

AD-A247 567



WL-TR-91-2128

COMPUTER GRAPHICS FOR BEARING
DYNAMIC ANALYSIS

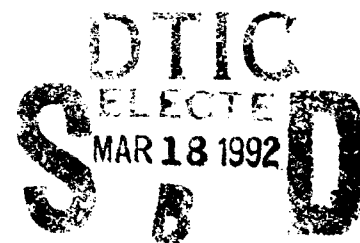
James D. Moore
Doug A. Clemons
SRS Technologies
990 Explorer Blvd., NW
Huntsville, AL 35806

December 30, 1991

Final Report for Period April 1991 to November 1991

Approved for public release; distribution unlimited.

AERO PROPULSION & POWER DIRECTORATE
WRIGHT LABORATORY
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433



92-06909

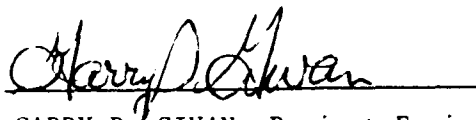


NOTICE

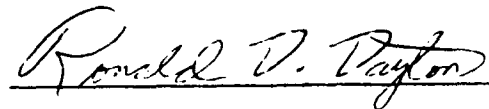
When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

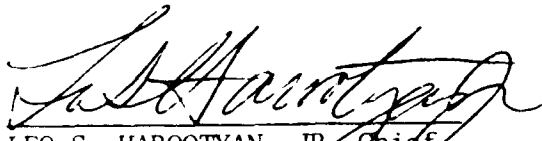
This technical report has been reviewed and is approved for publication.



GARRY D. GIVAN, Project Engineer
Lubrication Branch
Fuels and Lubrication Division
Aero Propulsion and Power Directorate



RONALD D. DAYTON, Chief
Lubrication Branch
Fuels and Lubrication Division
Aero Propulsion and Power Directorate



LEO S. HAROOTYAN, JR., Chief
Fuels and Lubrication Division
Aero Propulsion & Power Directorate

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify WL/POSL, WPAFB, OH 45433-6563 to help us maintain a current mailing list.

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

REPORT DOCUMENTATION PAGE

Form Approved
GMB No. 0704-1000

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b RESTRICTIVE MARKINGS		
2a SECURITY CLASSIFICATION AUTHORITY			3 DISTRIBUTION/AVAILABILITY OF REPORT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED		
2b DECLASSIFICATION/DOWNGRADING SCHEDULE			4 PERFORMING ORGANIZATION REPORT NUMBER(S) WL-TR-91-2128		
5a NAME OF PERFORMING ORGANIZATION SRS TECHNOLOGIES		5b OFFICE SYMBOL (if applicable)		7a NAME OF MONITORING ORGANIZATION AERO PROPULSION AND POWER DIRECTORATE (WL/POSU) WRIGHT LABORATORY	
6a ADDRESS (City, State, and ZIP Code) 990 Explorer Blvd., NW Huntsville, AL 35806		6b ADDRESS (City, State, and ZIP Code) Wright Patterson AFB, CA 45433-6563		9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-91-C-2133	
8a NAME OF FUNDING/SPONSORING ORGANIZATION		8b OFFICE SYMBOL (if applicable)		10 SOURCE OF FUNDING NUMBERS	
8c ADDRESS (City, State, and ZIP Code)		PROGRAM ELEMENT NO 65502F		PROJECT NO 3005	
		TASK NO 02		WORK UNIT ACCESSION NO 01	
11 TITLE (Include Security Classification) Computer Graphics for Bearing Dynamic Analysis					
12 PERSONAL AUTHOR(S) Moore, James D., and Clemons, Doug A.					
13a TYPE OF REPORT Final		13b TIME COVERED FROM 91/04/07 TO 91/11/07		14 DATE OF REPORT (Year, Month, Day) 91/12/30	
15 PAGE COUNT 40					
16 SUPPLEMENTARY NOTATION					
17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP	Rolling Element Bearings, Bearing Design, Computer Graphics, Animation, Thermal Analysis, Dynamic Analysis		
19 ABSTRACT (Continue on reverse if necessary and identify by block number) The enormous amount of data produced by computer codes used by the aerospace industry to design and evaluate high speed rolling element bearings makes it difficult and time consuming to synthesize and evaluate analysis results. The objective of this research effort was to determine if computer graphics and animation techniques can be used to develop software tools for post-processing dynamic analysis results and presenting the results in a clear concise graphical format. The software developed uses the output from a bearing dynamic modeling code to create an animation of the bearing such that the interactions of the bearing components can be viewed. During this study, a graphical postprocessor was developed that uses results from dynamic analysis to animate an operating ball bearing as viewed from an axial prospective. The software also includes various informational displays to present different types of bearing performance data in addition to bearing animation. The races and cages were displayed in two-dimensional graphics and the rolling elements were displayed in three-dimensional graphics. The software was used to review several bearing analysis data sets and it was found that use of the animation software improved comprehension of the analysis results and promoted better understanding of the					
20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS			21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED		
22a NAME OF RESPONSIBLE INDIVIDUAL Garry D. Givan			22b TELEPHONE (include Area Code) 513-255-1286		22c OFFICE SYMBOL WL/POSU

19 ABSTRACT (Continued)

kinetic phenomena occurring in the bearing. The technology developed will significantly improve the utilization of dynamic bearing analysis codes, by providing better understanding of bearing operating characteristics, leading to more reliable and cost effective rolling element bearing designs. Proposed future research will result in extending the software to provide complete three-dimensional graphics and to incorporate new informational displays and additional analysis capabilities.

FOREWORD

This report was prepared by SRS Technologies under Contract No. F33615-91-C-2133 entitled "Computer Graphics for Bearing Dynamic Analysis" for the Aero Propulsion and Power Directorate, Wright Laboratory, Air Force Systems Command, USAF. The research was administered under the technical direction of Mr. Garry D. Givan who served as the Air Force Project Manager.

The research was performed by the Aerospace Directorate at SRS Technologies Systems Technology Group during the April, 1991 - October, 1991 period. Mr. James D. Moore served as the SRS Technologies Project Manager. The personnel who supported this project were:

Doug Clemons

Laura Pullum

David Marty.

CONTENTS

SECTION	PAGE
FOREWORD	iii
LIST OF FIGURES	v
1.0 INTRODUCTION	1
2.0 SUMMARY	3
3.0 DEVELOPMENT OF BEARING ANALYSIS/ANIMATION CODE INTERFACE	5
3.1 Evaluation and Review of ADORE Output Dataset	5
3.2 Development of Interfacing Software	6
4.0 DEVELOPMENT OF ANIMATION SOFTWARE	16
4.1 Animation Code Display Features	16
4.2 Animation Software Development	18
4.3 Incorporation of an Interpolation Routine to View Dynamic Simulations with Accurate Velocity Representations	24
4.4 Development of Three-Dimensional Surface Display for Bearing Animation Software	28
5.0 CONCLUSION AND RECOMMENDATIONS FOR FUTURE WORK	30
6.0 BIBLIOGRAPHY	33

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

LIST OF FIGURES

NUMBER	TITLE	PAGE
3.1	Data Extracted from PRINT File.....	6
3.2	Data Flow for ADORE to Animation Interface	7
3.3	Example of Dynamic Analysis Output Conversion for Graphics Program by RD-GR-DAT.....	8
3.4	Data Required to Draw the Bearing Outer Race.....	9
3.5	Data to Draw the Inner Race and Data for Inner Race Mass Center Location.....	11
3.6	Data to Draw Cage Segments.....	12
3.7	Data to Calculate Ball Translations.....	13
3.8	Data to Orient Ball Angular Position.....	15
4.1	Graphics Screen Capture from Bearing Animation Software	17
4.2	Relationship of the 3-D Graphical Model, the Screen, and the Viewer	19
4.3	Illustration of New Surfaces Required to Allow 3-D Views from Any Prospective..	21
4.4	Animation Algorithms Evaluated to Display Bearing Motion.....	23
4.5	Time Flow Versus Integration Step During a Dynamic Bearing Simulation.....	25
4.6	Possible Solutions Reviewed to Address Variable Time Steps	26
4.7	Relationship of Display Data to Output Data with Interpolation Routine On	27
4.8	Ellipsoidal Comprehensive Stress Surface for Hertzian Contact	29
4.9	Rolling Element Race Hertzian Contact Stress Surface Display	29
5.1	Bearing Graphics Screen on a Compaq PC with VGA Graphics	31

1.0 INTRODUCTION

Successful design of high speed Rolling Element Bearings depends on a thorough understanding of the complicated interactions of rolling elements, cage, and races. State-of-the-art dynamic bearing simulations performed with software such as ADORE^{®(1)} (Advanced Dynamics of Rolling Elements) are capable of predicting the complex dynamic behavior of the bearing components. Dynamic simulation lets the user study the effects of changing various bearing design parameters in order to optimize the operating performance of the bearing. However, the tremendous amount of output data generated by the simulation makes it very difficult to assimilate and evaluate results from analysis. Plots of specific parameters such as acceleration, velocity and position are generated for each rolling element, race, and the cage. These plots are useful for interpreting the analysis. However, full understanding of bearing operational characteristics depends on visualizing the motion and interactions of all the bearing elements simultaneously. Traditionally, the designer has been required to construct a mental image of the component interactions from the various plots and numerical values output by the simulation. This process is difficult and time consuming, thus restricting utilization of the valuable results obtained through dynamic simulation.

The objective of this study was to investigate the feasibility of developing software capable of presenting results from dynamic analysis in the form of a computer animation. Animation of the operating bearing produces a composite view of all of the bearing components and their motion. The software developed during this effort has dramatically shown how this tool can greatly enhance the utilization of bearing dynamics codes and allow bearing designers to more quickly and thoroughly optimize bearing design parameters. The animation code developed imports data from dynamic analysis performed with the ADORE software and displays the results as a movie. The movie can be reviewed step-by-step or as an animation.

The software developed to demonstrate the feasibility of this concept is limited to reviewing results for analysis of ball bearings and the view of the animation is limited to an axial perspective. Future work has been proposed that will result in broadening the scope of applications and capabilities for the animation software as well as addressing additional bearing design concerns which are not addressed by dynamic analysis alone. The current software was developed with three-dimensional modeling of the rolling elements and two-dimensional displays for the races and cage. The proposed development includes modeling multiple bearing types with all components displayed using three-dimensional graphics. The designer will have

1. ADORE Copyright© 1983, Pradeep K. Gupta, Inc., The Computer Program ADORE is a Proprietary Software of Pradeep K. Gupta, Inc. (PKG).

the option of moving the viewing perspective of the bearing to any point in space. Using this feature, the designer will have the ability to study the complex three-dimensional kinetics of the bearing in close detail. The technology developed in this study can be applied to display results of thermal and stress analysis of bearings to supplement the dynamic analysis of the bearing. During this study various informational displays were developed to present "non-geometric" data such as cage loads and contact stresses. The software developed demonstrated that development of three-dimensional animation software for postprocessing of dynamic analysis is both feasible and a great technical advantage for a bearing designer.

The performance and life of bearings in rotating machinery is often the life limiting factor in many applications. The development of advanced bearing dynamic codes, such as ADORE, has given the designer a powerful capability to determine the cause of bearing anomalies and to optimize the bearing design for maximum life and reliable performance. The research performed has shown that graphical postprocessing of analysis results is the logical next step for advancement of bearing design technology. The software will greatly enhance utilization of dynamic analysis in the design process, as well as providing the designer with new tools to study the complex dynamic phenomenon occurring in high speed rolling element bearings.

