A Method for the Spatial Dynamic Simulation
of Reciprocating Compressors using the Digital Computer

by

Greg Sherman

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of
Master of Science
in
Mechanical Engineering

APPROVED:

Reginald G. Mitchiner, Chairman

Robert G. Leonard

Charles E. Knight

April 3, 1987
Blacksburg, Virginia
A Method for the Spatial Dynamic Simulation of Reciprocating Compressors using the Digital Computer

by

Greg Sherman

Reginald G. Mitchiner, Chairman

Mechanical Engineering

(ABSTRACT)

This thesis introduces the application of computer-aided engineering techniques to the dynamic analysis of reciprocating compressor designs. The analysis process is detailed in three steps.

The first step, an interactive pre-processor, develops the shaking forces and torques acting on the machine. The second step is a batch processed program that performs a dynamic simulation of the compressor in operation. The compressor and mounting are simulated as a rigid body with six degrees-of-freedom (X, Y, and Z translations and roll, pitch, and yaw) mounted to the ground with up to 25 arbitrarily oriented springs and dampers. Additionally, an eigenanalysis is performed that returns the natural frequencies and modes for the machine. The final step is an interactive post-processor where the user may examine the results of the eigenanalysis as well as the operating orbit of the machine.

A series of programs that implement the analysis process was developed. The specialized formulations for the six coupled, non-linear equations of motion are presented. Color computer graphics and animation are used for visual output displays. The dynamic simulation program is designed to function on many computer systems, from low-cost personal computers to large mainframes, while the pre- and post-processing programs are designed for the personal computer.

The programs were tested by comparing the predicted results with those of an Ingersoll-Rand model 242 two piston, two stage, 3 hp compressor. The correlation between the experimental and predicted results show that the programs can accurately simulate the dynamics of a reciprocating compressor operating at steady-state. Typically, the acceleration results agree for the six degrees-of-freedom in both the time and frequency domain to within 8%.
To Heather
Acknowledgements

I would like to express thanks to the following people who have made my graduate experience a little easier.

To Dr. Reginald G. Mitchiner, for serving as my advisor, for providing answers to every question, and for having an endless library with unlimited check-outs.

To Dr. Robert G. Leonard, for serving on my advisory committee, always providing encouragement, and listening to my problems.

To Dr. Charles E. Knight, for serving on my advisory committee. He was always available to provide additional insight.

To Dr. Arvid Myklebust, a seemingly endless source of knowledge (and stories).

To the graduate committee, and Dr. R.A. Comparin, for giving me the chance to prove myself to myself (and other non-believers).

To my fellow graduate students, late-night lab companions, and sometime poker partners (in no particular order), Brian Thatch, Steve Wampler, Doug Roach, Sandi Pennington, Dr. John Kosmatka, Mitch Keil, Brad Coffey, Bob Arenburg, Ed Moas, Jerry Schmidt, and Vince Lovejoy (for the heavy lifting). We spent a year fumbling to answer each others questions.

And to my father, who always supported this effort and never failed to say yes.
# Table of Contents

Nomenclature ................................................................. 1

Introduction ........................................................................ 2
- The Reciprocating Compressor .......................................... 3
- Designing A Compressor .................................................... 3
- The Analysis Process ........................................................ 6

Review of Literature ........................................................... 9
- Dynamic Simulation .......................................................... 9
- Personal Computer Graphics .............................................. 12
- Graphics Techniques ........................................................ 12

The Dynamics of a Compressor ............................................. 14
- Rigid Body Dynamics ....................................................... 14
  - The Linear Spring ......................................................... 15
  - The Dashpot ................................................................ 16
- General Equation of Motion .............................................. 16
- Two Degree-of-Freedom Newtonian Model ......................... 18
- Three Degree-of-Freedom Newtonian Model ....................... 23
- Expanded Position Matrix Method .................................... 25
- Compressor Kinematics .................................................... 27
- Thermodynamics ............................................................ 33
- Assumptions .................................................................. 33

Table of Contents v
# Appendix A. Pre-Processor User’s Guide

- Introduction .......................... 85
- Starting PRESYM ......................... 86
- Menu Structure .......................... 87
- Recalculation .......................... 88
- The PRESYM Main Menu .............. 89
- The EDIT Option ......................... 90
- Editing the Model ...................... 91
- Editing the Pistons .................... 91
- Editing the Counterweights .......... 92
- The FILES Option ...................... 92
- The DYNAMICS Option ................. 94
- The ANIMATE Option .................. 95
- The DRAW Option ...................... 96
- The OPTIONS Option .................. 96

# Appendix B. Processor User’s Guide

- Introduction .......................... 97
- Starting SYM .......................... 98
- Files Created by SYM .................. 99
- On Units ............................... 99
- The SYM Input File ................. 100
  - The COMMENT Card .................. 101
  - The TIME Card ...................... 101
  - The MASS Card ...................... 102
  - The SPRING / DAMPER PROPERTY Card 102
  - The SPRING / DAMPER LOCATION Card 103
  - The FORCE PROPERTY Card ........ 103
List of Illustrations

Figure 1. A two-piston, two-stage, transverse compressor ............................................. 4
Figure 2. A typical compressor and motor mounting arrangement .................................. 5
Figure 3. The compressor analysis process .................................................................... 8
Figure 4. A general dynamic system ............................................................................. 17
Figure 5. Two degree-of-freedom damped system ......................................................... 19
Figure 6. Free body diagram for two DOF system ......................................................... 20
Figure 7. Error in spring representation ....................................................................... 22
Figure 8. Three degree-of-freedom damped system ..................................................... 24
Figure 9. Typical slider-crank mechanism ................................................................... 28
Figure 10. Slider-crank mechanism with non-zero cylinder angle ................................. 30
Figure 11. Shaking forces produced by a slider-crank mechanism ................................. 32
Figure 12. Slider-crank angle conventions ................................................................. 36
Figure 13. Conventions used in spring formulation .................................................... 45
Figure 14. Conventions used in damping formulation ................................................ 48
Figure 15. A planar spring example ............................................................................ 52
Figure 16. Results of benchmark model 1 ................................................................. 57
Figure 17. Results of benchmark model 2 ................................................................. 59
Figure 18. Results of benchmark model 3 ................................................................. 62
Figure 19. The Ingersoll-Rand model 242 compressor ............................................. 67
Figure 20. Results of stiffness tests for neoprene mounts ........................................... 69
Figure 21. Results of stiffness tests for air springs ..................................................... 71
Figure 22. X acceleration at center of mass ................................................................. 75
Figure 23. Y acceleration at center of mass ................................................................. 76
Figure 24. Z acceleration at center of mass ................................................................. 77
Figure 25. X acceleration at corner of mounting plate ............................................... 78
Figure 26. Y acceleration at corner of mounting plate .......................... 79
Figure 27. Z acceleration at corner of mounting plate .......................... 80
Figure 28. Angle conventions used by PRESYM .............................. 93
## Nomenclature

The following symbols are used throughout this work.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F )</td>
<td>force vector (3-dimensional)</td>
</tr>
<tr>
<td>( K )</td>
<td>stiffness matrix</td>
</tr>
<tr>
<td>( M )</td>
<td>mass matrix</td>
</tr>
<tr>
<td>( P )</td>
<td>position vector (3-dimensional)</td>
</tr>
<tr>
<td>( U )</td>
<td>unit orientation vector (3-dimensional)</td>
</tr>
<tr>
<td>( V )</td>
<td>velocity vector (3-dimensional)</td>
</tr>
<tr>
<td>( J )</td>
<td>mass moment of inertia</td>
</tr>
<tr>
<td>( L )</td>
<td>connecting rod length</td>
</tr>
<tr>
<td>( R )</td>
<td>crank radius</td>
</tr>
<tr>
<td>( c )</td>
<td>coefficient of damping</td>
</tr>
<tr>
<td>( f_c )</td>
<td>crank inertia force</td>
</tr>
<tr>
<td>( f_p )</td>
<td>piston inertia force</td>
</tr>
<tr>
<td>( f_w )</td>
<td>counterweight inertia force</td>
</tr>
<tr>
<td>( k )</td>
<td>stiffness</td>
</tr>
<tr>
<td>( m )</td>
<td>mass</td>
</tr>
<tr>
<td>( u )</td>
<td>undeflected spring length</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>crank angle</td>
</tr>
<tr>
<td>( \beta )</td>
<td>cylinder angle</td>
</tr>
<tr>
<td>( \omega )</td>
<td>running speed, frequency</td>
</tr>
<tr>
<td>( \theta )</td>
<td>crank position, body orientation</td>
</tr>
</tbody>
</table>
Introduction

In the past, the engineering design process consisted of initial calculations, followed by the building and testing of a prototype, in turn followed by revision of the original calculations. Each design modification had to be carefully considered; the cycle continued until a final design was acceptable.

Often the engineer faced limits on time and funding which prevented designs from being fully explored. Although the final design might meet the original specifications, the chance to optimize and further refine a design was too expensive.

The introduction of the digital computer changed the way the engineer tackled the design and analysis process. While the cycle of design and analysis was unchanged, new methods no longer required extensive hand calculations or a large number of expensive prototypes.

The digital computer brought forth an era of design, analysis, and simulation of engineering systems through numerical techniques. Computer-aided engineering meant that the engineer could create and refine a design, then subject it to testing, without ever leaving the computer terminal.
**The Reciprocating Compressor**

A reciprocating compressor uses the motion of one or more pistons to compress a gas. The cylinders are commonly arranged in a transverse fashion with each piston operating on the same crankshaft. A cylinder may be oriented vertically, or at an angle about the crankshaft (as in a V-type arrangement).

A compressor will often operate in multiple stages. A stage is responsible for intake of gas (either from the previous stage or the atmosphere), compressing the gas, then exhausting the gas to the next stage. Each stage uses one or more pistons to compress the gas (see Figure 1).

A common arrangement for mounting a compressor and its driving motor is shown in Figure 2. The compressor and motor sit on a steel base connected to the ground by mountings. The mountings usually consist of a spring or a spring/damper vibration isolator.

**Designing A Compressor**

The process of designing a reciprocating compressor starts with a set of input and output specifications. Usually, the designer is given a required output gas pressure and rate of mass flow. Other design considerations may include the availability of power or a lack of operating space. The engineer must specify the number and size of the pistons, geometry of the internal parts, and other parameters.

The next step in the process is the analysis of the design. The analysis stage may explore thermodynamic characteristics, torque requirements, and forces transmitted to the compressor.
Figure 1. A two-piston, two-stage, transverse compressor
Figure 2. A typical compressor and motor mounting arrangement
mountings. The engineer may also investigate expected vibrations and methods for their control.

This thesis is primarily concerned with the analysis of the compressor design.

The Analysis Process

The analysis process, as defined by this project, requires the engineer to begin with a feasible compressor design. Such a design includes the number, size, and arrangement of pistons, specification of the running speed, and required pressure output. While the analysis process may suggest modifications to the initial design, it will not specify the initial parameters.

The first step in the analysis procedure is to examine the forces and torques produced by the motion of the internal parts. Next, the torque required to run the compressor must be computed. These forces and torques act on the body of the compressor/motor/base and are subsequently transmitted to the mountings. They also cause the compressor body to vibrate, a problem which may violate space constraints or cause fatigue problems in mountings and connected piping.

The next step in the analysis process is to examine the vibratory characteristics of the compressor. Given the forces and torques produced by the machine, a simulation of steady-state operation at running speed is performed. The engineer may specify the number, placement, and orientation of the springs and dampers that make up the compressor mounting.

The final step in the analysis process involves summarizing the information provided by the first two steps. This post-processing stage allows the engineer to graphically view the compressor motion, to explore the natural frequencies and mode shapes of the system, or to check for violations of space constraints.
The purpose of this thesis is the implementation of the compressor analysis process through use of the digital computer. Equations are derived that describe the forces generated by a compressor in steady-state operation. Further, the background necessary to produce the equations of motion for a compressor and its mountings is presented.

These concepts are implemented in software for the personal, mini-, and mainframe computer. The pre- and post-processing programs are menu-driven, use color graphics, and run on a personal computer. To provide output convenient to the engineer, results are displayed graphically or animated whenever possible. The processing program is designed to be system independent; the body of the code may be executed on many computer systems.

A summary of the analysis process as defined and implemented in this work is shown in Figure 3.
Figure 3. The compressor analysis process
Review of Literature

Because this study concentrates on both digital dynamic simulation of rigid bodies and on computer-aided engineering, the following discussion has been divided into two parts. The first part is concerned with the current state of software available for dynamic analysis. The second part outlines the use of graphics and animation on the personal computer.

Dynamic Simulation

Prior to the advent of digital computation, dynamic simulation of mechanical systems was characterized by the complex wiring of the analog computer. Traditionally, it was the job of the designer to formulate the differential equations of motion specific to the system being examined, and then to spend many hours in their implementation.

Often the designer was able to simplify his equations in terms of one or more generalized coordinates. However, due to the non-linear nature of dynamic equations of motion, a closed-form solution was usually impossible.

Today, although the hardware has improved, the technique has changed very little. Often the designer will describe the particular problem in terms of a specific set of differential equations, then
implement them using a high-level programming language such as FORTRAN. Indeed, although the fields of structural and electrical analysis have advanced through the introduction of digital computing, the field of dynamic simulation is in relative infancy [1].

In general, the digital computer is used for the solution of multi-body systems where each body possesses one or more degrees-of-freedom. However, the equations and equation derivatives needed to fully describe a non-linear system of only a few degrees-of-freedom can become extremely complex and almost unmanageable. Advances in symbolic algebraic manipulators, such as FORMAC, MACSYMA [2], and NEWEUL [3] have provided one method for reducing this chore.

Three objectives for computational methods of simulating dynamic systems have been described by Haug [1]. First, a digital computer program should be created that allows the engineer to conveniently describe the system of interest. Additionally, the program must automatically generate the equations of motion. Second, numerical algorithms for the solution of these non-linear equations should be implemented so the engineer may obtain the dynamic response of the system. Finally, computer graphics should be an integral part of the output used to provide results for the engineer.

Nikravesh [4] outlines three common methods for solving differential equations of motion where constraints exist. The first method involves a simple numerical integration of the equations without regard for the constraint violations that may occur as a result of numerical inaccuracy. The second method uses a feed-back approach to correct constraint violations for the next integration step. The final method, somewhat more complex, involves forming the equations in terms of dependent and independent generalized coordinates. Numerical integration is performed on the independent variables, then constraint equations are solved for the dependent coordinates. This final method is known as generalized coordinate partitioning.

Many codes have been developed and marketed that provide a generalized facility for describing and simulating dynamic systems. Some of these include ADAMS (Automated Dynamics Analysis of
Mechanical Systems), DADS (Dynamic Analysis and Design System) [5], IMP (Integrated Mechanisms Program), SINDYS (Simulation Program for Nonlinear Dynamics Systems), and MESA VERDE (Mechanism, Satellite, Vehicle, and Robot Dynamics). Although these codes are currently used, the history of most of them date back only to the early to mid 1980s.

Although a trend is developing towards making digital dynamic system analysis as comprehensive a field as finite element analysis, there is still a need for specialized codes designed for solving specific systems. Often these codes are developed for use by the novice where only the required input and output variables are known. Mitchiner [6] recently produced such a code for the description of the complex motion of centrifugal pendulum absorbers.

Another approach to dynamic simulation takes advantage of the specifics of the problem without sacrificing the generality of the code. Thatch [7] makes use of a pre-processing program that allows the engineer to define a complex mechanism in familiar terms. The program then produces input files for dynamic processors such as IMP. Using this method, the engineer need not be familiar with the use of the processor, only the problem at hand.

Although most of the simulation code available today is written for the mini-computer to mainframe market, the expanding power of the personal computer is making it the preferred machine for small to medium-sized applications. The cost of operating a personal workstation versus a larger time-shared system provides obvious economic benefits, if not improvements in performance.
Personal Computer Graphics

The general trend today in computer-aided design is toward standardized graphics programming libraries. Among these are GKS (Graphical Kernal System) and PHIGS (Programmers Hierarchical Interactive Graphics System). These packages offer the programmer independence from any one graphics device, as well as the assurance that his program will run on any system supporting the standard.

However, these packages are generally not suitable for the personal computer. There is substantial overhead required in making a graphics package device independent; this overhead tends to make performance suffer. Further, because their output consists of only of general primitives (lines, markers, text), they are unable to take advantage of some of the desirable aspects of the personal computer hardware.

Graphics Techniques

The greatest advantage to working in the personal computer environment is direct access to the video frame buffer. With this ability, the programmer can manipulate individual pixels on the screen, or act on the entire frame buffer as a whole. The amount of information required to perform the same functions on a typical mainframe to terminal connection would be unfeasible.

Complete control of the video hardware is essential for any animation technique. One method, described by Foley [8], involves use of the internal color look-up table. The technique begins with the loading of a series of images into the frame buffer, each image drawn using a different color
index. Animation is achieved by initially setting each color index to the background color, then cycling through the indices to toggle their color value. Each image appears sequentially for as long as a visible color is assigned to its index.

Another technique more suited to the personal computer is sometimes called double-buffering. With this method, the programmer defines a second frame buffer using the available memory of the computer. Images are loaded into this buffer, then transferred to the hardware frame buffer when complete. Often, the entire buffer contents may be moved with a single CPU instruction.

The double buffering technique offers the advantage of hiding the rasterizing of the vector data from the viewer. The drawing of lines cannot be seen; instead, the entire frame is presented at once. The viewer is able to perceive the current image while the program creates the next frame.

The programs developed for this work use the double-buffering technique.
Rigid Body Dynamics

The study of classical rigid body dynamics is based upon the laws of motion published in 1687 by Sir Issac Newton. In particular, Newton’s second law states

\[ \text{The time rate of change of linear momentum of a body is proportional to the force acting upon it and occurs in the direction in which the force acts.} \] [9]

This law is often stated simply as

\[ \text{Forces equals mass times acceleration.} \]

In equation form, the second law is written to describe translatory motion as

\[ \Sigma F = ma \] (1)

where \( \Sigma F \) is the sum of the forces acting on the body, and \( m \) and \( a \) are the mass and acceleration of the mass center of the body, respectively.

