMAT)
      CALL CLEAR(XZ,NMAT,NMAT)
      CALL CLEAR(XT,NMAT,NMAT)
      NODEM1=NODEFS-1
C
      DO 630 I=2,NODEM1
C COMPUTE MEMBRANE IFNSION
        TX=TENSN/MASS(I)/RLENGH(I)
        GS(I,I)=2.0*TX
        GS(I,I+1)=TX
        GS(I,I-1)=TX
      CONTINUE
630

```

```

C SET FIRST AND LAST BOUNDARY ELEMENTS
TX=TFNSN/MASS(1)/RIENGO(1)
GS(1,1)=-2.0*TX
GS(1,2)=TX
TX=TFNSN/MASS(NODES)/RIENGO(NODES)
GS(NODES,NODES)=-2.0*TX
GS(NODES,NODEFM1)=TX
CALL MOVE(GS,XZ,NODES,NMAT)

C
WRITE(6,9103)
9103 FORMAT(/,15H LATERAL STRING ,/)
IF(JPRNT(3).EQ.1) CALL PUTMAT(GS,NODES,NMAT,NODES)

C COMPUTE EIGENVALUES/EIGENVECTORS OF 2D MATRIX
CALL EIGPAC(NODES,NMAT,GS,WORK1,WORK2,EVFC,+1,ICNTL(3),XZ,AT)

C *****
C DAMPED LATERAL STRING MODEL
700 IF(ICNTL(4).IE.0) GO TO 800

C XW HOLDS DAMPED STRING SYSTEM A MATRIX
IF(NODES.GT.(2*NOMAX)) GO TO 800
CALL CLEAR(XW,NMAT,NMAT)
CALL CLEAR(G,NMAT,NMAT)
CALL CLEAR(WORKM,NMAT,NMAT)

C SET PURE INTEGRATOR ELEMENTS
DO 710 I=1,NODES
J=I*2
K=I*2-1
XW(J,K)=1.0
710 CONTINUE

C LOAD TENSILE TERMS
NODEM1=NODES-1
DO 720 I=2,NODEM1

```

```

TX=TFNSN/MASS(I)/RIENGN(I)
  J=I*2-1
  XA(J,J)=DAMP(I)/MASS(I)
  XW(J,J-1)=TX
  XW(J,J+3)=TX
  XW(J,J+1)=-2.0*TX
720  CONTINUE
C
C SET FIRST AND LAST BOUNDARY ELEMENTS
TX=TFNSN/MASS(1)/RIENGN(1)
XA(1,1)=DAMP(1)/MASS(1)
XW(1,2)=-2.0*TX
XA(1,4)=TX
TX=TFNSN/MASS(NODES)/RIENGN(NODES)
J=NODES*2-1
XA(J,J)=DAMP(NODES)/MASS(NODES)
XW(J,J-1)=TX
XW(J,J+1)=-2.0*TX
NODET2=NMAX
CALL MOVE(XW,G,NMAX,NMAT)
C
  WRITE(6,9104)
9104  FORMAT(/,22H LATERAL DAMPED STRING ,/)
  IF(IPRNT(4).EQ.1) CALL PUTMAT(XW,NMAX,NMAT,NMAX)
C
C COMPUTE EIGENVALUES/EIGENVECTORS OF 2D MATRIX
  CALL EIGPAC(NMAX,NMAT,XW,WORK1,WORK2,EVEC,-1,ICNTL(4),G,WORKM)
C *****
C
C UNDAMPED LONGITUDINAL BAR MODEL
C
800  IF(ICNTL(5).NE.0) GO TO 850
C
C GS IS UNDAMPED MODEL
C
  IF(NODES.GT.(2*NMAX)) GO TO 800
  CALL CLEAR(GS,NMAT,NMAT)

```

```

CALL CLEAR(XZ,NMAT,NMAT)
CALL CLEAR(XT,NMAT,NMAT)
NODEM1=NODEFS-1

C SET UP STIFFNESS MATRIX
DO 820 I=2,NODEM1
  GS(I,I)=-RKVEC(I)-RKVEC(I+1)
  GS(I,I-1)=RKVEC(I)
  GS(I,I+1)=RKVEC(I+1)
CONTINUE
820
C SET FIRST AND LAST BOUNDARY NODES
  GS(1,1)=-RKVEC(1)-RKVEC(2)
  GS(1,2)=RKVEC(2)
  GS(NODEFS,NODEFS)=-RKVEC(NODEFS)-RKVEC(NODES1)
  GS(NODEFS,NODEM1)=RKVEC(NODEFS)