2.0 SUMMARY

During this study, software was developed which could utilize the output from a bearing dynamic analysis to generate an animation of the operating bearing. The BDAA (Bearing Dynamic Analysis Animator) software can be used to view the motions and interactions of the various bearing components. The software also has additional displays for presenting specific performance data simultaneously along with the animation. The code facilitates review and comprehension of dynamic analysis results by presenting the data in a clear graphical format.

Development of this capability involved reviewing the output from an ADORE dynamic analysis, creating software to obtain the data needed for animation, and creating software to graphically display the data. Two programs were developed to extract the data and prepare input for the animation algorithm. One program was developed to read analysis results from the output file generated by ADORE. A second program converts the analysis results into the format required by the animation program. A third program, BDAA, displays the motions of the bearing components as an animation or frame-by-frame for detailed review. These programs were written in FORTRAN to maintain maximum machine independence. Machine independence was demonstrated by converting portions of the code to run on a personal computer.

The software includes various informational displays that were developed to present data on various operational characteristics of the bearing. These displays present additional data that cannot be reviewed by simply viewing the animation. Displays were developed to show; the magnitude of contact stress in the ball to race contacts, the contact area geometry, the magnitude/direction of cage to ball forces, and the magnitude/direction of cage to race forces. The quantitative data displays complement the qualitative animation to present the designer with an overall characterization of bearing performance.

Different approaches for creating the animation were investigated. The animation technique selected utilizes bit map graphics to create the animation. Using this technique each screen can be created and stored in computer memory. The screens are recalled from memory and displayed sequentially to produce the animation. This approach allows the "number crunching" required to generate each screen to be performed prior to displaying the animation. Implementation of this approach means that the complexity of the calculations required to generate the screen do not affect the animation display rate. Therefore, the screens can be very detailed and still be displayed quick enough to create an animation.

The code developed for this effort displays the rolling elements in 3-D and the races and cage are displayed in 2-D. The BDAA code was used to review results from several dynamic

analyses of ball bearings. It was found that this graphical postprocessing greatly supports utilization of dynamic analysis to improve understanding of bearing operating characteristics.

The feasibility of expanding the capabilities of the code to incorporate complete 3-D graphics and to display additional bearing performance characteristics was investigated. It was determined that these capabilities could be incorporated into the code provided bit map graphics were used for the animation algorithm. Future development of a complete 3-D version coupled with the creation of additional informational displays will result in the development of a powerful bearing design tool to support dynamic analysis of rolling element bearings.

3.0 DEVELOPMENT OF BEARING ANALYSIS/ANIMATION CODE INTERFACE

The bearing animation software developed during this study was developed as a postprocessor for displaying the results from dynamic simulation of an operating bearing. For this study, it was decided to create the animation software to be compatible with the Advanced Dynamics of Rolling Elements (ADORE) bearing dynamics code. This code represents the current state-of-the-art in bearing dynamic analysis and is widely used throughout the bearing design community. Therefore, it was necessary to create a software interface to extract the data needed to draw the bearing components and animate their motion from the simulation code output. The interface code also converts the data to a format suitable for use by the animation program. The following sections describe the methods and software developed to perform this task.

3.1 Evaluation and Review of ADORE Output Dataset

Animation of bearing dynamic performance involves at least three steps: simulation of the bearing using a bearing dynamics code, interpretation of the analysis results to convert the data into a format suitable for graphical display, and graphical display using software developed for this purpose. For this effort, it was decided to utilize data from the ADORE bearing simulation to develop a demonstration of the bearing animation software. Therefore, it was necessary to review in detail the output from ADORE and to write software which is capable of interpreting the output from an ADORE analysis and generating a file containing appropriate input for the animation code.

During the initial phase of this study, the output files from an ADORE bearing simulation were downloaded to a VAX computer for review. The objective of the review was to determine which data in the output files are needed for the animation. The ADORE software generates several output files that are available at the completion of an analysis. The output files consist of a main printed results file, from one to six plot data output files, and additional files required to restart and continue the analysis. It was determined that the data required for the animation could be obtained from the main "PRINT" output file.

The print file consists of two types of output: bearing geometry data which are constant and are output only once at the beginning of the file, and time step data which are output at user specified intervals throughout the solution. The dimensions for each of the bearing elements (inner race, outer race, cage, and rolling elements) can be obtained from the bearing geometry data. These geometry data are used to construct a graphical representation of each of the bearing elements. The size of the bearing is scaled to utilize the full screen. The time step data contain position data for each of the elements at each output solution step. The position data determine

where on the screen the bearing elements are placed at a particular time. Additionally, the time step output contains various "nongeometric" operating characteristics of the bearing which are important for evaluating the bearing performance. Figure 3.1 shows the types of data that are used by the postprocessor.

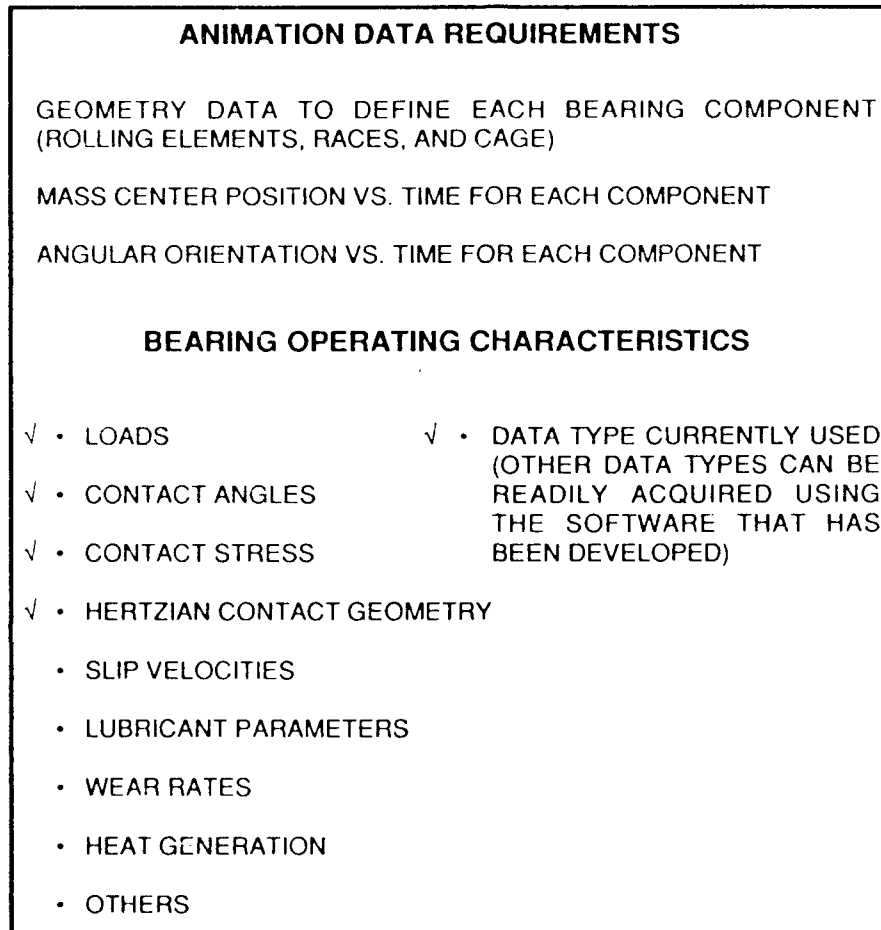


FIGURE 3.1 DATA EXTRACTED FROM PRINT FILE

During the preliminary phase of this study, the Print file was examined to "map" the location of each data item needed as input by the graphical postprocessor. Software was then written to read the print file generated by an ADORE analysis and extract needed data for use by the animation code.

3.2 Development of Interfacing Software

The software developed to interface the animation code to output from an ADORE analysis performs two functions. First, the data have to be extracted from the print file. After extraction, further processing is required to convert the raw data into the format required by

the animation code. These two steps are performed by the interface programs GETDATA and RD-GR-DAT that were developed for this effort.

These programs are written in FORTRAN and they perform several functions required to convert output from an ADORE analysis into a graphical model of the operating bearing. Figure 3.2 shows the data flow from the ADORE output file "PRINT" to the file "AP-GRAPH.DAT". The program GETDATA is a generic program that can extract alphanumeric data from any field in an ASCII computer file. GETDATA can work directly with the output file generated by ADORE. To use GETDATA a template is set up that tells the program where in the "PRINT" file important data is written. Important data include bearing geometry (number of rolling elements, pitch diameter, etc.), boundary conditions (applied loads, etc.) and analysis results (rolling element positions, reaction forces, etc.). These data extracted by GETDATA are output in a condensed format to the file

"GRAPH.DAT" for further processing. A template must be set up for each bearing type. During this study a template was set up for ball bearings. In future efforts, templates will be created for the additional bearing types modeled by ADORE. The ball bearing template will also be modified to extract the additional geometry and position data needed to define the races and cage in 3-D. GETDATA was written in a very generic format, therefore up-grading for future efforts will involve little effort. The second program in the ADORE/animation interface is RD-GR-DAT.

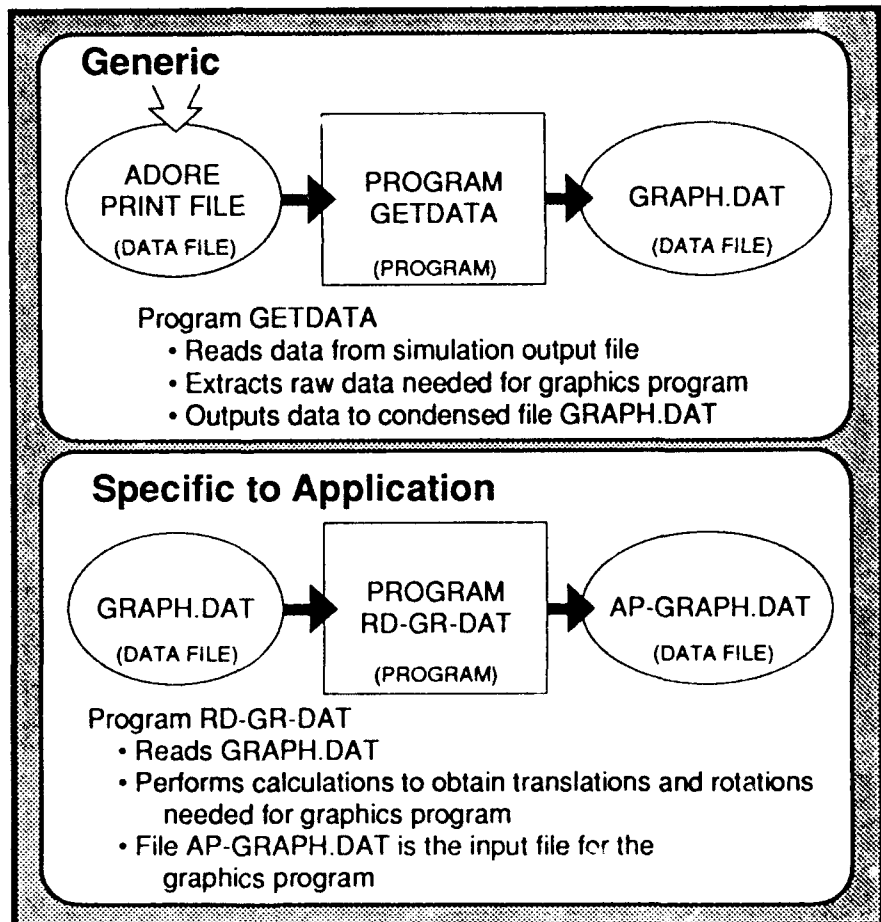


FIGURE 3.2 DATA FLOW FOR ADORE TO ANIMATION INTERFACE

This program performs processing on the raw data extracted from ADORE in order to convert the data into a format usable by the animation program. The majority of the data processing involves coordinate transformations to define all of the bearing components in a common reference coordinate system. ADORE uses many different coordinate frames to improve the computational efficiency of the code. The graphics program requires that each object in the animation be described in terms of an X, Y, and Z translation from a common reference coordinate system and an X, Y, and Z rotation about its own mass center. Figure 3.3 illustrates the transformations performed by RD-GR-DAT for each 3-D ball bearing rolling element. In continuing efforts, new routines will be developed to process ADORE output for the various types roller bearings modeled by ADORE. Also, the ball bearing routine will be upgraded to accommodate 3-D modeling of the races and cage.

