ANALYSIS OF TRUNK FLUTTER IN AN AIR CUSHION LANDING SYSTEM.

NOV 79  R B FISH

F33615-78-C-3412

UNCLASSIFIED

AFFDL-TR-79-3121
TRUNK FLUTTER ANALYSIS PROGRAM
USER'S MANUAL

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

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
NOTICE

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. HOWELL K. BREWER
Project Engineer 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 AFFDL/PEMB, WPAFB, 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.

AIR FORCE/96780/6 March 1980 — 130
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.
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.
# TABLE OF CONTENTS

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

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

vi
<table>
<thead>
<tr>
<th>Figure No.</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-5</td>
<td>Element coordinate transformation.</td>
<td>80</td>
</tr>
<tr>
<td>B-6</td>
<td>Transverse node displacement.</td>
<td>82</td>
</tr>
<tr>
<td>B-7</td>
<td>Force transformation.</td>
<td>83</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table No.</th>
<th>Table Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>THE CAPABILITIES OF THE TRUNK FLUTTER SIMULATION PROGRAM</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>A SUMMARY OF SUBROUTINES</td>
<td>12</td>
</tr>
<tr>
<td>B-1</td>
<td>A SUMMARY OF EIGENVALUE PROGRAM SUBROUTINES</td>
<td>69</td>
</tr>
</tbody>
</table>
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.*

---

**TABLE 1. THE CAPABILITIES OF THE TRUNK FLUTTER SIMULATION PROGRAM**

The program simulates the trunk behavior for variations in the following parameters.

<table>
<thead>
<tr>
<th>ACLS Parameters</th>
<th>Trunk Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>attachment points</td>
</tr>
<tr>
<td></td>
<td>cross section length</td>
</tr>
<tr>
<td></td>
<td>elasticity variations along the length</td>
</tr>
<tr>
<td></td>
<td>flexural stiffness variations along the length</td>
</tr>
<tr>
<td></td>
<td>density variations along the length</td>
</tr>
<tr>
<td></td>
<td>trim valve size</td>
</tr>
<tr>
<td></td>
<td>trunk orifice size and location</td>
</tr>
<tr>
<td>Fluid Parameters</td>
<td>fan characteristics</td>
</tr>
<tr>
<td></td>
<td>cushion volume</td>
</tr>
<tr>
<td></td>
<td>trunk volume</td>
</tr>
<tr>
<td></td>
<td>separation point</td>
</tr>
<tr>
<td></td>
<td>global damping</td>
</tr>
<tr>
<td>Operation Parameters</td>
<td>hard surface clearance</td>
</tr>
<tr>
<td>Flutter Suppression Parameters</td>
<td>strake at any location</td>
</tr>
<tr>
<td></td>
<td>minimum gap area</td>
</tr>
<tr>
<td></td>
<td>external spring at any location</td>
</tr>
<tr>
<td>Program Parameters</td>
<td>Simulation Parameters</td>
</tr>
<tr>
<td></td>
<td>time step</td>
</tr>
<tr>
<td></td>
<td>time limit</td>
</tr>
<tr>
<td></td>
<td>plotting options</td>
</tr>
<tr>
<td></td>
<td>Trunk Model Parameters</td>
</tr>
<tr>
<td></td>
<td>number of nodes</td>
</tr>
<tr>
<td>Options</td>
<td>separation point options*</td>
</tr>
<tr>
<td></td>
<td>a. diffuser model, i.e., separation at a fixed slope</td>
</tr>
<tr>
<td></td>
<td>b. fixed gap to separation point height</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>cushion - trunk pressures options</td>
</tr>
<tr>
<td></td>
<td>a. fixed cushion and trunk pressures**</td>
</tr>
<tr>
<td></td>
<td>b. variable cushion pressure fixed trunk pressure**</td>
</tr>
<tr>
<td></td>
<td>c. variable cushion and trunk pressures with fan characteristic**</td>
</tr>
<tr>
<td></td>
<td>pressure profile on trunk bottom options</td>
</tr>
<tr>
<td></td>
<td>a. nonvariable pressure profile**</td>
</tr>
<tr>
<td></td>
<td>b. variable pressure profile without trunk orifice flow**</td>
</tr>
<tr>
<td></td>
<td>c. variable pressure profile with trunk orifice flow**</td>
</tr>
</tbody>
</table>

* 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.
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 \( D_y_i(t) \) are then determined from \( y_i(t) \) by calling EQSIM

\[ D_y_i(t) = \frac{dy_i(t)}{dt} \]

c. The values \( y_{i1} \) at time \( t + dt/2 \) are then determined,

\[ y_{i1} = y_i + D_y_i \cdot dt/2 \]

d. The slopes \( D_y_{i1} \) at \( t + dt/2 \) 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

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

*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

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

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

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

3. NPLTM

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

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

Lable card for simulation output

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

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, IPXN, IFSEP, ILENG, INSEP

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)

IPXN - 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

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

SSLENG - Length of trunk with significant gap area for flow to atmosphere from cushion
TPERIM - Perimeter of trunk, around complete ACLS

9. ATC, ATRIM

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

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

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

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

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

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

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

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

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

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

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

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

PTK - Trunk pressure initial condition (or constant)
\[ P_{\text{FAN}} = CQ_0 + CQ_1 \cdot Q_{\text{FAN}} + CQ_2 \cdot Q_{\text{FAN}}^2 + CQ_3 \cdot Q_{\text{FAN}}^3 + CQ_4 \cdot Q_{\text{FAN}}^4 \]

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.* NVPLT, NPLT(I), I = 1,5 (I1, 4X, 5I2)

NVPLT - 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, DTPLT, XMIN, XMAX (6F12.5)

TPLSRT - Plot start time

TPLSTP - Plot end time

DTPLT - 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/sec$^2$

PI - $\pi$, 3.1415926535

PAT - Atmospheric pressure (absolute), Pat, 2116.8 lb/ft$^2$

RHO - Air density, $\rho$, 0.00234151 slugs/ft$^3$

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:

\[ I_f = \frac{\rho \ell}{A} = \text{AIFAN} \]

where

\[ I_f = \text{air inertance} \]

\[ \ell = \text{average flow path length in the fan, which may be approximated as the length of the fan} \]
\( \rho = \text{average air density} \)

\( A = \text{Cross section area of flow.} \)

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

\[
I = (J + 1) \times 4 + 1 \text{ for node No. } J
\]

\text{STATE}(I) = \text{X Velocity of node (J)}

\text{STATE}(I + 1) = \text{X Displacement of node (J)}

\text{STATE}(I + 2) = \text{Y Velocity of node (J)}

\text{STATE}(I + 3) = \text{Y Position of node (J)}

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

\[
I = (\text{NODES} + 2) \times 4 + \text{ICNTL}(1) + \text{ICNTL}(2) \times 2
\]

\text{STATE}(I) = \text{Dynamic cushion pressure}

\text{STATE}(I - 1) = \text{Dynamic trunk pressure}

\text{STATE}(I - 2) = \text{Dynamic fan flow}

\text{STATE}(I + 1) = \text{Trunk membrane length}

\text{STATE}(I + 2) = \text{Flow, trunk to cushion}

\text{STATE}(I + 3) = \text{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.

   \[ \text{ICNTL}(6) = 1 \]
   \[ \text{PEXT}(I) = \text{Node pressure (external) values. (Optional input)} \]

   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.

   \[ \text{ICNTL}(6) = 0 \text{ (DEFAULT MODEL)} \]

   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:

   \[ \text{ICNTL}(3) = 1 \]

   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.

\[ \text{IFSEP} = 2 \]

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.

\[ \text{IFSEP} = 1 \]

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.

\[ \text{IFSEP} = 3 \]
\[ \text{INSEP} = \text{separation node number} \]

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.

\[ \text{ICNTL}(3) = 1 \]
\[ \text{IFSEP} = 2 \]

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 equi-spaced 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 dynami-
cally, 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 dyna-
   mic 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).


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) ≠ 0).
4. ILLUSTRATIVE SIMULATION

The following describes the input data and the output printout 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.
<table>
<thead>
<tr>
<th>CARD NO.</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
</tr>
<tr>
<td></td>
<td>2.000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
</tr>
<tr>
<td></td>
<td>3.000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
</tr>
<tr>
<td></td>
<td>4.000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
</tr>
</tbody>
</table>

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
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
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
18    10.0, 0.1, 0
19    0.0, 0.0, 0.0
20    0.029, 1.041, 0.245, 0
21    -11274.0, 61.4279, -370792.0, 1.05604E-4, 1.574E-9
22    1.0, 0.50, 1.0, -0.2094395
23    2.775, 0.5, 2.775
24    330.0, 135.0, 1436.0
25    3XX73236404448
26    0.0, 0.99, 0.005, 0.0, 0.0
26    3XX757480000
27    0.0, 0.99, 0.005, 0.0, 0.0
26    2XX758270000
27    0.0, 0.99, 0.005, 0.0, 0.0

Figure 6. Input data test case. (Continued)
Figure 7. Program output.
Figure 7. Program output. (Continued)
Figure 7. Program output. (Continued)
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)

\[ \theta_i = \tan^{-1} \left( \frac{Y_{i+1} - Y_i}{X_{i+1} - X_i} \right) \]  \hspace{1cm} (A-1)

\[ \phi_i = \tan^{-1} \left( \frac{Y_i - Y_{i+1}}{X_i - X_{i+1}} \right) \]  \hspace{1cm} (A-2)
Figure A-1. Trunk representation for the dynamic simulation model.
Figure A-2. The 2D coordinate system.
\[ l_i = \sqrt{(X_{i+1} - X_i)^2 + (Y_{i+1} - Y_i)^2} \]  

(A-3)

Note: \( \tan^{-1} \) is a four quadrant function, therefore, if 
\[ \theta = \tan^{-1} \left( \frac{\Delta Y}{\Delta X} \right), \quad -\pi \leq \theta \leq \pi . \]

A.2 Force Relations

Spring Force (Figure A-3)

\[ F_{XK_i} = \cos(\phi_i) \cdot K_i \cdot \Delta \phi_i \]
\[ + \cos(\phi_{i+1}) \cdot K_{i+1} \cdot \Delta \phi_{i+1} \]  

(A-4)

\[ F_{YK_i} = \sin(\phi_i) \cdot K_i \cdot \Delta \phi_i \]
\[ + \sin(\phi_{i+1}) \cdot K_{i+1} \cdot \Delta \phi_{i+1} \]  

(A-5)

where

\[ \Delta \phi_i = \phi_i - \phi_{oi} \]

\[ \phi_{oi} (\phi_{oi} \text{ is initial } \phi_i) \]
Figure A-3. Trunk elasticity representation.
Bending Forces (Figure A-4)

TORQUE, $\tau = K_B \Delta \psi$

or

$$\tau = K_B (\psi_o - \psi) \quad (A-6)$$

since

$$\tau = F \times \ell$$

$$F \times \ell = K_B (\psi_o - \psi)$$

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

where

$$K_B = K_B^* \frac{2}{i + \frac{1}{2}}$$

so

$$F_1 = \frac{K_B^* (\psi_o - \psi)^2}{\ell (\frac{1}{i} + \frac{1}{i+1})} \quad (A-8)$$

$$F_2 = \frac{K_B^* (\psi_o - \psi)^2}{\ell (\frac{1}{i+1} + \frac{1}{i+1})} \quad (A-9)$$
\[
\begin{align*}
\text{FOR } \psi_i < \psi_{0i} \\
F_1 \text{ orthogonal to } v_i \\
F_2 \text{ orthogonal to } v_{i+1} \\
\psi_{0i} = \text{initial "no bending" node angle} \\
(\text{all angles are (+) counterclockwise rotation})
\end{align*}
\]

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)

\[
F_{x1} = -F_1 \cos(\beta_i) \tag{A-10}
\]

\[
F_{y1} = -F_1 \sin(\beta_i) \tag{A-11}
\]

\[
F_{x_{i+1}} = -F_1 \cos(\beta_{i+1}) - F_2 \cos(\beta_{i+1}) \tag{A-12}
\]

\[
F_{y_{i+1}} = -F_1 \sin(\beta_{i+1}) - F_2 \sin(\beta_{i+1}) \tag{A-13}
\]

\[
F_{x_{i+2}} = -F_2 \cos(\beta_{i+1} + \pi) \tag{A-14}
\]

\[
F_{y_{i+2}} = -F_2 \sin(\beta_{i+1} + \pi) \tag{A-15}
\]

Due to orthogonality:

\[
\beta_i = \beta_i - \frac{\pi}{2} \tag{A-16}
\]

\[
\beta_{i+1} = \beta_{i+1} - \frac{\pi}{2} \tag{A-17}
\]
From trigonometric identities:

\[ \cos(\theta_1 - \frac{\pi}{2}) = \sin(\theta_1) \]  \hspace{1cm} (A-18)

\[ \sin(\theta_1 - \frac{\pi}{2}) = -\cos(\theta_1) \]  \hspace{1cm} (A-19)

\[ \cos(\theta_1 + \frac{\pi}{2}) = -\sin(\theta_1) \]  \hspace{1cm} (A-20)

\[ \sin(\theta_1 + \frac{\pi}{2}) = \cos(\theta_1) \]  \hspace{1cm} (A-21)

So replacement of \( \phi \) terms with equivalent \( \theta \) terms gives:

\[ F_{X_1} = -F_1 \sin(\theta_1) \]  \hspace{1cm} (A-22)

\[ F_{Y_1} = -F_1 \left[ -\cos(\theta_1) \right] \]  \hspace{1cm} (A-23)

\[ F_{X_{i+1}} = -F_1 \left[ -\sin(\theta_i) \right] -F_2 \sin(\theta_{i+1}) \]  \hspace{1cm} (A-24)

\[ F_{Y_{i+1}} = -F_1 \cos(\theta_i) -F_2 \left[ -\cos(\theta_{i+1}) \right] \]  \hspace{1cm} (A-25)

\[ F_{X_{i+2}} = -F_2 \left[ -\sin(\theta_{i+1}) \right] \]  \hspace{1cm} (A-26)
\[ F_{y_{i+2}} = -F_2 \cos(\theta_{i+1}) \]  

(A-27)

Reducing the equations leads to:

\[ F_{x_i} = -F_1 \sin(\theta_i) \]  

(A-28)

\[ F_{y_i} = F_1 \cos(\theta_i) \]  

(A-29)

\[ F_{x_{i+1}} = F_1 \sin(\theta_i) - F_2 \sin(\theta_{i+1}) \]  

(A-30)

\[ F_{y_{i+1}} = -F_1 \cos(\theta_i) + F_2 \cos(\theta_{i+1}) \]  

(A-31)

\[ F_{x_{i+2}} = F_2 \sin(\theta_{i+1}) \]  

(A-32)

\[ F_{y_{i+2}} = -F_2 \cos(\theta_{i+1}) \]  

(A-33)

**Attached Spring Forces**

If an external spring is attached to node I to suppress flutter, it creates a generalized force \( F_G(I) \).
\[
F_G = -K_{\text{EXT}} \Delta D \tag{A-34}
\]

In \( X, Y \) components:

\[
F_{XG_i} = -K_{\text{EXT}} \Delta X_i \tag{A-35}
\]

\[
F_{YG_i} = -K_{\text{EXT}} \Delta Y_i \tag{A-36}
\]

where

\[
\Delta X_i = (X_{O_i} - X_i) \tag{A-37}
\]

\[
\Delta Y_i = (Y_{O_i} - Y_i) \tag{A-38}
\]

therefore

\[
F_{XG_i} = K_{\text{EXT}} (X_{O_i} - X_i) \tag{A-39}
\]

\[
F_{YG_i} = K_{\text{EXT}} (Y_{O_i} - Y_i) \tag{A-40}
\]
**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)