For rotation, the equation appears as

The Dynamics of a Compressor
\[ \Sigma M = J\alpha \]  

(2)

where \( \Sigma M \) is the sum of the moments acting on the body, \( J \) is the mass moment of inertia of the body about the axis of rotation, and \( \alpha \) is the angular acceleration about that axis.

These equations commonly appear as

\[ \Sigma F_x = m\ddot{x} \]
\[ \Sigma M_x = J\ddot{\theta}_x \]

(3)

where \( \ddot{x} \) represents the linear acceleration in the \( X \) direction and \( \ddot{\theta}_x \) represents the acceleration about the \( X \) axis. For spatial dynamics, there are corresponding equations describing \( Y \) and \( Z \) motion.

**The Linear Spring**

The linear spring represents a general stiffness element. When stretched, the spring exerts a force that is proportional to the applied displacement and opposite in direction. In equation form, the spring is represented as

\[ F = kx \]

(4)

where \( k \) is the spring constant for the given spring.

By convention, a spring that exerts a negative force is in compression; likewise, a positive force implies tension.

The Dynamics of a Compressor
**The Dashpot**

The dashpot represents a linear viscous damping element providing a resistance to velocity. It is more commonly called a damper.

When the two ends of the damper have some relative velocity, there is a force exerted that is proportional to the velocity and opposite in direction. In equation form, the damper is represented as

\[ F = c \dot{x} \]  

where \( c \) is the damping coefficient for the given damper.

**General Equation of Motion**

A general equation of motion describing rigid body dynamics will include contributions from each of the above elements. Applying Newton's second law, \( \Sigma F = m \ddot{x} \), we have for the general system shown in Figure 4

\[ m \ddot{x} = -c \dot{x} - kx \]  

or for a system with an applied, time-varying force \( F(t) \)

\[ m \ddot{x} = -c \dot{x} - kx + F(t) \]  

This general equation may be applied in matrix form for systems with more than one degree-of-freedom. The matrix equation for an unforced system appears as

\[ M \ddot{x} + C \dot{x} + Kx = 0 \]
Figure 4. A general dynamic system
and for the forced system as

\[ M\ddot{x} + C\dot{x} + Kx = F(t) \]  \hspace{1cm} (9)

The number of degrees-of-freedom a body possesses represents the number of coordinates required to fully describe its position and orientation; therefore, it is common to describe a real world system in terms of multiple degrees-of-freedom.

**Two Degree-of-Freedom Newtonian Model**

The classic example of a two degree-of-freedom system is shown in Figure 5.

The free body diagram for this system is shown in Figure 6.

The equations of motion for this system can be derived at the simplest linear level with the assumption that all angular displacements will be small, i.e. \( \sin \theta \approx \theta \). Applying Newton's equations of motion, \( \sum F_x = m\ddot{x} \), we have

\[ m\ddot{x} = F - (k_1 + k_2) x + (k_1 l_1 - k_2 l_2) \dot{\theta} - (c_1 + c_2) \dot{x} + (c_1 l_1 - c_2 l_2) \dot{\theta} \]  \hspace{1cm} (10)

and

\[ J\ddot{\theta} = F\theta + (c_1 l_1 - c_2 l_2) \dot{x} - (c_1 l_1^2 + c_2 l_2^2) \dot{\theta} \]
\[ - (k_1 l_1^2 + k_2 l_2^2) \theta + (k_1 l_1 - k_2 l_2) x \]  \hspace{1cm} (11)

Alternately, these equations may be written in the matrix form of equation (9) or
Figure 5. Two degree-of-freedom damped system
Figure 6. Free body diagram for two DOF system
For the given coordinate system, note that $x$ and $\theta$ are coupled in both the stiffness, $K$, and damping, $C$, matrices. This is evidenced by the presence of off-diagonal terms in both matrices. A coupled system is characterized by the motion of one degree-of-freedom affecting the motion of another.

Because these equations are formulated based on an assumption of small rotations, they are invalid when $\sin \theta \neq \theta$. We can modify these equations to provide a better reflection of the true motion of the system by including the non-linear terms. However, at the same time we are making a closed-form solution to the equations more difficult to achieve. The new equations appear as

\[
\begin{bmatrix} m & 0 \\ 0 & J \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} -(c_1 + c_2)(c_1 l_1 - c_2 l_2) & -(c_1 l_1^2 + c_2 l_2^2) \\ (c_1 l_1 - c_2 l_2) & -(k_1 + k_2)(k_1 l_1 - k_2 l_2) \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} F \\ Fe \end{bmatrix}
\]

(12)

Because we no longer have a linear relationship between $m$, $k$, $c$, $\theta$, and $x$, it is not possible to write these equations in a simple matrix form.

Although the second set of equations provides a more general and accurate representation of the system, there are still discrepancies between the mathematical model and the physical system. For example, while this formulation accounts for vertical spring deflection in the direction of the translational degree-of-freedom, it does not provide for the horizontal displacement of the spring caused by rotation of the mass. This error is illustrated in Figure 7.
Figure 7. Error in spring representation
Clearly, even the simplest two degree-of-freedom system presents difficulties in formulating accurate differential equations of motion.

Three Degree-of-Freedom Newtonian Model

The three degree-of-freedom system presented in Figure 8 is an extension of the two degree-of-freedom model of the previous discussion. It represents a body supported by four linear springs and dampers, an arrangement common to machinery mounting. This system has two rotational degrees-of-freedom, in addition to the single translational degree-of-freedom. The equations used to model the system are also extensions of the two degree-of-freedom case. Defining \((l_x, l_y)\) as the attachment point for spring \(i\), these equations appear as

\[
m\dddot{x} = -F
\]

\[
- (k_1 + k_2 + k_3 + k_4) \dot{x}
\]

\[
+ (k_3 l_{3y} + k_4 l_{4y} - k_1 l_{1y} - k_2 l_{2y}) \sin \theta_x
\]

\[
+ (k_2 l_{2x} + k_3 l_{3x} - k_1 l_{1x} - k_4 l_{4x}) \sin \theta_y
\]

\[
- (c_1 + c_2 + c_3 + c_4) \dot{z}
\]

\[
+ (c_3 l_{3y} + c_4 l_{4y} - c_1 l_{1y} - c_2 l_{2y}) \dot{\theta}_x
\]

\[
+ (c_2 l_{2x} + c_3 l_{3x} - c_1 l_{1x} - c_4 l_{4x}) \dot{\theta}_y
\]

\[
J_x \dddot{\theta}_x = -F \ddot{\theta}_y
\]

\[
- (k_1 l_{1y}^2 + k_2 l_{2y}^2 + k_3 l_{3y}^2 + k_4 l_{4y}^2) \sin \theta_y \cos \theta_y
\]

\[
- (k_1 l_{1y} + k_2 l_{2y} + k_3 l_{3y} + k_4 l_{4y}) \dot{z} \cos \theta_x
\]

\[
- (c_1 l_{1y} + c_2 l_{2y} + c_3 l_{3y} + c_4 l_{4y}) \dot{\theta}_x \cos \theta_x
\]

\[
- (c_1 l_{1y}^2 + c_2 l_{2y}^2 + c_3 l_{3y}^2 + c_4 l_{4y}^2) \dot{\theta}_x \cos \theta_x
\]
Figure 8. Three degree-of-freedom damped system
\[ J_y \ddot{\theta}_y = -F_{E_x} \]
\[ - (k_1 l_1^2 + k_2 l_2^2 + k_3 l_3^2 + k_4 l_4^2) \sin \theta_y \cos \theta_y \]
\[ - (k_1 l_1^2 + k_2 l_2^2 + k_3 l_3^2 + k_4 l_4^2) \cos \theta_y \]
\[ - (c_1 l_1^2 + c_2 l_2^2 + c_3 l_3^2 + c_4 l_4^2) \dot{\theta}_y \cos \theta_y \]
\[ \dot{c}_1 l_1^2 + c_2 l_2^2 + c_3 l_3^2 + c_4 l_4^2 \] \[ \cos \theta_y \]

Clearly, the equations presented here have become unwieldy to derive. In addition, the spring error present in the two degree-of-freedom example still affects the three degree-of-freedom formulation. Moreover, the equations presented here may only be used to describe a system that is similar to Figure 8. They represent a system with four springs and dampers mounted orthogonal to a plate; the only variables are the plate attachment points \((l_x, l_y)\) for each spring.

**Expanded Position Matrix Method**

It was noted in previous sections that a non-linear differential equation could not be written in matrix form. This was because the position vector, \(\{x\}\), contained non-linear terms. However, using an expanded position and velocity matrix that includes the non-linear terms, a useful matrix method may be derived.

As an example, consider a single spring from the three degree-of-freedom model. We begin the formulation of the expanded position matrix by deriving a spring force matrix. This matrix is formed by displacing each degree-of-freedom one unit, then measuring the resulting forces. The force in the \(j\)th degree-of-freedom due to displacing the \(i\)th degree-of-freedom becomes the \(f_{ij}\) entry in the matrix. For an arbitrary spring attachment point \((l_x, l_y)\) we have
\[
\begin{bmatrix}
-kz & -k_y \sin \theta_x & -k_x \sin \theta_y \\
-kzl_y & -k_y l_y \sin \theta_x \cos \theta_x & -k_x \sin \theta_x l_y \\
-kzl_x & -k_y \sin \theta_x l_x & -k_x l_x \sin \theta_y \cos \theta_y
\end{bmatrix}
\]  

(18)

This matrix is reduced to a simpler form by removing the terms that represent the body's degrees-of-freedom \((z, \theta_x, \theta_y)\):

\[
[S] = k
\begin{bmatrix}
-1 & -l_y & -l_x \\
-l_y & -l_y^2 & -l_y l_x \\
-l_x & -l_x l_y & -l_x^2
\end{bmatrix}
\]  

(19)

The removed terms are placed in the expanded position matrix:

\[
[XX] = \begin{bmatrix}
z & z & z \\
\sin \theta_x & \sin \theta_x \cos \theta_x & \sin \theta_x \\
\sin \theta_y & \sin \theta_y & \sin \theta_y \cos \theta_y
\end{bmatrix}
\]  

(20)

At each time step, the expanded position matrix, \(XX\), is constructed. The spring force matrix, \(S\), remains constant. When these matrices are multiplied, the resulting 3 x 3 matrix contains the spring force vector on the diagonal.

\[
[F]_{3x3} = [S][XX]
\]  

(21)

For this three degree-of-freedom system, \(F_{11}\) is the force in the \(Z\) direction, \(F_{22}\) is the torque about the \(X\) axis, and \(F_{33}\) is the torque about the \(Y\) axis.

For a system with multiple springs, the spring force matrix of each spring may be summed to form the system force matrix. Thus at each time step, only one matrix multiplication is required to form the force vector.
While this method provides for more efficient code, it is also dependent upon the geometry of the model. The formulation has taken the body attachment point of a spring or damper into consideration. Additional springs and dampers may be easily added. However, the springs and dampers are always assumed to be orthogonal to the plate.

**Compressor Kinematics**

A single-piston reciprocating compressor represents a classical slider-crank mechanism. Figure 9 shows a slider-crank and the associated nomenclature. The slider-crank in a compressor is made up of these major parts: the crank, the connecting rod, the piston, the crank pin, the piston pin, and the counterweight.

The slider-crank generates a predictable set of forces that are transmitted to the compressor block and mounting. These forces, known as *shaking forces*, are defined as the resultant of all forces acting on the frame of a mechanism due to inertia forces only [11].

To compute the shaking forces generated by the slider-crank, it is necessary to determine the inertia force created by the internal parts. For convenience, the equivalent weights of the crank, $W_C'$, and the connecting rod, $W_C''$, are lumped at the crank pin. At the piston pin, the piston and pin are lumped as $W_p'$ along with the equivalent weight of the connecting rod, $W_p''$. This allows us to represent the slider crank with a dynamically equivalent system of one rotating and one reciprocating weight.

Because the equations describing point-mass rotation and translation are easily derived, the lumping of masses in this manner greatly simplifies the modeling task. Certain parameters, such as the inertias of the connecting rod and crank, are ignored, *sometimes unacceptably.*
Figure 9. Typical slider-crank mechanism
The acceleration of the piston is written as a truncated Taylor series of the form

\[ A = -R \omega^2 \cos \theta + \frac{R}{L} \cos 2\theta \]  

(22)

where \( \omega \) is the rotational velocity of the crank and \( \theta \) is measured from top dead center (TDC). This approximation ignores the high order harmonics and thus may be unsuitable for some computer simulations.

The inertia force due to the piston is directed opposite to the acceleration and is expressed as

\[ f_p = \frac{W_p + W'_p}{g} R \omega^2 \cos \theta + \frac{W_p + W'_p}{g} \frac{R}{L} \omega^2 \cos 2\theta + \ldots \]  

(23)

A positive value for \( f_p \) indicates the force is directed away from the crank shaft.

Although the rotation angle, \( \theta \), is referenced from TDC, the crank angle, \( \alpha \), may not coincide with this angle. The crank angle represents the angular offset between a piston crank and the compressor crankshaft. The inertia force equations may be modified to reflect the crank angle as

\[ f_p = \frac{W_p + W'_p}{g} R \omega^2 \cos (\theta + \alpha) + \frac{W_p + W'_p}{g} \frac{R}{L} \omega^2 \cos (2\theta + \alpha) + \ldots \]  

(24)

The slider-crank will not always be configured in a vertical arrangement, as shown in Figure 10. Therefore, the inertia force equation must account for a non-zero cylinder angle, \( \beta \) as

\[ f_{p,x} = -f_p \sin \beta \]

\[ f_{p,y} = f_p \cos \beta \]  

(25)

A non-zero cylinder angle implies that the inertia force due to the piston has components in the \( X \) and \( Y \) directions.

The forces produced by the slider crank are shown in Figure 11.
Figure 10. Slider-crank mechanism with non-zero cylinder angle
Figure 11. Shaking forces produced by a slider-crank mechanism
The rotation of the equivalent crank and connecting rod weight at the crank pin is another source of shaking forces and moments. The magnitude of the force created by the centrifugal action of the weight is given by

$$f_c = \frac{W''c + W'c R_\omega^2}{g}$$  \hspace{1cm} (26)

Clearly, this force will have components in the $X$ and $Y$ directions that are functions of $\theta$. These components are given by

$$f_{c,x} = -f_c \sin \theta$$

$$f_{c,y} = f_c \cos \theta$$  \hspace{1cm} (27)

A similar set of shaking forces and moments are generated by the rotation of the counterweight. The centrifugal force is given by

$$f_w = \frac{W_w R_w \omega^2}{g}$$  \hspace{1cm} (28)

where $R_w$ is the radius of the counterweight. Because the counterweight is typically directed opposite to the crank, the counterweight angle, $\varphi$, is measured with a reference of $180^\circ$ from the crank angle, $\alpha$. The equations for the shaking forces are given by

$$f_{w,x} = f_w \sin (\theta + \varphi)$$

$$f_{w,y} = -f_w \cos (\theta + \varphi)$$  \hspace{1cm} (29)

In practice, a compressor may have more than one piston. Each piston is situated on the $Z$ axis at various positions along the crank shaft. This arrangement generates additional shaking moments about the $X$ and $Y$ axis due to piston inertia forces. For a given piston we have

$$M_x = (f_{p,x}) z$$

$$M_y = (f_{p,y}) z$$  \hspace{1cm} (30)

The Dynamics of a Compressor
where $z$ is the shaft position of the crank with respect to the center of mass of the compressor.

In order to use these shaking forces and moments, they are expressed in terms of $X$, $Y$, and $Z$ components.

\[
F_x = \sum_{i=1}^{n} (f_{p,x} + f_{c,x} + f_{w,x})_i \\
F_y = \sum_{i=1}^{n} (f_{p,y} + f_{c,y} + f_{w,y})_i \\
F_z = 0
\]  

(31)

The equations presented above define the forces and moments that are transmitted to the mountings of the compressor. They are used as forcing functions for exploring the dynamic response of the entire system.

**Thermodynamics**

**Assumptions**

In developing a simulation model of a reciprocating compressor, certain thermodynamic assumptions must be made. Many of these assumptions are based upon the actual performance of the compressor components; therefore, the validity of the assumption depends upon the quality of the compressor parts and their assembly.

The compressor is modeled with the assumption that all valving is ideal; that is, the valves open and close as expected. In studies by Spagnuolo [10], however, it was found that lubricating oil tends
to delay the opening of finger-type valves. This oil stiction means the valve is open for a shorter period of time during a crank cycle. Another valve assumption is that there is no gas leakage through the intake and exhaust ports when the valves are closed.

The model may assume that the interstage coolers are ideal. An ideal cooler will remove all heat added to the gas after one compression stage and before the next stage begins. In reality, the required capacity of such a cooler would not be feasible for typical applications. Therefore, the temperature of a gas entering the second stage will probably be higher in the actual compressor than in the model.

Other assumptions involve the properties of the gas being compressed. A model will generally assume that the gas temperature is constant throughout the cylinder and that temperature changes propagate immediately. Similarly, the model may assume that the gas pressure is constant throughout the cylinder and may be expressed as a function of volume only.

Note that these assumptions are applied in order to simplify the modeling task and are not uncommon in the study of compressors.

**Torque Computations**

The instantaneous torque required to move a single piston is a function of the crank angle, the piston inertia force, and the cylinder pressure. The cylinder pressure can be expressed as a function of volume as

\[
P = P_{in} \left[ \frac{V_{max}}{V} \right]^k
\]  

(32)
where $V_{\text{max}}$ is the maximum cylinder volume, $V$ is the instantaneous volume, and $k$ is the ratio of the specific heats of the gases. The maximum cylinder volume is found when the piston is at the point closest to the crankshaft (bottom dead center, or BDC).