The final product of the ADORE/animation interface is the file AP-GRAPH.DAT. This file contains, in a condensed format, the basic geometry data needed to draw each component of the bearing (i.e., race dimensions, rolling element dimensions, cage configuration, etc.). It also contains the location and orientation of each component for each time step in the ADORE output. AP-GRAPH.DAT serves as the input file for the animation program.

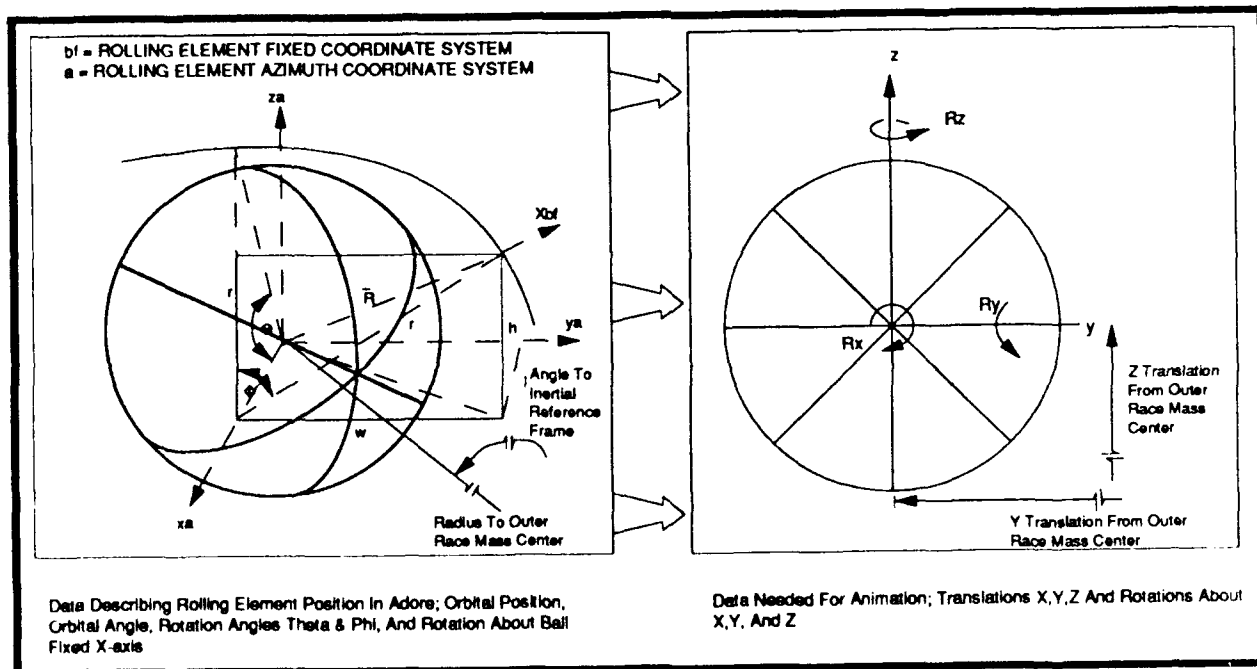
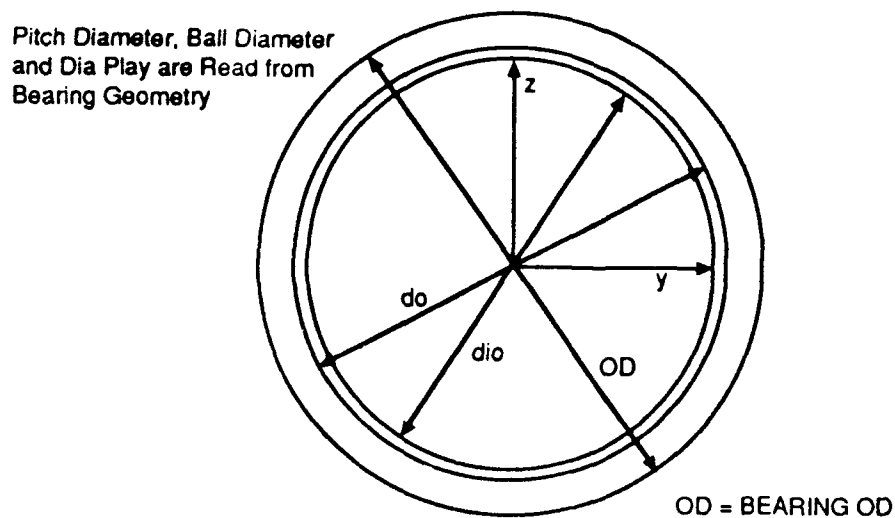


FIGURE 3.3 EXAMPLE OF DYNAMIC ANALYSIS OUTPUT CONVERSION FOR GRAPHICS PROGRAM BY RD-GR-DAT

The coordinate transformations performed by RD-GR-DAT utilize the data extracted from ADORE by GETDATA to calculate the translations and rotations of each bearing component (balls, races, and cage) from a fixed reference. The reference coordinate system used by the animation program is the same as the outer race coordinate system used in ADORE. Therefore, the location and orientation of each component are generated for animation in terms of delta-y and delta-z from the geometric center of the outer race and rotations about its own mass center. Calculating these rotations and translations can be performed through relatively simple transformation equations developed for each component.

The following descriptions and illustrations document which ADORE output data are used to draw the components and which data are used to calculate the component position and orientation. The data required for drawing and animating these inner and outer races are straight forward and can easily be extracted from the print file. Figure 3.4 illustrates how ADORE data is used to define the graphical features of the outer race.



$$do = \text{Pitch Diameter} + \text{Ball Diameter} + 1/2 \text{ DIA Play}$$

$$dio = do - 2 \times \text{shoulder height of outer race}$$

Note: The shoulder height parameter is not included in the output from ADORE. Therefore, the user must enter this data manually.

Note: Outer race always has its mass center at center of the screen

FIGURE 3.4 DATA REQUIRED TO DRAW THE BEARING OUTER RACE

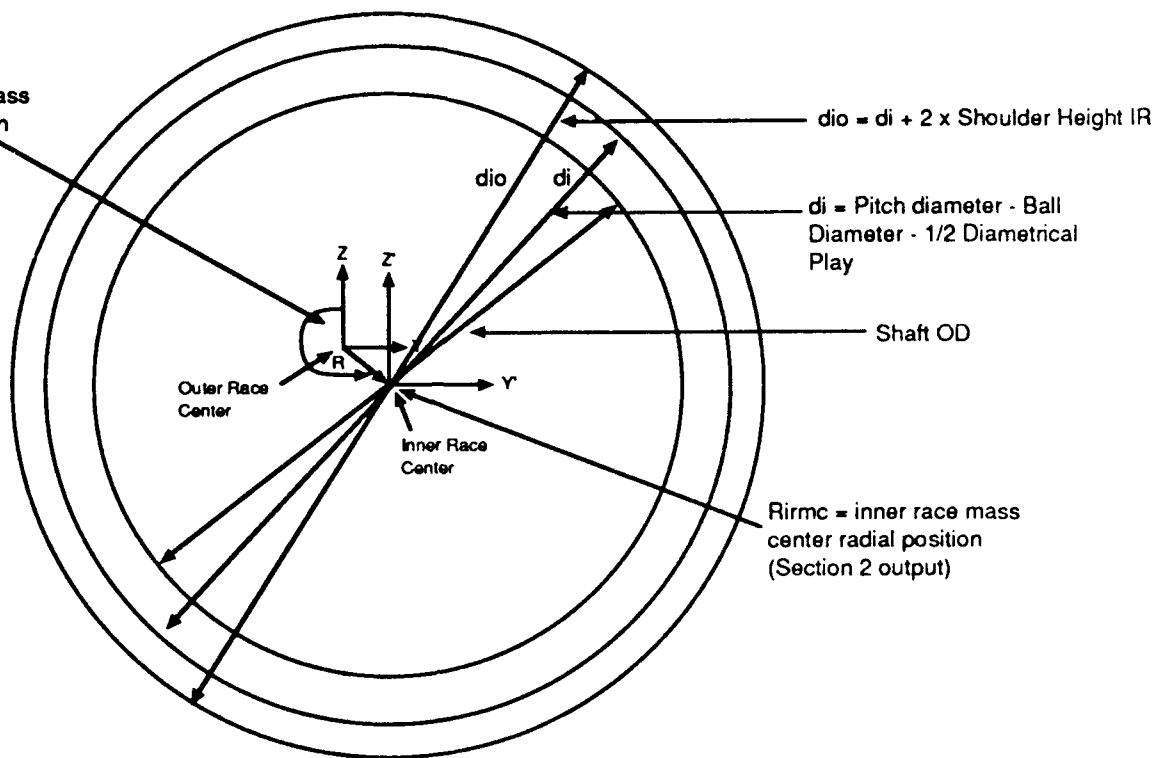
The features needed to display the race from an axial prospective include the outer and inner diameters (OD, dio) of the race and the bottom of the raceway groove (do). The bearing outer diameter can be read directly from the analysis output. However, the inner diameter and groove depth are not listed in the output file, but they may be calculated using the pitch diameter, ball diameter, and diametrical play. The outer race coordinate system coincides with the animation coordinate system; therefore, the race remains fixed during the animation. Thus, no translations need to be calculated for the outer race. The outer race is placed at the center of the screen for animation. Figure 3.5 illustrates the data needed to draw the inner race and to locate the inner race mass center with respect to the inertial coordinate system. The inner race features, when viewed from an axial prospective, include the bearing bore diameter (shaft OD), the inner race outer diameter (dio) and bottom of the raceway groove (di). The translations of the inner race with respect to the outer race are calculated using the ADORE output for the inner race orbital and radial mass center position at each output integration step. The rotation of the inner race about its own mass center is equivalent to the shaft rotation output by ADORE. Position and rotation data are updated at each time step during the animation.

The procedure for locating the cage mass center and cage orbital position is similar to the procedure for describing the position of the inner race. The cage inner diameter, outer diameter, and land diameter are read directly from the bearing geometry data. The cage rotation, orbital angle, and orbital radius are read from the integration step output and updated at each animation step. Thus, the Y and Z translations are calculated using the same method described for the inner race. The data needed to draw the individual cage segments is illustrated in Figure 3.6. The number of rolling elements and the ball pocket diameter are used in conjunction with cage diameter data to define the cage segments. Cylindrical cage pockets are modeled. The data and procedure for generating the cage segments are described in the illustration. The animation program uses this data to generate a graphical model of the visible cage segments. The cage is then drawn by the animation code in the proper location and orientation at each animation time step.