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

\[ F_{XD_i} = V_{x_i} \varepsilon_i \ell_i^2 \sqrt{M_i K_i} \quad (A-45) \]

\[ F_{YD_i} = V_{y_i} \varepsilon_i \ell_i^2 \sqrt{M_i K_i} \quad (A-46) \]
Figure A-6. Pressure force representation.
Figure A-7. Damping representation.
where

\[ \xi_i = \text{damping ratio at node } i \]
\[ M_i = \text{nodal mass at node } i \]
\[ K_i = \text{average stiffness at node } i \]

**Differential Equations:**

\[
\frac{d^2x_i}{dt^2} = \frac{(F_{x_d_i} + F_{x_k_i} + F_{x_p_i} + F_{x_b_i} + F_{x_G_i})}{M_i} \tag{A-47}
\]

\[
\frac{d^2y_i}{dt^2} = \frac{(F_{y_d_i} + F_{y_k_i} + F_{y_p_i} + F_{y_b_i} + F_{y_G_i})}{M_i} \tag{A-48}
\]

\[
\frac{dx_i}{dt} = x_i \tag{A-49}
\]

\[
\frac{dy_i}{dt} = y_i
\]

**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

\[
P_{ext} - P_{sep} = \rho \frac{V_{sep}^2}{2} \left[ 1 - \left( \frac{A_{sep}}{A_1} \right)^2 \right] \quad (A-53)
\]

at separation point

\[
P_{sep} = P_{at} = 0
\]

\[
P_1 = P_{ch} \quad V_1 = 0
\]

Therefore,

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

\[ V_{sep} = \sqrt{\frac{2P_{ch}}{\rho}} \]  \hspace{1cm} (A-55)

since

\[ Q = A_{sep} \sqrt{\frac{2P_{ch}}{\rho}} = A_{sep} \cdot V_{sep} \]  \hspace{1cm} (A-56)

therefore

\[ P_i = P_{ch} \left( 1 - \left( \frac{A_{sep}}{A_i} \right)^2 \right) \]  \hspace{1cm} (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:**

\[ \frac{d}{dt} (Q_{FAN}) = \frac{P(Q_{FAN}) - P_{tk}}{I_{FAN}} \]  \hspace{1cm} (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).
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:

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

\[
dW_i = \frac{i}{i} \cdot Q_{TC_i} \quad (A-63)
\]

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

then a new \( P_{i+1} \) is computed

\[
P_{i+1} = P_{ch} - \frac{i}{2} \cdot V_i^2 - \int_0^x \frac{V_i \cdot dW}{A_i} \, dx \quad (A-65)
\]
The flow-pressure pattern is computed under the trunk step-by-step out to the separation point. The flow computation is iterated by varying $W(l)$ 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

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

\[
P_p = -T \sin (\alpha_{p-1}) + T \sin (\alpha_p)
\]  

(B-1)

For a first order approximation,

\[
\sin (\alpha_{p-1}) \approx \frac{y_p - y_{p-1}}{y_{p-1}}
\]

\[
\sin (\alpha_p) \approx \frac{y_{p+1} - y_p}{y_p}
\]

Figure B-1. Lateral vibration model.
therefore,

\[ F_p = - \frac{T}{t_{p-1}} (y_p - y_{p-1}) + \frac{T}{t_p} (y_{p+1} - y_p). \]  (B-2)

From

\[ \bar{F} = MA \]

\[ F_p = \frac{d^2 y_p}{dt^2} \]

\[ \frac{d^2 y_p}{dt^2} = -\frac{T}{m_p} (y_p - y_{p-1}) + \frac{T}{m_p} (y_{p+1} - y_p), \]  (B-3)

let

\[ \lambda_p = \frac{2}{m_p} \frac{T}{t_p} : \text{ for } u_p = \lambda_{p-1} = \lambda \]

Then,

\[ \frac{d^2 y_p}{dt^2} + 2 \cdot \lambda_p y_p - \lambda_p^2 (y_{p+1} + y_{p-1}) = 0 \]  (B-4)

where the boundary conditions are:

\[ y_0 = y_{n+1} = 0. \]
Writing equation (B-4) in a Matrix form:

\[ \ddot{Y} + \lambda Y = 0 \]

\[
\begin{bmatrix}
\ddot{y}_1 \\
\ddot{y}_2 \\
\ddot{y}_3 \\
\vdots \\
\ddot{y}_n
\end{bmatrix} =
\begin{bmatrix}
2\lambda_1, -\lambda_1, \\
-\lambda_2, 2\lambda_2, -\lambda_2, \\
0, -\lambda_3, 2\lambda_3, -\lambda_3, 0, \ldots, 0 \\
0, \ldots, 0, -\lambda_n, 2\lambda_n
\end{bmatrix}
\begin{bmatrix}
y_1 \\
y_2 \\
y_3 \\
\vdots \\
y_n
\end{bmatrix} = 0
\]

Solution \( \det |\lambda| = 0 \) 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.
\[ \Gamma_{DP} = -V_p B_p \] (see Figure B-2) \hfill (B-6)

where

\[ B_p = \text{Damping constant of point } P \]
\[ V_p = \text{Velocity of point } P \]

so equation (B-4) becomes:

\[ \frac{d^2 y_p}{dt^2} = \frac{-T}{m_p} (y_p - y_{p-1}) + \frac{T}{m_p} (\dot{y}_{p+1} - \dot{y}_p) - \frac{B y_p}{m_p} \] \hfill (B-7)

where

\[ \frac{d}{dt} (\dot{y}_p) = \ddot{y}_p \]

Figure B-2. Nodal dampers (lateral).
The matrix formulation (B-5) becomes (using $k_p = k_{p-1}$ assumption):

\[
\begin{bmatrix}
\ddot{Y}_1 \\
\dot{Y}_1 \\
\ddot{Y}_2 \\
\dot{Y}_2 \\
\ddot{Y}_3 \\
\ddots \\
\ddots \\
\ddots \\
\ddots \\
\ddot{Y}_n \\
\dot{Y}_n
\end{bmatrix}
\begin{bmatrix}
\frac{B}{m_1}, 2\lambda_1, 0, -\lambda_1, 0, \\
1, 0, \\
0, -\lambda_2, \frac{B}{m_2}, 2\lambda_2, 0, -\lambda_2, 0, \\
0, 0, 1, 0, \\
0, 0, 0, -\lambda_3, \frac{B}{m_3}, 2\lambda_3, 0, -\lambda_3, 0, \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
0, 0, 0, 0, 0, 0, 1, 0 \\
0, 1, 0
\end{bmatrix}
\begin{bmatrix}
\dot{Y}_1 \\
Y_1 \\
\dot{Y}_2 \\
Y_2 \\
\dot{Y}_3 \\
Y_3 \\
\ddots \\
\ddots \\
\ddots \\
\ddots \\
\ddots \\
Y_n
\end{bmatrix}
\]

Solution of $\det \begin{bmatrix} 1 \end{bmatrix} = 0$ 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:

\[ F_i = -K_i X_i - K_{i+1} X_i + K_{i+1} X_{i+1} + K_i X_{i-1} \]  

(B-9)

\[ F_i = -(K_i + K_{i+1}) X_i + K_{i+1} X_{i+1} + K_i X_{i-1} \]  

(B-10)

From,

\[ \bar{F} = \bar{MA} \]
\[ F_i = m_i \frac{d^2x_i}{dt^2} \]  \hfill (B-11)

Therefore,

\[
\frac{d^2x_i}{dt^2} = -\frac{K_i}{m_i} x_{i-1} - \frac{K_{i+1}}{m_i} x_{i+1} + \left( \frac{K_i + K_{i+1}}{m_i} \right) x_i = 0 \]  \hfill (B-12)

Where boundary conditions are

\[ x_0 = x_{n+1} = 0 \]

In matrix form:

\[
\begin{bmatrix}
\vdots \\
\ddot{x} \\
\vdots
\end{bmatrix} + \begin{bmatrix}
1 & 1 \\
0 & 1 \\
\vdots & \vdots
\end{bmatrix} \begin{bmatrix}
x_0 \\
x_1 \\
\vdots
\end{bmatrix} = 0
\]

\[ \lambda = m^{-1}k \]
\[ \begin{bmatrix}
\ddot{x}_1 \\
\ddot{x}_2 \\
\ddot{x}_3 \\
\vdots \\
\ddot{x}_n
\end{bmatrix}
= 
\begin{bmatrix}
\frac{(K_1 + K_2)}{m_1}, & -K_2, & 0, \\
-K_2, & \frac{(K_2 + K_3)}{m_2}, & -K_3, & 0 \\
0, & -\frac{K_3}{m_3}, & \frac{(K_3 + K_4)}{m_3}, & -\frac{K_4}{m_4} \\
0, & \ldots, & \frac{0}{m_n}, & \frac{(K_n + K_{n+1})}{m_n}
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
\vdots \\
x_n
\end{bmatrix}
\]  
*(B-13)*

Solution of \( \det |A| = 0 \) 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.

\[ F_{Di} = -V_i \cdot B_i \]  
*(B-14)*

(see Figure B-4).
Equation (B-12) becomes

\[
\frac{d^2 X_i}{dt^2} - \frac{K_i}{m_i} X_{i-1} - \frac{K_{i+1}}{m_i} X_{i+1} + \left( \frac{K_i + K_{i+1}}{m_i} \right) X_i + \frac{B_i}{m_i} \dot{X}_i = 0
\]  

(B-15)
In matrix form:

\[
\begin{bmatrix}
\ddot{x}_1 \\
\ddot{x}_2 \\
\ddot{x}_3 \\
\ddots \\
\ddot{x}_n
\end{bmatrix} =
\begin{bmatrix}
\frac{B_1}{m_1} & \frac{(K_1 + K_2)}{m_1} & 0 & \frac{-K_2}{m_1} & 0 & \cdots \\
1 & 0 & \cdots \\
0 & \frac{-K_2}{m_2} & \frac{+B_2}{m_2} & \frac{(K_2 + K_3)}{m_2} & 0 & \frac{-K_3}{m_2} & 0 & \cdots \\
0 & 0 & 1 & 0 & \cdots \\
0 & \cdots & 0 & \frac{-K_n}{m_n} & \frac{B_n}{m_n} & \frac{(K_n + K_{n+1})}{m_n} & \cdots \\
0 & \cdots & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
\ddots \\
x_n
\end{bmatrix}
\]

(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*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

\[ K = \int \frac{B^T KB}{\ell} d\ell \]

\[ K = K_c \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \] in one dimension

\[ \bar{K}_i = \begin{bmatrix} \cos^2 \alpha & \sin \alpha \cos \alpha & -\cos^2 \alpha & -\sin \alpha \cos \alpha \\ \sin \alpha \cos \alpha & \sin^2 \alpha & -\sin \alpha \cos \alpha & -\sin \alpha \cos \alpha \\ -\cos^2 \alpha & \sin \alpha \cos \alpha & \cos^2 \alpha & \sin \alpha \cos \alpha \\ -\sin \alpha \cos \alpha & \sin \alpha \cos \alpha & \sin \alpha \cos \alpha & \sin^2 \alpha \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ u_j \\ v_j \end{bmatrix} \]

in two dimensions
\[ R = [K]_i \{ \delta \}_i \]

\[ U^T = [U_1 V_1 \ U_2 V_2] \]

\[ K = \sum_{i=1}^{N+1} K_i ; \ KU = R \quad \text{(B-17)} \]

\[ K = \text{global stiffness matrix} \]
\[ U = \text{displacement vector} \]
\[ R = \text{forcing load vector}. \]

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:

\[ \hat{k} = \frac{T*2}{(\hat{\chi}_i + \hat{\chi}_{i-1})} \quad \text{in Figure B-6} \quad \text{(B-18)} \]

since

\[ \hat{k} \cos(\alpha) \sim k_y \]
Figure B-6. Transverse node displacement.

\[ k \times \sin(\alpha) \approx k_x \]

with

\[ \alpha_i = \left( \frac{\alpha_i + \alpha_{i-1}}{2} \right); \quad \varphi = \frac{\ell_i + \ell_{i-1}}{2} \]

then

\[ P_T = \frac{T}{\varphi} (\dot{y}_i - \dot{y}_{i-1}) + \frac{T}{\varphi} (\dot{y}_{i+1} - \dot{y}_i) \]
or

\[ F_T = k(y_{i+1} - 2y_i + y_{i-1}) \quad (B-19) \]

(\( y \) is rotated relative to \( y \) by \( \alpha \).

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

Transverse displacement \( D \) transformation into \((X,Y)\) frame

\[ \Delta y \cos(\alpha_i) + \Delta x \sin(\alpha_i) = D \quad (\text{Figure B-7}) \quad (B-20) \]
\[
\hat{y}_i = y_i \left[ \cos (\hat{\alpha}_i) \right] + x_i \left[ \sin (\hat{\alpha}_i) \right]
\]

\[
F_x_i = \frac{T}{l_a} \sin \hat{\alpha}_i \left[ \begin{array}{c}
\hat{y}_{i-1} \\
\sin \hat{\alpha}_{i-1} x_{i-1} + \cos \hat{\alpha}_{i-1} y_{i-1}
\end{array} \right]
\] (B-21)

\[-2(\hat{y}_i) + \hat{y}_{i+1}\]

\[
F_y_i = \frac{T}{l_a} \cos \hat{\alpha}_i \left[ \hat{y}_{i-1} - 2\hat{y}_i + \hat{y}_{i+1} \right]
\] (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**

NODES - Number of nodes

3. **LS, RL, AX, BX, TENSN, HY**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LS</td>
<td>membrane length, stretched</td>
</tr>
<tr>
<td>RL</td>
<td>membrane length, unstretched</td>
</tr>
<tr>
<td>AX</td>
<td>horizontal distance between attachment point</td>
</tr>
<tr>
<td>BX</td>
<td>vertical distance between attachment point</td>
</tr>
<tr>
<td>TENSN</td>
<td>membrane preload tension</td>
</tr>
<tr>
<td>HY</td>
<td>trunk height</td>
</tr>
</tbody>
</table>

4. **MASS(I), I = 1, nodes**

MASS - lumped parameter nodal mass

5. **RKVEC(I), I = 1, nodes + 1**

RKVEC - elastic stiffness of element

6. **DAMP(I), I = 1, nodes**

DAMP - nodal damping in both x and y directions

7. **IXF**

IXF - element length select flag
(0, default; 1, read card 8)

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

RLENGO - element unstretched length
9. IXY

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

10. (Option) X(I), I = 1, nodes

X - X values of mass nodes

11. (Option) Y(I), I = 1, nodes

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.
<table>
<thead>
<tr>
<th>CARD NO.</th>
<th>010110050010101007</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>101010050010101007</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>9.75</td>
</tr>
<tr>
<td>4</td>
<td>1.0135,1.0135,1,1,0</td>
</tr>
<tr>
<td>5</td>
<td>0.86757,0.86757,0.86757</td>
</tr>
<tr>
<td>6</td>
<td>3926.0,3926.0,3926.0</td>
</tr>
<tr>
<td>7</td>
<td>0.010,0.010,0.010,0.010</td>
</tr>
<tr>
<td>8</td>
<td>0.887272,0.887272,0.887272</td>
</tr>
<tr>
<td>9</td>
<td>0.887272</td>
</tr>
<tr>
<td>10</td>
<td>0.13570,0.13570,0.13570</td>
</tr>
<tr>
<td>11</td>
<td>4.1501,4.1501,4.1501</td>
</tr>
<tr>
<td>12</td>
<td>0.9615,0.9615,0.9615</td>
</tr>
<tr>
<td>13</td>
<td>2.4289,2.4289,2.4289</td>
</tr>
</tbody>
</table>

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
ANALYSIS OF TRUNK FLUTTER IN AN AIR CUSHION LANDING SYSTEM.