The instantaneous volume of the cylinder is written as

$$V = Ax + V_c$$  \hspace{1cm} (33)

where $x$ is the piston displacement from its highest position in the cylinder, $A$ is the area of the piston face $\frac{\pi D^2}{4}$, and $V_c$ is the clearance volume. The clearance volume is the volume at the top of the cylinder that lies outside the travel of the piston.

The pressure force acting on the surface of the piston can now be expressed as

$$F_p = PA$$  \hspace{1cm} (34)

The total force acting on the piston is given by

$$F = F_p + f_p$$  \hspace{1cm} (35)

where $f_p$ is the inertia force described in the previous section.

For reference, Figure 9 on page 28 is duplicated on the following page. The angle $\psi$ represents the absolute position of the piston crank with respect to the cylinder. It is given by

$$\psi = \theta - \alpha - \beta$$  \hspace{1cm} (36)

where $\theta$ is the crankshaft angle, $\alpha$ is the initial crank angle, and $\beta$ is the cylinder angle.

For a piston that has a zero crank angle and cylinder angle, we have

$$\psi = 0$$  \hspace{1cm} (37)
Figure 12. Slider-crank angle conventions
We also define $\phi$ to be the instantaneous angle between the centerlines of the cylinder and the connecting rod

$$\phi = \sin^{-1}\left(\frac{R}{L} \sin \psi\right)$$

(38)

where $R$ is the crank length and $L$ is the connecting rod length.

The moment arm of the piston forces transmitted through the connecting rod about the crank shaft is given by

$$d = h \sin \phi$$

(39)

Finally, we can express the instantaneous torque requirement as opposite that of the implied torque by

$$T = - \frac{F}{\cos \phi} d$$

(40)

Note that $\phi$, $F$, and $h$ may all be expressed as a function of the crankshaft angle.

The torque required by the compressor at running speed is supplied by the motor. This torque is applied to the system as a moment about the $Z$ axis. Along with the shaking forces and moments, it serves as a complete description of the forces affecting the compressor mounting.
The Compressor Pre-Processing Module

The function of a pre-processor is to allow the engineer to fully describe, edit, and check the design before submitting a model for subsequent processing. The pre-processor developed for this study carries out each of these functions; moreover, it has been expanded to provide the user with additional information specific to compressor design.

Because the simulation program has been designed to be independent of the details of the system being modeled, the pre-processor must condense the compressor design into a format understood by the processor. In particular, the pre-processor needs to describe the compressor solely in terms of the shaking forces and moments induced by the steady-state motion of the internal parts.

Pre-Processor Input

The designer must be aware of certain input parameters. The first and most important of these is the required exit air pressure. The pre-processor accepts both the surrounding atmospheric pressure and compressor output pressure as input parameters.

Next, the designer may perturb the remaining variables to conform to manufacturing specifications, space and size requirements, or available driving power.
These parameters are required for all compressor designs, regardless of the piston arrangement:

- Number of Stages
- Number of Cylinders
- Input Pressure PSIA
- Output Pressure PSIA
- Crank Speed REV/MIN
- Input Ambient Temperature °F
- Ratio of Specific Heats

These parameters are required for each piston regardless of the piston arrangement:

- Cylinder Diameter INCHES
- Cylinder Angle DEGREES
- % Clearance
- Operating Stage
- Reciprocating Weight LBS

These parameters apply only to a transverse piston arrangement:

- Crank Radius INCHES
- Connecting Rod Length INCHES

These parameters apply to pistons in a transverse piston arrangement:

- Crank Angle DEGREES
- Axial Position INCHES
- Rotating Weight LBS
- Counterbalance Weight LBS
- Counterbalance Radius INCHES

The Compressor Pre-Processing Module
• Counterbalance Angle  

These parameters apply only to an axial piston arrangement:

• Swash Plate Radius  INCHES

• Piston Half Stroke  INCHES

**Pre-Processor Output**

When all input parameters have been specified, the designer may simulate the motion of the compressor for a single rotation at speed. The pre-processor computes the total torque requirement and the total shaking forces and moments due to piston and counterweight motion. As an option, the designer may view the shaking forces and moments caused by a single piston. All output is in a graphical format so that the variance of the forces as a function of the crankshaft angle may be observed.

A second pre-processor output function allows the designer to verify the physical layout of the compressor. The animation option computes and stores a sequence of 36 frames showing the full rotational motion of one or all pistons. If only one piston is selected, the corresponding displacement, inertia force, and torque are plotted simultaneously as a function of the crankshaft angle. Subsequently selecting another piston allows the same set of plots to be overlayed.

Optionally, the compressor position at any single crankshaft angle may be drawn. The instantaneous torque is displayed.
In addition to the graphical presentations provided by the pre-processor, a numerical analysis file must be created. The analysis file contains a Fourier transformed representation of the shaking forces and moments so that the simulator may reproduce the time histories of these forces and moments.

To create the analysis file, the pre-processor computes and sums the shaking forces, moments, and applied torques into 3 force and 3 moment components. These 6 data sets are computed at 256 evenly spaced angles around a single crank rotation. The summations are then shifted into the frequency domain through use of a fast Fourier transform (FFT) algorithm.

The use of the FFT allows the processor to reconstruct the forcing functions computed by the pre-processor. The FFT algorithm returns sine and cosine coefficients; the forcing function may then be expressed (in time) as

\[
 f(t) = \frac{A_0}{2} + \sum_{k=1}^{n-1} (B_k \sin k\omega t + A_k \cos k\omega t)
\]  

(41)

where \( B_k \) are the sine coefficients, \( A_k \) are the cosine coefficients, and \( \omega \) is the frequency of the waveform. Note that for the periodic motion associated with compressor motion, we have \( \omega t = 0 \).

By reading the sine and cosine coefficients for the first 10 harmonics, the simulator can conveniently approximate any complex function. The pre-processor stores these coefficients in the analysis file along with data describing the application points of each force, the starting and ending times of application, the orientations of each force, and the forcing frequency.

A completed model may be stored on a diskette for later recall and modification. The file that contains the model is independent of the analysis file and is written in a format that only the pre-processor understands.

The Compressor Pre-Processing Module
**Editing Facilities**

The pre-processor provides a complete, interactive editing facility that allows the designer to conveniently describe the compressor design. Each of the input parameters required for the computation of shaking forces and moments may be entered or modified to reflect perturbations in the model.

Because certain parameters are pertinent to the entire compressor model whereas others apply to specific pistons, the functions of editing the model and pistons are separated. In addition, the designer has the ability to attach arbitrary counterweights at any position along the crank to reduce the magnitude of shaking forces.
The Dynamic Processing Module

The function of the processor is to simulate the dynamic response of the compressor body. As input, the processor accepts the shaking force and moment descriptions provided by the pre-processor. Also, the processor must be aware of the mass and axial moments of inertia of the compressor configuration in order to complete the equations of motion.

The user is able to describe the mounting arrangement by attaching springs and dampers at arbitrary locations around the compressor body. The formulations used to compute the forces generated by these springs and dampers are described below.

A secondary function of the processor is the computation of the rigid body natural frequencies and mode shapes of the system with the given springs and mass. A six degree-of-freedom system will have six natural frequencies. These frequencies correspond to the undamped, unforced frequency of vibration seen by the body when an appropriate initial displacement is applied.

When a body is excited at one of its natural frequencies, the resulting displacement is called the mode shape. A mode shape may include one or more of a body's degrees-of-freedom and is usually presented in a normalized form.
**Spring Formulation**

As described in Chapter 3, the force due to the displacement of a spring is given by

\[ F = kx \]  

(42)

This formulation can be expanded if we assume the spring to be grounded at the origin of some arbitrary coordinate system. If the opposite end of the spring is displaced along the \( X \) axis, the force in the \( X \) direction is given by

\[ F_x = k(u - x) \]  

(43)

where \( x \) is the position of the end of the spring and \( u \) is the static length of the spring. Note that with this formulation, the sign convention of the force, \( F \), matches that of the displacement, \( x \).

This equation can be expanded further to account for any position along the \( X \) axis. If \( x_1 \) is defined as the grounded end of the spring and \( x_2 \) as the displaced end, we have

\[ F_x = k[u - (x_2 - x_1)] \]  

(44)

For the purpose of modeling, however, we cannot assume that a spring will be oriented along an axis of some pre-defined coordinate system. Therefore, we define \( P_1 \) and \( P_2 \) as vectors defining the grounded and displaced ends of the spring, respectively (see Figure 13).

The spring force equation now appears as

\[ F_x = k[u - |P_2 - P_1|] \]  

(45)

This equation returns a scalar quantity that describes the magnitude of the spring force along the axis of the spring.
Figure 13. Conventions used in spring formulation
This force must now be separated into three spatial components. Since the given information includes the positions of the spring ends, we can describe the orientation of the spring in terms of a unit vector, \( \mathbf{U}_s \). Thus, we have

\[
\mathbf{U}_s = \frac{\mathbf{p}_2 - \mathbf{p}_1}{|\mathbf{p}_2 - \mathbf{p}_1|}
\]

(46)

We can then describe the spring force vector as

\[
\mathbf{F} = F_s \mathbf{U}_s
\]

(47)

By substituting equations (45) and (46) into (47), we have the complete formulation of the spring force vector in terms of position, stiffness, and static length.

\[
\mathbf{F} = k \left[ u - |\mathbf{p}_2 - \mathbf{p}_1| \right] \left[ \frac{\mathbf{p}_2 - \mathbf{p}_1}{|\mathbf{p}_2 - \mathbf{p}_1|} \right]
\]

(48)

These equations are applied by the dynamic processor. At each time step, the position of the body-attached end of the spring, \( \mathbf{p}_2 \), is calculated. The location of the grounded end, \( \mathbf{p}_1 \), as well as the stiffness, \( k \), are given as input. The force vector for the spring is computed and applied to the body.

**Damping Formulation**

The derivation of the forces due to damping is similar to that for the spring. Just as the spring assumes no lateral stiffness, we must be careful to insure the damper force is computed based only on the velocity component that acts along the axis of the damper.
As discussed in Chapter 3, the force exerted by a damper is given by

\[ F = c\dot{x} \]  

(49)

If we allow the damper to lie along the \( X \) axis of some coordinate system with one end fixed, the force in the \( X \) direction is

\[ F_x = -c\dot{x} \]  

(50)

where \( \dot{x} \) is the velocity of the opposite end.

Just as with the spring formulation, however, we cannot assume that the damper will lie along a particular axis. Neither can we assume that the velocity vector of the free end is parallel to the orientation vector of the damper.

By taking the dot product of the velocity vector and the orientation vector, we obtain the magnitude of the velocity component that lies along the axis of the damper

\[ \nu = V_2 \cdot (P_2 - P_1) \]  

(51)

Therefore, we can express the damping force as

\[ F_d = c\nu \]  

(52)

To express the damping force in terms of spatial components, we use the same equations used for the spring (refer to Figure 14).

The damper orientation is expressed as

\[ U_d = \frac{P_2 - P_1}{|P_2 - P_1|} \]  

(53)

We can then describe the damping force vector as
Figure 14. Conventions used in damping formulation
By substituting equations (52) and (53) into (54), we have the complete formulation of the damping force vector in terms of velocity, position, and damping coefficient.

\[ F = F_d U_d \]  

(Eq. 54)

The general eigenvalue problem [12] is defined as

\[ A \mathbf{x} = \lambda \mathbf{B} \mathbf{x} \]  

(Eq. 56)

where \( \lambda \) is an eigenvalue and \( \mathbf{x} \) is the corresponding eigenvector. For a dynamics problem, the eigenvalue problem can be stated as

\[ K \mathbf{x} = \omega^2 M \mathbf{x} \]  

(Eq. 57)

where \( K \) is the system stiffness matrix, \( M \) is the system mass matrix, \( \omega \) is a natural frequency, and \( \mathbf{x} \) is the corresponding mode shape.
Mass Matrix

For a body that possesses all six degrees-of-freedom, the mass matrix is

\[
M = \begin{bmatrix}
m & 0 & 0 & 0 & 0 & 0 \\
0 & m & 0 & 0 & 0 & 0 \\
0 & 0 & m & 0 & 0 & 0 \\
0 & 0 & 0 & I_{xx} & I_{yx} & I_{zx} \\
0 & 0 & 0 & I_{xy} & I_{yy} & I_{zy} \\
0 & 0 & 0 & I_{xz} & I_{yz} & I_{zz}
\end{bmatrix}
\]

where \( m \) is the mass of the body and \( I_{ij} \) are the principal moments of inertia and products of inertia.

If the coordinate system is defined to lie along the three principal axis of the body, the off-diagonal terms of the inertia tensor drop off. The mass matrix then appears as

\[
M = \begin{bmatrix}
m & 0 & 0 & 0 & 0 & 0 \\
0 & m & 0 & 0 & 0 & 0 \\
0 & 0 & m & 0 & 0 & 0 \\
0 & 0 & 0 & I_{xx} & 0 & 0 \\
0 & 0 & 0 & 0 & I_{yy} & 0 \\
0 & 0 & 0 & 0 & 0 & I_{zz}
\end{bmatrix}
\]

Central Difference Stiffness Matrix

The stiffness matrix for the body is a function of the number of springs attached, their individual stiffnesses, and their orientation. An element of the stiffness matrix is defined as the force seen by the \( i \)th degree-of-freedom when the \( j \)th degree-of-freedom is given a unit displacement, or

\[
k_{ij} = \frac{F_j}{x_i}
\]
where $x_i = 1$.

In order to perform an eigenanalysis to obtain real mode shapes and natural frequencies, the stiffness matrix must be real and symmetric. However, if the spring formulation previously described is applied in conjunction with equation (60), a non-symmetric equation results.

As an example, consider a spring that lies in the $XY$ plane as shown in Figure 15. Theoretically, this spring should have no stiffness in the $Z$ direction. However, if we displace the spring one unit out of the $XY$ plane, in reality a $Z$ force results. The stiffness matrix would appear as

$$K = \begin{bmatrix} x & x & x \\ x & x & x \\ 0 & 0 & x \end{bmatrix}$$

which is clearly not symmetric.

One approach in overcoming this problem is use of the central finite difference method as described by James, Smith, and Wolford [13]. This method, as applied to the above example, would displace the spring some value, $\Delta z$, both into and out of the $XY$ plane. The stiffness would then be given by the difference in the two forces divided by the total displacement, or

$$k = \frac{F(\Delta z) + F(-\Delta z)}{\Delta z}$$

(62)

The error in this formulation is on the order of $\Delta x^2$. Therefore, the stiffness can be expressed more accurately as

$$k = \lim_{\Delta z \to 0} \frac{F(\Delta z / 2) - F(-\Delta z / 2)}{\Delta z}$$

(63)

For a computer implementation of this equation, a suitable value for $\Delta z$ is in the range $10^{-5}$ to $10^{-7}$.
Figure 15. A planar spring example
Stiffness Transformation

A second method for constructing a stiffness matrix is common in finite element practice. First, a local coordinate system is attached to each spring. The spring is forced to lie along a principle axis of that system (for example, the $X$ axis) so that the local stiffness matrix appears as

$$K = \begin{bmatrix} k & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (64)$$

Using some transformation method (e.g. direction cosines, Euler angles, etc.), a matrix is constructed that rotates the spring to match the given orientation.

$$\{x\}' = [T] \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad (65)$$

This same matrix is then applied to the local stiffness matrix to give the global stiffness matrix for the single spring.

$$[K]' = [T]^T \begin{bmatrix} k & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} [T] \quad (66)$$

The process is performed on each spring, then the matrices are summed to form the system stiffness matrix.

When the stiffness and mass matrices have been assembled, an eigensolver routine may be used to compute the natural frequencies and mode shapes. The processor uses a Jacobi iteration scheme; any suitable method or mathematics package may be used.
Simulation Techniques

The mainline of the simulation program consists of a single loop in which a numeric integration routine is called. The integrator used is a fourth-order Runge-Kutta algorithm with a fixed time step as described by James, Smith, and Wolford [13]. This algorithm offers acceptable accuracy for most applications. However, the user may wish to employ other algorithms, such as a variable step Runge-Kutta or an Adams Predictor-Corrector. For that reason, the integration routines have been kept external and are independent of the main program.

As with any numerical integrators, a routine that computes the derivative of the desired function is required. The routine used in the simulation program determines all forces and torques acting on the body due to springs, dampers, or external applications. The integrated function is then expressed simply as

\[ \ddot{x} = M^{-1}F \]  

(67)

where \( \ddot{x} \) is a vector describing the translational and angular acceleration of the body.

At each time step, the body moves or rotates a finite amount based upon the equation of motion and the value of the time step. As the simulation progresses, the integrator is used to compute the current orientation of the body in space. The integration process continues until a specified time is reached.
Benchmark Results

As part of the testing of the simulation code, various models were run in all modes on a variety of processors. These processors included

- IBM PC under PC-DOS 3.1
- IBM AT under PC-DOS 3.1
- DEC MicroVax II under VMS 4.3
- DEC VAX 11/780 under VMS 4.4
- IBM 4341 under VM/CMS 4.03
- IBM 3084 under VM/CMS 4.03
- IBM 3090 under VM/CMS 4.03

Each of these systems support the ANSI FORTRAN-77 standard to the degree required by the simulation code. Details of the code's implementation on the above systems is presented in Appendix 4.

The code used for each system was identically prepared. Where possible, the same executable file was used. For multi-user systems, the CPU load was never higher than 20%.

The following pages present the results of tests with three benchmark models. Run times are given in real seconds as opposed to CPU seconds.

Refer to Appendix 3 for details on the format of model files.
Model 1 - One Degree-of-Freedom

The one degree-of-freedom model represents a single spring-mass-damper system. A single sinusoidal force is applied to the mass.