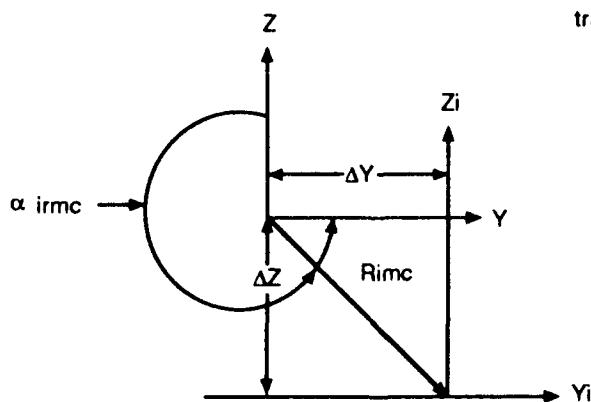
The data needed to animate the rolling elements are also extracted and processed by the interface codes GETDATA and RD-GR-DAT. The initial plans for the Phase I animation called for a 2-D representation of the ball. However, after investigating different approaches and possible visual cues to demonstrate the ball motion in 2-D no satisfactory method was determined. Therefore, it was decided to develop a 3-D spherical representation of the rolling elements. Using this approach the complex 3-D motion of the ball can be accurately portrayed in animation. The 3-D approach also directly supports proposed enhancement to the code to model the entire bearing in 3-D. The data needed to locate and orient the 3-D ball are illustrated in Figures 3.7 and 3.8. Figure 3.7 shows which data are extracted from the print

α_{irmc} = inner race mass center orbital position (Section 2 output)

Pitch diameter, ball diameter, shaft OD, and diametrical play are read from bearing geometry data.
 α_{irmc} & R_{irmc} are read from Section 2 integration step data



Use α_{irmc} and R_{irmc} to find the mass center translations for the inner race



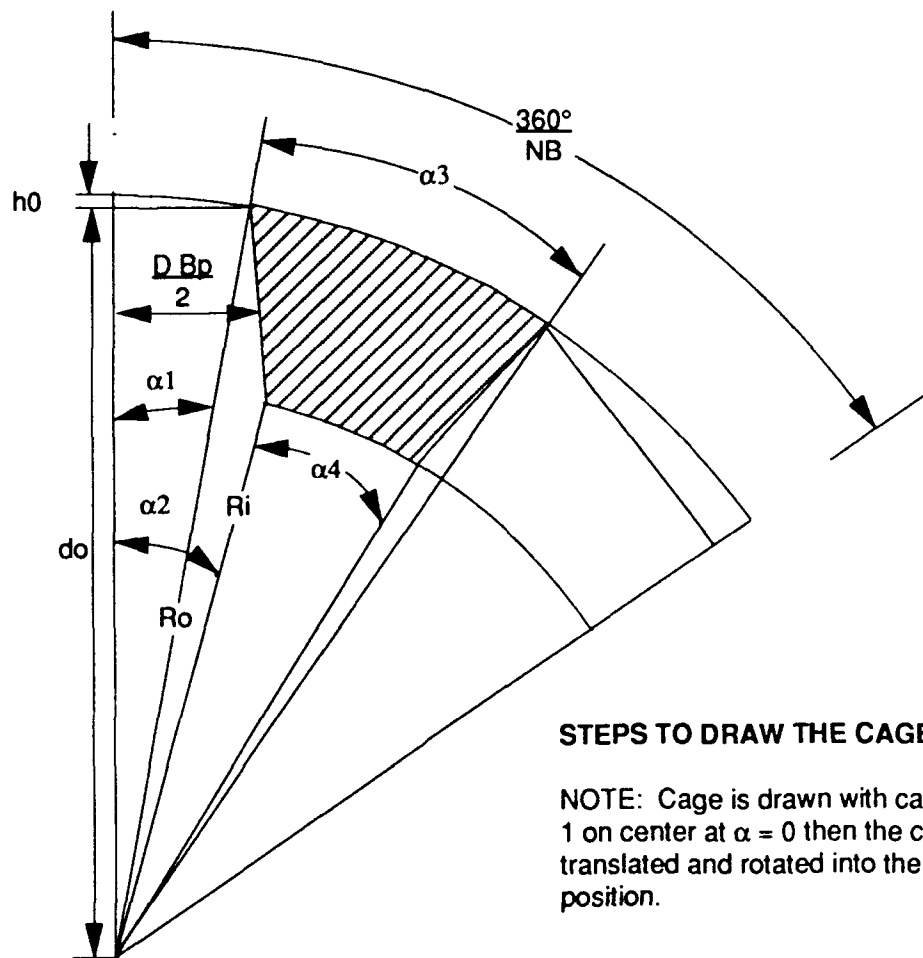
$$\Delta Y^* = -R_{irmc} \times \sin(\alpha_{irmc})$$

$$\Delta Z^* = R_{irmc} \times \cos(\alpha_{irmc})$$

* must recalculate at each time step

Note: User must enter shoulder height, this value is not in the ADORE output

FIGURE 3.5 DATA TO DRAW THE INNER RACE AND DATA FOR INNER RACE MASS CENTER LOCATION



STEPS TO DRAW THE CAGE

NOTE: Cage is drawn with cage pocket 1 on center at $\alpha = 0$ then the cage is translated and rotated into the proper position.

- 1) Solve For α_1 , α_2 , α_3 , & α_4

$$\alpha_1 = \sin^{-1} (Dbp / (2 Ro))$$

$$\alpha_2 = \sin^{-1} (Dbp / (2 Ri))$$

$$\alpha_3 = (360 / NB) - 2\alpha_1$$

$$\alpha_4 = (360 / NB) - 2\alpha_2$$

Dbp = Diameter Of The Ball Pocket

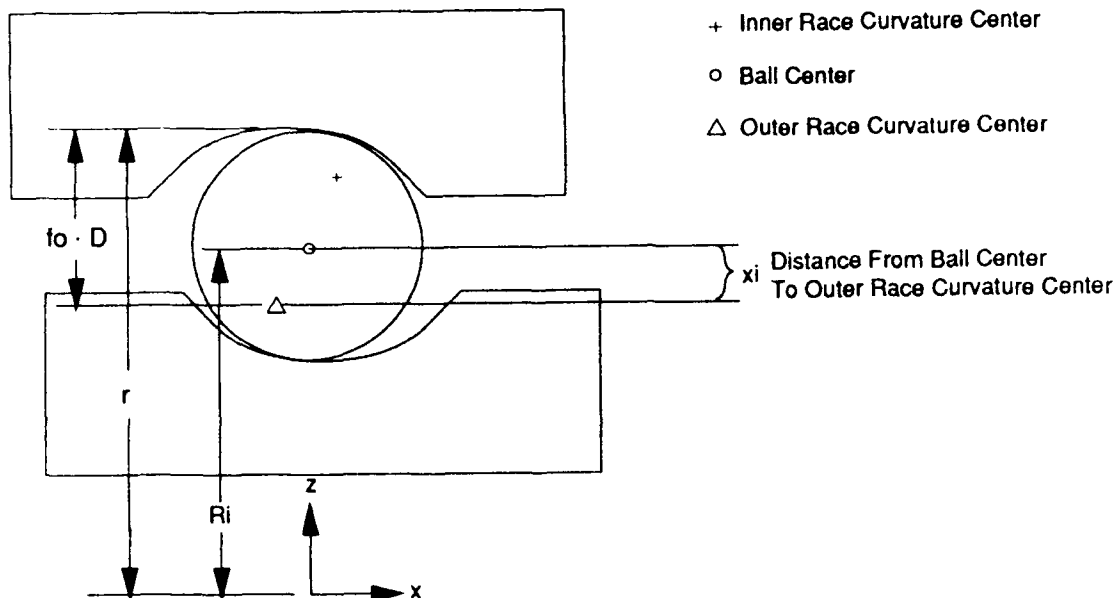
Ro = The Cage Outer Diameter

Ri = The Cage Inner Diameter

NB = Number Of Balls

- 2) Draw Outer Arc Segments With Radius Ro
- 3) Draw Inner Arc Segments With Radius Ri
- 4) Connect Arc Segments With Straight Lines
- 5) Repeat NB Times Adding Angle NB/360 Each Increment
- 6) Rotate and Translate Cage To Proper Position

FIGURE 3.6 DATA TO DRAW CAGE SEGMENTS



STEPS TO FIND BALL POSITION FOR EACH BALL

α = Outer Race Contact Angle
 δ_{oi} = Contact Deflection At Outer Race/Ball Contact
 fo = Outer Race Curvature Factor
 $r = (\text{pitch diameter} + \text{ball diameter} + .5 \cdot \text{diametrical play}) \cdot .5$
 D = Ball Diameter
 α_{bi} = Ball Orbital Angle Output From Print File

- 1) Solve For X_i
 $X_i = \cos \alpha (fo - .5)D + \delta_{oi}$
- 2) Solve For R_i
 $R_i = r - fo(D) + x_i$
- 3) Translate Ball To Orbital Position
 $\Delta y = -R_i \sin \alpha_{bi}$
 $\Delta z = R_i \cos \alpha_{bi}$

* Note: Ball Radial Position Is Not Output Directly In Print File. However, The Position Can Be Obtained Using Steps 1 & 2