NOV 79  R B FISH
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.
<table>
<thead>
<tr>
<th>ONE POSITION</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>9.409</td>
<td>0.4000</td>
</tr>
<tr>
<td>1.357</td>
<td>0.5945</td>
</tr>
<tr>
<td>1.840</td>
<td>1.7465</td>
</tr>
<tr>
<td>1.170</td>
<td>2.3815</td>
</tr>
<tr>
<td>1.692</td>
<td>2.3131</td>
</tr>
<tr>
<td>9.3</td>
<td>2.8000</td>
</tr>
<tr>
<td>8.453</td>
<td>2.6777</td>
</tr>
<tr>
<td>8.102</td>
<td>2.7044</td>
</tr>
<tr>
<td>8.295</td>
<td>2.7323</td>
</tr>
<tr>
<td>4.751</td>
<td>2.6240</td>
</tr>
<tr>
<td>8.451</td>
<td>2.4486</td>
</tr>
<tr>
<td>5.910</td>
<td>2.1678</td>
</tr>
<tr>
<td>5.717</td>
<td>1.4824</td>
</tr>
<tr>
<td>5.742</td>
<td>1.3695</td>
</tr>
<tr>
<td>5.715</td>
<td>1.0847</td>
</tr>
<tr>
<td>0.091</td>
<td>0.0877</td>
</tr>
<tr>
<td>8.370</td>
<td>0.3799</td>
</tr>
<tr>
<td>4.743</td>
<td>1.3739</td>
</tr>
</tbody>
</table>

Continuous string frequencies Hz

| 5.6947 |
| 16.777 |
| 29.45 |
| 31.787 |
| 32.574 |
| 47.07 |

Figure B-9. Sample output data.
Figure B-9. Sample output data. (Continued)
## B.5 Principal Nomenclature

<table>
<thead>
<tr>
<th>Program Variable Name</th>
<th>Symbol</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, XW, XZ(I,J)</td>
<td></td>
<td>Matrix, global stiffness</td>
</tr>
<tr>
<td>AS, G, GS(I,J)</td>
<td></td>
<td>Matrix, global stiffness</td>
</tr>
<tr>
<td>AT, WORKM, XT(I,J)</td>
<td></td>
<td>Matrix, global stiffness</td>
</tr>
<tr>
<td>AX</td>
<td>a</td>
<td>Horizontal distance between attachment points</td>
</tr>
<tr>
<td>BX</td>
<td>b</td>
<td>Vertical distance between attachment points</td>
</tr>
<tr>
<td>COSALP(I)</td>
<td>\cos(\theta_i)</td>
<td>Cosine of ALPHA(I)</td>
</tr>
<tr>
<td>COSTHE(I)</td>
<td>\cos(\theta_i)</td>
<td>Cosine of THETA(I)</td>
</tr>
<tr>
<td>DAMP(I)</td>
<td>D_i</td>
<td>Damping ratio of node (I)</td>
</tr>
<tr>
<td>ESTIF(I,J)</td>
<td></td>
<td>Matrix, element stiffness</td>
</tr>
<tr>
<td>EVEC(I)</td>
<td></td>
<td>Work vector space</td>
</tr>
<tr>
<td>ICNTL(I)</td>
<td></td>
<td>Program control vector</td>
</tr>
<tr>
<td>IPRNT(I)</td>
<td></td>
<td>Printer control vector</td>
</tr>
<tr>
<td>IXF</td>
<td></td>
<td>Flag, element initial length selection</td>
</tr>
<tr>
<td>IXY</td>
<td></td>
<td>Flag, X, Y position selection</td>
</tr>
<tr>
<td>LS</td>
<td>\ell_s</td>
<td>Trunk length</td>
</tr>
<tr>
<td>L1</td>
<td>\ell_1</td>
<td>Trunk inner length</td>
</tr>
<tr>
<td>L2</td>
<td>\ell_2</td>
<td>Trunk outer length</td>
</tr>
<tr>
<td>MASS(I)</td>
<td>m_i</td>
<td>Nodal mass</td>
</tr>
<tr>
<td>NODES</td>
<td></td>
<td>Number of nodes</td>
</tr>
<tr>
<td>PHI1</td>
<td>\phi_1</td>
<td>Trunk angle, inner</td>
</tr>
<tr>
<td>PHI2</td>
<td>\phi_2</td>
<td>Trunk angle, outer</td>
</tr>
<tr>
<td>Program Variable Name</td>
<td>Symbol</td>
<td>Explanation</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>R1</td>
<td>$R_1$</td>
<td>Trunk radius, inner</td>
</tr>
<tr>
<td>R2</td>
<td>$R_2$</td>
<td>Trunk radius, outer</td>
</tr>
<tr>
<td>RKVEC(I)</td>
<td>$R_{K_i}$</td>
<td>Trunk elasticity, element (I)</td>
</tr>
<tr>
<td>RL</td>
<td>$R_L$</td>
<td></td>
</tr>
<tr>
<td>RLENG(I)</td>
<td>$R_{L_i}$</td>
<td>Trunk element length (I)</td>
</tr>
<tr>
<td>RLENGO(I)</td>
<td>$R_{L_0i}$</td>
<td>Trunk element initial length (I)</td>
</tr>
<tr>
<td>SINALP(I)</td>
<td>$\sin(\theta_i)$</td>
<td>Sine of ALPHA(I)</td>
</tr>
<tr>
<td>SINTHE(I)</td>
<td>$\sin(\theta_i)$</td>
<td>Sine of THETA(I)</td>
</tr>
<tr>
<td>TENSN</td>
<td>$T$</td>
<td>Tension in trunk</td>
</tr>
<tr>
<td>W</td>
<td>$\omega$</td>
<td>String frequency</td>
</tr>
<tr>
<td>WORK1(I)</td>
<td></td>
<td>Matrix, work space</td>
</tr>
<tr>
<td>WORK2(I)</td>
<td></td>
<td>Matrix, work space</td>
</tr>
<tr>
<td>X(I)</td>
<td>$X_i$</td>
<td>X position of node (I)</td>
</tr>
<tr>
<td>Y(I)</td>
<td>$Y_i$</td>
<td>Y position of node (I)</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Program Variable Name</th>
<th>Symbol</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>a</td>
<td>Horizontal distance between inner and outer trunk attachment point</td>
</tr>
<tr>
<td>AGAP(I)</td>
<td>A_i</td>
<td>Flow gap trunk to ground</td>
</tr>
<tr>
<td>AIFAN</td>
<td>I_f</td>
<td>Fan inertance</td>
</tr>
<tr>
<td>ATC</td>
<td>A_{tc}</td>
<td>Area trunk to cushion</td>
</tr>
<tr>
<td>ATCF(I)</td>
<td>A_{tcf_i}</td>
<td>Area trunk to cushion element</td>
</tr>
<tr>
<td>ATRIM</td>
<td>A_{tr}</td>
<td>Area trim valve</td>
</tr>
<tr>
<td>B</td>
<td>b</td>
<td>Vertical distance between inner and outer trunk attachment point</td>
</tr>
<tr>
<td>CGAP</td>
<td>C_g</td>
<td>Discharge coefficient gap flow</td>
</tr>
<tr>
<td>CKK</td>
<td></td>
<td>Polytropic expansion coefficient</td>
</tr>
<tr>
<td>COSPHI(I)</td>
<td>\cos(\phi_i)</td>
<td>Cosine of PHI (I)</td>
</tr>
<tr>
<td>COSTHE(I)</td>
<td>\cos(\theta_i)</td>
<td>Sine of THETA (I)</td>
</tr>
<tr>
<td>CQ0</td>
<td>\alpha_0</td>
<td>Fan polynomial coefficient 1</td>
</tr>
<tr>
<td>CQ1</td>
<td>\alpha_1</td>
<td>Fan polynomial coefficient 2</td>
</tr>
<tr>
<td>CQ2</td>
<td>\alpha_2</td>
<td>Fan polynomial coefficient 3</td>
</tr>
<tr>
<td>CQ3</td>
<td>\alpha_3</td>
<td>Fan polynomial coefficient 4</td>
</tr>
<tr>
<td>CQ4</td>
<td>\alpha_4</td>
<td>Fan polynomial coefficient 5</td>
</tr>
<tr>
<td>Program Variable Name</td>
<td>Symbol</td>
<td>Explanation</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>CTC</td>
<td>( C_{tc} )</td>
<td>Discharge coefficient trunk to cushion, flow other than trim.</td>
</tr>
<tr>
<td>CTRIM</td>
<td>( C_{tr} )</td>
<td>Discharge coefficient trim value</td>
</tr>
<tr>
<td>DAMP(I)</td>
<td>( D_i )</td>
<td>Damping ratio X, Y at node (I)</td>
</tr>
<tr>
<td>DAMPR</td>
<td></td>
<td>Damper rest value</td>
</tr>
<tr>
<td>DERY(I)</td>
<td>( \frac{d}{dt} (S_i) )</td>
<td>Derivatives of state variables</td>
</tr>
<tr>
<td>DQ</td>
<td>( dq )</td>
<td>Incremental flow trunk-channel</td>
</tr>
<tr>
<td>DTIME</td>
<td>( dt )</td>
<td>Integration time step</td>
</tr>
<tr>
<td>DVCH</td>
<td>( \frac{d}{dt} (V_{ch}) )</td>
<td>Cushion volume rate of change</td>
</tr>
<tr>
<td>DW(I)</td>
<td>( dw_i )</td>
<td>Incremental mass flow trunk-channel element (I)</td>
</tr>
<tr>
<td>FORCXB(I)</td>
<td>( F_{xb_i} )</td>
<td>Force, X direction, bending (I)</td>
</tr>
<tr>
<td>FORCXD(I)</td>
<td>( F_{xd_i} )</td>
<td>Force, X direction, damping (I)</td>
</tr>
<tr>
<td>FORCXX(I)</td>
<td>( F_{xk_i} )</td>
<td>Force, X direction, elasticity (I)</td>
</tr>
<tr>
<td>FORCXP(I)</td>
<td>( F_{xp_i} )</td>
<td>Force, X direction, pressure (I)</td>
</tr>
<tr>
<td>FORCYB(I)</td>
<td>( F_{yb_i} )</td>
<td>Force, Y direction, bending (I)</td>
</tr>
<tr>
<td>FORCYD(I)</td>
<td>( F_{yd_i} )</td>
<td>Force, Y direction, damping (I)</td>
</tr>
<tr>
<td>FORCYK(I)</td>
<td>( F_{yk_i} )</td>
<td>Force, Y direction, elasticity (I)</td>
</tr>
<tr>
<td>FORCYP(I)</td>
<td>( F_{yp_i} )</td>
<td>Force, Y direction, pressure (I)</td>
</tr>
<tr>
<td>GAPMIN</td>
<td></td>
<td>Minimum gap area</td>
</tr>
<tr>
<td>HY</td>
<td></td>
<td>Trunk height</td>
</tr>
<tr>
<td>ICFLAG</td>
<td></td>
<td>Flag, trunk shape selection</td>
</tr>
<tr>
<td>Program Variable Name</td>
<td>Symbol</td>
<td>Explanation</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>ICNTL(I)</td>
<td></td>
<td>Control vector for options</td>
</tr>
<tr>
<td>ICS</td>
<td></td>
<td>Cushion separation node</td>
</tr>
<tr>
<td>IFSEP</td>
<td></td>
<td>Flag, separation point selection</td>
</tr>
<tr>
<td>ILENG</td>
<td></td>
<td>Flag, segment length selection</td>
</tr>
<tr>
<td>INODE</td>
<td></td>
<td>Node number of lowest trunk point</td>
</tr>
<tr>
<td>INSEP</td>
<td></td>
<td>Stake location node</td>
</tr>
<tr>
<td>ISEP</td>
<td></td>
<td>Separation point node</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>Trunk membrane length</td>
</tr>
<tr>
<td>L1</td>
<td>L_1</td>
<td>Inner trunk length (inner attachment point to bottom)</td>
</tr>
<tr>
<td>L2</td>
<td>L_2</td>
<td>Outer trunk length (outer attachment point to bottom) (L_1, L_2 used only by subroutine TRUNK)</td>
</tr>
<tr>
<td>NODES</td>
<td></td>
<td>Number of mass nodes</td>
</tr>
<tr>
<td>PAT</td>
<td>P_a</td>
<td>Atmospheric pressure</td>
</tr>
<tr>
<td>PCH</td>
<td>P_ch</td>
<td>Cushion pressure</td>
</tr>
<tr>
<td>PCRIT</td>
<td></td>
<td>Critical flow pressure</td>
</tr>
<tr>
<td>PEXT(I)</td>
<td></td>
<td>External pressure on trunk</td>
</tr>
<tr>
<td>PHI(I)</td>
<td>Φ_i</td>
<td>Angle PHI at node (I)</td>
</tr>
<tr>
<td>PLOST</td>
<td></td>
<td>Pressure loss due to momentum</td>
</tr>
<tr>
<td>PRESUR(I)</td>
<td></td>
<td>Pressure differential on trunk element</td>
</tr>
<tr>
<td>PTK</td>
<td>P_tk</td>
<td>Trunk pressure</td>
</tr>
<tr>
<td>QCA</td>
<td>Q_{CA}</td>
<td>Flow, cushion to atmosphere</td>
</tr>
<tr>
<td>QEXIT</td>
<td>Q_{EXIT}</td>
<td>Flow at exit (separation)</td>
</tr>
<tr>
<td>QFAN</td>
<td>Q_{FAN}</td>
<td>Fan flow</td>
</tr>
<tr>
<td>QGAP</td>
<td>Q_{GAP}</td>
<td>Gap flow</td>
</tr>
<tr>
<td>QIN</td>
<td>Q_{IN}</td>
<td>Flow to trunk</td>
</tr>
<tr>
<td>Program Variable Name</td>
<td>Symbol</td>
<td>Explanation</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>QINT</td>
<td>$Q_{\text{INT}}$</td>
<td>Momentum change integral</td>
</tr>
<tr>
<td>QOUT</td>
<td>$Q_{\text{OUT}}$</td>
<td>Flow, out of trunk</td>
</tr>
<tr>
<td>QTA</td>
<td>$Q_{\text{TA}}$</td>
<td>Flow, trunk to atmosphere</td>
</tr>
<tr>
<td>QTC</td>
<td>$Q_{\text{TC}}$</td>
<td>Flow, trunk to cushion</td>
</tr>
<tr>
<td>QTOT</td>
<td>$Q_{\text{TOT}}$</td>
<td>Total gap flow</td>
</tr>
<tr>
<td>QTRIM</td>
<td>$Q_{\text{TR}}$</td>
<td>Flow, trim valve</td>
</tr>
<tr>
<td>RBVEC(I)</td>
<td>$R_{\text{BVEC}}(I)$</td>
<td>Bending stiffness, node (I)</td>
</tr>
<tr>
<td>RHO</td>
<td>$\rho$</td>
<td>Air density</td>
</tr>
<tr>
<td>RITER</td>
<td>$R_{\text{ITER}}$</td>
<td>Iteration parameter</td>
</tr>
<tr>
<td>RKEEXTX</td>
<td>$R_{\text{KEEXTX}}$</td>
<td>External spring stiffness, X</td>
</tr>
<tr>
<td>RKEEXTY</td>
<td>$R_{\text{KEEXTY}}$</td>
<td>External spring stiffness, Y</td>
</tr>
<tr>
<td>RKVEC(I)</td>
<td>$R_{\text{KVEC}}(I)$</td>
<td>Trunk elasticity</td>
</tr>
<tr>
<td>RLENG(I)</td>
<td>$R_{\text{LENG}}(I)$</td>
<td>Trunk element length</td>
</tr>
<tr>
<td>RLENGO(I)</td>
<td>$R_{\text{LENGO}}(I)$</td>
<td>Trunk element unstretched length</td>
</tr>
<tr>
<td>RMASS(I)</td>
<td>$R_{\text{MASS}}(I)$</td>
<td>Nodal mass</td>
</tr>
<tr>
<td>SIE(I)</td>
<td>$\psi_{\text{i}}$</td>
<td>Angle SIE</td>
</tr>
<tr>
<td>SINPHI(I)</td>
<td>$\text{SINE}(\phi_{\text{i}})$</td>
<td>Sine of PHI(I)</td>
</tr>
<tr>
<td>SINTHE(I)</td>
<td>$\text{SINE}(\theta_{\text{i}})$</td>
<td>Sine of THETA(I)</td>
</tr>
<tr>
<td>SSSLENG</td>
<td>$\text{SSSLENG}$</td>
<td>Trunk gap length</td>
</tr>
<tr>
<td>STATE(I)</td>
<td>$S_{\text{i}}$</td>
<td>State variable (I)</td>
</tr>
<tr>
<td>TEMPAT</td>
<td>$\text{TEMPAT}$</td>
<td>Atmospheric temperature</td>
</tr>
<tr>
<td>THETA(I)</td>
<td>$\alpha_{\text{i}}$</td>
<td>Angle THETA(I)</td>
</tr>
<tr>
<td>TIME</td>
<td>$t$</td>
<td>Simulation time</td>
</tr>
<tr>
<td>TKVOL</td>
<td>$V_{\text{lk}}$</td>
<td>Trunk volume</td>
</tr>
<tr>
<td>TREST</td>
<td>$T_{\text{REST}}$</td>
<td>Damper reset time</td>
</tr>
<tr>
<td>Program Variable Name</td>
<td>Symbol</td>
<td>Explanation</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>TSEP</td>
<td>$T_{sep}$</td>
<td>Separation point angle</td>
</tr>
<tr>
<td>VCH</td>
<td>$V_{ch}$</td>
<td>Volume of cushion</td>
</tr>
<tr>
<td>VEL(I)</td>
<td>$V_i$</td>
<td>Velocity of flow at node (I)</td>
</tr>
<tr>
<td>VEXIT</td>
<td>$V_{exit}$</td>
<td>Initial estimate of flow velocity at separation point</td>
</tr>
<tr>
<td>VSEP</td>
<td>$V_s$</td>
<td>Velocity at separation point</td>
</tr>
<tr>
<td>W(I)</td>
<td>$W_i$</td>
<td>Mass flow at node (I)</td>
</tr>
<tr>
<td>X(I)</td>
<td>$X_i$</td>
<td>X position of node (I)</td>
</tr>
<tr>
<td>XEXITO</td>
<td>$X_{exit}$</td>
<td>X position of spring at rest</td>
</tr>
<tr>
<td>XZETA</td>
<td>$\zeta_x$</td>
<td>X damping ratio</td>
</tr>
<tr>
<td>Y(I)</td>
<td>$Y_i$</td>
<td>Y position of node (I)</td>
</tr>
<tr>
<td>YASEP</td>
<td>$Y_s$</td>
<td>Flow separation point area</td>
</tr>
<tr>
<td>YCSEP</td>
<td></td>
<td>Cushion separation point area</td>
</tr>
<tr>
<td>YDMIN</td>
<td></td>
<td>Minimum trunk Y displacement</td>
</tr>
<tr>
<td>YEXITO</td>
<td></td>
<td>Y position of spring at rest</td>
</tr>
<tr>
<td>YGAPM</td>
<td></td>
<td>Minimum gap area</td>
</tr>
<tr>
<td>YGRNDS</td>
<td></td>
<td>Hard surface clearance</td>
</tr>
<tr>
<td>YSEPX</td>
<td></td>
<td>Separation point gap</td>
</tr>
</tbody>
</table>
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
C SET UP DEVICE NUMBERS
NTYP=6
NINP=5
IPTAPC=4
ICD=NINP
IPR=NTYP
N=100