C
DO 830 I=1,NODES
DO 830 J=1,NODES
  GS(I,J)=GS(I,J)/MASS(I)
CONTINUE
830
CALL MOVE(GS,XZ,NODES,NMAT)

C
WRITE(6,9105)
9105 FORMAT(/,20H LONGITUDINAL STRING ,/)
  IF(IPRNT(5).EQ.1) CALL PUTMAT(GS,NODEFS,NMAT,NODEFS)

C
C COMPUTE EIGENVALUES/EIGENVECTORS OF 2D MATRIX
CALL ETGPAC(NODES,NMAT,GS,WORK1,WORK2,FVFC,+1,ICNTL(5),XZ,XT)
C *****
C DAMPED LONGITUDINAL BAR MODEL
C
850 IF(ICNTL(6).LE.0) GO TO 900
C
  IF(NODEFS.GT.(2*NDDMAX)) GO TO 900
  CALL CLEAR(G,NMAT,NMAT)
  CALL CLEAR(WORK4,NMAT,NMAT)

```

```

      CALL CLFAP(XW,NMAT,NMAT)
C SET PURE INTEGRATOR ELEMENTS
DO 860 I=1,NODES
  J=I+2
  K=I+2-1
  XW(J,K)=1.0
  CONTINUE
  NODEM1=NODES-1
C
C SET UP DAMPED STIFFNESS MATRIX
DO 870 I=2,NODEM1
  J=I+2-1
  XW(J,J)=-DAMP(I)
  XW(J,J-1)=RKVEC(I)
  XW(J,J+3)=RKVEC(I+2)
  XW(J,J+1)=-RKVEC(I+1)-RKVEC(I)
  CONTINUE
C SET FIRST AND LAST BOUNDARY ELEMENTS
XW(1,1)=-DAMP(1)
XW(1,2)=-RKVEC(1)-RKVEC(2)
XW(1,4)=RKVEC(2)
J=NODES+2-1
XW(J,J)=-DAMP(NODES)
XW(J,J-1)=RKVEC(NODES)
XW(J,J+1)=-RKVEC(NODES)-RKVEC(NODES+1)
C
DO 880 I=1,NMAX,2
DO 890 J=1,NMAX
  K=I/2+1
  XW(I,J)=XW(I,J)/MASS(K)
  CONTINUE
  CALL MOVE(XW,G,NMAX,NMAT)
C
  WRITE(6,9106)
  FORMAT(/,27H LONGITUDINAL DAMPED STRING ,/)
  IF(IPRNT(6).EQ.1) CALL PUTMAT(XW,NMAX,NMAT,NMAX)
C

```

```

      UFF FICENVALUES/FICENVECTORS OF 2D MATRIX
      CALL EIGSPAC(NMAX,NMAT,XW,WORK1,WORK2,EVEC,-1,ICNTL(6),G,WORKM)

```

```

SUBROUTINE TRUNK(ISHAPE)
C TRUNK GEOMETRY CALCULATIONS
C
REAL L,L1,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L12,R1,R2,PHI1,PHI2
COMMON/GEOMET/A,R,HY,L,L1,L2,L3,L4,L5,L6,L7,L8,L9,L10,L11,L12,R1,R2,PHI1,PHI2
DATA RTOL/0.01/
C
IF (HY.LT.0.0) GO TO 11
C *****
C ITERATION FOR R2
C COMPUTE INNER RADIUS OF CURVATURE
R2=SQRT(A**0.25+HY*HY)
C
C ITERATION LOOP FOR L2,L1,R1,R2
DO 10 I=1,50
PHI2=ARS(COS(AMAX1(-1.0,AMIN1(1.0,((R2-HY)/R2))))))
SINPH2=SIN(PHI2)
C COMPUTE OUTER RADIUS OF CURVATURE
R1=((A-R2*SINPH2)**2+(R+HY)**2)/(2.*(R+HY))
PHI1=ARS(COS(AMAX1(-1.0,AMIN1(1.0,((R1-HY-B)/R1))))))
XS=A-R2*SINPH2
IF (XS.LE.0.0) PHI1=6.2831852-PI1
L2=L-PHI1*R1
C R2S IS RESULTANT RADIUS FOR COMPUTED L2 IN ITERATION
IF (ARS(PHI2) .LT.1.0E-2) PHI2=1.0E-2
R2S=L2/PHI2
C TEST IF TOLERANCE .GT. ERROR
IF (ABS(R2-R2S).LE.RTOL) GO TO 50
R2=(R2+R2S)*0.5
10 CONTINUE
C *****
C ITERATED 50 TIMES WITHOUT SUCCESS, ERROR RETURN
11 CONTINUE
WRITE(6,9001)
9001 FORMAT(10X,26H INFEASIBLE TRUNK GEOMETRY ,//)
ISHAPE=0
RETURN
C TRUNK OK, RETURN

```

L1=L2
TSHAPE=1
RETURN
END

50