The model file used is given below.

```
C C SYM program benchmark 1
C
T 0.0  2.0  0.005  2  2  10.0  32
M 10.0 25.0 25.0 25.0
S 1  400.0 10.0  1.0
L 1  0.0  0.0  0.0  0.0  -1.0  0.0
F 1  2  0.0  3.0  10.0  50.2655  0.0  0.0
R 1  0.0  0.0  0.0  0.0  -1.0  0.0
```
Figure 16. Results of benchmark model 1
Model 2 - Three Degrees-of-Freedom

The three degree-of-freedom model represents a plate mounted on four springs and dampers. A single sinusoidal force is applied.

The model file used is given below.

```
SYM program benchmark 2
T 0.0 2.0 0.005 2 2 10.0 32
M 10.0 25.0 25.0 25.0
S 1 100.0 10.0 1.0
L 1 10.0 0.0 10.0 10.0 -1.0 10.0
L 1 -10.0 0.0 10.0 -10.0 -1.0 -10.0
L 1 -10.0 0.0 -10.0 -10.0 -1.0 -10.0
F 1 2 0.0 3.0 10.0 50.2655 0.0
R 1 0.0 0.0 0.0 0.0 -1.0 0.0
```
Figure 17. Results of benchmark model 2
Model 3 - Six Degrees-of-Freedom

The six degree-of-freedom model represents a typical compressor loading and mounting. Three complex forces and moments are applied to the body mounted on four springs.

The model file used is given below.

<table>
<thead>
<tr>
<th>C</th>
<th>T</th>
<th>M</th>
<th>S</th>
<th>L</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYM program benchmark 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T0.0 2.0 0.005 3 2 10.0 32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M 10.0 25.0 25.0 25.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S 1 800.0 1.0 1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S 2 7600.0 1.0 1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L 1 10.0 0.0 10.0 10.0 -1.0 10.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L 1 10.0 0.0 -10.0 10.0 -1.0 -10.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L 1 -10.0 0.0 10.0 -10.0 -1.0 10.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L 2 10.0 0.0 10.0 11.0 0.0 11.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L 2 10.0 0.0 -10.0 11.0 0.0 -11.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L 2 -10.0 0.0 -10.0 -11.0 0.0 -11.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L 2 10.0 0.0 10.0 -10.0 0.0 10.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L 2 -10.0 0.0 -10.0 -10.0 0.0 10.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F 1 3 1.0 10.0 1.0 50.2655 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F 2 3 0.0 10.0 1.0 50.2655 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F 3 3 0.0 10.0 1.0 50.2655 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F 4 3 0.0 10.0 1.0 50.2655 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Dynamic Processing Module
The Dynamic Processing Module
Figure 18. Results of benchmark model 3
The Compressor Post-Processing Module

The function of the post-processor is to summarize and organize the results found during the dynamic analysis. The designer is primarily interested in vibrations and displacements, although natural frequencies and mode shapes are important in specifying running speed.

There are four functional units to the post processor.

- Function one animates the motion of the compressor body as a function of time. As an option, the user may exaggerate the rotations or translations of the body. The body is drawn in one of three orthographic views or an isometric view.
- Function two animates the mode shapes of the body and displays the corresponding natural frequencies. The animation is exaggerated to emphasize the mode shape. As with function one, the user may select from one of four views.
- Function three creates XY plots that detail the motion of the compressor body. Three plots of translation and three plots of rotation are presented.
- Function four allows the user to determine the translation at any point on the body. This function may be used to guarantee that any space constraints are met.
Experimental Verification

With any modeling technique, it is necessary to verify the results of a mathematical simulation by comparison with a known system. For this work, experimental procedures were devised to test the results found by both the pre-processor and the dynamic simulator.

Mass and Moments of Inertia

Whereas the mass of a machine is relatively easy to find in the laboratory, the principal mass moments of inertia are not. Because the moments of inertia are important to the basic equations of motion, a method was devised to estimate their value. This method combines the collection of experimental data with an inverse eigensolver process.

We recognize the typical stiffness matrix for a system as

\[
K = \begin{bmatrix}
k_{11} & k_{12} & k_{13} \\
k_{21} & k_{22} & k_{23} \\
k_{31} & k_{32} & k_{33}
\end{bmatrix}
\]  

(68)

The eigenvalue problem for translational modes can be stated as
\[ K - \omega^2 M = \begin{bmatrix} k_{11} - \omega^2 m & k_{12} & k_{13} \\ k_{21} & k_{22} - \omega^2 m & k_{23} \\ k_{31} & k_{32} & k_{33} - \omega^2 m \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \]  \tag{69}

or in equation form as

\[
\begin{align*}
  k_{11} x_1 + k_{12} x_2 + k_{13} x_3 - \omega^2 m x_1 &= 0 \\
  k_{21} x_1 + k_{22} x_2 + k_{23} x_3 - \omega^2 m x_2 &= 0 \\
  k_{31} x_1 + k_{32} x_2 + k_{33} x_3 - \omega^2 m x_3 &= 0
\end{align*}
\tag{70}
\]

where \( m \) is the mass of the system.

Note that \( \{x_1, x_2, x_3\}^T \) represent a single mode shape corresponding to natural frequency \( \omega \).

In the laboratory, it is possible to excite the translational modes and record the natural frequencies of the machine and its mounting. Therefore, for equations (70), we can solve for the value of \( m \) as

\[
\begin{align*}
  m &= \frac{k_{11} + k_{12} \frac{x_2}{x_1} + k_{13} \frac{x_3}{x_1}}{\omega_1^2} \\
  m &= \frac{k_{21} \frac{x_1}{x_2} + k_{22} + k_{23} \frac{x_3}{x_2}}{\omega_2^2} \\
  m &= \frac{k_{31} \frac{x_1}{x_3} + k_{32} \frac{x_2}{x_3} + k_{33}}{\omega_3^2}
\end{align*}
\tag{71}
\]

Note that for carefully recorded data, the values returned by the three equations will agree.

A similar set of equations is used to determine the moments of inertia.
The Ingersoll-Rand 242 Compressor

As part of the testing process, an Ingersoll-Rand model 242 reciprocating compressor was modeled.

This compressor is a two-stage, two-piston compressor. The pistons are arranged in a vee fashion and ride on the same crank throw (see Figure 19).

The geometry of the compressor was available from either Ingersoll-Rand literature or through laboratory measurements. That data is given below.

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight of Large Piston</td>
<td>1.750 LBF</td>
</tr>
<tr>
<td>Weight of Small Piston</td>
<td>1.080 LBF</td>
</tr>
<tr>
<td>Weight of Large Pin</td>
<td>0.340 LBF</td>
</tr>
<tr>
<td>Weight of Small Pin</td>
<td>0.200 LBF</td>
</tr>
<tr>
<td>Weight of Connecting Rod</td>
<td>0.720 LBF</td>
</tr>
<tr>
<td>Length of Connecting Rod</td>
<td>6.813 IN</td>
</tr>
<tr>
<td>Crank Radius</td>
<td>1.395 IN</td>
</tr>
<tr>
<td>Diameter of Large Piston</td>
<td>4.000 IN</td>
</tr>
<tr>
<td>Diameter of Small Piston</td>
<td>2.500 IN</td>
</tr>
<tr>
<td>Piston Offset</td>
<td>1.250 IN</td>
</tr>
</tbody>
</table>

The piston offset represents the distance between the centerlines of the cylinders. This offset leads to a shaking moment about both the X and Y axes.
Figure 19. The Ingersoll-Rand model 242 compressor
The compressor operates at 790 RPM and provides 100 psi output. This data was used, in conjunction with the pre-processor, to compute the shaking forces, shaking moments, and input torques associated with the compressor.

The compressor mounting is similar to Figure 2 on page 5. The springs used were neoprene-in-shear mounts from Vibration Mounting and Controls, Inc. (model RD-2) with an advertised axial spring rate of 760 lbs/in. The axial and transverse stiffnesses of the mounts were tested using an Instron compression/tension testing rig. The mounts were found to have an axial spring rate of 760 lbs/in and a transverse spring rate of 260 lbs/in in the range of applied loading (see Figure 20).
Figure 20. Results of stiffness tests for neoprene mounts
To determine the moments of inertia of the compressor and mounting, the mounts were replaced with softer air springs having less internal damping. Again using the Instron rig, the spring rates were found to be 240 lbs/in in the axial direction and 60 lbs/in in the transverse direction. Although the air springs had non-linear spring rates, these values were valid for the range of applied loading. (see Figure 21).
Figure 21. Results of stiffness tests for air springs
Using a PCB model 302A low-frequency accelerometer, accelerations for excited principal modes were captured. A Zonic DMS-5003 FFT processor was used to store the results and transform them into the frequency domain, thus giving the natural frequency for the mode.

Before using the inverse eigensolver method to determine moments of inertia, the method was first tested for the computation of weight. The weight (mass) of the compressor and mounting was found by lifting the system with a load cell supported on a hydraulic jack. Next, using the known spring stiffnesses and the natural frequencies found for the translational modes, equations (71) were applied. The three values computed for \( m \) agreed with each other to within 1%; they agreed with the load cell weight to within 3%.

Applying equations (72) to the same stiffness data, the moments of inertia were computed for the rotational modes. The results were

\[
\begin{array}{ll}
\text{WEIGHT} & 620 \text{ LBF} \\
X \text{ MOM-OF-INERTIA} & 390 \text{ LBF-}\text{S}\times\text{S} \\
Y \text{ MOM-OF-INERTIA} & 370 \text{ LBF-}\text{S}\times\text{S} \\
Z \text{ MOM-OF-INERTIA} & 85 \text{ LBF-}\text{S}\times\text{S}
\end{array}
\]

The mass and mass moments of inertia of the system, along with the mounting data for the neoprene springs, were included in an analysis file for the simulation program. The analysis file is given below.
<table>
<thead>
<tr>
<th>Name</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
<th>Value 4</th>
<th>Value 5</th>
<th>Value 6</th>
<th>Value 7</th>
<th>Value 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.5</td>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>R2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>R3</td>
<td>0.0</td>
<td>0.0</td>
<td>-0.5</td>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>R4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>-0.5</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>R5</td>
<td>0.0</td>
<td>-0.5</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>R6</td>
<td>0.0</td>
<td>0.0</td>
<td>-1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Experimental Verification 73
To verify the results of the simulation, accelerations were recorded for the three principal directions at two points on the system; the center of mass and an arbitrary point at the corner of the steel mounting plate. The arbitrary point was offset from the center of mass to verify the simulation of rotational degrees-of-freedom.

For the physical tests, a population of 1024 samples was recorded at a rate of 2 milliseconds per sample. The results of 10 such measurements were averaged in the frequency domain. The signal from the accelerometer was sampled for 2.048 seconds. The simulation was run for 2.0 seconds at a time step of 2 milliseconds.

The graphs on the following pages present a comparison of the physical test results (top) and those predicted by the simulation (bottom). The acceleration units (Y axis) are g's (386.4 in/s²). Beside each graph is a table presenting the magnitudes (in g's) of the first 5 harmonics of the compressor running speed (e.g. first harmonic = 1 X running speed, second harmonic = 2 X running speed, etc.).

Note that for most cases, the correlation for the lower harmonics is good, usually within 8%. The simulation provides correlation to within 15% to 62% for third harmonics. Fourth and fifth harmonics are non-existent for most simulation results. However, the magnitude of the acceleration signal at those frequencies is generally low and was taken to be insignificant.
Figure 22. X acceleration at center of mass
Figure 23. Y acceleration at center of mass
Figure 24. Z acceleration at center of mass
Figure 25. X acceleration at corner of mounting plate
Figure 26. Y acceleration at corner of mounting plate
Figure 27. Z acceleration at corner of mounting plate
Two distinctive differences between the actual and predicted results need to be addressed. First, note that the curves representing the predicted results tend to be cleaner than those representing the actual results. The physical system contains high frequency harmonics superimposed on top of the dominant acceleration signal. The absence of these harmonics in the model may be attributed to several reasons:

- As noted, a finite number of terms is used to approximate the Taylor series representing piston acceleration. Because piston acceleration is the basis for the shaking forces and moments, these terms may be significant to the ultimate system response.
- Small fluctuations in the torque are a result of the reactions of the poles of the electric motor. These variations occur at a multiple of the running frequency, depending on the number of poles. In this model, they occur at four times running frequency.
- The model file may not contain enough of the coefficients necessary to fully describe the forcing functions.
- Finite clearances between mechanical parts can lead to high-frequency, low-amplitude forces.

Although these frequencies are missing from the output spectrum, the amplitude of the dominant acceleration signal is comparable for the predicted and actual results.

A second difference is noted in the prediction of Z (axial) accelerations. In both cases, note the Z acceleration is predicted to be lower than the actual results (see Figure 24 and Figure 27). Again, this difference may be attributed to several reasons:

- The significant effects of the compressor driving motor are disregarded in the modeling process. The windings of an electric motor are usually skewed to prevent reverse rotation. This can lead to a harmonic forcing along the axis of the motor.
- Because the idealized slider-crank model defines no explicit Z direction forces, the only predicted acceleration in this direction is due to side affects from rotation about the X or Y axes.
In summary, the modeling and simulation methods provide an acceptable reflection of the true system performance. A recognized problem exists in the source and modeling of high frequency force components. However, these components generally fall into a range of acceptably low magnitudes.
Recommendations for Further Research

This thesis served as an introduction to the complex task of compressor modeling, the major concern being the simulation of the vibratory response during steady-state operation. There are several possible extensions of this work that should be addressed.

The simulation of compressor start-up and shut-down is critical. During this transient phase, the compressor mountings are often forced through one or more natural frequencies of vibration. This can lead to possible damage to the machine or surroundings. Note that the shaking forces and moments computed in this work are for a constant running speed. Therefore, to accurately model the start-up and shut-down phenomenon, the force equations must be integrated simultaneously with the equations of motion.

In the section outlining the thermodynamics of the compressor, a number of modeling assumptions were presented. Whereas these assumptions tend to make the modeling task simpler, they also tend to limit the accuracy of the model. Therefore, allowing the engineer to more accurately describe the thermodynamics of the design would be a worthwhile extension of the model.

Other modeling assumptions were presented in the section describing compressor kinematics. These assumptions included the lumping of the masses of the connecting rod, crank, pistons, and pins, again to simplify the modeling task. A more accurate model would account for the distribution of the mass of the connecting rod and crank, and the effects of the inertia of these components.
These extensions serve to enhance some of the shortcomings of the modeling procedure presented in this work.
References


Appendix A. Pre-Processor User’s Guide

This appendix presents a user’s guide for using the PRESYM pre-processor program on the IBM PC.
Introduction

The PRESYM pre-processor program allows the engineer to model a compressor design for the purpose of vibration analysis. Modules are included that offer animation of the compressor configuration, XY plots of shaking forces, and computation and display of torque requirements. The user may model either transverse or axial reciprocating compressors.

The output of the PRESYM program consists of an analysis file used to communicate with the SYM dynamic simulation program. The analysis file includes data that describe the complex shaking forces and moments, spring and damper attachments, and mass and moments of inertia of the compressor body.

PRESYM offers the engineer the ability to add, modify, and remove counterweights for machine balance. Counterweights may be added at cylinder locations, or at any arbitrary location along the crank shaft.

This manual describes the use of the PRESYM program and introduces the SYM and POSTSYM programs.

Starting PRESYM

The PRESYM program requires an IBM PC/XT/AT with 320 KB main memory and a color/graphics (CGA) or enhanced graphics (EGA) adapter.

To start PRESYM, place the program diskette in drive A. Type
A>PRESYM  <cr>

at the prompt.

After PRESYM loads, the graphics manager displays

   enter graphics card -->

If your system has a CGA adapter, enter 0 <cr>.
If your system has a EGA adapter, enter 1 <cr>.
It is important to enter the proper adapter code as PRESYM will adjust all screen and printer output based on that value.

PRESYM will initialize the graphics manager and display the main screen. The screen layout consists of four sections:

- The status area used for displaying the model title, time and date, graphics status, and any prompts or messages.
- The menu area used to display menus and highlight menu selections.
- The plot area used for some intermediate data plots.
- The main display area used for model display and animation.

Menu Structure

The PRESYM menus are controlled by the cursor keys on the right of the keyboard. By pressing the down arrow key (↓), the next lowest option is highlighted; by pressing the up arrow key (↑), the next highest option is highlighted. The highlighted option is selected by pressing the enter key.
At each menu, pressing the Alt-H key sequence will cause *PRESYM* to generate a hardcopy of the current screen. If no graphics printer is attached, the Alt-H command is ignored.

All menus (with the exception of the main menu) have the option

**BACKSPACE**

This option causes *PRESYM* to exit to the next highest menu level. At the top level (the main menu), selecting

**EXIT**

causes *PRESYM* to end.

**Recalculation**

Because calculation of shaking forces, moments, and torque requirements may take some time, *PRESYM* will only perform these calculations when requested. Whenever a menu option that requires computation is selected, *PRESYM* prompts

**OK TO RECALCULATE? [N]#**

If the last set of calculations performed is still valid (i.e. no changes have been made to the model), press the enter key. Otherwise, enter Y <cr>.

*Appendix A. Pre-Processor User's Guide*
The PRESYM Main Menu

These are the options found on the main PRESYM menu and a short description.

EDIT
The EDIT option allows the user to specify or change the configuration of the model. When EDIT is selected, the graphic display is replaced by an interactive, spread-sheet style input screen.

FILES
The FILES option is used to recall and store models. The option also includes the ability to create the analysis file for dynamic processing. Another FILES option allows the model memory to be cleared.

DYNAMICS
The DYNAMICS option is used to compute and graphically display shaking forces, shaking moments, and torque requirements. Plots are made in either the time domain or frequency domain and for one or all pistons.

ANIMATE
ANIMATE allows the compressor pistons to be displayed and animated through the complete 360° cycle.

DRAW
The DRAW option displays all pistons in the main display area. The angle of the crank is given as an option.