C INPUT INTEGRATION CONTROL PARAMETERS
100 READ(INP,1) FITR,TSTEP,STIME
110 WRITE (IPK,6)
    WRITE (IPR,4) N,STIME,FILN,TSTEP
    IFWR=0

C SET STATES OUT OF BOUNDS
DO 120 I=1,N
    YMIN(I)=1.75E0
    YMAX(I)=1.75E0
    F(I)=0.0
120 Y(I)=0.0

C FIND PRINTING VECTOR LENGTH.
READ (ICD,7) IPRNT
READ(NINP,7) NPRTY
DO 130 I=1,Y
    IF (IPRNT(I)) 140,140,130
130 CONTINUE
I=10
140 NPRNT=I-1

C ON FIRST CALL TO EOSIM, NJDOT IS NEGATIVE
    NJDOT=-1
    NJ=STIME

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
10 IF(NPRNT)410,410,310
310 IPR1=1
   WRITE (IPR,1) (TH,I=1,NPRNT)
   WRITE (IPR,12) (IR,I=1,NPRNT)
   WRITE (IPR,9) (TH,IPRNT(I),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 YM(N)=Y(I)
   430 IF (Y(I)=YMAX(I)) 450,450,440
440 YMAX(I)=Y(I)
450 CONTINUE
C I9ST FOR COMPLETE
C GET OUT IF T IS NOT BETWEEN TIME AND TTIME
   TX=T+0.5*STEP
   IF((STIME-TX)*(FTIME-TX))460,480,480
C CALL INTEGRATION ROUTINE FOR TIME STEP
460 CALL WDIF
   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
500 IF(YMAX(I))500,510,500
510 CONTINUE
C
C IF PLOTTING OTHER CALL OUTPUT ROUTINES
S20 IF(NPLOT4.GE.1) CALL PLOTT4
600 WRITE(NTYP,9010)
9010 FORMAT(9X,18H END OF DYSIS JOB ,/)
END
SUBROUTINE RK4IF
C RUNGE-KUTTA 4TH ORDER INTEGRATION ROUTINE
C
REAL SY(100),YO(100),Y1(100),Y2(100)
COMMON T,TSTEP,Y(100),DY(100),TIME,TIME,FNOWN,IFWRT,N,
1 IPR,ICD,ICN,TNEXT,PNEXT,TBACK
EQUIVALENCE (DT,TSTEP),(N,NSYS)
C
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)
   YO(I)=DY(I)
   10 Y(I)=H*DY(I)+Y(I)
C
   T=T+H
   CALL EQSIM
C
   DO 20 I=1,NSYS
      Y1(I)=DY(I)
   20 Y(I)=SY(I)+H*DY(I)
C
   CALL EQSIM
C
   DO 30 I=1,NSYS
      Y2(I)=DY(I)
   30 Y(I)=SY(I)+DT*DY(I)
C
   T=T+H
   CALL EQSIM
   H=H/3.0
C
   DO 40 I=1,NSYS
      PRT1=2.0*(Y1(I)+Y2(I))
      PRT2=YO(I)+DY(I)
   40
SUBROUTINE EGSIN
C DYSYS DYNAMIC INTEGRATION STATE EQUATION SUBROUTINE
C
C******************************************************************************
C DYSYS STATE EQUATIONS FOR ACLS TRUNK
C LUMPED PARAMETER MEMBRANE MODEL
C UNITS ARE IN FT,SLUG,SECOND SYSTEM
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 FORCE COMPONENT VECTOR(X,Y)
C FORCXD,FORCYD = DAMPER FORCE COMPONENTS
C FORCXL,FORCYL = SPRING FORCE COMPONENTS
C FORCXO,FORCYO = PRESSURE FORCE COMPONENTS
C FORCXB,FORCYB = BENDING FORCE COMPONENTS
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
C LOGICAL LDATS(16)
C

DIMENSION RIENGO(42), RLNGO(42)
DIMENSION RLENGO(42)
DIMENSION RMASS(40), RKVFLC(42)
DIMENSION RAMP(42), X(48), Y(48)
DIMENSION PRE运(42)
DIMENSION SINTHE(42), COSSFHE(42)
DIMENSION COSPHI(42), SINPHI(42)
DIMENSION FORCE(42), FORCY(42)
DIMENSION FORCX(42), FORCY(42)
DIMENSION FORCN(42), FORCY(42)
DIMENSION FEXT(42), AGAP(44)
DIMENSION RFACT(5)
DIMENSION RKSAV(42), RLSAV(42)
DIMENSION RTHETA(42)
DIMENSION ICNTL(16)
DIMENSION PH1(42), STE(42), RBVEC(42)
DIMENSION FORCXR(42), FORCY(42)
DIMENSION XLAB(40), FORCXR(42), FORCY(42)
DIMENSION W(42), VEL(42)
DIMENSION ATCF(42)
DIMENSION DK(42)

C

COMMON TME, OTE, STATE(100), DER2(100), STIME, FTIME, NEWDT, IFWRIT, N,
1 PDR, TCD, TCM, TANGE, PEKXT, TRACK
COMMON/GFMT/A, H, HY, I, HY, PII, PIH, R1, R2, I1, L, TSHAPE
COMMON/100/STNTP, NTP, NPT, TAF
COMMON/PLOT/NUM, TPL, SCR, TPLSTP, NPLTN, TPILOT, XPLTG(5, 200)
1, NMIN, NVPLT, XMIN, XMAX, NMAX, NSTORE

C

EQUVALENC (TX, THX)
EQUVALENC (RLNGO(1), RLNGO(1))
EQUVALENC (QFXTT, QOUT)
EQUVALENC (CGAP, CGAP)

C

ORIFICE FLOW EQUATION

QFLOW(X,Y)=X*SORT(ARS(Y))*SIGN(STTRHO,Y)

C
C FAN DYNAMICS PRESSURE VS. FLOW EQUATION
   PFA*X(I)=C00+C01*Q+C02*Q^2+C03*Q^3+C04*Q^4

C ***************************************************************
C KOSTF ENTRY PRINT
C ***************************************************************
C
C IF(N#DT) = -1 ON FIRST CALL, =1 ON 4TH CALL PER TIME STEP ELSE 0
C IF(N#DT)1,2,2
C
C INPUT INITIAL DATA
C CALLED FOR SIMULATION SET UP ONCE
1 C
   CONTINUE
   NMAX=100
   NSTONE=100
   NMAX=40
   NMAX1=41
   NMAX2=NMAX#2
   PI=3.1415927
   Y014=5.0
   G=37.174
   PCTOT=0.0
   RNH=0.0
   OUCH=0.0
   CKK=1.4
   PAT=14.7*144.0
   TF#PAT=70.0
   RH=1.24/(440.0+TK#PAT)
   ST#RH=SQRT(2.0/RH)
   QTC=0.0
   QTRIM=0.0
   OCA=0.0
   OFAN=0.0
C CLEAR VECTORS TO ZERO FOR STATE AND DERIVATIVES
C DERV(I)=0.0
C DO 10 I=1,NMAX

10 CONTINUE
STATE(I)=0.0
CONTINUE

DO 15 I=1,NMAX
PEXT(I)=0.0
FORCX(I)=0.0
FORCY(I)=0.0
FORCZ(I)=0.0
FORCP(I)=0.0
FORCX(I)=0.0
FORCY(I)=0.0
FORCZ(I)=0.0
PREU(I)=0.0
RLEUV(I)=0.0
CONTINUE

C READ FOR LABEL CARD
READ(NINP,9430)(XIAR(I),I=1,40)
9430  FORMAT(40A2)
WRITE(NOUT,9431)(XLAH(I),I=1,40)
9431  FORMAT(/,5X,40A2,/)  