```

SUBROUTINE MOVE(A,AS,N,M)
C GENERAL MATRIX MOVE ROUTINE
  DIMENSION A(M,M),AS(M,M)
  DO 10 I=1,N
  DO 10 J=1,N
    AS(I,J)=A(I,J)
  CONTINUE
  RETURN
END
10

```

```

SUBROUTINE CMINV(A,N,D,L,M)
C COMPLEX MATRIX INVERSION
C
      COMPLEX A,D,RIGA,HOLD
      DIMENSION A(1),L(1),M(1)
C
      NM=N*N
      D=CMPLX(1.0,0.0)
      NK=-N
      DO 80 K=1,N
      NK=NK+N
      L(K)=K
      M(K)=K
      KK=NK+K
      RIGA=A(KK)
      DO 20 J=K,N
      IZ=N*(J-1)
      DO 20 I=K,N
      IJ=IZ+I
      10 IF(CABS(RIGA)-CABS(A(IJ))) 15,20,20
      15 RIGA=A(IJ)
      L(K)=I
      M(K)=J
      20 CONTINUE
C
C      INTERCHANGE ROWS
C
      J=L(K)
      IF(J=K) 35,35,25
      25 KI=K-N
      DO 30 I=1,N
      KI=KI+N
      HOLD=-A(KI)
      JI=KI-K+J
      A(KI)=A(JI)
      30 A(JI)=HOLD
C
C      INTERCHANGE COLUMNS

```

```

C
35 I=M(K)
   TF(I-K) 45,45,38
38 JP=N*(I-1)
   DO 40 J=1,N
   IK=NK+J
   JI=JP+J
   HOFD=-A(JK)
   A(JK)=A(JI)
40 A(JI)=HOFD
C
C   DIVIDE COLUMN BY MINUS PIVOT (VALUE OF PIVOT ELEMENT IS
C   CONTAINED IN RIGA)
C
45 TF(RIGA) 48,46,48
   D=CMPLX(0.0,0.0)
   RETURN
48 DO 55 I=1,N
   TF(I-K) 50,55,50
50 IK=NK+I
   A(IK)=A(IK)/(-RIGA)
55 CONTINUE
C
C   REDUCE MATRIX
C
   DO 65 I=1,N
   IK=NK+I
   HOFD=A(IK)
   IJ=I-N
   DO 65 J=1,N
   IJ=IJ+N
   TF(I-K) 60,65,60
60 TF(J-K) 62,65,62
62 KJ=IJ-I+K
   A(IJ)=HOFD*A(KJ)+A(IJ)
65 CONTINUE
C
C   DIVIDE ROW BY PIVOT

```

```

C
      KJ=K-N
      DO 75 J=1,N
      KJ=KJ+N
      IF(J=K) 70,75,70
      70 A(KJ)=A(KJ)/RIGA
      75 CONTINUE
C
      PRODUCT OF PIVOTS
      D=D*RIGA
C
      REPLACE PIVOT BY RECIPROCAL
C
      A(KK)=CMPLX(1.0,0.0)/RIGA
      80 CONTINUE
C
      FINAL ROW AND COLUMN INTERCHANGE
      K=N
      100 K=(K-1)
      IF(K) 150,150,105
      105 I=L(K)
      IF(I=K) 120,120,108
      108 J=N*(K-1)
      JR=N*(I-1)
      DO 110 J=1,N
      JK=JQ+J
      HOLD=A(JK)
      JI=JR+J
      A(JK)=A(JI)
      110 A(JI)=HOLD
      120 J=M(K)
      125 IF(J=K) 100,100,125
      KI=K-N
      DO 130 I=1,N
      KI=KI+N
      HOLD=A(KI)
      JI=KI-K+J

```