OPTIONS
The OPTIONS menu allows the view of the model to be changed from the default isometric. Other views are along the three principal axes.
The EDIT Option

Selecting the EDIT option displays the \textit{PRESYM} editing screen. At the bottom of the display, the function key commands are displayed:

\begin{verbatim}
F1 Edit Model  F2 Edit Pistons  F3 Edit Counterweights  Esc Exit
\end{verbatim}

The option needed is selected by pressing a function key. The Esc key returns \textit{PRESYM} to the main screen and menu.

The \textit{PRESYM} editor operates in a spreadsheet fashion. The highlighting cursor is moved from one parameter to the next with the arrow keys on the extreme right of the keyboard. When the up or down arrow key is pressed, the highlighted parameter is entered into the model database and the next parameter is selected.

The carriage return key functions identically to the down arrow key.

When editing piston or counterweight data, the PgDn key selects the next piston or counterweight record. Similarly, the PgUp key selects the previous record. Both keys will wrap around to the top of the list of pistons or counterweights.

When any edit screen is activated, an input window becomes visible at the bottom of the display. The data for the highlighted parameter is entered here and then displayed.
**Editing the Model**

These parameters are required as part of the overall model definition.

- Number of Stages
- Number of Cylinders
- Crank Radius \( \text{INCHES} \)
- Connecting Rod Length \( \text{INCHES} \)
- Input Pressure \( \text{PSIG} \)
- Output Pressure \( \text{PSIG} \)
- Crank Speed \( \text{RPM} \)
- Input Ambient Temperature \( ^\circ\text{F} \)
- Ratio of Specific Heats

**Editing the Pistons**

These parameters are required for each piston definition.

- Cylinder Diameter \( \text{INCHES} \)
- Cylinder Angle \( \text{DEGREES} \)
- % Clearance
- Operating Stage
- Reciprocating Weight \( \text{LBS} \)
- Crank Angle \( \text{DEGREES} \)
- Axial Position \( \text{INCHES} \)
- Rotating Weight \( \text{LBS} \)
- Counterbalance Weight \( \text{LBS} \)
- Counterbalance Radius \( \text{INCHES} \)
• Counterbalance Angle

DEGREES

The cylinder and crank angles are measured positive counter-clockwise with zero being vertical up. The counterweight angle is measured with zero being 180° from the crank angle. See Figure 28.

Editing the Counterweights

These parameters are required for each counterweight definition.

• Weight
  LBS
• Axial Position
  INCHES
• Radius
  INCHES
• Angle
  DEGREES

The FILES Option

There are four items that make up the FILES menu. They are

RECALL MODEL This option allows the users to recall a compressor model that has been stored on disk. The prompt

ENTER FILE NAME#

is displayed. Only the file name needs to be entered; PRESYM looks for a file with the extension .MOD.

STORE MODEL This option allows the user to store the model that is in memory. The prompt
Figure 28. Angle conventions used by PRESYM

\[ a = \text{CYLINDER ANGLE} \]
\[ b = \text{CRANK ANGLE} \]
\[ c = \text{COUNTERWEIGHT ANGLE} \]
ENTER FILE NAME#

is displayed. Again, only enter a file name; PRESYM will create a file with the extension .MOD.

ANALYSIS

This option creates the analysis file. The analysis file stores the data describing the shaking forces and moments affecting the compressor body. The prompt

ENTER FILE NAME#

is displayed. Enter only a file name; PRESYM adds the extension .ANA.

CLEAR MEMORY

This option clears the entire model database from memory and erases all default values. It may be used after a model has been stored and a new model is to be started.

The DYNAMICS Option

The DYNAMICS menu is used for the computation of shaking forces, shaking moments, and torque requirements. The user may optionally perform these calculations for one or all pistons to examine individual effects. These options are on the DYNAMICS menu:

SHAKING FORCE

This option is used to compute the shaking forces and moments for one or all pistons. The sub-menu is displayed:

ALL
BACKSPACE
PISTON 1
PISTON 2


Appendix A. Pre-Processor User's Guide
Select the desired option. A second sub-menu is displayed:

TIME DOM
FREQ DOM

The TIME DOM option presents the shaking forces and moments as a function of the crank shaft angle. The FREQ DOM option displays the magnitudes of the frequency components found in the curves.

**TORQUE REQ**

This option computes the torque requirements for one or all pistons. The sub-menu is displayed:

ALL
BACKSPACE
PISTON 1
PISTON 2

Select the desired option. As the torque is computed, *PRESYM* displays the crank shaft angle in the status area. The torque curve is presented in both the time and frequency domains.

**The ANIMATE Option**

The ANIMATE option allows the user to verify the geometry of the compressor model. The sub-menu is displayed:

ALL
BACKSPACE
PISTON 1
PISTON 2

By selecting an individual piston, *PRESYM* will concurrently display the torque, inertia force, and displacement curves of the piston in the plot area. When a selection is made, *PRESYM* computes

Appendix A. Pre-Processor User’s Guide 97
and stores 36 animation frames. Animation of the frames may be interrupted by pressing the space bar.

Due to performance and memory limitations, PRESYM will only show up to 5 pistons simultaneously.

The DRAW Option

When the DRAW option is selected, PRESYM displays the prompt

ENTER CRANK ANGLE#

Enter the position of the crank (in degrees) at which you wish to view the piston configuration.

The OPTIONS Option

The items available on the OPTIONS menu currently include

VIEW This option allows the user to view the image in the main display area from a different position. Possible views include isometric, XY plane, YZ plane, and XZ plane.
Appendix B. Processor User’s Guide

This appendix presents a user’s guide for using the $SYM$ dynamic processor program on the IBM PC or VAX/VMS system.
Introducción

El programa de simulación $SYM$ es una herramienta general de modelado dinámico y análisis. Su principal uso es en la investigación del movimiento de un solo cuerpo con seis grados de libertad. El usuario puede especificar la adición de elementos de resistencia y amortiguación generales y aplicar una variedad de fuerzas variables con el tiempo.

$SYM$ y sus programas complementarios, $PRESYM$ y $POSTSYM$, forman parte de un sistema completo de análisis de compresores recíprocos. El programa $SYM$, sin embargo, es independiente de los detalles del diseño del compresor.

$SYM$ está diseñado para correr en cualquier sistema que soporte el estándar ANSI FORTRAN-77. Algunas extensiones no estándar son necesarias, pero estas se encuentran en la mayoría de las implementaciones FORTRAN-77.

Starting SYM

El método utilizado para iniciar $SYM$ depende del sistema operativo que se esté utilizando.

Para un sistema IBM PC tipo, instale el disco de programa e inserte en el drive y escriba

```
A>SYM <cr>
```

al prompt DOS.

Para un sistema VAX/VMS, inicie el programa $SYM$ escribiendo

For an IBM PC type system, install the program diskette and enter

```
A>SYM <cr>
```

at the DOS prompt.

For a VAX/VMS system, begin the SYM program by entering
at the VMS prompt.

When SYM loads, it sends the following prompt to the terminal

```
ANALYSIS FILE#
```

Enter the name of the prepared analysis file. The extension for this file must be .ANA.

Files Created by SYM

The SYM program creates two output files during the course of the simulation. The first, xxx.LST, contains data for the user which includes a summary of the model. The second, xxx.OUT, contains data for the post-processing program, POSTSYM. The xxx is the name of the analysis file specified when SYM was loaded.

The size these two files will depend on the model simulation job being executed, the running time, and the time step used. In any event, insure the local disk storage device has ample space available.

On Units

The SYM program has been designed to be independent of the measurement system (units) used in the model. Any system of units convenient to the designer may be used; however, use of units should be consistent.
For instance, to use the metric system, valid units would be kilograms (kg) for mass, meters (m) for length and distance, and Newtons (N) for force.

To use an English system, valid units would be slugs for mass, inches (in) for length and distance, and pounds (lb) for force. For this system, a body’s mass can be expressed as its weight in pounds divided by the gravitational constant (386.4 in/sec²).

All frequencies (forces, natural frequencies) are given in radians/sec.

The SYM Input File

The SYM input file is used to communicate the model configuration to the processor. The file consists of a various data cards, each serving a particular input function. Each input card has a single character mnemonic in column 1 defining it’s function.

The following lists the available SYM input cards and their syntax.

Note: All data is read by the processor input module in a free format; the formats given below are for presentation only. However, the function mnemonic must always appear in column 1 and all variables must be defined, even if they are zero.
**The COMMENT Card**

The data on a COMMENT card is disregarded by the processor. A comment card must have a C or a blank in the first column.

**The TIME Card**

The TIME card is used to specify the simulation time and job type. The format of the TIME card is

<table>
<thead>
<tr>
<th>T</th>
<th>TSTA</th>
<th>TEND</th>
<th>TSTP</th>
<th>IJOB</th>
<th>ISSV</th>
<th>PER</th>
<th>NSSV</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2-10</td>
<td>11-20</td>
<td>21-30</td>
<td>31-40</td>
<td>41-50</td>
<td>51-60</td>
<td>61-70</td>
</tr>
</tbody>
</table>

where

- **TSTA** is the start time for the simulation
- **TEND** is the end time for the simulation
- **TSTP** is the simulation time step
- **IJOB** is the job code
  - 1 = eigenanalysis only
  - 2 = full time simulation
  - 3 = steady-state single cycle simulation
- **ISSV** is the variable to be tested for steady-state
- **PER** is the steady-state tolerance
- **NSSV** is the initial number of time steps per cycle

See the following sections for detailed information concerning IJOB and other TIME card parameters.
The MASS Card

The MASS card defines the mass of the body and its principal moments of inertia about the center of gravity of the body. The format of the MASS card is

<table>
<thead>
<tr>
<th>M</th>
<th>BMASS</th>
<th>XJ</th>
<th>YJ</th>
<th>ZJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2-10</td>
<td>11-20</td>
<td>21-30</td>
<td>31-40</td>
</tr>
</tbody>
</table>

where

- **BMASS** is the mass of the body
- **XJ** is the moment of inertia about the X axis
- **YJ** is the moment of inertia about the Y axis
- **ZJ** is the moment of inertia about the Z axis

The SPRING / DAMPER PROPERTY Card

The SPRING/ DAMPER PROPERTY card is used to create a table of available springs and dampers. This table is accessed by the SPRING / DAMPER LOCATION card. The format of the SPRING / DAMPER PROPERTY card is

<table>
<thead>
<tr>
<th>S</th>
<th>ISPR</th>
<th>STF</th>
<th>DMP</th>
<th>UNF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2-8</td>
<td>9-20</td>
<td>21-32</td>
<td>33-44</td>
</tr>
</tbody>
</table>

where

- **ISPR** is the spring /damper property code
- **STF** is the spring constant
- **DMP** is the coefficient of damping
- **UNF** is the static length of the spring

Appendix B. Processor User's Guide
You may specify a null value for either STF or DMP. If a null value is given for STF, UNF may also be null.

_The SPRING / DAMPER LOCATION Card_

The SPRING/ DAMPER LOCATION card is used to attach springs and dampers to the body. This card accesses the table created by SPRING / DAMPER PROPERTY cards. The format of the SPRING / DAMPER LOCATION card is

<table>
<thead>
<tr>
<th>L</th>
<th>ISPR</th>
<th>BDX</th>
<th>BDY</th>
<th>BDZ</th>
<th>GRX</th>
<th>GRY</th>
<th>GRZ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2-8</td>
<td>9-20</td>
<td>21-32</td>
<td>33-44</td>
<td>45-56</td>
<td>57-68</td>
<td>69-80</td>
</tr>
</tbody>
</table>

where

- **ISPR** is the spring/damper property code
- **BDX** is the local X coordinate of the body attachment
- **BDY** is the local Y coordinate of the body attachment
- **BDZ** is the local Z coordinate of the body attachment
- **GRX** is the global X coordinate of the ground attachment
- **GRY** is the global Y coordinate of the ground attachment
- **GRZ** is the global Z coordinate of the ground attachment

_The FORCE PROPERTY Card_

The FORCE PROPERTY card is used to create a table of available forces. The table is accessed by the FORCE LOCATION card. The format of the FORCE PROPERTY CARD is

<table>
<thead>
<tr>
<th>F</th>
<th>IFOR</th>
<th>ITYP</th>
<th>STIM</th>
<th>ETIM</th>
<th>AMP</th>
<th>FREQ</th>
<th>ACL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2-8</td>
<td>9-20</td>
<td>21-32</td>
<td>33-44</td>
<td>45-56</td>
<td>57-68</td>
<td>69-80</td>
</tr>
</tbody>
</table>

where
IFOR is the force property code
ITYP is the force type
1 = step force
2 = sinusoidal force
3 = complex force
STIM is the time the force is initially applied
ETIM is the time the force is removed
AMP is the amplitude of the force
FREQ is the frequency of the force
ACL is the frequency acceleration of the force

FOR STEP FORCES: the FREQ and ACL parameters are ignored. The force is applied as a constant value of AMP.

FOR SINUSOIDAL FORCES: the ACL parameter represents the change in frequency per second. The frequency of the applied force will be zero at the start of the simulation and will increase at a rate of ACL until the given frequency is reached.

FOR COMPLEX FORCES: the AMP parameter represents an optional multiplier of the amplitude defined by the FFT coefficients. The ACL parameter is used to define the number of coefficient pairs used in the force description. The coefficient pairs must follow on the next ACL cards in the input file.

The FORCE LOCATION Card

The FORCE LOCATION card is used to apply forces to the body. This card accesses the table created by FORCE PROPERTY cards. The format of the FORCE LOCATION card is

<table>
<thead>
<tr>
<th>R</th>
<th>IFOR</th>
<th>BDX</th>
<th>BDY</th>
<th>BDZ</th>
<th>ORX</th>
<th>DRY</th>
<th>ORZ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2-8</td>
<td>9-20</td>
<td>21-32</td>
<td>33-44</td>
<td>45-56</td>
<td>57-68</td>
<td>69-80</td>
</tr>
</tbody>
</table>

where
IFOR is the force property code
BDX is the local X coordinate of the force application
BDY is the local Y coordinate of the force application
BDZ is the local Z coordinate of the force application
ORX is the X component of the force orientation
ORY is the Y component of the force orientation
ORZ is the Z component of the force orientation

A Sample Input File

This file defines a single mass mounted on a spring and damper. A small sinusoidal force is applied.

```
c analysis time card
   0.0  10.0  0.1  3

model parameter card
   1.0  10.0  10.0  10.0

spring / damper property cards
   1   100.0  10.0  1.0

spring / damper location cards
   1   0.0  0.0  0.0  0.0  0.0  0.0 -1.0

force property cards
   1   2   0.0  10.0  10.0  30.0  0.0

force location cards
   1   0.0  0.0  0.0  0.0  0.0  0.0  1.0
```
**Modeling Techniques**

The *SYM* program formulates spring and damping forces in a non-linear fashion. Therefore, motion in one degree-of-freedom will affect the forces in the other degrees-of-freedom.

*SYM* does not, however, account for the lateral stiffness of a spring. All springs are considered to have stiffness only along their axis. Likewise, a damper restricts only the component of velocity that lies along its axis. Lateral stiffnesses may be modeled using two spring/damper elements orthogonal to the primary element. *This is necessary to provide some constraint against rigid body motion in directions that are not grounded in the physical system.*

**Simulation Modes**

The *SYM* program operates in one of three analysis modes as determined by the IJOB parameter of the TIME card.

**Eigensolver Analysis (IJOB = 1)**

In this mode, the *SYM* program computes only the eigenvalues and eigenvectors for the input file. The results are sent to the listing file and the output file. No further processing of the model is performed.
Full Time Simulation (IJOB = 2)

The full time simulation mode performs a complete analysis starting and ending at the specified times. In addition, the eigenanalysis is performed. All results are sent to the listing and output files.

Status messages are sent to the terminal as the simulation proceeds.

Steady-State One Cycle Simulation (IJOB = 3)

The steady-state mode allows $SYM$ to store the data found during one cycle of periodic motion. The period of the model is determined by the frequency of the first defined force; therefore, the first force must exist and must be of either type 2 (sinusoidal) or 3 (complex).

The simulation starts with the time specified on the TIME card. The time step is also computed from the frequency of the first force defined in the input file. The value of the time step is equivalent to NSSV steps per force cycle.

Steady-state is determined by computing the root mean squared (RMS) of the values of one coordinate or all coordinates during a cycle. This value is compared to the previous RMS after each cycle. When the difference between two consecutive RMS values is less than a given percentage of the current RMS, steady-state is detected.

\[
\left| \frac{A_{\text{CURRENT}} - A_{\text{PREVIOUS}}}{A_{\text{CURRENT}}} \right| \times 100 < PER
\]

(73)

Steady-state will not be detected until at least 2 cycles have occurred.
Once steady-state is detected, *SYM* forces a new time step equivalent to 128 steps per cycle. The simulation is advanced until the start of a cycle is found; 128 additional time steps then define a single cycle. Output is to both the listing and output files.

The *TIME* card parameters are used to define the steady-state characteristics. Recall the *TIME* card definition as

<table>
<thead>
<tr>
<th>T</th>
<th>TSTA</th>
<th>TEND</th>
<th>TSTP</th>
<th>IJOB</th>
<th>ISSV</th>
<th>PER</th>
<th>NSSV</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2-10</td>
<td>11-20</td>
<td>21-30</td>
<td>31-40</td>
<td>41-50</td>
<td>51-60</td>
<td>61-70</td>
</tr>
</tbody>
</table>

The last three parameters on the time card specify the method of steady-state detection.

**ISSV** specifies which of the coordinates are checked for steady-state characteristics. Values 1-3 select X, Y, Z translation; values 4-6 select X, Y, Z rotation. A value of 0 selects all coordinates for simultaneous testing.

**PER** specifies the percentage of the current average that is used to test against the difference in consecutive averages.

**NSSV** specifies how many time steps are used per cycle prior to detection of steady-state. Up to 128 time steps per cycle are allowed. Steady-state will not be detected until at least 2 cycles of NSSV steps have occurred.