C INPUT CONTROL VECTOR
ICNTL(1) ON = 1 FOR CUSHION PRESSURE DYNAMICS
ICNTL(2) ON = 1 FOR TRUNK-FAN PRESSURE DYNAMICS
ICNTL(3) ON = 1 FOR TRUNK FLOW INTO CHANNEL
ICNTL(4) ON = 1 FOR SEPARATION POINT INTERPOLATION
ICNTL(5) ON = 1 FOR SPECIAL NO TRUNK MOTION EXECUTION
ICNTL(6) ON = 1 FOR STATIC PRESSURE LOAD CASE
C READ(NOUT,9006)(ICNTL(I),I=1,16)
9006  FORMAT(9I1)
WRITE(NOUT,9432)(ICNTL(I),I=1,16)
9432  FORMAT(5X,16I3,/)  
DO 12 I=1,16
LDATS(1)=.FALSE.
IF (ICNTL(1),GT,0) LDATS(1)=.TRUE.
17 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,9441)
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 IFLAG = 0 FOR READ X,Y ELSE COMPUTE
C IFXN = PRINT AND PLOTTER VS STEP NUMBER
C IFSFP = SEPARATION POINT SELECTION FLAG =2 FOR DIFFUSER ELSE 1
C ILENG = ELEMENT LENGTH CONTROL FLAG
C INSEP = SEPARATION POINT NODE SET
C
READ(<!NP,9005) NODES,ICFLAG,IFXN,IFSEP
1,ILENG,INSEP
9005 FORMAT(1015)
NDEFS1=NODFS+1
NDEFS2=NODFS+2
NSTATE=NODFS*4
N=NODFS*4+ICNTL(1)+ICNTL(2)*2
WRITE(NTYP,9402)
9402 FORMAT(/,5X,31H NODES,NSTATE,ICFLAG,IFXN,IFSEP,12H INSEP,ILENG )
WRITE(NTYP,9120) NODS,NSTATE,ICFLAG,IFXN,IFSEP
1,INSEP,ILENG
9120 FORMAT(10110,/)
C
C GEOMETRY CONSTANTS
READ(NINP,9000)A,R,L,H Y
HY=HY
WRITE(NTYP,9401)
9401 FORMAT(5X,11H A,R,L,H Y)
WRITE(NTYP,9110)A,R,L,H Y
C
READ(NINP,9000)SSL ENG,TPFRIM
WRITE(NTYP,9421)
9421 FORMAT(5X,* SSL ENG,TPFRIM *)
WRITE(NTYP,9110)SSL ENG,TPFRIM
READ(NINP,9000)ATC,AT RIM
WRITE(NTYP,9428)
WRITE(NTYP,9110)ATC,AT R IM
9428 FORMAT(5X,* ATC,AT R IM *)
READ(NINP,9000)(ATCF(I),I=1,NODES1)
WRITE(NTYP,9425)
9425 FORMAT(5X,*,AREA TRUNK=GAP EACH ELEMENT *)
CALL PUTVEC(ATCF,NODES1)
C
C II.FNG, I=READ DATA CARDS , N= COMPUTE I/NODES
IF(ILFNG)7,7,8
8 READ(NINP,9000)(RLENG0(I),I=1,NODES1)
GO TO 9
7 DO 6 I=1,NODES1
RLENG0(I)=L/FLOAT(NODES1)
CONTINUE
9 CONTINUE
WRITE(NTYP,9408)
9408 FORMAT(5X,11H SPRING I, 0 )
CALL PUTVEC(RLENG0,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(TCFLG)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)
9000  FORMAT(AG10,4)
   GO TO 45
C
C  OPTIONAL TRUNK SHAPE INITIAL CONDITIONS
C  ONLY GOOD FOR EQUISPACED NODES
25  CALL TRUNK
   NX=1FTX(FLAT(NODES1)*(PHI2*R2/I.))
   NZ=NODES-4X
   WRITE(NTYP,9120)NX,NZ
C
C  COMPUTE RIGHT SECTOR POINTS
   XCNTR=R2*STN(PHI2)
   YCNTR=HY=R1
   TX=2.0*ASIN(4*ENGG(1)*0.5/R1)
   ANG=ATAN2((X(NODES2)-XCNTR),(Y(NODES2)-YCNTR))
   DT  40  I=1,NZ
   ANG=ANG+TX
   J=NODES2+1
   X(J)=XCNTR+R1*STN(ANG)
   Y(J)=YCNTR+R1*STN(ANG)
CONTINUE
C
C  COMPUTE LEFT SECTOR POINTS
   YCNTR=YCNTR+R1=R2
   ANG=1.570796-PHI2
   TX=2.0*ASIN(4*ENGG(1)*0.5/R2)
   DT  43  I=1,NX
   ANG=ANG+TX
   J=I+1
   X(J)=XCNTR-R2*STN(ANG)
   Y(J)=YCNTR-P2*STN(ANG)
43 CONTINUE
C
C FORCE NODES ABOVE GROUND LEVEL.
45 DO 46 I=1,NODES2
   Y(I)=AMIN1(YMIN,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 70 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,FR,4))
20 CONTINUE
   WRITE(NTYP,9414)
9414 FORMAT(/)
C
C COMPUTE THETA NODE ANGLES
   DO 21 I=1,NODES1
21 THETA(I)=ATAN2((Y(I+1)-Y(I)),(X(I+1)-X(I)))
   IF(FLAG=I)XN
   WRITE(NTYP,9409)
9409 FORMAT(5X,12H NODE ANGLES )
   CALL PUTVFC(THETA,NODES1)
C
C INPUT MASS IN POUNDS
   READ(NINP,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)
9405 FORMAT(5X,12H NODE "MASS"S )
   CALL PUTVFC(RMASS,NODES)
C
C INPUT MEMBRANE STIFFNESS L/H/FT
READ(NINP,0000)(RKVFEC(I),I=1,NODES)
C SAVE SPRING DATA
DO 4 I=1,NODES
RKSAV(I)=RKVFEC(I)
4 CONTINUE
WRITE(NYP,9406)
9406 FORMAT(5X,17H SPRING CONSTANTS )
C
C C
C INPUT MEMBRANE BENDING STIFFNESS L.B/RAD
READ(NINP,0000)(RRVFEC(I),I=1,NODES)
WRITE(NYP,9415)
9415 FORMAT(5X,13H BENDING STIFFNESS )
C
C INPUT MEMBRANE DAMPING L.B-SEC/FT
READ(NINP,0000)(DAMP(I),I=1,NODES)
WRITE(NYP,9407)
9407 FORMAT(5X,13H NODE DAMPING )
C
C DAMPER RESFT TIME AND FACTOR
ITH=0
READ(NINP,0000)TRST,DAMPP
WRITE(NYP,9429)
9429 FORMAT(5X,* TREST,DAMPR *)
WRITE(NYP,9110) TRST,DAMPR
C
C INPUT EXTERNAL STIFFNESS COEFFICIENTS
READ(NINP,0001) IEXT,PKFXTX,RKFXTY
9001 FORMAT(12,2I10,5)
C
C SAVE I,C, FOR SPRING ATTACHMENT
XEXTO=X(IEX)
YEXTO=Y(IEX)
IF(IEXT,NE,0) WRITE(NYP,9002) IEXT,RKEXTX,RKFXTY
9002 FORMAT(5X,25H EXTERNAL SPRING AT NODE ,12.5H RKX=,
1 F12.5,CH PKYE,F12.5,/

C
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,901) C00,C01,C02,C03,C04
WRITE(Ntyp,9427)
9427 FORMAT(5X,*C00,C01,C02,C03,C04 *)
WRITE(Ntyp,9902) C00,C01,C02,C03,C04
9901 FORMAT(5F15.5)
9902 FORMAT(5X,6(513,6,2X))
READ(N inp,9000) CTC,CTRIM,CGAP,TSFP
WRITE(Ntyp,9470)
9470 FORMAT(5X,* CTC,CTRIM,CGAP,TSFP *)
WRITE(Ntyp,9110) CTC,CTRIM,CGAP,TSFP
READ(Ninp,9000) YGRNDS,S RATIO,YDMIN
WRITE(Ntyp,9403)
9403 FORMAT(5X,* YGRNDS,S RATIO,YDMIN *)
WRITE(Ntyp,9110) YGRNDS,S RATIO,YDMIN

C
READ(Ninp,9000)PTK,PCH,QGAP
C IF DYNAMIC FAN SET INITIAL CONDITION
IF(IdatS(1)) STATE(N)=PCH
IF(IdatS(2)) STATE(N-1)=PTK
IF(IdatS(2)) STATF(N-2)=QGAP
IF(IdatS(2)) QFAN=QGAP
C SET EQUILIBRIUM FLOW INPUT AS FUNCTION OF PCH AND GAP
QIN=0.0
QFIT=0.0
C IF DYNAMIC CUSHION PRESSURE SFT 1,C,
IF(IdatS(1)) STATE(N)=PCH
WRITE(Ntyp,9404)
9404 FORMAT(5X,* PTK,PCH,QGAP *)
WRITE(Ntyp,9110)PTK,PCH,QGAP
9110 FORMAT(5X,10F12.5,/)
C         IF(.NOT.,LOADS(6)) GO TO 2
C         INPUT OUTER PRESSURE PROFILE
READ(NTYP,9200)(PFXT(I),I=1,NODES)
WRITE(NTYP,9410)
9410   FOMAT(5X,1RH NODF EXF PRESSURE )
   CALL PUTVEC(PFXT,NODES1)
C
C******************************************************************************
C ONE ENTRY PER STEP
C******************************************************************************
C
C LIMIT Y NODE DISPLACEMENT TO GROUND
2   DO 2500 I=1,NODES
   J=I+4
   STATE(J+2)=AMIN1(STATE(J+2),YDYN)
2500 CONTINUE
C TEST IF LAST CALL FOR STEP ELSE GOTO 3
   IF(NEWNT)30,3,30
C
C ****
30   YMAX=0.0
C FIND LOWEST NODE POINT ON TRUNK
   DO 5000 I=2,NODES1
      IF(Y(I),LT,YMAX) GO TO 5000
      INODE=I
      YMAX=Y(I)
5000 CONTINUE
C
C INODE= NUM OF MIN GAP NODE
C COMPUTE GAP AREAS
C
5001   DO 5005 I=1,NODES2
     AGAP(I)=YMAX(I)-YGRND(I)
5005   CONTINUE
C
   ICS=1
   YCSFP=AGAP(IC)
   END
C GAP AT LOWEST POINT
YCAPM=YCAP(1,INDF)
C IF LOADS(N) TRUE STATIC PRESSURE INPUT, SKIP DYNAMICS
IF(LOADS(N)) GO TO 5081
C
C IF NO GAP JUMP TO SPECIAL NO FLOW CASE
IF(YCAPM.EQ.0.0) GO TO 5054
YF=YCAPM*10.0
C LOOK BACK TO FIND CUSHION SEPARATION POINT
DO 5010 I=1,INDF
IF(YCAP(I),Lt.,YF) GO TO 5020
C REMEMBER POINT
ICS=I
YCSFP=YCAP(I)
5010 CONTINUE
C
C **************************************************************
C SEPARATION POINT SELECTION ROUTINE
5020 ISEP=INDF
YSEP=YCAP(ISEP)
C ISEP .EQ. 1 FOR FIXED GAP CASE
C ISEP .EQ. 2 FOR DIFFUSER CASE
C ISEP .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=YCAPM/SRATIO
DO 5030 I=INDF,NOFS2
IF(YCAP(I),Lt.,YSEPX) GO TO 5030
ISEP=I
C Y AT SEPARATION POINT = YSEPX VALUE = YCAPM/SRATIO
YSEP=AINV1(YSEPX,YCAP(I))
GO TO 5051
5030 CONTINUE
CALL FRPDP(1)
C
C ********
C SEPARATION POINT FROM DIFFUSER MODEL
C & DEGREE SLOPE OR NORE FOR SEPARATION
C LOOK FROM IN#ST NODE OUT#AR
5031 ISFP=IN#SP
    YASFP=AGAP(ISFP)
    M1=5040 I=ISFP,NODF52
    IF(THETA(I),GT,TSFP) GO TO 5040
    GO TO 5041
5040 CONTINUE
    CALL ERROR(7)
    YASFP=AGAP(I)
    GO TO 5050
C
C ********
C FORCE SEPARATION POINT TO BE AT NODE IN#SP
5031 ISFP=IN#SP
    YASFP=AGAP(ISFP)
    GO TO 5051
C
C ********
C INTERPOLATION SCHEME FOR SEPARATION POINT GAP
5050 IF (.NOT.,LSTAB(4)) GO TO 6200
    TEMP1=THETA(ISFP)-THETA(ISFP)
    TEMP2=THETA(ISFP)-THETA(ISFP+1)
    TEMP1=(AGAP(ISFP+1)-AGAP(ISFP))*TEMP1/TEMP2
    YASFP=AGAP(ISFP+TEMP1)
    GO TO 5051
C
C ********
C TRUNK FLOW SET SEPARATION POINT
6200 IF (.NOT.,LSTAB(3)) GO TO 5051
C LOOK FOR SEPARATION POINT
DO 6210 I=1,NODF
    J=NODF3+1
    ISX=J
    IF (ATCF(J),GT,0,0) GO TO 6250
6210 CONTINUE
    GO TO 8051
Continued

CALL FRMPP(3)

6750 IF(1SX.CF.ISFP) G1 TO 5051
   ISFP=1SX
   YAS=P=AGAP(ISFP)

C

C ******
C PRESSURE = PCH FOR NODES 1 TO ICS
C PRESSURE = PCH FOR NODES ISFP+1 TO 1MAX
C PRESSURE = F(VFL) FOR NODES ICS+1 TO ISFP
C MINIMUM GAP FLOW AREA AT NODE = IMODE

5051 ICS1=ICS+1
   ISFP1=ISFP+1
   ICS1=INT(ICS1,HODES2)
   ISFP1=INT(ISFP1,HODES2)

C

C COMPUTE VELOCITY AND FLOW AT EXIT POINT
C USE SEPARATION GAP TO CONTROL TOTAL FLOW Q=CF(P)
VEXIT=SQRT(V/2.0*PCH/RIID)
QEXIT=VEXIT*YASFP
G1 TO 5045

C

C ******
C SPECIAL NO FLOW CASE

5054 CONTINUE
   QEXIT=0.0
   ICS=HODES
   ISFP1=INODES+1

C

C ******
C LOAD OUTER PRESSURE DATA ARRAY
C LOAD CUSHION PRESSURE FROM INNER EDGE TO CUSHION SEPARATION
5055 DD 5060 T=1,ICS
   PEXIT(1)=PCH
5060 CONTINUE
C

C LOAD ATMOSPHERE PRESSURE FROM SEPARATION POINT TO END
C INITIAL MASS INWT OF CUSHION
   *(I)=WHAB*CIC*YASFP*SORT(2.0/RH0*AMAX1(0.0,(PCU-PLST)))
   RTER=0.10

C USING ESTIMATE OF CUSHION PRESSURE FIRST ITERATE PROFILE
6010  QTNT=0.0
       QTNT=QTNT+(1)/RH0
       DO 6050  I=2,1SEP
       DO=0.0
       WRITE(I)=RH0
       IF(ACCF(I),60,0,0) GO TO 6040
C USE PRESSURE AT I-1 AND ESTIMATE OF P(I)
   PAVF=(PFXT(I)+PFXT(I-1))*0.5
C COMPUTE FLOW INTO CONTROL VOLUME FROM TRUNK
   DO=DFLOM(CIC,(PAT-PAVF))*ACCF(I)
   WRITE(I)=RH0*RO
C COMPUTE MASS FLOW AT EXIT OF CONTROL VOLUME
6040  QTNT=QTNT+DO
       VEL(I)=QTNT/AGAP(I)
       #1=0.0*W(I-1)
C COMPUTE MOMENTUM CHANGE PRESSURE DROP
   QINT=1.0/AGAP(I)*VEL(I-1)*DO(I-1)*QINT
C COMPUTE NEW ESTIMATE OF VOLUME PRESSURE
   PFXT(I)=PCU-0.5*RH0*VEL(I-1)**2-QINT
   PFXT(I)=AMAX1(PFXT(I),PCRIT)
6050  CONTINUE
C
C TEST IF EXIT POINT PRESSURE IS WITHIN HOUNDS SET
   IF((PFXT(1SEP),GT,-1.0),AND,(PFXT(1SEP),LT,1.0)) GOTO 6075
C ERROR ON EXIT PRESSURE , ITERATE
   ITCTE=ITCTE+1
C ALLOW ONLY TEN ITERATIONS ON PRESSURE
   IF(ITCTE,GE,11) GOTO 6075
   IF(ITCTE,GE,5) RTER=RTER+0.00
C CORRECT INITIAL FLOW FROM CUSHION TO ZERO P EXIT
   #1=1.0*CTC*RTER*RH0*DFLOM(YASFP,PFXT(1SEP))
   GT TO 6015
6075  CONTINUE
C ***
C IF NOORE CONTACT SET TO TRUNK PRESSURE
5011 IF N02=1,NDPFS1
IF(A(GAP(1),J,F,0.,0.)=PTK
5022 CONTINUE
5090 CONTINUE
C
C HOW HAVE PRESSURE PROFILE FOR TRUNK
57 75 I=1,NDPFS1
75 PRESSURE(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)
QFAN=AMIN(QFAN,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
ACAP=8ASFP*2.0*SSLENG
PCT=STATE(N)
PCH=AMAX1(PCH,0.,0.)
QTA=0.0
QCA=ACAP*QFLOW(CGAP,PCH)
QTC=TC*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+QF=(1./RPHO)*TPFR1K
QCA=1./RPHO*2.0*SSLENG
C *****************************************
C SAVE AUXILIARY VARIABLES IF ANY
C SAVE TOTAL LENGTH OF TRUNK SEGMENTS
1490 XL=0.0
   DO 1500 I=1,NODES1
       XL=XL+RLNEG(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)=QCA
   PCTOT=PCTOT+PCH
   RNUM=RNUM+1.0
   PCAVE=PCTOT/RNUM
   STATE(N+5)=PCAVE
   STATE(N+6)=QTOT
   STATE(N+7)=QTA
C
ISEP1=ISEP+1
C TFST PRINT OUT FLAG
1510 IF(IFXN,NE,IFLAG) GO TO 1650
   WRITE(NTYP,9056)TIME
   WRITE(NTYP,9412)
9412 FORMAT(5X,14H NODE Y VALUFS )
      CALL PUTVEC(Y,NODES2)
      WRITE(NTYP,9413)
9413 FORMAT(5X,*ICS,ISEP,INODE,YASEP,YGAPM,AGAP(ISEP),VFEXIT*,
      1 *,QEXIT,PCH,PTK,QFAN*)
      WRITE(NTYP,9055)ICS,ISEP,INODE,YASEP,YGAPM,AGAP(ISEP),
      1VFEXIT,QEXIT,PCH,PTK,QFAN
9055 FORMAT(5X,315,RF12.5)
      NP1=N+1
      NP7=N+7
      IF(.OR.,LDATS(2)) WRITE(NTYP,9050) (STATE(JK),JK=NP1,NP7)
9050 FORMAT(5X,*XL,QT,QTIM,QCA,PCAVE,QTOT,QTA*,/,8F15.5)
C