```
A(KI)=-A(JI)  
130 A(JI) =HRI,D  
GO TO 100  
150 K=0  
RETURN  
END
```

```

      SUBROUTINE ELEMK(X,A,STIF)
      C ELEMENT STIFFNESS MATRIX FORMULATION (2D)
      C A IS LOCAL ROTATION ANGLE
      DIMENSION X(4,4)
      C
      C COMPUTE TRANSCENDENTALS ONCE
      CA=COS(A)
      SA=SIN(A)
      CASA=CA*SA
      SSA=SA*SA
      CSA=CA*CA
      C
      C ROTATION MATRIX
      X(1,1)=CSA
      X(1,2)=CASA
      X(1,3)=-CSA
      X(1,4)=-CASA
      X(2,1)=CASA
      X(2,2)=SSA
      X(2,3)=-CASA
      X(2,4)=-SSA
      X(3,1)=-CSA
      X(3,2)=-CASA
      X(3,3)=CSA
      X(3,4)=CASA
      X(4,1)=CASA
      X(4,2)=-SSA
      X(4,3)=CASA
      X(4,4)=SSA
      C
      C MULTIPLY ROTATION MATRIX BY STIFFNESS
      DO 10 I=1,4
      DO 10 J=1,4
      X(I,J)=X(I,J)*STIF
      CONTINUE
      10
      C
      RETURN
      END

```

```

SUBROUTINE CLEAR(A,N,M)
C GENERAL MATRIX CLEAR TO ZERO
  DIMENSION A(1)
  N4EM*4
  DO 100 I=1,NM
    A(I)=0.0
  CONTINUE
  RETURN
  END
100

```

```

SUBROUTINE PUTMAT(A,N,M,L)
C MATRIX AND VECTOR PRINT OUT
  DIMENSION A(M,M)
  IF(L=1) 100,20,50
20  WRITE(6,9002)(A(I,1),I=1,N)
    WRITE(6,9001)
    GO TO 100
50  DO 75 I=1,N
    WRITE(6,9002)(A(I,J),J=1,L)
    WRITE(6,9001)
75  CONTINUE
100 CONTINUE
9001 FORMAT(/)
9002 FORMAT(5X,10G12,5)
    RETURN
  END

```

```

SUBROUTINE PUTFIG(X,Y,N,IJK)
C EIGENVALUE OUTPUT ROUTINE
  DIMENSION X(1),Y(1)
C
  WRITE(6,9005)
9005  FORMAT(/,10H FREQUENCY OF MODES )
  IF(IJK) 2,1,1
C ONF DIMENSIONAL CASE
1    WRITE(6,9011)
9011  FORMAT(/,5X,45H      HZ      RAD/SEC      REAL  ,/)
      GO TO 3
2    WRITE(6,9010)
9010  FORMAT(/,5X,68H      HZ      RAD/SEC      REAL  ,/)
1    IMAG
3    DO 50 I=1,N
      IF(ABS(X(I)).LE.1.0E-05) X(I)=0.0
      IF(ABS(Y(I)).LE.1.0E-05) Y(I)=0.0
50    CONTINUE
C
  DO 100 I=1,N
C W=SQRT(RE**2+IM**2)
      W=SQRT(X(I)**2+Y(I)**2)
C IF NO DAMPING TAKE SORT
      IF(IJK.GT.0.0) W=SQRT(W)
C CONVERT TO HERTZ
      Z=W*0.15915494
      IF(IJK) 70,60,60
60    WRITE(6,9000) Z,W,X(I)
      GO TO 100
70    WRITE(6,9000) Z,W,X(I),Y(I)
9000  FORMAT(5X,G12.5))
100   CONTINUE
      RETURN
  END

```

```

SUBROUTINE EIGPAC(N,M,A,WORK1,WORK2,EVFC,IJK,IFLAG,B,C)
C ROUTINE TO CALL SSP ROUTINES FOR EIGENVALUES
C N IS ARRAY SIZE TO DN, M IS STORAGE SIZE
C A,R CONTAIN SAME MATRIX
C WORK1,WORK2,C ARE WORK AREAS
  INTEGER EVFC
  DIMENSION A(1),WORK1(1),WORK2(1),EVFC(1)
  DIMENSION R(1),C(1)
C FORM HESSENBERG UPPER ALMOST TRIANGULAR MATRIX
20  CALL HSRG(N,A,M)
C
C COMPUTE EIGENVALUES WITH QR ALGORITHM
  CALL ATFIC(N,A,WORK1,WORK2,EVEC,M)
C
  CALL PUTFIG(WORK1,WORK2,N,IJK)
  IF(ABS(IFLAG)-2) 100,25,100
C CALL EIGENVECTOR ROUTINE IF NEEDED
25  CALL VFCPAC(N,M,R,WORK1,WORK2,A,C)
100  RETURN
      END

```