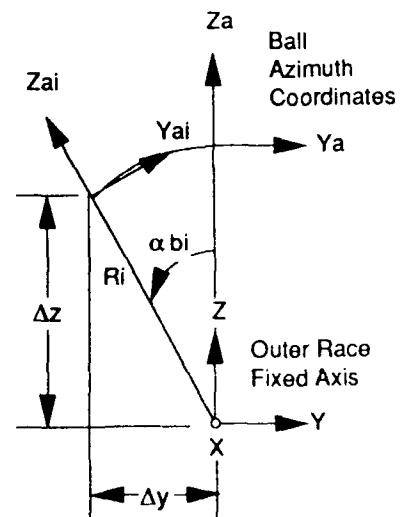
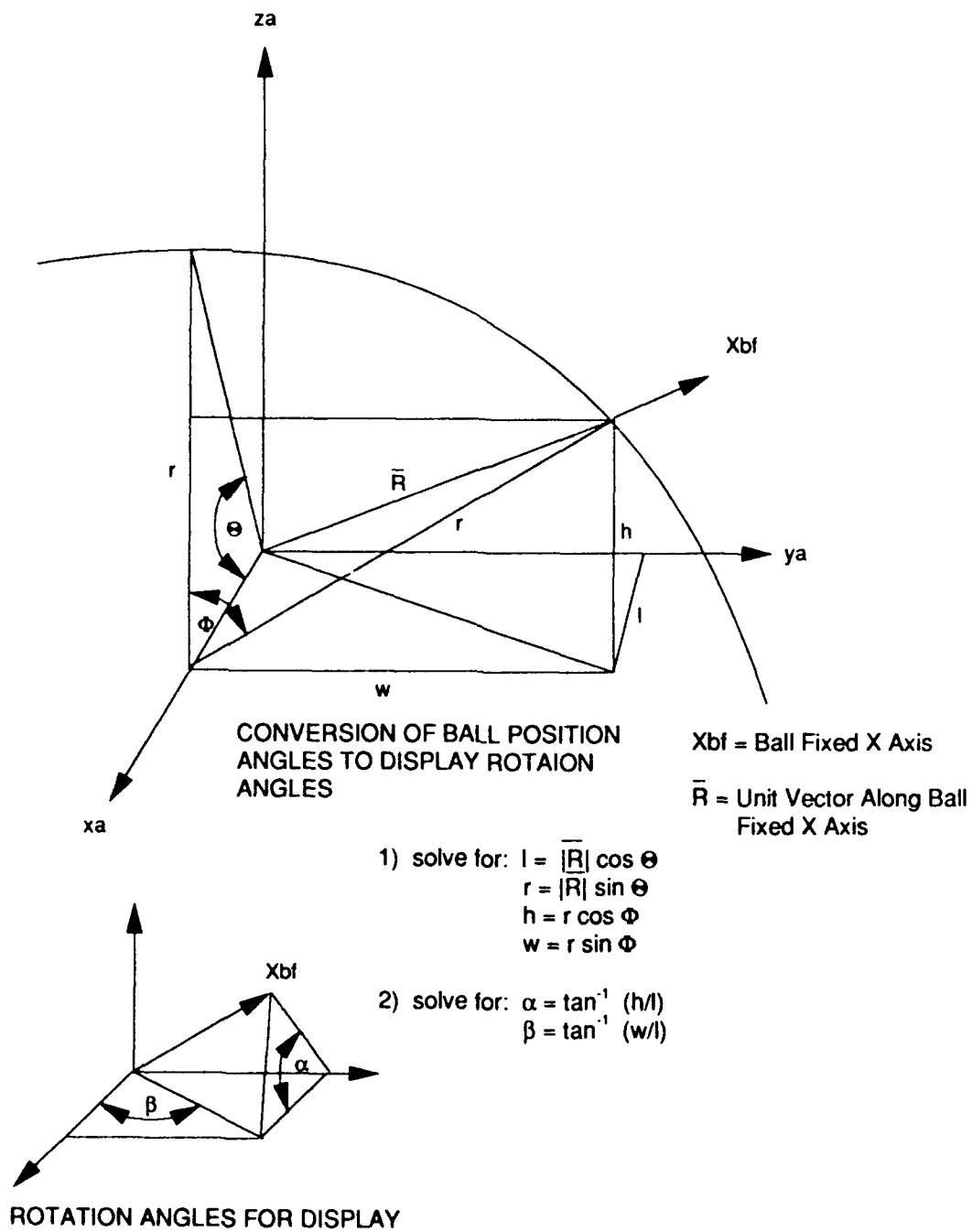


FIGURE 3.7 DATA TO CALCULATE BALL TRANSLATIONS

file and how it is used to determine the location of the ball mass center. The orbital angle for each ball can be read directly from the print output. The radius to the ball position must be determined from other data. The radius which describes the ball position is the radius from the outer race geometric center to the ball mass center. This choice is for convenience since the outer race is either fixed or has a prescribed motion. For each ball, at each time step, the radial distance from the ball center of mass to the outer race curvature center is calculated. Once this distance is determined the distance from the ball center to the outer race center can be determined as shown in the figure. By calculating the ball position in this manner, the variations in orbital radius which occur as a function of contact angle are accurately reflected in the animation.

Figure 3.8 shows the data used to orient the angular position of the ball once its position in space is determined. To determine the orientation, the angles Theta and Phi are read from the print file. These angles represent rotations about the ball azimuth coordinates. These rotations are then converted into rotations in the display coordinates. Once the ball is oriented, an additional rotation is performed about the ball-fixed x-axis. This rotation represents the rolling of the ball. Each sphere is graphically represented in 3-D by displaying longitudinal lines on the surface of the ball. The intersection of these lines occurs along the ball-fixed $\pm x$ axis. During the animation, the motion of the intersection point clearly shows the complex 3-D ball movement. This feature is further described in Section 4.1.

The programs GETDATA and RD-GR-DAT constitute the interface between the dynamic simulation and the graphical postprocessor. To view the bearing animation the user must first perform a simulation of the bearing using ADORE. The "print" file generated by ADORE must then be transferred to the workstation or personal computer where the animation software is installed. The programs GETDATA and RD-GR-DAT are then executed. These programs read the print file and generate an input file for the animation code. The animation code can then be executed to view the operating characteristics of the bearing in the form of an animation or step-by-step for detailed study. The process is relatively simple and can be performed quickly. To further simplify the process, the programs could easily be combined in a batch file or a single executable file. However, for this study the programs were maintained independently to modularize the code for increased flexibility. The animation software developed during this study could easily be converted to animate results from simulation of any dynamic process. All that is required to utilize the graphics code to animate new processes is to create a new version of RD-GR-DAT. The GETDATA program is generic and could be used to extract data from any ASCII output file generated by a dynamic simulation.



- 3)*Ball Is Traslated To Correct Orbital and Radial Positions
- 4) Ball Is Rotated To Correct Orientation By Applying α and β Rotations
- 5) Ball Is Rotated About Its Fixed X Axis

FIGURE 3.8 DATA TO ORIENT BALL ANGULAR POSITION

4.0 DEVELOPMENT OF ANIMATION SOFTWARE

The animation software has been developed to serve as a design tool for bearing engineers. The software presents bearing performance data and operating characteristics in a clear and concise graphical format. Typically, the output from dynamic analysis consists of discrete values for individual parameters output at specified time intervals throughout the simulation. The format of the output is usually tables of numbers or plots of specific parameters vs. time. From this output the user must determine how specific parameters are related by constructing a mental image of the motions and interactions occurring in the bearing. The animation software presents the data graphically, updating all of the parameters simultaneously, in the form of a slow speed motion picture. This allows the user to directly view how the various bearing components are interacting. This capability greatly complements utilization of the output generated by dynamic analysis to design bearings or investigate anomalies which may occur during bearing operation. The following sections describe the various features of the software and the techniques used to develop the code.

4.1 Animation Code Display Features

During this study, an operational prototype of the bearing animation software was developed. The program is written in FORTRAN and is operational on an Apollo workstation. Figure 4.1 illustrates the layout and features of the screen as it currently exists. The bearing display utilizes the largest portion of the screen. The inner and outer races of the bearing can be represented as simple cylinders when viewed from an axial prospective. An additional ring is displayed for each race which represents the portion of the race from the bottom of the groove to the shoulder height of the bearing. The main portion of the race ring and the exposed portion of the groove are displayed in different colors. Therefore, the changes in contact angle of the ball as it orbits can be observed by viewing the amount of the groove color displayed under the ball. The cage segments are displayed in their true relative position with respect to the balls and races. The motions of the cage and inner race are highlighted by mass center position indicators. The motion of these indicators is scaled to exaggerate the motion for better viewing. The 3-D balls are illustrated as spheres with longitudinal lines to indicate angular orientation. Each ball has a number associated with it and these numbers are displayed on the screen and move with the balls. The numbers are used to cross reference to the other graphical displays on the screen. For example, the ball/cage contact force is shown on the lower left corner of this display. The bar graph shows the magnitude of ball/cage contact forces as a function of time. By watching this graph, the user can determine if cage-to-ball interactions are occurring in an orderly pattern or if they are more random. Watching the changes in magnitude of the bars is

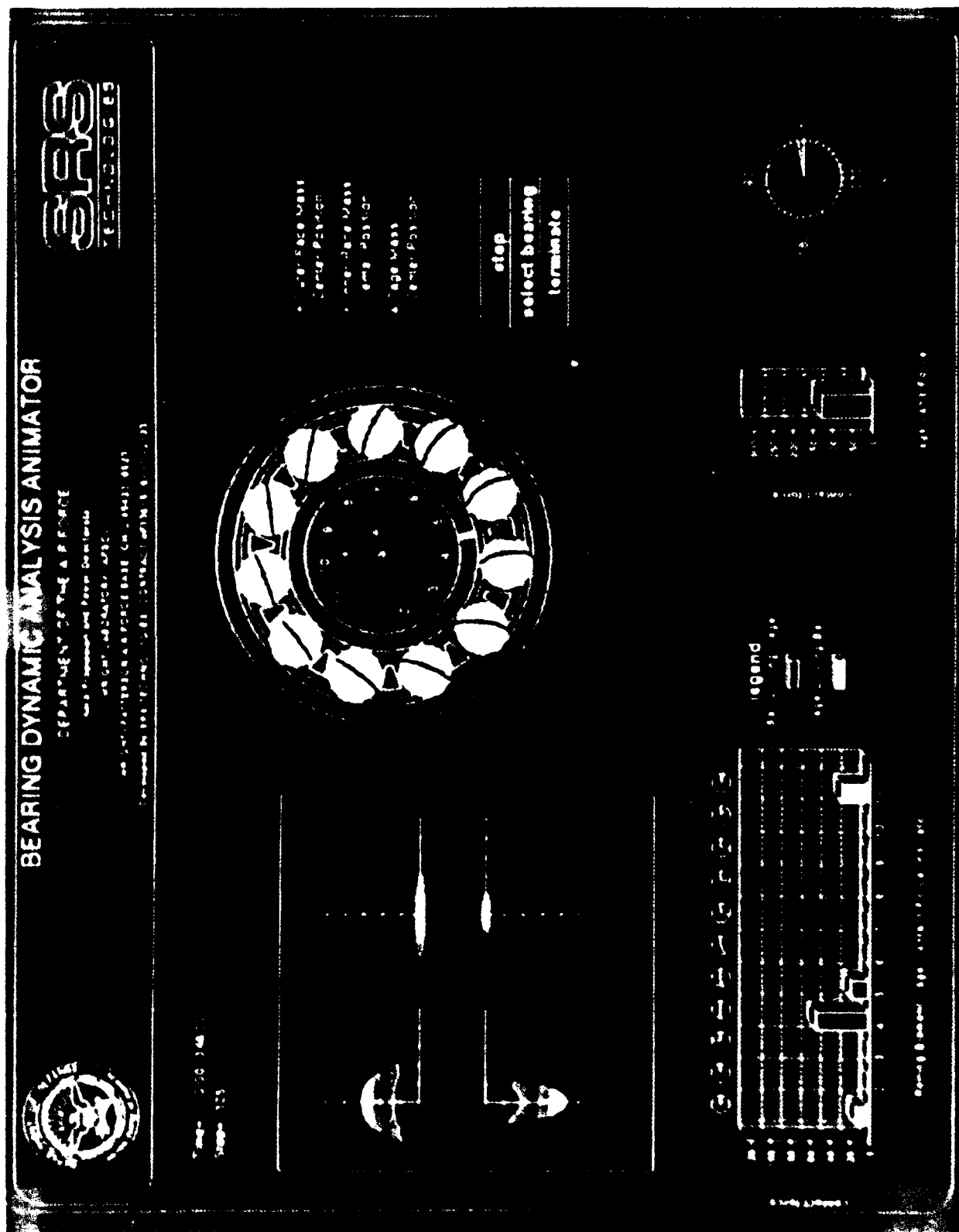


FIGURE 4.1 GRAPHICS SCREEN CAPTURED FROM BEARING ANIMATION SOFTWARE

useful for helping to determine if the cage is in a stable or an unstable operating condition. The color of the bars on the graph change to indicate whether the contact represents the ball driving the cage or the cage driving the ball. Each bar on the graph has a contact angle indicator which is displayed above the bar. This indicator gives a more precise indication of the ball contact angle within the cage pocket. A similar bar chart and contact angle indicator are displayed on the lower right hand corner of the screen for the cage/guide land contact force and angle.