WRITE(NTYP,9410)
CALL PUTVEC(PEXT,NODES2)
9056 FORMAT(5X,*TIME*,F12.5)
IFLAG=0
1650 IFLAG=IFLAG+1
C
****
C PIOTING ROUTINE
IF(NPLTM,GE,1) CALL PSTORQ(NODES)
C
C RESET DAMPING RATION IF AT EQUILIBRUM TIME
C FIRST CALL, TEST TIME FOR RESET/STEP
IF(TIME,LT,TREST) GO TO 1700
IF(TIP)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
AXVEG(I)=AXS(I)*RLNG(I)/RLNGO(I)
1800 CONTINUE
C
C ********************************************************
C
C DIFFERENTIAL EQUATIONS
C CALLED FOUR TIMES PER STEP
C
C ********************************************************
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)=STATX(I+2)
150 CONTINUE
C SPECIAL NO MOTION EXECUTION
D7 160 I=1,N
160 DFRY(I)=0.0
IF(LOADS(5)) GO TO 1013
C
C *****
D7 50 I=1,MODES
C COMPUTE SPRING LENGTHS
RLEN(I)=SORT((Y(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
SINTHETA(I)=SIN(TX)
COSTHETA(I)=COS(TX)
50 CONTINUE
C
C IF FIRST CALL COMPUTE BENDING STIFFNESS INITIAL CONDITION
IF(NNEW)51,53,55
51 DO 52 I=1,NODES
52 SF(I)=THETA(I+1)-PHI(I)
53 CONTINUE
C
C *****
C COMPUTE FORCES
53 DO 60 I=1,NODES
C FIND SPRING FORCES AT NODE
C RK1,RK2 ARE SPRING FORCE MAGNITUDES FOR 2 ATTACHED SPRINGS, =STIF
C FORCE = SPRING CONSTANT * DELTA LENGTH/LENGTH
RK1=(RLEN(I)-RLEN(I+1))*RKVFC(I)
C FIND DAMPER FORCES AT NODE
J=1*4+1
C FORCE = 2.0 * ZETA * WAVE N * VELOCITY
XZETA=2.0*SORT(R*MAG((I)*RXY(T)/RXY(T))
FCYX=T=STATE(J))*DAMP(I)*XZETA
FCYD(T)=STATE(J+2)*DAMP(I)*XZETA
60 CONTINUE
C COMPUTE PRESSURE FORCES
C PRESSURE X, Y COMPONENTS, FORCE = P*A ACTING ON LENGTH/2 ON SIDES OF NODE
DO 125 T=1, NNODES
FCYPT=T=PRESUR*0.5*(RXY(T)*SIN(T)+RXY(T+1)*SINTHE
1(T+1))
FCYPT=T=PRESUR*0.5*(RXY(T)*COSTHE(T)+RXY(T+1)*COSTHE
1(T+1))
125 CONTINUE
C ZERO BENDING FORCES
DO 129 T=1, NNODES
FCYXR(T)=0.0
FCYBT(T)=0.0
129 CONTINUE
C COMPUTE BENDING FORCES
DO 130 T=1, NNODES
IF(RAVFC(T),0,0) GO TO 130
C COMPUTE ANGULAR DISPLACEMENT FROM EQUILIBRIUM
TX=STFT(T)-(THET(T+1))+PHI(T))
C IF SIE(T) ANGLE INCREASE TORQUE IS NEGATIVE
TX=TX*RAVFC(T)
C COMPUTE LENGTH NORMALIZATION FACTOR
RX1=2.0/(RXY(T)*(RXY(T)+RXY(T+1))))
RX2=2.*/(RLENG(I+1)*(RLENG(I)+RLENG(I+1)))
C IF FIRST NODE SKIIP FORCE AT ATTACHMENT POINT
  IF(I,FO,1) GO TO 132
C COMPUTE BENDING EQUIVALENT FORCES IN X,Y TERMS
  FORCX(I-1)=FORCXY(I-1)+RX1*(-TX)*SINTH(I)
  FORCY(I-1)=FORCXY(I-1)+RX1*TX*COSTH(I)
132 FORCX(I)=FORCXP(I)+RX1*TX*SINTH(I)+RX2*TX*SINTH(I+1)
  FORCY(I)=FORCYP(I)+RX1*(-TX)*COSTH(I)+RX2*(-TX)*COSTH(I+1)
C IF LAST NODE SKIP FORCE AT ATTACHMENT POINT
  IF(I,FO,NODES) GO TO 130
  FORCX(I+1)=FORCXY(I+1)+RX2*(-TX)*SINTH(I+1)
  FORCY(I+1)=FORCYP(I+1)+RX2*TX*COSTH(I+1)
130 CONTINUE
C C COMPUTE FORCES DUE TO EXTERNAL STIFFNESS TERMS
  IF(TEXT,EX,0) GO TO 140
  FORCXM(TEXT)=-RFKFTX*(X(TEXT)-XFTO)
  FORCYM(TEXT)=-RFKFTV*(Y(TEXT)-YFTO)
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,NODES
1001 CONTINUE
  J=I*4+1
  C X VELOCITY
    DERV(J)=(FORCXY(I)+FORCXX(I)+FORCXP(I)+FORCYI(I)+FORCG(I))/RMASS(I)
  C X POSITION
    DERV(J+1)=STATE(I)
  C Y VELOCITY
    DERV(J+2)=(FORCYP(I)+FORCYK(I)+FORCYP(I)+FORCYH(I)+FORCYG(I))/
              1/RMASS(I)
  C Y POSITION
    DERV(J+3)=STATE(J+2)
C
C **CONSTRAINTS ON MEMBRANE**
C CHECK FOR GROUND CONTACT OF TRUNK
1050 DO 2000 I=1, N600
   J=I*4+2
C TEST FOR CONTACT
   IF (STATE(J+2).LT.YMIN) GO TO 2000
C FORCE MAXIMUM ON Y DISPLACEMENT
   STATE(J+2)=YMIN(STATE(J+2),YMIN)
C GROUND CONTACT ENERGY ABSORPTION
C FORCE ONLY IF REBOUND
   DELY(J+1)=MIN(0,DELY(J+1))
   STATE(J+1)=MIN(0,STATE(J+1))
2000 CONTINUE
C RETURN
END
SUBROUTINE TRUNK
C TRUNK SHAPE ITERATION
REAL L,1,1,1,2
COMMON/GEOMET/A,R,MY1,MY,PHI1,PHI2,R1,R2,L1,L2,ISHAp
DATA RTOL,P17/0.01,6.28318/
C
IF(HY)11,11,1
1 R2=SQRT(A**A+0.25+HY*HY)
RH Y=R+Hy
NMAX=100
DO 10 I=1,NMAX
PHI2=AR S(ACOS(AMAX1(-1.0,AMIN1(1.0,(R2-HY)/R2))))
SINPH2=STN(PHI2)
R1=((A-R2*STNPH2)**2+(BHY*RHY))/(BHY+BHY)
PHI1=ABS(ACOS(AMAX1(-1.0,AMIN1(1.0,(R1-HY-B)/R1))))
XS=A-R2*STNPH2
IF(XS)5,5,6
5 PHI1=PHI2-PHI1
6 L2=PHI1*R1
IF(ABS(PHI2).LT.1.0F-2) PHI2=1.0F-2
R2S=0.2/PHI2
IF(ABS(R2-R2S).LE.RTOL) GO TO 50
R2=(R2+R2S)*0.5
10 CONTINUE
C
C ERROR RETURN
11 TSHAPE=0
C
C GOOD RETURN
50 L1=L-L2
ISHAPE=1
RETURN
END
SUBROUTINE PULLVEC(A,m)
C SUBROUTINE FOR VECTOR OUTPUT
DIMENSION A(m)
COMMON/OLIST/NYP, NT, P, I, TAPEN
WRITE (5X, 9010) (A(1), 1=1, m)
9010 WRITE (NYP, 9010) (A(1), 1=1, m)
9000 RETURN
END
SUBROUTINE PLOTTER
C PLOTTING CONTROL SUBROUTINE
DIMENSION IDUM(5)
COMMON/PLOT/NPLT(10),TPLSR,TPLSTP,NPLT,NPLOT,XPLTX(5,200)
COMMON/IOLIST/NTYP,NINP,IPTAPE
C
C MARK END OF PLOT DATA FILE
ENDFILE IPTAPE
DO 1000 IJK=1,NPLT
WRITE(NTYP,9100)
9100 FORMAT(141)
C READ PLOTTER CONTROL CARD
READ(NINP,9010) NVPLT,(NPLT(I),I=1,5)
READ(NINP,9060) TPLSR,TPLSTP,NPLOT,XMIN,XMAX
9060 FORMAT(6F12,5)
DO 1 I=1,5
1 IDUM(I)=NPLT(I)
CONTINUE
WRITE(NTYP,1020)(IDUM(I),I=1,5)
1020 FORMAT(5X,21H PLOTS 1-5 = STATES= ,5I4,/) 
9010 FORMAT(11,4X,512)
IF(NVPLT.EQ.0) GO TO 1000
IF(NVPLT.GT.5) NVPLT=5
C CALL PACKER TO READ REQUESTED VARIABLES INTO XPLTX
CALL PACKER(I)
C
IF((XMIN.NE.0,0).AND.,(XMAX.NE.0,0)) GO TO 100
XMAX=XPLTX(1,1)
XMIN=XPLTX(1,1)
DO 10 K=1,NVPLT
10 DO 10 J=1,I
IF((K.EQ.1).AND.,(J.EQ.1)) GO TO 10
C FIND MINIMUM AND MAXIMUM OF DATA
XMAX=A.MAX1(XMAX,XPLTX(K,J))
10 XMIN=A.MIN1(XMIN,XPLTX(K,J))
IF(XMAX.EQ.XMIN) XMAX=XMAX+1.0
XMAX=XMAX
CALL PRINT TO PUT DATA ON PRINTER.

X0=MAX+0.01*(MAX-MIN)
X1=MIN+0.01*(MAX-MIN)

CALL PRINT TO PUT DATA ON PRINTER.

1000 CONTINUE
2000 RETURN
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),TPLSTP,TPLSTP,NPLTV,DTPLT1,XPLTX(5,200)
1+DIM,NPLTV,XMAX,MAX,STORF
COMMON/IDISI/NYPE,NIP,IPTAPE
C READ TAPE 4 TO START OF SIMULATION
READ(4,TITL0)=0,0
ISTOP=0
I=0
N=MAX=200
C READ TIME STEP VARIABLES AND TIME
10 READ(IPTAPE),(X(IJK),IJK=1,NSTORF),XTIME
C IF END OF FILE STOP READING
20 CONTINUE
C IF TIME .LT. PLOT TIME READ NEXT
IF((XTIME,L.T.,TPLSTP)) GO TO 10
C IF TIME .GT. PLOT TIME RETURN
60 IF(XTIME,G.T.,TPLSTP) RETURN
C IF TIME SINCE LAST POINT EQUAL OR POLE SAVE
IF(XTIME.EQ.,(TIMOLD+DTPLT1)) GO TO 30
GJ=10
100 ISTOP=1
30 TIMOLD=XTIME
I=I+1
70 J=J+1,NPLOT
C LOAD APPROPRIATE CURVE WITH DATA
C NPLT(J) CONTAINS STATE NUMBER TO BE PLOTED
NJ=NPLT(J)
50 XPLTX(J,I)=X(NJ)
C IF ARRAY FULL RETURN
IF((I,L.T.,NPAX),AND,(ISTOP.EQ.,0)) GO TO 10
RETURN
END
SUBROUTINE PRNTPT(NPNTS)
C PRINT PLOT PROGRAM
C
INTEGER LINE, XFIG, BLANK, DOT, T, CHAR
INTEGER SOFF
DIMENSION LINE(120), XFIG(5)
C
DIMENSION XPLAB(11)
COMMON /PLTN/PN(10), IPLSRT, IPLSTP, NPLTM, DPTLOT, XPLOTX(5,200)
I NDM, NCURVE, XMIN, XMAX, NMAX, NSTORE
COMM/ULISI/NTYP, NINT, IPTAPF
C
C SKI PLOT CURVE SYMBOLS
DATA BLANK, DOT/4H , 6H, /
DATA XFIG/4H* , 4MX, 4H+, 4HD, , 4HS /
DATA T/4HT /
DATA SOFF/4H* /
C
C ROUTINE WILL PRINT 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.NMAX) NPNTS=NMAX
IF (NPNTS.LE.0) GO TO 300
IF (NCURVE.LE.0) GO TO 300
WRITE (NTYP,1005) XMIN, DX, XMAX
1005 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, 16H CURVE MARKERS=) 1=* 2=x 3=+ 4=0 5=8 ,
  1 12H =OFF SCALE ,/) 
DO 5 I=1, 11
  5 XPLAB(I)=XMIN+DX*FLOAT(I-1)*10, 0
  WRITE (NTYP,9000)(XPLAB(I), I=1,11)
9000 FORMAT(1X, 11G10.3)
C LOAD LINK WITH BLANKS AND SET MARKERS
DO 10 I=2, 101
  10 LINE(I)=DOT
LINE(1) = 1
LINE(102) = 1
C PRINT FIRST MARKER LINE
*WRITE (4, 1001) (LINE(I), I = 1, 102)
1001   DO I = 5X, 102A, 10, 1
   WRITE = TP1 .SHR
   WRITE = (TP1 - TP1 .SHR) / FLOAT(NPNTS) * 10.0
   I = 0
C PLOT ARRAY OF CURVES
DO 100 J = 1, NPNTS
DO 20 I = 2, 119
20   LINE(I) = BLANK
C CLEAN LINE TO BLANKS
C SET BOARDER FOR PLOT
LINE(1) = DOT
LINE(102) = 0
C FOR EACH CURVE SET UP LINE
DO 200 J = 1, CURVE
   VAL = XPLTN(J)
   CHAR = XFIG(J)
   KDEX = FIX(FIXVAL - 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 IF LINES POINTS POSITION IN LINE
26   LINE(KDEX) = CHAR
200   CONTINUE
C PRINT LINE OF PLOT
   IT = 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
   IF(LINE(I), FU, BLANK) LINE(I) = DOT
45   CONTINUE
**Problem**

```fortran
  WRITE(NTYP,1001)(LINE(K),K=1,102),XTIME
  XTIME=XTIME+DTIME
  GO TO 100
30   CONTINUE
   WRITE(NTYP,1001)(LINE(K),K=1,102)
100  CONTINUE
   DO 33 IJK=2,101
33   LINE(IJK)=DOT
   WRITE(NTYP,1001)(LINE(K),K=1,102),TPLSTP
   RETURN
300  WRITE(NTYP,1002)
1002  FORMAT(5X,45H ERROR CONDITION ON PLOT CONTROL NUMBERS)
   RETURN
END
```
SUBROUTINE PSTORE(NODES)
COMMON/IDLIST/NNT, NNP, IPTAPE
COMMON/PLT/NPLT(10), TPLSHA, TPLSHA, NPLT, DTPLSHA, XPLOX(5, 200)
I, NNP, NPLT, XM, XMAX, XMMP, NSTORE
COMMON TIME, DTIME, STATE(100), DERY(100), STIME, FTIME, NOUT, IF=RT, N,
IPR, ICD, ICN, TNEXT, PNFXT, THACT
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
PROGRAM FMAEVEC (INPUT, OUTPUT, TAPF5=INPUT, TAPF6=OUTPUT)