```

SUBROUTINE EVECTR(A,VEC,X,Y,N,M)
C ROUTINE TO COMPUTE EIGENVECTORS
C SOLVES FOR COMPLEX EIGENVECTORS OF A REAL MATRIX
C A,WORK,MAT ARE ALL FUNCTIONS OF VECPC ARRAY SIZE
      COMPLEX X(M),Y(M),VEC(M)
      COMPLEX A(40,40)
      COMPLEX WORK(40)
      COMPLEX MAT(1600),DET
C
      IF(N.LE.1) RETURN
      FORMAT(5X,5(2G12.5))
      VEC(N)=CMPLX(1.0,0.0)
      NM1=N-1
      DO 100 I=1,NM1
      DO 100 J=1,NM1
      K=(J-1)*NM1+I
      MAT(K)=A(I,J)
      CONTINUE
      IF(NM1-1) 200,200,300
      MAT(1)=CMPLX(1.0,0.0)/MAT(1)
      GO TO 400
100  CONTINUE
200  MAT(1)=CMPLX(1.0,0.0)/MAT(1)
      GO TO 400
C
C MATRIX INVERSION
300  CALL CMINV(MAT,NM1,DFT,X,Y)
      IF(DFT.EQ.CMPLX(0.0,0.0)) WRITE(6,9000)
      FORMAT(5X,17H DETERMINANT ZERO )
C
400  DO 500 I=1,NM1
      VEC(I)=CMPLX(0.0,0.0)
      DO 500 J=1,NM1
      K=(J-1)*NM1+I
      VEC(I)=VEC(I)+A(J,N)*MAT(K)
      CONTINUE
      RETURN
500  END

```

```

SUBROUTINE VFCPAC(N,M,A,EIGR,EIGI,X,Y)
C EIGENVECTOR COMPUTATION/OUTPUT CONTROL ROUTINE
C N IS NUMBER OF NODES, M IS DIMENSION
C NOTE * AC MUST BE DIMENSIONED TO (M*M)
C NOTE * X,Y MUST BE DIMENSIONED TO (M*2)
      COMPLEX X(R1),Y(R1)
      COMPLEX AC(40,40)
      DIMENSION A(M,M),EIGR(M),EIGI(M)
C
      NTYP=6
C SOLVE FOR EACH VECTOR
      DO 1000 K=1,N
        IF(K.EQ.1) GO TO 3
C SKIP SECOND HALF OF COMPLEX CONJUGATE PAIR
        IF((EIGR(K).EQ.EIGR(K-1)).AND.(EIGI(K).EQ.-EIGI(K-1)))GO TO 1000
3      CONTINUE
C LOAD EIGENVECTORS INTO COMPLEX VECTOR
      DO 5 I=1,N
        X(I)=CMPLX(EIGR(I),EIGI(I))
5      CONTINUE
C FORM COMPLEX MATRIX FROM REAL A
      DO 10 I=1,N
        DO 10 J=1,N
          AC(I,J)=CMPLX(A(I,J),0.0)
10      CONTINUE
C
C SUBTRACT EIGENVALUE FROM DIAGONAL OF A MATRIX
      DO 100 I=1,N
        AC(I,I)=AC(I,I)-X(K)
100      CONTINUE
C
C CALL EIGENVECTOR SOLVER
      CALL EVECPR(AC,Y,X(N+1),Y(N+1),N,M)
C
      WRITE(NTYP,9000) X(K),Y(1)
      WRITE(NTYP,9001)(Y(J),J=2,N)
9000  FORMAT(//.5X,11H EIGENVALUE ,2X,2F15.5,/,5X,14H EIGENVECTOR
1,2F15.5)

```

```

9001  FORMAT(19X,2F15.5)
C  NORMALIZE REAL PARTS OF DISPLACEMENT EIGENVECTORS
XMAX=0.0
DO 200 I=2,N,2
  IF(ABS(REAL(Y(I))).GT.XMAX) GO TO 200
  XMAX=REAL(Y(I))
200  CONTINUE
  WRITE(6,9010)
9010  FORMAT(/,42H REAL DISPLACEMENT NORMALIZED EIGENVECTORS ,/)
DO 300 I=2,N,2
  XI=REAL(Y(I))/XMAX
  WRITE(6,9001)XI
300  CONTINUE
1000  CONTINUE
      RETURN
      END

```