The animation code also includes an optional display to present Hertzian contact stresses in the ball to race interfaces. The user has the option of displaying stresses for any of the rolling elements. The display generates a 3-D surface representing the magnitude of stress at any point in the contact ellipse. The outer race contact ellipse is displayed in the upper half of the display and the inner race contact is displayed in the lower half of the display. The user may specify any viewing orientation to examine the surface. The default is an edge on view in the left half of the display and a top down view in the right half of the screen. The magnitude of the stress is displayed with color contours.

Control of program features is via menu commands that can be selected with a mouse. The mouse commands allow the user to choose various optional information displays. Commands are also provided for viewing the display as a continuous animation or frame-by-frame. The frame-by-frame option allows forward and backward stepping for studying the bearing motion in detail. The animation features allow the user to continuously view the motion from the start of the simulation to the end. This allows velocity variation in the components to be observed and enhances overall comprehension of the dynamic interactions occurring in the bearing.

4.2 Animation Software Development

During the initial phase of this study, it was decided to utilize an Apollo 9000 Series 400-T workstation for development of the software. High level programming of the code was done using FORTRAN. This approach was chosen to allow for utilization of the graphics code development tools available on the workstation while maintaining maximum machine independence by using a standard high level language such as FORTRAN. The mathematics and data manipulation routines have all be developed in FORTRAN. The graphical output to the screen is performed using FORTRAN calls to graphics primitive routines in the system library. These routines perform simple graphics functions such as drawing a line between two specified points or filling a polygon defined by an array of points. The FORTRAN code that was developed can create all of the graphics needed to animate the bearing operation by using repetitive calls to a few simple graphics primitives. Most computer systems have similar graphics libraries that can be accessed from standard FORTRAN codes. For example, portions of the code developed for this effort were transferred to an IBM compatible personal computer and compiled using the

Lahey EM-32 FORTRAN compiler and graphics libraries. This exercise demonstrated that graphics software developed using this approach can be created in a relatively machine independent form while still maintaining the graphics capabilities to generate the 3-D animations.

Developing the graphical display capabilities, required to create an animation from dynamic analysis output, involved two major steps. First, each component of the bearing must be mathematically described in 3-D space by defining the coordinates of points on the surface of the component. The next step is to calculate the projection of the components, as defined by the surface points, onto a 2-D plane for display on the monitor. Figure 4.2 illustrates the relationship between the 3-D mathematical model, the screen projection, and the viewer. The graphics routines allow the code to manipulate the image to display the bearing in many different ways. The animation code includes rotation and translation routines that manipulate the mathematical model of the bearing components to change the orientation and location of the object in space. Similar routines can be used to vary the distance between the bearing components and the screen and the position of the viewer. However, the software developed in this effort was limited to viewing the bearing from an axial prospective only. Therefore, the viewpoint and distance from the bearing to the screen are fixed and chosen to optimize the display from this prospective.

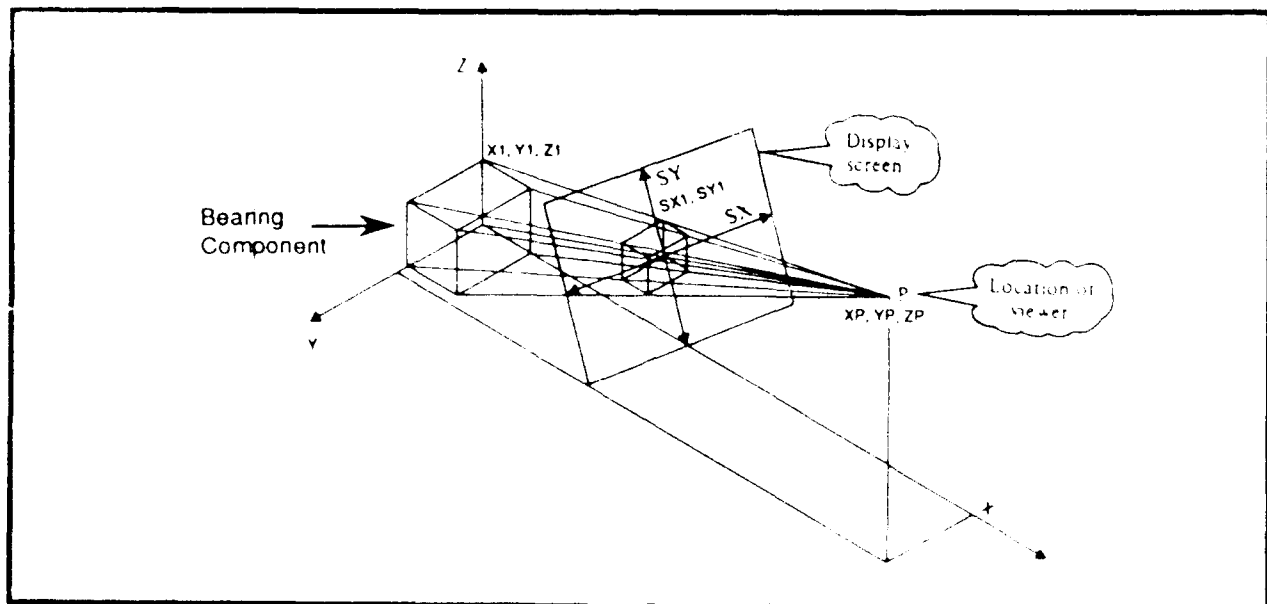


FIGURE 4.2 RELATIONSHIP OF THE 3-D GRAPHICAL MODEL, THE SCREEN, AND THE VIEWER

The Bearing Dynamic Analysis Animator (BDAA) code creates the 3-D mathematical model in two steps. First, the interface codes extract mass center position data, orientation data, and geometry data for each component of the bearing from an ADORE output file. The mass center coordinates are converted into X, Y, and Z coordinates in a reference coordinate system that has the Y- Z plane defined in the outer race plane and the X coordinate projects axially from the bearing. The position and orientation data defines where in this 3-D coordinate system each component is located. The geometry data (ball radius, race O.D. and I.D., etc.) is then used to generate an array of points located on the surface of each of the bearing components. Graphical images of the components can then be created by drawing gridlines between the points defining the surfaces. Solids are modeled by using a polygon filling routine to fill the surfaces defined by the points with color.

During this study, FORTRAN routines were developed for 3-D rotations and translations of surface definition points. These routines are implemented to model the balls in a ball bearing in 3-D. To rotate or move a bearing component to a new orientation in space, the FORTRAN routines are applied to each point defining a component surface on a point-by-point basis. As illustrated below new values are calculated for each point by successive multiplication by the desired transformation matrix:

$$[X_i' Y_i' Z_i' 1] = [X_i Y_i Z_i 1] [T_1] [T_2] [T_3]$$

The fourth coordinate is called a homogeneous coordinate and it is included only because it helps do some of the transformations more efficiently. Each of the transformation matrices performs a rotation about one of the coordinate axis or a translation along one of the axis. By sequentially applying these transformations the bearing components can be redrawn in any new position. The transformations are applied on a point-by-point basis. Therefore, the routines developed for this study can be used directly in future efforts involving 3-D modeling of all of the bearing components.

The major upgrade required for a full 3-D implementation is modifying the routines that generate the surfaces to include the entire bearing. In the current version only the parts of a ball bearing, as viewed from an axial prospective were generated. Figure 4.3 illustrates how allowing a user definable prospective increases the number of surfaces that are required for a solid model. In this study the races were displayed as 2-D disks. For user definable prospectives, all of the surfaces of the 3-D races and cage will be generated. However, it should be noted that the current software models all of the surfaces in 3-D. In other words, with the existing software, it is possible to rotate the view and see the bearing as illustrated in Figure 4.3-A. Therefore, the same software can be used for full 3-D modeling, but it will have to be

upgraded to utilize new data extracted by the ADORE interface to create the additional surfaces not currently needed for the axial prospective only version of BDAA.

Reviewing Figure 4.3-B it can be seen that some of the surfaces defining a bearing component are not be visible when viewed from an arbitrary prospective. The graphics code must have provisions for identifying the surfaces which are not visible. The techniques for identifying these surfaces are called hidden line removal procedures. Failure to remove the hidden surfaces results in an X-ray image being projected on the screen. This is confusing and unacceptable for a graphical postprocessor. For this effort a hidden line removal routine was created based on the painters algorithm. The painters routine draws objects by starting with the object furthest from the viewer and then progressively "paints" the screen outward towards the viewer, as a result, objects in the background get covered by the fore-

ground image. In the axial view only version BDAA large areas such as the entire face of a race could be defined as a surface. However, for full 3-D viewing capability it will be required to define surfaces in much more detail. To accomplish this the number of polygons used to define a surface will be increased to achieve the resolution needed. Another advantage of using small polygons to define the surface is that the coordinates of each polygon can be used to calculate a normal vector to the surface. The normal vector can be used to determine the orientation between the surface and viewer or between the surface and a simulated light source. Color shading can then be used to enhance the 3-D effect.

The graphical display software developed during this study allows each of the bearing components to be drawn and displayed on the monitor in any arbitrary location and orientation. However, additional programming was required to generate an animation, thus, allowing the user to view the motion of the bearing components. Animating the bearing motion involves creating a new display screen for each integration step output from the dynamic simulation. Each display screen is updated with new position and orientation data, for each component,

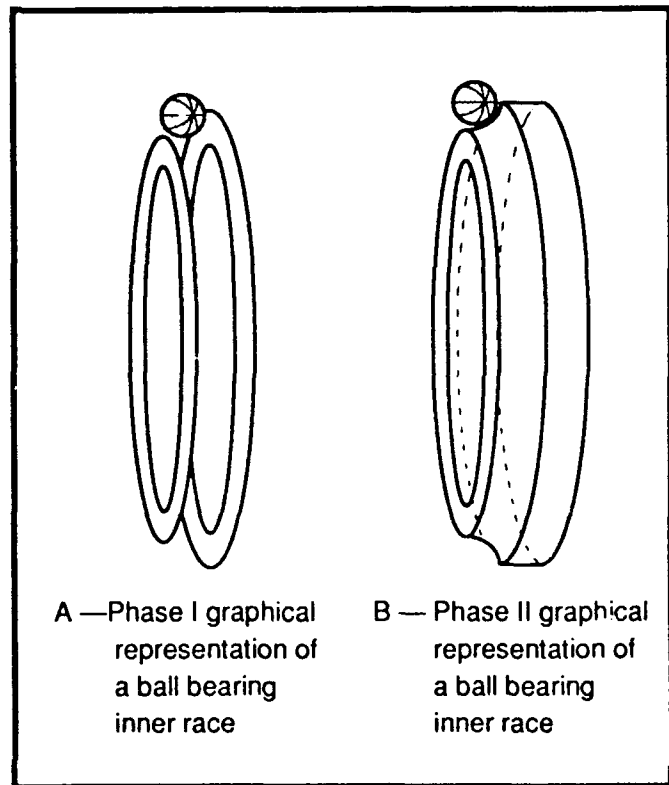


FIGURE 4.3 ILLUSTRATION OF NEW SURFACES REQUIRED TO ALLOW 3-D VIEWS FROM ANY PERSPECTIVE

extracted from the simulation output. Thus, each screen represents a small increment in time. The screens are displayed sequentially to create the animation. If the screens are displayed rapidly enough, the mind perceives the series of discrete images as smooth motion. However, if the display rate is too slow, the individual images are noticed and the motion becomes jerky. Therefore, to achieve acceptable animation the software must be capable of updating the screen very rapidly.