To insure steady-state is reached and all start-up transients are removed, *SYM* applies an exponentially decaying damping in the X, Y, and Z directions. The damping has an initial value equivalent to critical damping for the system. After five cycles, the damping decays to 10% of critical and continues to approach zero.
The SYM Output Files

A SYM run creates two files: the listing file and the output file.

The Listing File

The listing file contains a summary of the input model, the mass and stiffness matrices for the model, the natural frequencies and normalized mode shapes for the model, and the simulation run time. This file is meant to be read by the program user.

The Output File

The output file contains the simulation results. It is designed to be read by an output processor, such as the POSTSYM post-processing program. The format of the output file is as follows:

LINE 1
Contains the six natural frequencies of the system in radians/sec.

LINE 2-7
Contains the normalized modes of the system. Each column contains the modal vector corresponding to the frequency in line 1. Lines 2-4 have the relative X, Y, and Z displacements; lines 5-7 have the relative X, Y, and Z rotations.

LINE 8-
Contain the displacement and velocity of the body in time. At each time step, two lines are printed: the first contains the simulation time and six displacements; the second contains six velocities. For a full-time simu-
lation (IJOB = 2), the time ranges from the start time (TSTA) to the end time (TEND). For a steady-state simulation (IJOB = 3), the time ranges from 0 to the length of a force cycle.

For an eigenanalysis (IJOB = 1), no lines are printed after line 7.
Appendix C. Post-Processor User’s Guide

This appendix presents a user’s guide for using the POSTSYM pre-processor program on the IBM PC.
Introduction

The POSTSYM post-processing program allows the compressor analyst to review the results of a steady-state, dynamic simulation of a compressor design. Modules are included for the animation of the rigid body motion, XY plots of the body translation and rotation, and XY plots of arbitrary point translation.

The output from POSTSYM is graphic only. It serves as an interface between the data produced by the POSTSYM companion program, SYM, and the designer. The program layout and design are similar to the pre-processor, PRESYM.

This manual describes the use of the POSTSYM program as part of a package including the SYM and PRESYM programs.

Starting POSTSYM

The POSTSYM program requires an IBM PC/XT/AT with 320 KB main memory and a color/graphics (CGA) or enhanced graphics (EGA) adapter.

To start POSTSYM, place the program diskette in drive A. Type

A>POSTSYM <cr>

at the prompt.
After POSTSYM loads, the graphics manager displays

\textit{enter graphics card} -->

If your system has a CGA adapter, enter 0 <cr>. If your system has a EGA adapter, enter 1 <cr>. It is important to enter the proper adapter code as POSTSYM will adjust all screen and printer output based on that value.

POSTSYM will initialize the graphics manager and display the main screen. The screen layout consists of four sections:

- The status area used for displaying the model title, time and date, graphics status, and any prompts or messages.
- The menu area used to display menus and highlight menu selections.
- The plot area used for some intermediate data plots.
- The main display area used for model display and animation.

\textbf{Menu Structure}

The POSTSYM menus are controlled by the cursor keys on the right of the keyboard. By pressing the down arrow key (\textdagger), the next lowest option is highlighted; by pressing the up arrow key (\textdaggerdbl), the next highest option is highlighted. The highlighted option is selected by pressing the enter key.

At each menu, pressing the Alt-H key sequence will cause POSTSYM to generate a hardcopy of the current screen. If no graphics printer is attached, the Alt-H command is ignored.

All menus (with the exception of the main menu) have the option
This option causes POSTSYM to exit to the next highest menu level. At the top level (the main menu), selecting

EXIT

causes POSTSYM to end.

The POSTSYM Main Menu

These are the options found on the main POSTSYM menu and a short description.

FILES

This option is used to load a SYM output file into the POSTSYM database. The file load routine is called automatically if certain other menu selections are made before a file has been loaded.

ANIMATE

Animates the mode shapes of the model or the results of a simulation. The simulation may be either periodic or full time.

PLOT

Plots the output of the SYM processor. Displacements of the center of gravity of the body may be plotted or, optionally, of any point on the body.

OPTIONS

Change the size of the displayed model, the rotation and translation scales, or the view of the model.
The FILES Option

When FILES is selected, the FILES menu is displayed.

RECALL MOD
BACKSPACE

The only available option, RECALL MOD, is used to load a SYM output file into the database. When this option is selected, POSTSYM prompts

ENTER FILE NAME

Enter the file name you used when the model was created. Do not enter an extension; POSTSYM automatically looks for .OUT.

The ANIMATE Option

The ANIMATE menu is used to animate mode shapes of the system and the motion of the body. The ANIMATE sub-menu is displayed:

SIMULATION
MODE SHAPES
BACKSPACE

SIMULATION The SIMULATION option animates the motion of the body for the duration of the simulation. If the simulation was a steady-state single cycle, that cycle is animated. If the simulation was a full time job, the entire simulation is animated. The displacement and rotation scales set through the OPTIONS menu affect the animation output.
While the animation frames are being computed, POSTSYM displays

**PREPARING VIEW FOR ANIMATION**

The preparation may be interrupted at any time by pressing the space bar.

When the animation begins, it may be interrupted by pressing the space bar.

**MODE SHAPES**

A mode shape represents the motion of the body when a certain natural frequency is excited. POSTSYM will animate the six mode shapes and display the corresponding natural frequency.

The mode shape menu is displayed:

```
MODE 1
MODE 2
MODE 3
MODE 4
MODE 5
MODE 6
LIST
BACKSPACE
```

The first six options animate the respective mode shapes. The animation is not affected by the displacement and rotation scales set on the OPTIONS menu. The LIST option lists the mode shapes (eigenvectors) and the corresponding natural frequencies on the main display area.
The PLOT Option

The PLOT option allows the results of the simulation to be plotted against time or crank angle (depending on the simulation job). Also, the translations of an arbitrary point may be plotted.

The PLOT sub-menu is displayed:

BODY
ACCLEROMTR
BACKSPACE

BODY

The BODY option plots the displacements and rotations for the body for the duration of the simulation.

ACCLEROMTR

When the ACCLEROMTR option is selected, POSTSYM prompts

ENTER ACCELEROMETER POSITION#

Enter the coordinates of the point (with respect to the center of gravity of the body) where translation plots are desired.

Neither of the PLOT options are affected by the displacement or rotation scales set under the OPTIONS menu.

The OPTIONS Option

The OPTIONS sub-menu is displayed:
The DISPLACMNT and ROTATION options allow you to set the relative scale of body motion for purposes of animation. By increasing the scales, the motion of the body is exaggerated. By zeroing the scales, the motion is eliminated.

When DISPLACMNT is selected, POSTSYM prompts:

ENTER DISPLACEMENT SCALE [xxxx]#

where xxxx is the current displacement scale.

When ROTATION is selected, POSTSYM prompts:

ENTER ROTATION SCALE [xxxx]#

where xxxx is the current rotation scale.

The SIZE option allows you to change the size of the machine representation used to animate the motion of the body. When SIZE is selected, POSTSYM prompts:

ENTER SIZES#

Enter the X, Y, and Z lengths of the body. The values are automatically scaled to fit properly on the display area.

The VIEW option allows the angle at which the animation is viewed to be changed. When VIEW is selected, POSTSYM highlights the view in the upper right of the plot area. The cursor keys are used to highlight the other views; the enter key selects the current view.
Appendix D. SYM Program Listing

The following pages present the FORTRAN code used to implement the dynamic processor program, SYM.

System Dependencies

Whereas the body of the SYM program is written using the ANSI FORTRAN-77 standard, some of the code is dependent upon the particular system. These dependencies are outlined below.

INCLUDE Files

The ability to include external files as part of the source program at compile time is a useful tool. However, the INCLUDE metacommand is applied differently on each system where SYM has been implemented.

MS FORTRAN77

The format of the INCLUDE metacommand is

$\text{INCLUDE: 'fn.ext'}$
where fn.ext is the file being included. The $ must appear in column 1.

VAX/VMS FORTRAN

The format of the INCLUDE metacommand is

```
INCLUDE 'fn.ext'
```

where fn.ext is the file being included. The INCLUDE must start in or past column 7.

VS FORTRAN77

The format of the INCLUDE metacommand is

```
INCLUDE (member)
```

where member is an entry in a macro library (MACLIB). The macro library must be made available at compile time with the GLOBAL MACLIB command. The INCLUDE must start in or past column 7.

The OPEN Statement

The OPEN statement allows the executing program to assign a logical unit number to an external file. Whereas the format is virtually identical on all three systems, the implementation is different.

MS FORTRAN77

The basic implementation of the OPEN statement is

```
OPEN (UNIT=nn,FILE='fn.ext',STATUS='stat')
```
where nn is the unit number, fn.ext is the file name, and stat is the current file status (OLD, NEW, UNKNOWN). The file name used is exactly as specified and may include any valid MS-DOS extensions.

VAX/VMS FORTRAN

The basic implementation of the OPEN statement is

```
OPEN (UNIT=nn,FILE='fn.ext',STATUS='stat')
```

where nn is the unit number, fn.ext is the file name, and stat is the current file status (OLD, NEW, UNKNOWN). The file name used is exactly as specified and may include any valid VMS extensions.

VS FORTRAN

The basic implementation of the OPEN statement is

```
OPEN (UNIT=nn,FILE='ft',STATUS='stat')
```

where nn is the unit number, ft is the file type, and stat is the current file status (OLD, NEW, UNKNOWN). The file name used will have name FILE, type fn, and mode A. Only a single contiguous character string may be used for ft.
The XTIME Subroutine

The XTIME subroutine is required by the main SYM program to display and print the system time during execution. The strings returned by XTIME are also used to compute elapsed time for a simulation.

The format of the XTIME subroutine header must be

```
SUBROUTINE XTIME (TVAR)
CHARACTER*8 TVAR
```

The time is returned in the variable TVAR in the form hh:mm:ss. While the exact implementation of XTIME will be different on each system, an example from VM/CMS is presented below.

```
SUBROUTINE XTIME (TVAR)
CHARACTER*8 TVAR
INTEGER TIMAR(10)
CALL DATIM (TIMAR)
WRITE (TVAR,100) TIMAR(5),TIMAR(4),TIMAR(3)
100  FORMAT (I2,':',I2,':',I2)
      IF (TVAR(1:1).EQ. ' ') TVAR(1:1) = '0'
      IF (TVAR(4:4).EQ. ' ') TVAR(4:4) = '0'
      IF (TVAR(7:7).EQ. ' ') TVAR(7:7) = '0'
      RETURN
END
```

Note in this example the call to subroutine DATIM. DATIM is a VM/CMS subroutine that returns the system time; it is not available on other systems.
PROGRAM SYM

VERSION 1.0

AUTHOR Greg Sherman

DATE SPRING 1987

DESCRIPTION dynamic simulation of spatial body with applied springs, dampers, and forces

HARDWARE DEC VAX 11/780 - MicroVAX

IBM PC/AT/XT

IBM 30xx - IBM 43xx

SYSTEM VAX/VMS - MS-DOS - VMS/CMS

LANGUAGE ANSI FORTRAN-77 WITH COMMON EXTENSIONS

FILES NEEDED global declarations in SYM.H

ADDTL NOTES * JACOBI subroutine courtesy of J. Kosmatka

PROGRAM SYM

IMPLICIT REAL*8 (A-H,O-Z)

$INCLUDE: 'SYM.H'

COMMON / TIME / TS,TJ,TP,TPP,IJOB,ISSV,SSP,NSSV

COMMON / MATRIX / M,MI

REAL*8 XI(12),XO(12)

REAL*8 M(6,6),MI(6,6)

CHARACTER*8 TIM1,TIM2,TIM3

DATA X / 12 * 0.000 /

DATA XO / 12 * 0.000 /

C open all files and read model

CALL FPREP

CALL GETMOD (IERR)

IF (IERR.NE.0) STOP ' '

C initialize listing file

CALL SUMARY

CALL BLDMM (M,MI)

CALL BLDSTF

IF (IJOB.EQ.1) GOTO 900

CALL XTIME (TIM1)

WRITE (*,*) TIM1,' : BEGIN SIMULATION'

5 T = TS

ISTP = 1

C main simulation loop

10 IF (IJOB.EQ.2.OR.IJOB.EQ.4) THEN

WRITE (40,1000) T,X(I),I=1,6,X(I),I=7,12

1000 FORMAT (F8.4,6E12.4/8X,6E12.4)

ENDIF

CALL RKGX (T,TP,X,XO,12)
DO 100 I = 1,12
   X(I) = XO(I)
100   CONTINUE

C handle a steady-state run

IF (IJOB.EQ.3.OR.IJOB.EQ.5) THEN
   CALL SSCHK (X,ISSV,SSP,NSSV,XRMS,IFLG)
   IF (IFLG.EQ.1) THEN
      CALL XTIME (TIM2)
      WRITE (*,*) TIM2, ' : STEADY-STATE FOUND & STEP ',ISTP
   ENDIF
   ELSE
      IJOB = 2
      TP = TPP
      GOTO 5
   ENDIF
ENDIF

ISTP = ISTP + 1

IF (T.LE.TF) GOTO 10

C end of main simulation loop

IF (IJOB.EQ.3.OR.IJOB.EQ.5) THEN
   WRITE (*,1020)
   WRITE (60,1020)
1020 FORMAT (' STEADY-STATE NOT REACHED IN SPECIFIED TIME')
ENDIF

500   CALL XTIME (TIM2)
   WRITE (*,*) TIM2, ' : END SIMULATION'
   CALL ELASPE (TIM1,TIM2,TIM3)
   WRITE (60,1010) TIM1,TIM2,TIM3
1010 FORMAT (' simulate started at ',A/ + ' simulation ended at ',A/ + ' elapsed time ',A)

900   CLOSE (40)
   CLOSE (50)
   CLOSE (60)

   STOP 'SYM END RUN'
END
C --- S S T A T E ------------------------------------------ SHERMAN ---
C
run a single period at steady-state

input
X current position vector
XO current velocity vector
T current time

output
to out file

referenced
RKGN
XTIME

SUBROUTINE SSTATE (X,XO,T)
IMPLICIT REAL*8 (A-H,O-Z)

$INCLUDE: 'SYM.H'
REAL*8 X(12),XO(12)
CHARACTER*8 TIMI

C compute new time step from frequency of force

TP = (2.0DO * ACOS (-1.0DO) / FRQ(I)) / 128.0DO

C go until start of cycle

100 TH = ACOS (COS (T * FRQ(I)))
CALL RKGN (T,TP,X,XO,12)
DO 110 I = 1,12
X(I) = XO(I)
110 CONTINUE
IF ((ABS (TH)-DTH).GT.0.0) GOTO 100
CALL XTIME (TIMI)
WRITE (*,*) TIMI,' : BEGIN STEADY-STATE CYCLE'

C go for one cycle

SST = 0.0DO

DO 300 II = 1,128
WRITE (40,1000) SST,(X(I),I=1,6),(X(I),I=7,12)
1000 FORMAT (F8.4,6E12.4/8X,6E12.4)
CALL RKGN (T,TP,X,XO,12)
DO 200 I = 1,12
X(I) = XO(I)
200 CONTINUE
SST = SST + TP

300 CONTINUE

CALL XTIME (TIMI)
WRITE (*,*) TIMI,' : END STEADY-STATE CYCLE'

RETURN
END

Appendix D. SYM Program Listing
SUBROUTINE FPREP

CHARACTER*80 AFILE

WRITE (*,100)

100 FORMAT (' ANALYSIS FILE# ',$)

READ (*,110) AFILE

110 FORMAT (A)

CALL MODFIL (AFILE,'OUT')
OPEN (UNIT=40,FILE=AFILE,STATUS='NEW')

CALL MODFIL (AFILE,'ANA')
OPEN (UNIT=50,FILE=AFILE,STATUS='OLD')

CALL MODFIL (AFILE,'LST')
OPEN (UNIT=60,FILE=AFILE,STATUS='NEW')

RETURN
END
check for steady-state characteristics

input
- \( X \) - current value of data
- \( \text{PER} \) - percentage of value to flag steady-state
- \( \text{NVAL} \) - number of values in a cycle

output
- \( \text{XRMS} \) - root mean square of last 32 values
- \( \text{IFLG} \) - result flag
- 0 = no steady-state
- 1 = steady-state reached

referenced
- none

SUBROUTINE 'SSCHK (X,ISSV,PER,NVAL,XRMS,IFLG)

IMPLICIT REAL*8 (A-H,O-Z)
REAL*8 X(12)
REAL*8 CDAT(128)

DATA IPTR / 1 /
DATA NCYC / 0 /
DATA PAVG / 0.000 /

IFLG = 0

store value in circular buffer

IF (ISSV.EQ.0) THEN
    CDAT(IPTR) = X(1) + X(2) + X(3) + X(4) + X(5) + X(6)
ELSE
    CDAT(IPTR) = X(ISSV)
ENDIF

IPTR = IPTR + 1

compute sum of cycle values to test

IF (IPTR.GT.NVAL) THEN
    NCYC = NCYC + 1
    AVG = 0.000
    DO 100 I = 1,NVAL
        AVG = AVG + CDAT(I) * CDAT(I)
    100 CONTINUE
    AVG = SQRT (AVG)
    DVAL = ABS (AVG - PAVG)
    CHK = ABS (AVG * PER / 100.000)
    IF (DVAL.LE.CHK) IFLG = 1
    PAVG = AVG
    IPTR = 1
ENDIF

RETURN
END
find the derivative of the current X vector

input
  T current time value
  N order of system (not used)
  X initial orientation vector

output
  XD derivative of orientation vector

SUBROUTINE DERIV (T,X,XO,N)

IMPLICIT REAL*S (A-H,O-Z)

REAL*S X(12),XO(12)

COMMON / MATRIX / M,MI

REAL*S M(6,6),MI(6,6)

REAL*S OR(4,4),F(6)

DO 100 I = 1,6

XO(I) = X(I+6)

100 CONTINUE

CALL BLDORN (X(1),X(4),OR)