REAL L9, L1, L2
REAL MASS
INTEGER FVFC

DIMENSION RLFLNG(21), RLFLNGO(21)
DIMENSION RLFLNGO(21)
DIMENSION DAMP(21), X(22), Y(22)
DIMENSION THFTA(21)
DIMENSION FVFC(100)
DIMENSION WORK1(40), WORK2(40)
DIMENSION PKVEC(21)
DIMENSION MASS(20)
DIMENSION FSTIF(4, 4)
DIMENSION COSTHE(21), SINTHE(21)
DIMENSION ICNTL(10)
DIMENSION SINALP(40),COSALP(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, HY, RL, L1, L2, P1, R2, PH11, PH12

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
C STORAGE EQUIVALENCY MAPPING
C AT = WORKM = XT
C A = XW = XZ
C AS = G = GS
C
C*********************************************************************************************
C
C INPUTINITIAL DATA
1 CONTINUE
   NITP=6
   WINP=5
C NMAT IS STORAGE ARRAY LIMIT
   NMAT=40
   NMATV=21
   NUMAX=NMAT/2
   NODAX=NMAT/4
   PI=3.141592
   PI02=1.570796
GO=37.174
WRITE(NYTP,9003)
9003 FORMAT(1H1)
9002 FORMAT(/1/
9050 FORMAT(5Y,10G17.5)
9001 FORMAT(/1/
C
C CLEAR MATRICES
CALL CLEAR(MASS,NMAX,1)
CALL CLEAR(FKANL,NMAX,1)
CALL CLEAR(DAMP,NMAX,1)
CALL CLEAR(RLENGTH,NMAX,1)
C
C INPUT CONTROL INFORMATION
READ(NINP,9015)(ICNTIL(I),I=1,10),(IPRNT,J),J=1,10
9015 FORMAT(2011)
C
C ICNTL 1 = 1,FIELDVALUES 2,EIGENVECTORS
C ICNTL 1 = COMPUTE 2 DIMENSIONAL FIELDVALUES
C ICNTL 2 = COMPUTE 2 DIMENSIONAL DAMPED FIELDVALUES
C ICNTL 3 = COMPUTE TRANSVERSE STRING UNDAMPED EIGENVALUES
C ICNTL 4 = COMPUTE TRANSVERSE DAMPED EIGENVALUES
C ICNTL 5 = COMPUTE LONGITUDINAL UNDAMPED EIGENVALUES
C ICNTL 6 = COMPUTE LONGITUDINAL 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=NOOFS+1
NODES2=NOOFS+2
MAXN=NOOFS+2
MAXP=MAXN+2
WRITE(NYTP,9051) NODES
9051 FORMAT(5X,8H NODES = ,I3,/
READ (NINP,9000)IS,PL,AX,EX,TENS
1,MY
WRITF(NINTP,9056)
9056 FORMAT(5X,12H LENGTH 0 ,12H LENGTH 1C ,12H A POINT ,12H A PO)
IINT ,12H TENSION ,3H HY )
WRITF(NINTP,9050)I.S.AX.BY.TE SN
1,HY
WRITF(NINTP,9001)
C
C INPUT MASS IN POUNDS
READ(NINTP,9000)(MASS(I),I=1,NODES)
C INPUT MEMBRANE STIFFNESS (L/FT
READ(NINTP,9000)(RKVEC(I),I=1,NODES1)
C INPUT MEMBRANE DAMPING (L.sec/FT
READ(NINTP,9000)(DAMP(I),I=1,NODES)
C
C CHANGE MASS TO SLUGS
DO 5 I=1,NODES
5 CONTINUE
C
C SET EQUIL SPACED ELEMENT LENGTHS
DO 6 I=1,NODES1
6 CONTINUE
C
C RLengo SPACING OPTION
READ(NINTP,9015) TXF
C INPUT RLengo IF TXF = 1
IF(TXF.EQ.1) READ(NINTP,9000)(RLengo(T),I=1,NODES1)
C
C NORMALIZE SPRING CONSTANTS
DO 7 I=1,NODES1
7 CONTINUE
C
C TRANSFORM DAMPING RATION INTO DAMPER VALUE
DO 8 I=1,NODES
8 CONTINUE
C PRINT INITIAL VECTORS
  WRITE(NTYP,9053)
  9053 FORMAT(SY,11H MASS NDOFS)
  CALL PUTMAT(MASS,NDOFS,NMAT,1)
  WRITE(NTYP,9054)
  9054 FORMAT(SY,20H SPRING COEFFICIENTS )
  CALL PUTMAT(KP,VEC,NDOFS,NMAT,1)
  WRITE(NTYP,9055)
  9055 FORMAT(SY,13H NODE DAMPING )
  CALL PUTMAT(DAMP,NDOFS,NMAT,1)
  WRITE(NTYP,9057)
  9057 FORMAT(SY,OH LENGTH 0 )
  CALL PUTMAT(RL=1D0,NDOFS1,NMAT,1)
  C
  *****************************************************************************
  C
  C SET BOUNDARY NDOFS
  X(1)=0.0
  Y(1)=0.0
  Y(NDOFS2)=-AX
  X(NDOFS2)=AX
  C
  C X,Y POINT OPTION
  READ(NINP,9015)IXY )
   IF(IXY.EQ.1) GO TO 17
  C
  C INPUT NODE COORDINATES FOR INITIAL STIFFNESS CALCULATION
  16 READ(NINP,9000)(X(I),I=2,NDOFS1)
  READ(NINP,9000)(Y(I),I=2,NDOFS1)
  9000 FORMAT(AG10.5)
   GO TO 45
  C
  C COMPUTE TRUNK SHAPE
  17 CALL TRUNK(ISHAPF)
    NX=FIX(FLAT(NODES)*(PHI2*R2/PL))
    NZ=NDOFS-NX
  C
C COMPUTE RIGHT SECTOR POINTS
  XCNTR=R2*STN(PHI2)
  YCNTR=HY=R1
  TX=2.0*ASIN(RLNGD01)*0.5/R1
  ANG=ATAN2((X(NODES2)-XCNTR),(Y(NODES2)-YCNTR))
  DO 19 I=1,N
  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=XCNTR+R1=R2
  ANG=1.570796-PHI2
  TX=2.0*ASIN(RLNGD01)*0.5/R2
  DO 19 I=1,N
  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
  WRITE(NIYP,9010)
9010 FORMAT('///,5x,15H NODE POSITIONS,,7X,2H X,,10X,2H Y ,/')
  DO 20 I=1,NODES2
  WRITE(NIYP,9020) X(I),Y(I)
9020 FORMAT(2(2X,F8.4))
20  CONTINUE
WRITE(NIYP,9001)
C
DO 50 I=1,NODES1
C COMPUTE SPRING LENGTHS
  RLNGD01=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
COSTHE(I)=COS(TX)
SINTHE(I)=SIN(TX)
GO CONTINUE
C
C COMPUTE LOCAL SLOPE ANGLE, ALPHA
DO 55 I=1,NODES
TX=ATAN2((Y(I+2)-Y(I)),(X(I+2)-X(I)))
SINALP(I)=SIN(TX)
COSALP(I)=COS(TX)
55 CONTINUE
C
***********************************************************************
C
C CONTINUOUS STRING FREQUENCIES
C NOTE : ONLY GOOD FOR EQUAL SPACED NODES, PLEASE USE STRING LATERAL FREQU
C
RVASt=0.0
DO 70 I=1,NODES
RVASt=RVASt+MASS(I)
70 CONTINUE
WRITE(6,9052)
9052 FORMAT(/,5X,30H CONTINUOUS STRING FREQUENCIES ,3H HZ,/) DO 80 I=1,NODES
W=FINTAT(I)*SORT(TENSNR/(RVASt)/(PL*2.0)) WRITE(6,9050)W
80 CONTINUE
C
*****************************************************************************
C
C 2 DIMENSIONAL UNDAMPED MATRIX MODEL.
C IF(INTRL(I),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)
N=0
C
DO 2000 IJK=1,NODES1
C ADD DIRECT STIFFNESS SUBMATRIX INTO GLOBAL MATRIX G
C
CALL ELEMK(ESTTF(1,1),THETA(TJK),=PKVEC(IJK))
C
DO 100 I=1,4
DO 100 J=1,4
TK=I+K-2
TF((IK,LT,1),DR,(IK,GT,NMAX)) GO TO 100
JK=J+K-2
TF((JK,LT,1),DR,(JK,GT,NMAX)) GO TO 100
WORK(IK,JK)=WORKM((IK,JK)+ESTTF(T,J))
100 CONTINUE
K=K+2
2000 CONTINUE
C
C ADD TENSION SPRING EFFECT MODEl INTO STIFFNESS MATRIX
DO 450 I=1,NODFS
J=(I-1)*7+1
TX=TENS*I2.0/(RLENG(T)+RLENG(I+1))*SINALP(T)
TY=TENS*I2.0/(RLENG(T)+RLENG(I+1))*COSALP(T)
C
TF(I,EQ0.1) GO TO 440
G(J,J-2)=G(J,J-2)+TX*SINAP(I-1)
G(J,J-1)=G(J,J-1)+TX*COSALP(I-1)
G(J+1,J-7)=G(J+1,J-7)+TY*SINAP(T-1)
G(J+1,J-1)=G(J+1,J-1)+TY*COSALP(T-1)
C
440 G(J,J)=G(J,J)+TX*SINAP(T)-2,0
G(J,J+1)=G(J,J+1)+TX*COSALP(T)-2,0
G(J+1,J)=G(J+1,J)+TY*SINAP(T)-2,0
G(J+1,J+1)=G(J+1,J+1)+TY*COSALP(T)-2,0
C
TF(I,ED,NODES) GO TO 450
G(J,J+7)=G(J,J+7)+TX*SINAP(T+1)
G(J,J+9)=G(J,J+9)+TX*COSALP(T+1)
G(J+1,J+2)=G(J+1,J+2)+TY*SINAP(T+1)
G(J+1,J+1)=G(J+1,J+3)+TY*COSALP(I+1)
450 CONTINUE
C SUM LONGITUDINAL AND LATERAL FORCE COMPONENTS
DO 420 I=1,NMAX
K=(I-1)/7+1
DO 420 J=1,NMAX
G(I,J)=(G(I,J)+WORKM(I,J))/MASS(K)
420 CONTINUE
CALL MOVF(G,WORKM,NMAX,NMAT)
C WRITF(6,9101)
9101 FORMAT(/,20H UNDAMPED MATRIX,/) IF(IPRNT(1),EQ,1) CALL PUMAT(G,NMAX,NMAT,NMAX)
C COMPUTE EIGENVALUES/EIGENVECTORS OF 2D MATRIX
CALL EIGPAC(NMAX,NMAT,G,WORK1,WORK2,FVEC,+?,ICNTL(1),WORKM,XW)
C**********************************************************************
C 2 DIMENSIONAL DAMPED MEMBRANE MATRIX MODEL
C 202 IF(ICNTL(2),LE,0) GO TO 615
C A IS DAMPED 2*N STATE COEFFICIENT MATRIX
IF(NODES,GT,NMAX) GO TO 615
CALL CLEAR(A,NMAT,NMAT)
CALL CLEAR(AS,NMAT,NMAT)
CALL CLEAR(AT,NMAT,NMAT)
I0=1
J0=1
C SET DAMPING AND PURE INTEGRATOR COEFFICIENTS
DO 1010 I=1,NODES
A(I0,J0)=DAMP(I)
A(I0+1,J0)=1.0
A(I0+3,J0+2)=1.0
A(I0+2,J0+2)=DAMP(I)
DO 1020 I=1,NODES1
   JDM=(I-1)*4+5
C
C ADD STIFFNESS TERMS FOR ALL SPRINGS
   DO 1030 J=1,4
       DO 1030 K=1,4
           L=(J*2)+10
           IF((L,L.T.1),OR,(L,GT,NMAX2)) GO TO 1030
           W=(K*2)+10+1
           IF((L,LT.1),OR,(L,GT,NMAX2)) GO TO 1030
           CALL ELEM(ESTIF(J,1),THETA(T),PKVEC(T))
           DO 1030 J=1,4
               DO 1030 K=1,4
                   L=(J*2)+10
                   IF((L,L.T.1),OR,(L,GT,NMAX2)) GO TO 1030
                   W=(K*2)+10+1
                   IF((L,LT.1),OR,(L,GT,NMAX2)) GO TO 1030
                   A(L,W)=A(L,W)+ESTIF(J,1)
       C
   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/(RLEN(T)+RLEN(I+1))*STNALP(I)
       TY=TENSN*2.0/(RLEN(T)+RLEN(I+1))*COSALP(T)
   C
       IF(I,EO,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(T-1)
           A(J+2,J-3)=A(J+2,J-3)+TX*STNALP(T-1)
           A(J+2,J-1)=A(J+2,J-1)+TY*COSALP(T-1)
       1022 A(J,J+1)=A(J,J+1)+TX*STNALP(T) *(-2,0)
           A(J,J+3)=A(J,J+3)+TY*COSALP(T) *(-2,0)
           A(J+2,J+1)=A(J+2,J+1)+TX*STNALP(T) *(-2,0)
           A(J+2,J+3)=A(J+2,J+3)+TY*COSALP(T) *(-2,0)
   C
       IF(I,EO,NODES) GO TO 1025
           A(J,J+5)=A(J,J+5)+TX*STNALP(T+1)
A(J,J+7)=A(J,J+7)+TY*COSALP(T+1)
A(J+2,J+5)=A(J+2,J+5)+TY*SINALP(T+1)
A(J+2,J+7)=A(J+2,J+7)+TY*COSALP(T+1)

CONTINUE
DO 1040 I=1,NMAX?,2
K=T/4+1
DO 1040 J=1,NMAX?
A(I,J)=A(I,J)/MASS(K)

CONTINUE
CALL MOVE(A,AS,NMAX2,NMAT)

WRITE(6,9102)

9102 FORMAT(*/17H 2D DAMPED MATRIX ,/)
IF(IPRT(2),FO,1) CALL PUTMAT(A,NMAX?,NMAT,NMAX2)

C COMPUTE EIGENVALUES/EIGENVECTORS OF 2D MATRIX
CALL EIGPAC(NMAX2,NMAT,A,WORK1,WORK2,EVER,-2,ICNTL(2),AS,AT)

C**************************************************************************
C UNDAMPED LATFRAL STRING MODEL
C 615 IF(ICNTL(3),LE,0) GO TO 700
C GS HOLDS STRING SYSTEM MATRIX
IF(NDOFS,GT,(2*NMAX)) GO TO 800
CALL CLEAR(GS,NMAT,NMAT)
CALL CLEAR(XZ,NMAT,NMAT)
CALL CLEAR(XT,NMAT,NMAT)
NDEM1=NDOFS-1