During this study two animation algorithms were developed and evaluated. These algorithms are illustrated schematically in Figure 4.4. The first and simplest approach to animating the bearing motion is to read position data for a time step, create a display screen for this time step, display the screen for a specified amount of time, and then repeat the process for the next time step. The drawback to this approach is that the calculations required to create the display screen must be performed in the short time period between the display of each individual screen. The more detailed the screen becomes the less feasible this approach. During this study, it was found that this approach could be successfully utilized for the relatively simple graphics screens required when the viewing prospective is limited to an axial view. This is based on running the software on a relatively fast workstation. On a slower computer this approach might not be acceptable even for an axial view. As screen detail is increased the calculations required to update the screen become more time consuming thus reducing the capability of the computer to generate new screens fast enough for smooth animation. Thus, with the objective of creating more detailed 3-D animations in the future, an alternative animation routine was developed.

The animation algorithm ultimately implemented in the BDAA code utilizes bit mapping and bit map storage techniques to improve the display rate capability of the animation. Bit maps are binary representations of the screen which define the colors displayed at each individual pixel of the display screen. Bit maps can be manipulated like files on a computer and stored in computer RAM memory or on disk memory. Using bit map storage it is possible to perform all of the calculations needed to create each screen prior to displaying any of the screens. The bit map algorithm involves reading position data for each time step, creating the display screen for this time step, storing the screen as a bit map, and repeating the process until all of the animation frames have been created and stored. Once the screens are stored they can be recalled and displayed rapidly regardless of the computation time required to generate the screens. To view the bearing analysis results the code pages through the stored screens sequentially. The level of detail in the bit maps does not affect the rate at which the screens can be displayed. Therefore, the level of detail can be increased to any level required to display the engineering data needed by the user. The amount of detail is limited only by the amount of memory installed

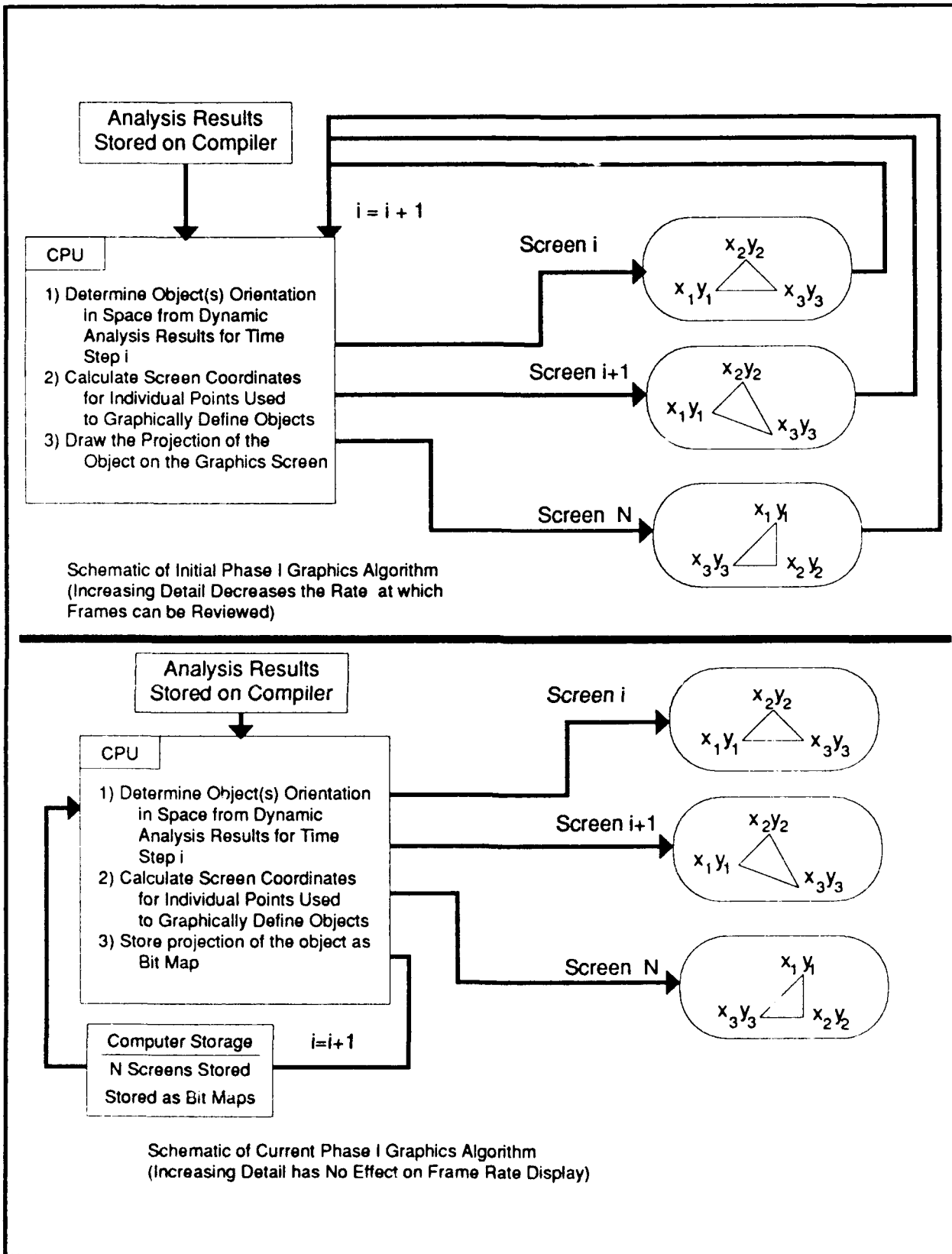


FIGURE 4.4 ANIMATION ALGORITHMS DEVELOPED TO DISPLAY BEARING MOTION

in the computer and the resolution of the display. It is anticipated that using the bit map approach for animation will be required for future full 3-D implementations of the BDAA code.

4.3 Incorporation of an Interpolation Routine to View Dynamic Simulations with Accurate Velocity Representations

The animation software works by drawing all of the bearing components, in their proper relative positions, at each time step output by the simulation. Typically, the illusion of motion is achieved in an animation by displaying sequential screens at a constant rate. In order for velocities to be correctly represented, each bit map must represent a uniform time step. On the other hand, dynamic simulations often use a variable time step in order to maintain accuracy and minimize computer run time. Typically, output from the simulation is obtained at specified integration steps during the analysis and the time increment between these output can vary as the analysis progresses. Figure 4.5 illustrates how the time increment between integration steps can vary as the simulation progresses. The initial animation developed did not compensate for the changing time steps resulting in jerky animation. The inner race appeared to speed up and slow down when in reality it was rotating with constant velocity. To correct this problem two approaches were investigated. Figure 4.6 demonstrates the two techniques for hypothetical output.

The first approach considered involves changing the display rate between frames of the animation. To change the display rate a pause function is introduced between screen displays. The length of the pause is scaled proportionally to represent the time step between two outputs from the dynamic simulation. A long pause is introduced between two frames that represent a large time step and no pause is introduced between the two frames representing the smallest time step in the output.

The alternative approach to using variable frame rates is to create a new set of data by interpolating between output data points at uniform time steps. The interpolated data can then be displayed at a constant frame rate without distorting the velocities of the bearing components.

The advantage of the first approach is that the actual data is not varied in any way. Therefore, the possibility of introducing errors through interpolation is eliminated. However, using the pause function reduces the maximum rate at which the frames can be displayed. This could be a significant problem if the simulation output has large variations in time step. The second approach allows the frames to be displayed at the fastest possible rate. However, it is possible that interpolation could introduce errors into the results. Based on these pros and cons it was decided to implement the interpolation method first for evaluation. A subroutine was created to interpolate between output time steps from ADORE to new uniform time steps. Figure 4.7 illustrates the linear interpolation used to calculate new values for the output parameters.

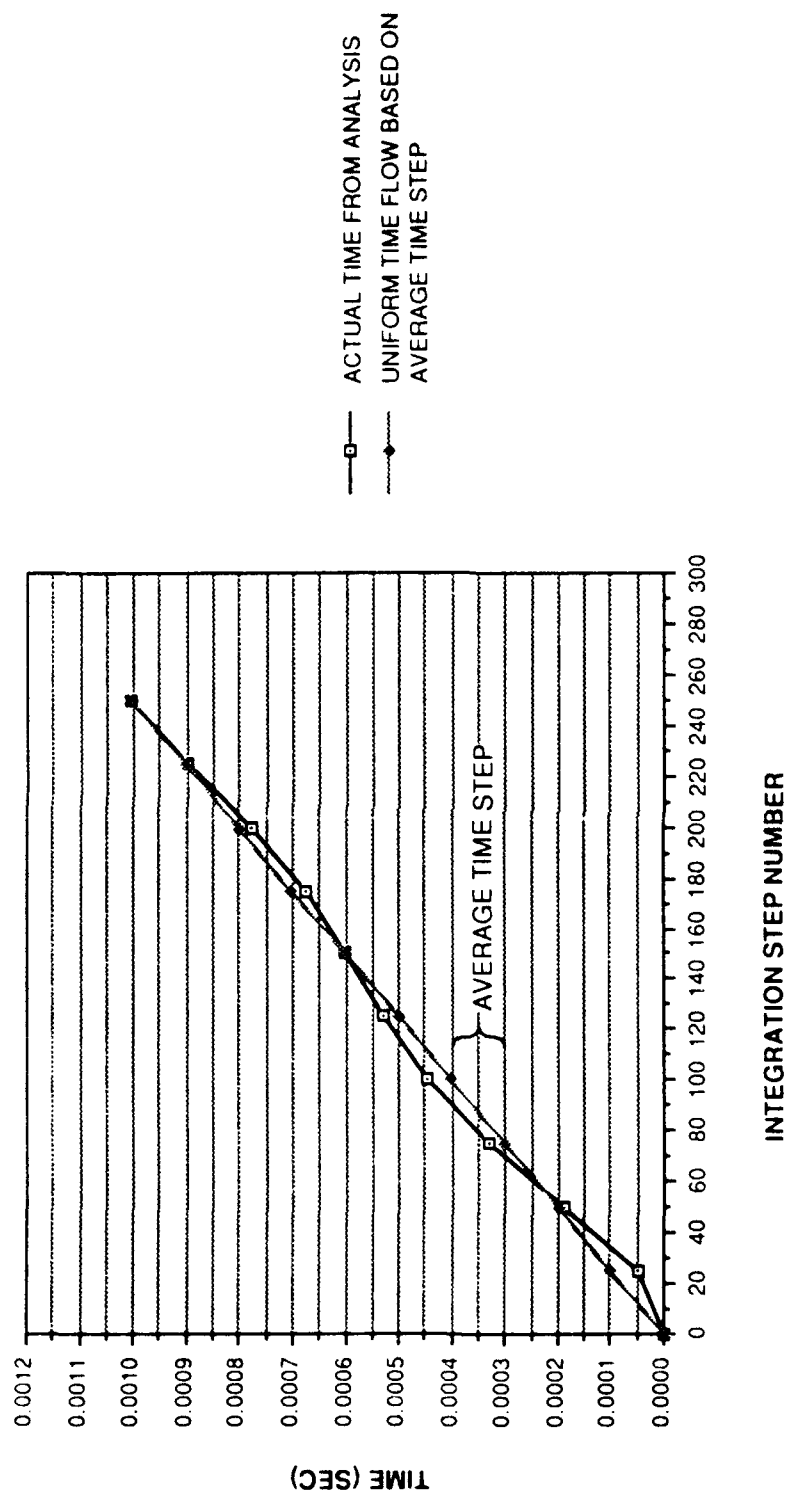


FIGURE 4.5 TIME FLOW VERSUS INTEGRATION STEP DURING A DYNAMIC SIMULATION

VARIABLE FRAME RATE METHOD

INTEGRATION STEPS OUTPUT IN PRINT FILE	TIME AT CORRESPONDING INTEGRATION STEP	TIME STEP BETWEEN INTEGRATION STEP	NUMBER OF TIMES TO CALL PAUSE FUNCTION	ACTUAL PHYSICAL TIME WHEN FRAME WOULD BE DISPLAYED
0	0.00			0
25	0.10	0.10	10	0.35
50	0.20	0.10	10	0.7
75	0.33	0.13	21	1.16
100	0.40	0.07	0	1.41
125	0.51	0.11	14	1.80

SCALE DELTA TIME TO DETERMINE NUMBER OF PAUSES NEEDED BETWEEN FRAME DISPLAYS

ASSUME MAXIMUM FRAME DISPLAY RATE IS 0.25 SECOND/FRAME.
DEVELOP A PAUSE FUNCTION WITH A SMALL DELAY.
ASSUME A VALUE OF 0.01 SECOND.