CALL BLDF (T,OR,X(7),X(10),F)

CALL MULMAT (MI,F,6,6,1,XO(7))

RETURN

END
SUBROUTINE BLD (T,OR,RD,THD,FV)

IMPLICIT REAL*8 (A-H,O-Z)

REAL*8 OR(4,4),FV(6)
REAL*8 THD(3),RD(3)

$INCLUDE: 'SYM.H'

COMMON / TIME / TS,TF,TP,TPP,IJOB,ISSV,SSP,NSSV
REAL*8 F(3),M(3),SKS(3,3)

CALL ZERMAT (FV,1,6)
DO 100 I = 1,NS

C calculate force / moment vector due to spring
IF (FSTF(I,NE.0.000) THEN
    CALL SPRFOR (OR,SLO(I,1),SGR(I,1),ULEN(I),FSTF(I),F,M)
    CALL ADDMAT (FV(I),F,1,2,FV(I))
    CALL ADDMAT (FV(4),M,1,3,FV(4))
ENDIF

C calculate force / moment vector due to damper
IF (FDMP(I,NE.0.000) THEN
    CALL DMPFOR (OR,RO,THD,SLO(I,1),SGR(I,1),FDMP(I),F)
    CALL ADDMAT (FV(I),F,1,2,FV(I))
    CALL XPROD (SLO(I,1),F,M)
    CALL ADDMAT (FV(4),M,1,3,FV(4))
ENDIF

100 CONTINUE

C compute force / moment vector due to applied forces
DO 200 I = 1,NA

IF (IFTYP(I,EQ.FSTEP) THEN
    CALL STPFOR (T,AFV(I,1),STIM(I),ETIM(I),AMP(I),F)
ELSE IF (IFTYP(I,EQ.FSINE) THEN
    CALL SINFOR + (T,AFV(I,1),STIM(I),ETIM(I),AMP(I),FRQ(I),ACL(I),F)
ELSE IF (IFTYP(I,EQ.FCMXP) THEN
    CALL CPXFOR + (T,AFV(I,1),STIM(I),ETIM(I),FRQ(I),STF(1,1),CTF(1,1),

Appendix D. SYM Program Listing 131
NCOEF(I),F)
ELSE
CALL ZERMAT (F,1,3)
ENDIF
CALL ADDMAT (FV(1),F,1,3,FV(1))
CALL XPROD (ALO(1,1),F,H)
CALL ADDMAT (FV(4),M,1,3,FV(4))

200 CONTINUE

C apply critical damping if necessary to reduce transients
IF (IJOB.EQ.3.OR.IJOB.EQ.4.OR.IJOB.EQ.5) THEN
CALL CDAMP (T;RD,F)
CALL ADDMAT (FV(1),F,1,3,FV(1))
ENDIF
RETURN
END
SUBROUTINE SPRFOR (OR, SPLO, SPGR, ULEN, SPSTF, FS, MV)

REAL*8 OR(4, 4), SPLO(4), SPGR(4), FS(3), MV(3)
REAL*8 SPBD(4), DELT(3), UVEC(3)

find body attached point in global coordinates
CALL MULMAT (OR, SPLO, 4, 4, 1, SPBD)

DELT(1) = SPBD(1) - SPGR(1)
DELT(2) = SPBD(2) - SPGR(2)
DELT(3) = SPBD(3) - SPGR(3)

DLEN = SQRT (DELT(1)*DELT(1) + DELT(2)*DELT(2) + DELT(3)*DELT(3))
SPFO = (ULEN - DLEN) * SPSTF

compute unit vector
UVEC(1) = DELT(1) / DLEN
UVEC(2) = DELT(2) / DLEN
UVEC(3) = DELT(3) / DLEN

compute force components
FS(1) = UVEC(1) * SPFO
FS(2) = UVEC(2) * SPFO
FS(3) = UVEC(3) * SPFO

compute moment due to spring
CALL XPROD (SPLO, FS, MV)

RETURN
END
SUBROUTINE DMPFOR (OR, RD, THD, SPLO, SPGR, CDMP, FD)

IMPLICIT REAL*8 (A-H, O-Z)

REAL*8 OR(4,4), RD(3), THD(3)
REAL*8 SPLO(4), SPGR(4), FD(3)
REAL*8 SPBD(4), DELT(3), UVEC(3), DVEL(3)

CALL MULMAT (OR, SPLO, 4, 4, 1, SPBD)

CALL XPROD (THD, SPLO, DVEL)

DVEL(1) = RD(1) + DVEL(1)
DVEL(2) = RD(2) + DVEL(2)
DVEL(3) = RD(3) + DVEL(3)

DVEL(1) = SPBD(1) - SPGR(1)
DVEL(2) = SPBD(2) - SPGR(2)
DVEL(3) = SPBD(3) - SPGR(3)

OLEN = SQRT (DVEL(1)**2 + DVEL(2)**2 + DVEL(3)**2)

UVEC(1) = DVEL(1) / OLEN
UVEC(2) = DVEL(2) / OLEN
UVEC(3) = DVEL(3) / OLEN

VPRJ = DVEL(1) * (UVEC(1)) +
       DVEL(2) * (UVEC(2)) +
       DVEL(3) * (UVEC(3))

DPFO = -1.0DO * VPRJ * CDMP

FD(1) = UVEC(1) * DPFO
FD(2) = UVEC(2) * DPFO
FD(3) = UVEC(3) * DPFO

RETURN
END
SUBROUTINE SINFOR (T, OVEC, TS, TE, AMP, FRQ, ACL, FV)

IMPLICIT REAL*8 (A-H, O-Z)
REAL*8 OVEC(3), FV(3)
REAL*8 UVEC(3)

IF (T.LE.TS.OR.T.GT.TE) THEN
  FOR = 0.000
ELSE IF (ACL.EQ.0.0) THEN
  FOR = AMP * SIN ((T - TS) * FRQ)
ELSE
  FRQL = (T - TS) * ACL
  IF (FRQL.LT.FRQ) THEN
    FOR = AMP * SIN (0.600 * ACL * (T - TS) ** 2)
  ELSE
    FOR = AMP * SIN ((T - TS) * FRQ)
  ENDIF
ENDIF

ULEN = SQRT (OVEC(1)**2 + OVEC(2)**2 + OVEC(3)**2)
FV(1) = FOR * OVEC(1) / ULEN
FV(2) = FOR * OVEC(2) / ULEN
FV(3) = FOR * OVEC(3) / ULEN

RETURN
END
SUBROUTINE STPFO R (T,OV E C, TS, TE, AMP, FV)
IMPLICIT REAL*8 (A-H, O-Z)
REAL* 8 OVEC(3), FV(3)
REAL* 8 UV E C(3)

IF (T.LT.TS.OR.T.GT.TE) THEN
  FOR = 0.000
ELSE
  FOR = AMP
ENDIF

ULEN = SQRT (OVEC(1)*OVEC(1)+OVEC(2)*OVEC(2)+OVEC(3)*OVEC(3))
FV(1) = FOR * OVEC(1) / ULEN
FV(2) = FOR * OVEC(2) / ULEN
FV(3) = FOR * OVEC(3) / ULEN
RETURN
END
compute current value of applied complex force

input

T  time
OVEC  force orientation vector
TS  starting time of force
TE  ending time of force
AMP  amplitude of force
FRQ  frequency of force
ST  array of sine coefficients
CT  array of cosine coefficients
N  number of coefficients

output

FV  force vector

referenced

INVFFT

SUBROUTINE CPXFOR (T, OVEC, TS, TE, FRQ, ST, CT, N, FV)
IMPLICIT REAL*8 (A-H, O-Z)
REAL*8 OVEC(3), FV(3)
REAL*8 ST(1), CT(1)
REAL*8 UVEC(3)
data
pt / -1.0 /

IF (T.GE.TS.AND.T.LE.TE) THEN
    THETA = (T - TS) * FRQ
    CALL INVFFT (ST, CT, THETA, N, FOR)
ELSE
    FOR = 0.0D0
ENDIF

ULEN = SQRT (OVEC(1)**2 + OVEC(2)**2 + OVEC(3)**2)
FV(1) = FOR * OVEC(1) / ULEN
FV(2) = FOR * OVEC(2) / ULEN
FV(3) = FOR * OVEC(3) / ULEN

RETURN
END
apply critical damping for system

input  T  time
       RD  body velocity

output  F  damping force vector

 referenced  ZERMAT

SUBROUTINE CDAMP (T, RD, F)
IMPLICIT REAL*8 (A-H, O-Z)
$INCLUDE: 'SYM.H'

REAL*8    RD(3), F(3)
DATA ICYCLE / 5 /
DATA DECAY / 0.1D0 /

C determine the end time of cycle ICYCLE
   ET = FLOAT (ICYCLE) * 2.0D0 * ACOS (-1.0D0) / FRQ(1)

C compute decay rate based on DECAY parameter
   A = LOG (DECAY) / ET
   SCAL = EXP (A*T)

C compute force vector
   F(1) = (-CCRIT(1)) * RD(1) * SCAL
   F(2) = (-CCRIT(2)) * RD(2) * SCAL
   F(3) = (-CCRIT(3)) * RD(3) * SCAL

RETURN
END
C --- INVF FFT ------------------------------------------ SHERMAN ---
C perform an inverse FFT
C
C input       ST    array of sine coefficients
C CT       array of cosine coefficients
C THETA  crank angle = freq * t
C N    number of coefficients
C
C output       SUM    value of function at THETA
C
C referenced  none
C
SUBROUTINE INVFFT (ST,CT,THETA,N,SUM)
IMPLICIT REAL*8 (A-H,O-Z)
REAL*8 ST,CT

SUM = CT(1) / 2.0

DO 100 K = 1,N-1
SUM = SUM +
+ CT(K+1) * COS ((K+1)*THETA/2.000) +
+ ST(K+1) * SIN ((K+1)*THETA/2.000)

100 CONTINUE

RETURN
END

Appendix D. SYM Program Listing 139
SUBROUTINE BLDSTF

IMPLICIT REAL*8 (A-H,O-Z)

$INCLUDE: 'SYM.H'

COMMON / MATRIX / M,M1
REAL*8 M(6,6),M1(6,6)
REAL*8 KSYS(6,6),KSPR(6,6)
REAL*8 MI(3),M2(3),FI(3),F2(3)
REAL*8 RPOS(6),OR(4,4)
REAL*8 HK(6),D(6),Z(6,6)
CHARACTER*8 TIM1

DATA DELT / 1.0D-06 /

CALL XTIME (TIM1)
WRITE (*,*) TIM1,' : ASSEMBLING STIFFNESS MATRIX'
CALL LSTOUT (S,1)
WRITE (60,1000)
1000 FORMAT (' STIFFNESS MATRICES')
CALL ZERMAT (KSYS,6,6)
DO 500 I = 1,NS
C
C construct the displacement stiffness matrix for one spring
DO 100 II = 1,6
CALL ZERMAT (RPOS,6,1)
RPOS(II) = DELT / 2.0
CALL BLDORN (RPOS(1),RPOS(4),OR)
CALL SPRFOR (OR,SLO(1,1),SGR(1,1),ULEN(1),FSTF(1),F1,M1)
RPOS(II) = (-DELT) / 2.0
CALL BLDORN (RPOS(1),RPOS(4),OR)
CALL SPRFOR (OR,SLO(1,1),SGR(1,1),ULEN(1),FSTF(II),F2,M2)
KSPR(1,II) = ABS (F1(1) - F2(1)) / DELT
KSPR(2,II) = ABS (F1(2) - F2(2)) / DELT
KSPR(3,II) = ABS (F1(3) - F2(3)) / DELT
KSPR(4,II) = ABS (M1(1) - M2(1)) / DELT
KSPR(5,II) = ABS (M1(2) - M2(2)) / DELT
KSPR(6,II) = ABS (M1(3) - M2(3)) / DELT
100 CONTINUE
C
C clear entries smaller than delta value as insignificant
DO 110 II = 1,6
DO 120 JJ = 1,6
IF (KSPR(II,JJ),LE.DELT) KSPR(II,JJ) = 0.000
120 CONTINUE
110 CONTINUE
C
C add spring stiffness matrix to system

Appendix D. SYM Program Listing
CALL ADDMAT (KSYS, KSPR, 6, 6, KSYS)

WRITE (60, 1010) I, ((KSPR(JJ), JJ=1, 6), II=1, 6)
1010 FORMAT (' SPRING ', I2(5(3X, 6E12.4/), 3X, 6E12.4))

500 CONTINUE

WRITE (60, 1020) ((KSYS(JJ), JJ=1, 6), II=1, 6)
1020 FORMAT (' SYSTEM ', 5(3X, 6E12.4/), 3X, 6E12.4)

C compute X Y Z critical damping

CCRIT(1) = 2.000D0 * SQRT (M(1,1) * KSYS(1,1))
CCRIT(2) = 2.000D0 * SQRT (M(2,2) * KSYS(2,2))
CCRIT(3) = 2.000D0 * SQRT (M(3,3) * KSYS(3,3))

C find eigenvalues and eigenvectors and normalize

CALL JACOBI (KSYS, M, Z, D, HK, 6, DELT, 15, 0, 6, IERR)
CALL NORMAL (Z, 6)

IF (IERR.EQ.1) THEN

C

WRITE ('*', 1025)

C

WRITE (60, 1025)

1025 FORMAT (STIFFNESS MATRIX NOT POSITIVE DEFINITE')

DO 600 II = 1, 6

D(II) = 0.000D0

600 CONTINUE

ELSE

DO 610 II = 1, 6

D(II) = SQRT (D(II))

610 CONTINUE

ENDIF

C list file dump

CALL LSTOUT (S, 1)

WRITE (60, 1050) D(II), II=1, 6

1050 FORMAT (N A T U R A L F R E Q U E N C I E S'/3X, 6E12.4)

WRITE (60, 1040) (II, (REAL (Z(JJ), JJ=1, 6), II=1, 6)

1040 FORMAT (' H O M O D E S H A P E S'/


+ 5(3, 6E12.4/), 3, 6E12.4)

WRITE (40, 1050) (D(II), II=1, 6)

1050 FORMAT (6E12.4)

WRITE (40, 1060) ((Z(JJ), JJ=1, 6), II=1, 6)

1060 FORMAT (5(6E12.4/, 6E12.4)

RETURN

END
C --- N O R M A L ------------------------------- SHERMAN ---
C
normalize a matrix of eigenvectors
C
input X square matrix of columnar eigenvectors
N order of matrix
C
output X normalized vectors
C
referred none
C
SUBROUTINE NORMAL (X,N)
IMPLICIT REAL*8 (A-H,O-Z)
REAL*8 X(N,N)
DO 200 J = 1,6
   XMAX = X(1,J)
   DO 180 I = 2,6
      IF (ABS (X(I,J)).GT.ABS (XMAX)) XMAX = X(I,J)
      CONTINUE
   180 CONTINUE
   DO 190 I = 1,6
      X(I,J) = X(I,J) / XMAX
   190 CONTINUE
200 CONTINUE
RETURN
END

Appendix D. SYM Program Listing
C --- B L D M ----------------------------------- SHERMAN ---

build the mass matrix M

input none

output M mass matrix
MI inverse of mass matrix

referenced ZERMAT
LSTOUT

SUBROUTINE BLDM (M,MI)
IMPLICIT REAL*8 (A-H,O-Z)
COMMON / MODEL / xMS,xIX,xIY,xIZ
REAL*8 M(6,6),MI(6,6)

zero the matrices

CALL ZERMAT (M,6,6)
CALL ZERMAT (MI,6,6)

populate the matrix

M(1,1) = xMS
M(2,2) = xMS
M(3,3) = xMS
M(4,4) = xIX
M(5,5) = xIY
M(6,6) = xIZ

populate the inverse matrix

MI(1,1) = 1.0DD / xMS
MI(2,2) = 1.0DD / xMS
MI(3,3) = 1.0DD / xMS
MI(4,4) = 1.0DD / xIX
MI(5,5) = 1.0DD / xIY
MI(6,6) = 1.0DD / xIZ

CALL LSTOUT (S,1)
WRITE (60,1000) ((M(I,J),J=1,6),I=1,6)
1000 FORMAT (' M A S S M A T R I X'/(6X,6E12.4/))

RETURN
END
C --- MODFIL ----------------------------------------- SHERMAN ---
C modify a file name by adding an extension
C input AFILE file name
AEXT file extension
C output AFILE extension appended and uppercase
C referenced none
C
SUBROUTINE MODFIL (AFILE, AEXT)
CHARACTER*80 AFILE
CHARACTER*3 AEXT

C find position for extension
100 DO 130 I = 1, 80
   IF (AFILE(I:I).EQ.'.').OR.AFILE(I:I).EQ.' ') GOTO 140
130 CONTINUE
C concatenate file extension
140 AFILE(I:I) = '.
   DO 150 J = 1, 3
      AFILE(I+J:I+J) = AEXT(J:J)
150 CONTINUE
RETURN
END

Appendix D. SYM Program Listing 144
C --- SUMMARY ------------------------------- SHERMAN ---

dump summary of model database to listing file

input none

output to list file

referenced LSTOUT

SUBROUTINE SUMMARY

IMPLICIT REAL*8 (A-H,O-Z)

$INCLUDE:'SYM.H'

COMMON / TIME / TS,TF,TP,TPP,IJOB,ISSV,SSP,NSSV

CHARACTER*26 ANTPY(5)

DATA ANTPY / 'EIGENANALYSIS', 'FULL TIME SIMULATION',
+ 'STEADY-STATE SINGLE PERIOD', 'FULL TIME W/ CRIT DAMPING',
+ 'STEADY-STATE FULL TIME' /