DO 630 I=2,NDOFM1
C COMPUTE MEMBRANE TENSION
TX=TENSN/MASS(I)/RLENGN(T)
GS(I,I)=2.0*TX
GS(I,I+1)=TX
GS(I,I-1)=TX

630 CONTINUE
C SET FIRST AND LAST BOUNDARY ELEMENTS
TX=TENSIN/MASS(1)/RLENGO(1)
GS(1,1)=2.0*TX
GS(1,2)=TX
TX=TENSIN/MASS(NODES)/RLENGO(NODES)
GS(NODES,NODES)=2.0*TX
GS(NODES,NODES+1)=TX
CALL MOVE(GS,XZ,NODES,NMAT)
C WRITE(6,9103)
9103 FORMAT(/,15H LATERAL STRING ,/
IF(IPNTE(3),EQ,1) CALL PUTMAT(GS,NODES,NMAT,NODES)
C COMPUTE EIGENVALUES/EIGENVECTORS OF 2D MATRIX
CALL EIGPAC(NODES,NMAT,GS,WORK1,WORK2,FVFC,+1,ICNTL(3),XZ,AT)
C *******************************************************************
C DAMPED LATERAL STRING MODEL.
C 700 IF(ICNTL(4),IE,0) GO TO 800
C XW HOLDS DAMPED STRING SYSTEM A MATRIX
IF(NODES,GT,(2*NDMAT)) GO TO 800
CALL CLEAR(XW,NMAT,NMAT)
CALL CLEAR(G,NMAT,NMAT)
CALL CLEAR(OMKM,NMAT,NMAT)
C SET PURE INTEGRATOR ELEMENTS
DO 710 I=1,NODES
J=I+1
K=I+2
XW(J,K)=1.0
710 CONTINUE
C LOAD TENSILE TERMS
NODEM1=NODES+1
DO 770 I=2,NODEM1
CALL CLEAR(XZ, NWAT, NWAT)
CALL CLEAR(XT, NWAT, NWAT)
NODES = 1, NODFS = 1

C
C SET UP STIFFNESS MATRIX
DO 220 I = 2, NODFS
   GS(I, 1) = RKVEC(I) - RKVEC(I + 1)
   GS(I, I - 1) = RKVEC(I)
   GS(I, I + 1) = RKVEC(I + 1)
220 CONTINUE

C SET FIRST AND LAST BOUNDARY NODES
GS(1, 1) = RKVEC(1) - RKVEC(2)
GS(1, 2) = RKVEC(2)
GS(NODFS, NODFS) = RKVEC(NODFS) - RKVEC(NODES1)
GS(NODFS, NODFS) = RKVEC(NODFS)

C
DO 330 I = 1, NODFS
330 CONTINUE

C WRITE(6, 9105)
9105 FORMAT(/, 20H LONGITUDINAL STRING ,/)
WRITE(5, 9015) CALL PUTNWAT(GS, NODFS, NWAT, NODFS)

C
C COMPUTE EIGENVALUES/EIGENVECTORS OF 2D MATRIX
CALL EIGPAC(NODES, NWAT, GS, WORK1, WORK2, EVFC, +1, ICNTL(5), XZ, XT)

C
C******************************************************************************

C DAMPED LONGITUDINAL BAR MODEL
C
850 IF (ICNTL(6), LE, 0) GO TO 900

C IF (NODFS, GT, (2 * NDWAX)) GO TO 900
CALL CLEAR(G, NWAT, NWAT)
CALL CLEAR(WORKM, NWAT, NWAT)
CALL, CI, FAP(XW, NMAT, NMAT)

C
C SET PURE INTEGRATOR ELEMENTS
DO 860 I=1, NODFS
   J=I+2
   K=I+2
   XW(J,K)=1.0
   CONTINUE
860
   NODFS=1=NODFS

C
C SET UP DAMPED STIFFNESS MATRIX
   DO 870 I=2, NODFS+1
      J=I+2
      XW(J,J)=-DAMP(T)
      XW(J,J-1)=RKVEC(I)
      XW(J,J+1)=RKVEC(I+1)
      CONTINUE
870

C
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=NODFS+2-1
   XW(J,J)=-DAMP(NODFS)
   XW(J,J-1)=RKVEC(NODFS)
   XW(J,J+1)=RKVEC(NODFS)=RKVEC(NODFS+1)

C
DO 880 I=1, NMAX, 2
DO 880 J=1, NMAX
   K=I/2+1
   XW(I,J)=XW(I,J)/MASS(K)
   CONTINUE
880

WRITE(6, 9106)
9106
FORMAT(/, 2TH LONGITUDINAL DAMPED STRING, /)
IF(IPRNT(6), E), 1) CALL PUTMAT(XW, NMAX, NMAT, NMAX)
COMPUTE EIGENVALUES/EIGENVECTORS OF 2D MATRIX
CALL EIGAP(MAX, MAT, XW, WORK, ICNTL(6), G, WORK)

CALL EXIT
SUBROUTINE TRUNK(ISHAPE)
C TRUNK GEOMETRY CALCULATIONS
C
REAL L,L1,L2,L5,L6,IP,MASS
COMMON/GEOVET/A,R,HY,L,L1,L2,R1,R2,PHI1,PHI2
DATA RTOL/0.01/
C
IF(HY,IP,0,0) GO TO 11
C*******************************************************************************
C ITERATION FOR R2
C COMPUTE INNER RADIUS OF CURVATURE
R2=SQRT(A*A*0.25+HY*HY)
C
C ITERATION LOOP FOR L2,L1,R1,R2
DO 10 I=1,50
PHI2=ARS(ACOS(A*M(1.0,0,A*M(1.0,(R2-HY)/R2))))
SINPH2=SIN(PHI2)
C COMPUTE OUTER RADIUS OF CURVATURE
R1=(A-R2*SQNP(H2)**2+(R+HY)**2)/(2.*(R+HY))
PHI1=ARS(ACOS(A*M(-1.0,0,A*M(1.0,(R1-HY-8)/R1))))
XS=A-R2*SQNP(H2)
IF(XS,IE,0,0) PHI1=6.2831852-PHI1
L2=L-PHI1*R1
C R2S IS RESULTANT RADIUS FOR COMPUTED L2 IN ITERATION
IF(ARS(PHI2),1.1,0.E-2) PHI2=1.0E-2
R2S=L2/PHI2
C TEST IF TOLERANCE GT ERROR
IF(ARS(R2-R2S),0,E,RTO1) GO TO 50
R2S=(R2+R2S)**0.5
10 CONTINUE
C*******************************************************************************
C ITERATE 50 TIMES WITHOUT SUCCESS,ERROR RETURN
11 CONTINUE
WRITE(6,4001)
4001 FORMAT(10X,26H INFFASIBLE TRUNK GEOMETRY,/) ISHAPE=0 RETURN
C TRUNK (K,RETURN
SUBROUTINE MOVE( A, AS, N, M )
C GENERAL MATRIX MOVE ROUTINE
DIMENSION A(M,N), AS(M,N)
DO 10 T=1,N
DO 10 J=1,N
AS( I, J ) = A( T, J )
10 CONTINUE
RETURN
END
SUBROUTINE CMINV(A,N,D,L,M)
C COMPLEX M AthX INVERSION
C
COMPLEX A,N,D,RIGA,HOLD
DIMENSION A(1),L(1),M(1)
C
LΚ=Ν+Ν
Ν=COMPLEX(1.0,0.0)
ΝΚ=-Ν
ΟΟ 80 K=1,N
ΝΚ=ΝΚ+Ν
Κ(Κ)=Κ
Μ(Κ)=Κ
ΚΚ=ΝΚ+Κ
RIGA=A(ΚΚ)
ΟΟ 20 J=Κ,N
ΙΖ=Ν*(J-1)
ΟΟ 20 T=Κ,N
ΤΖ=ΤΖ+Ι
10 IF(CARS(RIGA)=CARS(A(IJ))) 15,20,20
15 RIGA=A(IJ)
Κ(Κ)=Ι
Μ(Κ)=Ι
20 CONTINUE
C
C INTERCHANGE ROWS
C
Κ=Κ(J)
ΤΦ(J=Κ) 35,35,25
25 ΚΙ=Κ=N
ΟΟ 30 T=1,N
ΚΙ=ΚΙ+Ν
ΗΟΙ.Ο=Α(ΚΙ)
ΚΙ=ΚΙ-Κ+Ι
Α(ΚΙ)=Α(ΙΙ)
30 Α(ΙΙ)=ΗΟΙ.Ο
C
C INTERCHANGE COLUMNS
35 I=H(K)
TF(I=K) 45, 45, 3R
3A JP=N*(I-1)
D 40 I=1,N
IK=NP+J
JI=JP+J
HOLD=A(IK)
A(IK)=A(IJ)
40 A(IJ)=HOLD

DIVIDE COLUMN BY MINUS PIVOT (VALUE OF PIVOT ELEMENT IS
CONTAINED IN RICA)

45 TF(RIGA) 4R, 46, 4R
46 D=CMPLX(0,0,0,0)
RETURN
4A DO 55 I=1,N
TF(I=K) 50, 55, 50
50 IK=NP+I
A(IK)=A(IK)/(RIGA)
55 CONTINUE

REDUCE MATRIX

DO 65 I=1,N
IK=NP+I
HOLD=A(IK)
JI=I-N
DO 65 J=1,N
TJ=I-J+N
TF(I=K) 60, 65, 60
60 TF(J=K) 62, 65, 62
62 K=I,J=T+K
A(IJ)=HOLD*A(KJ)+A(IJ)
65 CONTINUE

DIVIDE ROW BY PIVOT
C
KJ=K=N
00 75 J=1,N
KJ=KJ+N
IF(J=K) 70,75,70
70 A(KJ)=A(KJ)/BICA
75 CONTINUE
C
PRODUCT OF PIVOTS
D=D*BICA
C
REPLACE PIVOT BY RECIPROCAL
A(KK)=CMPLX(1,0,0)/BICA
80 CONTINUE
C
FINAL ROW AND COLUMN INTERCHANGE
C
K=N
100 K=(K-1)
   IF(K) 150,150,105
105 T=L(K)
   IF(I=K) 120,120,108
108 J2=N*(K-1)
   JR=N*(I-1)
   DO 110 J=1,N
   JK=JO+J
   HOLD=A(JK)
   JI=JR+J
   A(JK)=A(JI)
110 A(JI)=HOLD
120 J=M(K)
   IF(J=K) 100,100,125
125 K1=K=N
   DO 130 I=1,N
   KI=KI+N
   HOLD=A(KI)
   JI=KI*K+I
   130 CONTINUE
130 \texttt{A(KI)} = \texttt{A(JT)}
150 \texttt{GO TO 100}
150 \texttt{RETURN}

SUBROUTINE ELMK(X,A,STIF)
C ELEMENT STIFFNESS MATRIX FORMULATION (2D)
C A IS LOCAL ROTATION ANGLE
DIMENSION X(4,4)
C
C COMPUTE TRIGONOMETRICALS 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 MULTPLY ROTATION MATRIX BY STIFFNESS
DO 10 J=1,4
DO 10 I=1,4
X(I,J)=X(I,J)*STIF
10 CONTINUE
C
RETURN
END
SUBROUTINE CLEAR(A,N,M)
C GENERAL MATRIX CLEAR TO ZERO
DIMENSION A(1)
M = N
0) 100 I=1,NM
A(I) = 0.0
100 CONTINUE
RETURN
END
SUBROUTINE PUTMAT(A,M,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
75 WRITE(6,9002)(A(I,J),J=1,L)
WRITE(6,9001)
75 CONTINUE
100 CONTINUE
9001 FORMAT(1)
9002 FORMAT(5X,10G12.5)
RETURN
END
SUBROUTINE PUTFG(X,Y,N,IJK)
C FICENVALUE OUTPUT ROUTINE
DIMENSION X(1), Y(1)
C
WRITE(6,9005)
9005 FORMAT( //, 10H FREQUENCY OF MODES )
IF(IJK) 2,1,1
C DNF 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(ARS(X(I)), LE, 1.0E-05) X(I)=0.0
IF(ARS(Y(I)), LE, 1.0E-05) Y(I)=0.0
50 CONTINUE
C
DO 100 I=1,N
C W=SORT(RE**2+IM**2)
W=SORT(X(I)**2+Y(I)**2)
C IF NO DAMPING TAKE SORT
IF(IJK,GT,0,0) W=SORT(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(5(5X,C12.,5))
100 CONTINUE
RETURN
END
SUBROUTINE EIGPAC(V,M,A,WORK1,WORK2,FWFC,IJK,IFLAG,B,C)
C ROUTINE TO CALL SSP ROUTINES FOR EIGENVALUES
C N IS ARRAY SIZE TO DO, M IS STORAGE SIZE
C A,A CONTAIN SAME MATRIX
C WORK1,WORK2,C ARE WORK AREAS
INTEGER FWFC
DIMENSION A(1),WORK1(1),WORK2(1),EVFC(1)
DIMENSION R(1),C(1)
C FORM HESSENBURG UPPER ALMOST TRIANGULAR MATRIX
20 CALL HSSBGN(M,N,A)
C COMPUTE EIGENVALUES WITH OR ALGORITHM
CALL ATFIG(N,A,WORK1,WORK2,EVFC,M)
C
CALL PUTFIG(WORK1,WORK2,N,IJK)
IF(JABS(IFLAG)=2) 100,25,100
C CALL EIGENVECTOR ROUTINE IF NEEDED
25 CALL VFPCAC(N,M,R,WORK1,WORK2,A,C)
100 RETURN
END
SUBROUTINE EIGVEC(A,VEC,Y,N,N,M)
C ROUTINE TO COMPUTE EIGENVECTORS
C SOLVES FOR COMPLEX EIGENVECTORS OF A REAL MATRIX
C A,WORK,WTM ARE ALL FUNCTIONS OF VEC PAC ARRAY SIZE
C COMPLEX X(Y),Y(M),VEC(N)
C COMPLEX A(40,40)
C COMPLEX WORK(40)
C COMPLEX WTM(1600),DET
C
IF(N,LE,1) RETURN
9001 FORMAT(B5,Y,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)
100 CONTINUE
IF(NM1=1) 700,200,300
200 MAT(1)=CMPLX(1.0,0.0)/MAT(1)
GO TO 400
C
C MATRIX INVERSION
300 CALL CMINV(MAT,NM1,DET,X,Y)
IF(DET,F0,CMPLX(0.0,0.0)) WRITE(6,9000)
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(I,N)*MAT(K)
500 CONTINUE
RETURN
END
SUBROUTINE VECPRAC(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 (N*2)
COMPLEX X(R1),Y(R1), 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 T=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 T=1,N
AC(I,I)=AC(I,I)-Y(K)
100 CONTINUE
C
C CALL EIGENVECTOR SOLVER
CALL EVECTR(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,M
9000 FORMAT(//,5X,11H EIGENVALUE ,2X,2F15.5, //,5X,14H EIGENVECTOR
1,2F15.5)
C NORMALIZE REAL PARTS OF DISPLACEMENT EIGENVECTORS
XMAX=0.0
DO 200 I=1,N
IF(AABS(REAL(Y(I))).LT.XMAX) GO TO 200
XMAX=REAL(Y(I))
200 CONTINUE
WRITE(*,'(6,010)')
9010 FORMAT(7,47H REAL DISPLACEMENT NORMALIZED EIGENVECTORS ,/)
DO 300 I=1,N,2
XT=REAL(Y(I))/XMAX
WRITE(*,'(6,0001)')XT
300 CONTINUE
1000 CONTINUE
RETURN
END