CALL LSTOUT (S,1)
WRITE (*,2000) IJOB,ANTPY(IJOB),TS,TF,TP
WRITE (60,2000) IJOB,ANTPY(IJOB),TS,TF,TP

2000 FORMAT (' ANALYSIS TYPE ',I1,' : ',A//
+ ' START TIME : ',F10.4/
+ ' END TIME : ',F10.4/
+ ' TIME STEP : ',F10.4//)

WRITE (60,1000) NS

1000 FORMAT (' SPRING / DAMPER S',T74,I2)

WRITE (60,1010) (J,(SLO(I,J),I=1,3),(SGR(I,J),I=1,3),J=1,NS)

1010 FORMAT (T09,'LOCAL BODY ATTACHMENT',
+ T45,'GLOBAL GROUND ATTACHMENT'/
+ ' NO',T15,'X',T27,'Y',T39,'Z',T51,'X',T63,'Y',T75,'Z'/
+ (I3,6E12.4))

WRITE (60,1020) (I,FSTF(I),FDMP(I),ULEN(I),I=1,NS)

1020 FORMAT (I,' TYPE',I1,' ST TIME EN TIME AMP',
+ ' FREQ ACL'/
+ (I3,5E10.2))

RETURN

END

Appendix D. SYM Program Listing 145
--- LSTOUT ------------------------------------------ SHERMAN ---

write a string to the listing file

input
STR character string
IFLG function
0 = write string
1 = eject page and do not write string

output
to list file

referenced
none

SUBROUTINE LSTOUT (STR,IFLG)
CHARACTER*80 STR
DATA IPGNUM / 0 /

IF (IFLG.EQ.1) THEN
   IPGNUM = IPGNUM + 1
   WRITE (60,1000) IPGNUM
1000 FORMAT ('SYM SIMULATION LISTING FILE',T69,'PAGE ',I2//)
ELSE
   WRITE (60,1010) STR
1010 FORMAT (1X,A)
ENDIF

RETURN
END
```plaintext
-- build the body orientation matrix

input  R  position of center of gravity
       TH  vector of rotations

output OR  orientation matrix

referenced ROTMAT
          MULMAT

SUBROUTINE BLDORN (R,TH,OR)

IMPLICIT REAL*8 (A-H,O-Z)
REAL*8   R(3),TH(3),OR(4,4)
REAL*8   T1(4,4),T2(4,4),T3(4,4)
INTEGER  IMAP(3)
DATA     IMAP / 1, 2, 3 /

CALL ROTMAT (IMAP(1),(-TH(IMAP(1))),T1)
CALL ROTMAT (IMAP(2),(-TH(IMAP(2))),T2)
CALL MULMAT (T2,T1,4,4,4,OR)
CALL ROTMAT (IMAP(3),(-TH(IMAP(3))),T3)
CALL MULMAT (T2,T3,4,4,4,OR)

OR(1,4) = R(1)
OR(2,4) = R(2)
OR(3,4) = R(3)
OR(4,4) = 1.000

RETURN
END

Appendix D. SYM Program Listing

147
C --- ROTMAT --------------------------------------------------------- SHERMAN ---

build a rotation matrix for one degree of freedom

input

M degree of freedom

1 = \( x \) rotation

2 = \( y \) rotation

3 = \( z \) rotation

THETA angle of rotation

output

A rotation matrix

SUBROUTINE ROTMAT (M,THETA,A)

IMPLICIT REAL*8 (A-H,O-Z)

REAL*8 A(4,4)

DO 110 I = 1,4
    DO 100 J = 1,4
        A(I,J) = 0.000
    100 CONTINUE
    110 CONTINUE

A(4,4) = 1.000
A(M,4) = 1.000

M1 = MOD (M,3) + 1
M2 = MOD (M1,3) + 1
C = COS (THETA)
S = SIN (THETA)

A(M1,M1) = C
A(M2,M2) = C
A(M1,M2) = S
A(M2,M1) = -S

RETURN
END
SUBROUTINE GETMOD (IERR)

IMPLICIT REAL*8 (A-H,O-Z)

$INCLUDE:'SYM.H'

COMMON / TIME / TS,TF,TP,TPP,IJOB,ISSV,SSV,NSSV
COMMON / MODEL / xMS,xIX,xIY,xIZ

CHARACTER=1 ATYP
CHARACTER=80 BUF
REAL=8 TYP(3,25),FOR(6,25)
REAL=8 COEF(128,10)

C initialize variables

IERR = 0
ICRD = 0
NS = 0
NA = 0

c start of read loop

100 READ (50,110,END=900,ERR=900) BUF
110 FORMAT (A)
    ICRD = ICRD + 1
    ATYP = BUF(1:1)
    BUF(1:1) = '

C parse the model card

IF (ATYP.EQ.'M') THEN
    READ (BUF,*,END=500,ERR=500) xMS,xIX,xIY,xIZ

C parse a spring property card and store in table

ELSE IF (ATYP.EQ.'S') THEN
    READ (BUF,*,END=500,ERR=500) ITYP,(TYP(J,ITYP)),J=1,3

C parse a spring location card and apply properties

ELSE IF (ATYP.EQ.'L') THEN
    READ (BUF,*,END=500,ERR=500) ITYP,
    +     (SLO(I,NS+1),I=1,3),
    +     (SGR(I,NS+1),I=1,3)
    SLO(I,NS+1) = 1.000
    FSTF(NS+1) = TYP(1,ITYP)
    FDMP(NS+1) = TYP(2,ITYP)
    ULEN(NS+1) = TYP(3,ITYP)
    NS = NS + 1

C parse a force property card and store in table

ELSE IF (ATYP.EQ.'F') THEN
    READ (BUF,*,END=500,ERR=500) IFOR,JFTYP
    IF IFJFTYP.EQ.1.OR.JFTYP.EQ.2 THEN
        READ (BUF,*,END=500,ERR=500) IFOR,(FOR(J,IFOR)),J=1,6
    ELSE
        READ (BUF,*,END=500,ERR=500) IFOR,(FOR(J,IFOR)),J=1,6
        DO 200 I = 1,INT (FOR(J,IFOR))
**Appendix D. SYM Program Listing**

```fortran
K = 2 * I - 1
READ (50,*),END=900,ERR=900) COEF(K,IFOR),COEF(K+1,IFOR)
CONTINUE
ENDIF
C parse a force location card and apply properties
ELSE IF (ATYP.EQ. 'R') THEN
  READ (BUF,*,END=500,ERR=500) IFOR,
  + (ALO(I,NA+1),I=1,3),
  + (AFLV(I,NA+1),I=1,3)
  ALO(4,NA+1) = 1.0
  IFYP(NA+1) = FOR(3,IFOR)
  STIM(NA+1) = FOR(2,IFOR)
  ETIM(NA+1) = FOR(3,IFOR)
  AMP(NA+1) = FOR(4,IFOR)
  FRQ(NA+1) = FOR(5,IFOR)
  ACL(NA+1) = FOR(6,IFOR)
  IF (IFYP(NA+1).EQ.3) THEN
    NCOEF(NA+1) = INT (ACL(NA+1))
    DO 300 I = 1,NCOEF(NA+1)
    STI(I,NA+1) = COEF(I*2-1,IFOR)
    CTF(I,NA+1) = COEF(I*2,IFOR)
  CONTINUE
  END IF
  NA = NA + 1
C parse the time card
ELSE IF (ATYP.EQ. 'T') THEN
  READ (BUF,*,END=500,ERR=500) TS,TF,TP,IJOB
  IF (IJOB.EQ.3.OR.IJOB.EQ.5) THEN
    READ (BUF,*,END=500,ERR=500) TS,TF,TP,IJOB,ISSV,SSP,NSSV
  END IF
C parse a comment card
ELSE IF (ATYP.EQ. 'C'.OR.ATYP.EQ. ' ') THEN
C handle unknown card
ELSE
  WRITE (*,*) 'SYM: UNKNOWN CARD TYPE - CARD ',ICRD
  IERR = 1
ENDIF
GOTO 100
C end of read loop
500 WRITE (*,*) 'SYM: ERROR IN INPUT - CARD ',ICRD
     IERR = 1
     GOTO 100
C set time step for steady-state analysis
900 IF (IJOB.EQ.3.OR.IJOB.EQ.5) THEN
    TPP = TP
    TP = 2.0D0 * ACOS (-1.0D0) / (FRQ(1) * FLOAT (NSSV))
ENDIF
C close file and return
CLOSE (50)
RETURN
END
```
SUBROUTINE JACOBI(A,B,X,EIGV,D,N,RTOL,NMAX,IFPR,IOUT,IERR)

THIS SUBROUTINE IS USED TO SOLVE THE GENERALIZED EIGENPROBLEM
USING THE JACOBI ITERATION TECHNIQUE.

\[ [A]x = [EIGV][B]x \]

*** EIGENVALUE (EIGV) = SQUARE OF NATURAL FREQUENCY

Buckling Load

A: STIFFNESS MATRIX (ASSUMED POSITIVE SEMI-DEFINITE)
B: MASS MATRIX (ASSUMED POSITIVE DEFINITE) (N,N)
X: EIGENVECTORS STORED COLUMNWISE (N,N)
EIGV: EIGENVALUES (N)
D: WORKING VECTOR (N)
N: SIZE OF THE STIFFNESS AND MASS MATRIX
RTOL: CONVERGENCE TOLERANCE (USUALLY SET TO 10.0**-12)
NMAX: MAXIMUM NUMBER OF SHEEPS ALLOWED (USUALLY SET TO 15)
IFPR: FLAG FOR PRINT DURING ITERATION
IOUT: OUTPUT DEVICE NUMBER (USUALLY SET TO 6)
IERR: ERROR RETURN
= 0 NO ERROR
= 1 NON POSITIVE DEFINITE DETECTED

IMPLICIT DOUBLE PRECISION(A-H,O-Z)
DIMENSION A(N,N), B(N,N), X(N,N), EIGV(N), D(N)

IERR = 0

INITIALIZE EIGENVALUE AND EIGENVECTOR MATRICES
DO 10 I = 1,N
10 IF(A(I,I),GT.0.000.AND.B(I,I),GT.0.000) GO TO 4
WRITE(*,6020)
IERR = 1
RETURN

4 D(I) = A(I,I)/B(I,I)

EIGV(I) = D(I)
DO 20 I = 1,N
DO 20 J = 1,N
X(I,J) = 0.000
20 X(I,I) = 1.000
IF(N.EQ.1) RETURN

INITIALIZE SHEEP COUNTER AND BEGIN ITERATION
NSHEEP = 0
NR = N - 1
40 NSHEEP = NSHEEP + 1
IF(IFPR.EQ.1) WRITE(*,6000) NSHEEP

CHECK IF OFF-DIAGONAL ELEMENT IS LARGE ENOUGH TO REQUIRE ZEROING
EPS = (1.0D0**NSHEEP)**2.00D0
DO 210 J = 1,NR
JJ = J + 1
DO 210 K = JJ,N
TT = A(J,J)*A(K,K)
TB = A(J,J)*A(K,J)
IF(TT.EQ.0.000.AND.TB.EQ.0.000) EPTOL = 0.000
IF(TT.EQ.0.000.AND.TB.EQ.0.000) GO TO 42
EPTOL = ABS(TT/TB)
210 TT = B(J,J)*B(J,K)
TB = B(J,J)*B(K,K)
IF(TT.EQ.0.000.AND.TB.EQ.0.000) EPTOLB = 0.000
IF(TT.EQ.0.000.AND.TB.EQ.0.000) GO TO 44
EPTOLB = TT/TB
42 TT = B(J,J)*B(J,K)
TB = B(J,J)*B(K,K)
IF(TT.EQ.0.000.AND.TB.EQ.0.000) EPTOLB = 0.000
IF(TT.EQ.0.000.AND.TB.EQ.0.000) GO TO 44
EPTOLB = TT/TB
44 IF(EPTOLA.LT.EPS.AND.(EPTOLB.LT.EPS)) GO TO 210

CALCULATE THE ROTATION ELEMENTS (CA,CB) IF ZEROING IS REQUIRED
AKK = A(J,K)*B(J,K) - B(K,K)*A(J,J)
AJJ = A(J,J)*B(J,K) - B(J,J)*A(J,K)
AB = A(J,J)*B(K,K) - A(K,K)*B(J,J)
CHECK = (AB*AB + 4.0DO*AKK*AJJ)/4.0DO
IF(CHECK) 50,60,60
C50 WRITE(*,6020)
50 IERR = 1
RETURN
60 SGCH = SQRT(CHECK)
D1 = AB/2.0DO + SGCH
D2 = AB/2.0DO - SGCH
DEN = D1
IF(ABS(D2).GT.ABS(D1)) DEN = D2
IF(DEN) 80,70,80
70 CA = 0.000
CG = A(J,K)/A(K,K)
GO TO 90
80 CA = AKK/DEN
CG = AJJ/DEN
C PERFORM THE GENERALIZED ROTATION TO ZERO THE OFF-DIAGONAL ELEMENT
90 IF(N-2) 100,190,100
100 JP1 = J + 1
JM1 = J - 1
KP1 = K + 1
KM1 = K - 1
IF(JM1) 130,110,110
110 DO 120 I = 1,JM1
AJ = A(I,J)
BJ = B(I,J)
AK = A(I,K)
BK = B(I,K)
AI(I,J) = AJ + CG*AK
BI(I,J) = BJ + CG*BK
AI(I,K) = AK + CA*AJ
120 BI(I,K) = BK + CA*BJ
130 IF(IJP1) 140,140,160
140 DO 150 I = KPl,N
AJ = AIJ,I)
BJ = BIJ,I
AK = AIK,I)
BK = BIK,I)
AIJ,I) = AJ + CG*AK
BIJ,I) = BJ + CG*BK
AIK,I) = AK + CA*AJ
150 BIK,I) = BK + CA*BJ
160 IF(IJP1) 170,170,190
170 DO 180 I = JPl,KM1
AJ = A(J,I)
BJ = B(J,I)
AK = A(K,I)
BK = B(K,I)
AI(J,I) = AJ + CG*AK
BI(J,I) = BJ + CG*BK
AI(K,I) = AK + CA*AJ
180 BI(K,I) = BK + CA*BJ
190 AK = AIK,K)
BK = B(K,K)
AI(K,K) = AK + 2.0DO*CA*AIJ,KJ + CA*CA*AIJ,JJ
BI(K,K) = BK + 2.0DO*CA*B(J,K) + CA*CA*B(J,J)
AI(J,J) = AJ + 2.0DO*CG*AIJ,JK + CG*CG*AK
BI(J,J) = BJ + 2.0DO*CG*B(J,K) + CG*CG*BK
AI(J,K) = 0.0DO
BI(J,K) = 0.0DO
C UPDATE THE EIGENVECTOR MATRIX AFTER EACH ROTATION
DO 200 I = 1,N
XJ = X(I,J)
XK = X(I,K)
XI(I,J) = XJ + CG*XK
200 XI(I,K) = XK + CA*XJ
210 CONTINUE
C UPDATE THE EIGENVALUES AFTER EACH SHEEP
DO 220 I = 1,N
IF(A(I,I).GE.0.0DO.AND.B(I,I).GT.0.0DO) GO TO 220
C WRITE(*,6020)

Appendix D. SYM Program Listing 152
IERR = 1
RETURN

220 EIGV(I) = A(I,I)/B(I,I)
IF(IFPR.EQ.0) GO TO 230
WRITE(*,6050)
WRITE(*,6010) (EIGV(I),I = 1,N)

C C CHECK FOR CONVERGANCE
250 DO 240 I = 1,N
TOL = RTOL*DIIJ
DIF = ABS(EIGV(I) - D(I))
IF(DIF.GT.TOL) GO TO 280
CONTINUE

C C CHECK ALL OFF-DIAGONAL ELEMENTS TO DETERMINE IF ANOTHER SHEEP
EPS = RTOL**2
DO 250 J = 1,NR
JJ = J + 1
DO 250 K = JJ,N
TT = A(J,K)*A(J,K)
TB = A(J,J)*A(J,K)
IF(TT.EQ.0.000.AND.TB.EQ.0.000) EPSA = 0.000
IF(TT.EQ.0.000.AND.TB.EQ.0.000) GO TO 252
EPSA = ABS(TT/TB)
IF(TT.EQ.0.000.AND.TB.EQ.0.000) EPSB = 0.000
EPSB = TT/TB
EPSB = (B(J,K)*B(J,K))/(B(J,J)*B(J,J))
IF(EPSA.LE.EPSB) GO TO 254
GO TO 250
CONTINUE

C C FILL OUT BOTTOM TRIANGLE OF RESULTANT MATRICES AND SCALE VECTORS
255 DO 260 I = 1,N
DO 260 J = 1,N
A(J,I) = A(I,J)
B(J,I) = B(I,J)
BB = SQRT(B(J,J))
DO 270 K = 1,N
X(K,J) = X(K,J)/BB
CONTINUE
IF(ICHK.EQ.0) GO TO 320
GO TO 250

C C ARRANGE EIGENVALUES AND EIGENVECTORS IN ASCENDING ORDER
NM1 = N - 1
300 ICHK = 0
DO 310 I = 1,NM1
IF(EIGV(I).LE.EIGV(I + 1)) GO TO 310
ICHK = ICHK + 1
EIGV(I) = EIGV(I + 1)
EIGV(I + 1) = EIGV(I)
DO 305 J = 1,N
X(J,I) = X(J,I)
X(J,I + 1) = X(J,I + 1)
CONTINUE
IF(ICHK.EQ.0) GO TO 320
GO TO 300

320 RETURN

C C UPDATE D MATRIX AND START NEW SHEEP, IF ALLOWED
280 DO 290 I = 1,N
D(I) = EIGV(I)
IF(NSHEEP.LT.NSMAX) GO TO 40
GO TO 255

C C FORMAT STATEMENTS
6000 FORMAT(' SHEEP NUMBER IN *JACOBI* = ',I4)
6010 FORMAT'(2X,6E13.5)
6020 FORMAT(' STIFFNESS MATRIX IS NOT POSITIVE DEFINITE ')
6030 FORMAT(' CURRENT EIGENVALUES IN *JACOBI* ARE; ')
END

Appendix D. SYM Program Listing 153
The vita has been removed from the scanned document