MINIMUM TIME STEP = 0.07

MAXIMUM TIME STEP = 0.13

EXAMPLE FOR STEP 1: $\frac{0.25}{0.07} = \frac{X}{0.1}$ X = 0.35

0.35 - 0.25 = 0.1 THEREFORE CALL PAUSE 10 TIMES

NUMBER OF PAUSES = 0.1/0.01

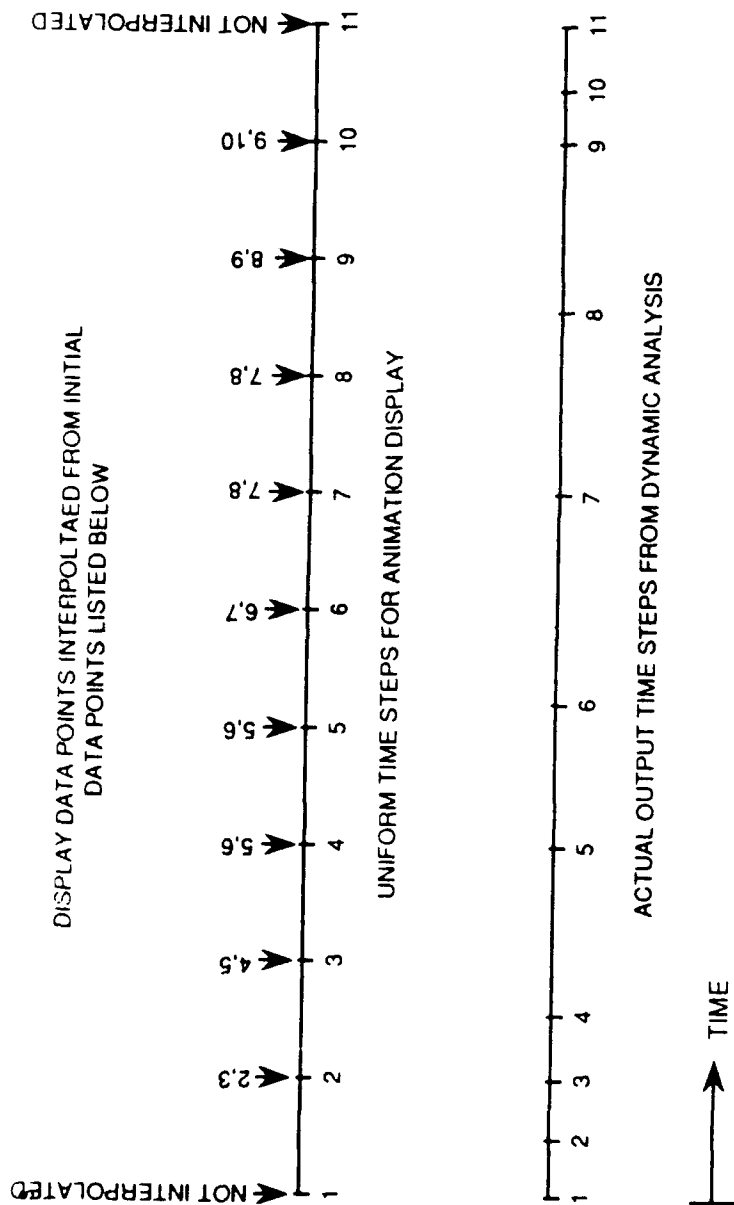
INTERPOLATION METHOD

INTEGRATION STEPS OUTPUT IN PRINT FILE	TIME AT CORRESPONDING INTEGRATION STEP	HYPOTHETICAL PARAMETER FOR ANIMATION	UNIFORM TIME STEP FOR ANIMATION BASED ON AVERAGE OF ACTUAL TIME STEPS	INTERPOLATED VALUE OF HYPOTHETICAL PARAMETER	INTEGRATION STEPS USED FOR INTERPOLATION
0	0.00	1	0.000	1.00	0
25	0.10	2	0.102	2.02	25, 50
50	0.20	3	0.204	3.03	50, 75
75	0.33	4	0.306	3.81	50, 75
100	0.40	5	0.408	5.07	100, 125
125	0.51	6	0.510	6.00	125

CALCULATE POSITIONS AND OTHER PARAMETERS

FOR NEW UNIFORM TIMES BASED AVERAGE TIME STEPS,
INTERPOLATE VALUE USING THE TWO NEAREST STEPS.

FIGURE 4.6 POSSIBLE SOLUTIONS REVIEWED TO ADDRESS VARIABLE TIME STEPS



PA = ANY PARAMETER FROM ACTUAL OUTPUT DATA
 PV = INTERPOLATED VALUE FOR PA AT NEW UNIFORM TIME STEP
 T = TIME

EXAMPLE:

$$P_{V3} = \left[\left(\frac{P_5 - P_4}{T_5 - T_4} \right) (T_{V3} - T_4) \right] + P_4$$

FIGURE 4.7 RELATIONSHIP OF DISPLAY DATA TO OUTPUT DATA WITH INTERPOLATION ROUTINE ON

The subroutine searches the original data set to find the data points nearest the new uniform time step. The interpolation routine then interpolates new values and uses the new values to generate the display screen. Experience using this routine showed that as long as data from the dynamic model are output frequently enough for animation (at least every 25th integration step for our test cases), then the interpolation does not distort the data. The routine was incorporated into the animation software as a user specifiable option. Therefore, the original data can be viewed in addition to the interpolated data for comparison. Based on the success of this routine the pause method was not coded for further evaluation.

4.4 Development of Three-Dimensional Surface Display for Bearing Animation Software

Throughout development of the bearing animation software, emphasis was placed on developing features that contribute to making the program a bearing design tool. The animation alone is a useful tool for reviewing and assimilating data from bearing dynamic analysis. Viewing the animation gives the user an excellent qualitative understanding of the various interactions between the bearing elements. The effects of changing various design parameters can be comparatively accessed by viewing the results from parametric analysis generated with dynamic bearing simulations. However, the animation alone does not provide all of the quantitative results that are important for bearing design and performance studies. Therefore, a considerable portion of this effort was devoted to developing and evaluating various auxiliary displays that present additional quantitative analysis results and make the software a more complete design tool.

For example, a three-dimensional surface display has been developed and incorporated into the bearing animation software. The display can be used to view any bearing parameter that can be represented as a surface. For evaluation purposes, contact stresses at the inner and outer rolling element to raceway interactions were displayed. Figure 4.8 illustrates the ellipsoidal compressive stress surface for a Hertzian point contact. The actual display screen is shown in Figure 4.9. The interface program RD-GR-DAT was modified to output the maximum contact stress, semimajor axis, and semiminor axis of the contact ellipse for each of the rolling elements at both races. This data, imported from the ADORE print file, are sufficient to define the contact stress surface. For display during the animation, the bearing graphics program determines the stress at uniform grid points throughout the contact ellipse.

The magnitude of stress is treated like a z-coordinate to define a surface. The resulting three-dimensional surface is displayed by connecting the points on the surface with lines to form a wire frame grid. The lines are color coded to produce color contours on the surface. The viewing angle of the surface can be arbitrarily specified by the user. Specifying the viewing

angle as directly down the Z axis results in a conventional two-dimensional color contour plot of the stresses in the contact ellipse.

The surface generation routine is currently functional as a user specifiable option in the program. In the current code, the surface generation capability is limited to viewing Hertzian contact stresses in the ball

to race contacts. However, in future versions we plan to greatly expand the list of various parameters that can be displayed using this routine. Possible parameters for display include hydrodynamic film thickness, hydrodynamic pressure and friction force distributions, isothermal contours predicted by thermal analysis, and subsurfaces stress distribution calculated from stress models.

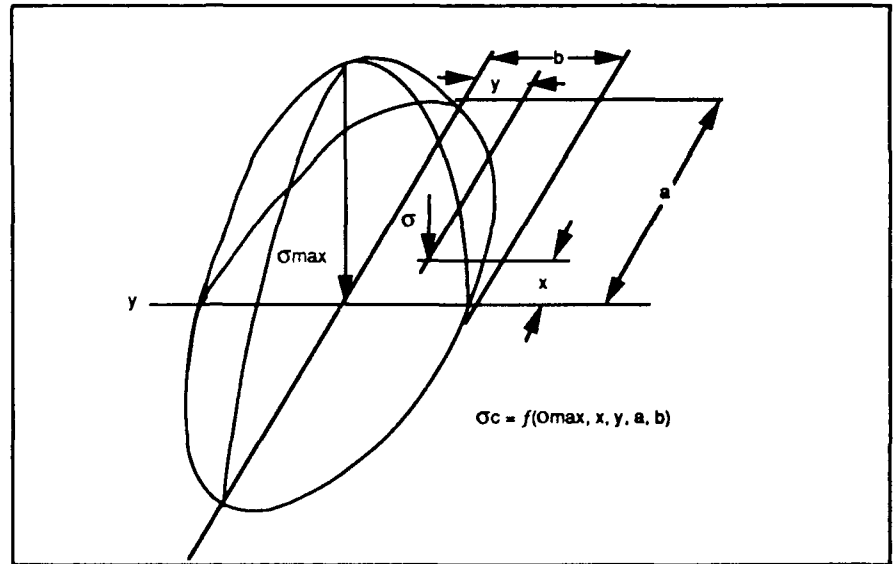


FIGURE 4.8 ELLIPSOIDAL STRESS SURFACE FOR HERTZIAN CONTACT



FIGURE 4.9 ROLLING ELEMENT/RACE HERTZIAN CONTACT STRESS SURFACE DISPLAY

5.0 CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

The objectives of this research effort were to develop computer graphics software capable of displaying an animated pictorial of an operating bearing from an axial prospective and to evaluate the benefits of this capability. Emphasis was placed on determining the feasibility of extending the capabilities of the software to accommodate full 3-D graphics with user definable viewing perspectives. The feasibility of incorporating additional bearing performance characteristics, generated by thermal or stress analysis, was also investigated.

The Bearing Dynamic Analysis Animator (BDAA) code was developed to accomplish these objectives. The BDAA code provides graphical postprocessing of results obtained from bearing dynamic simulations performed with ADORE. The code demonstrated the feasibility and usefulness of this capability. The BDAA software allows the user to visually review the dynamic interactions of the bearing components as viewed from an axial prospective. It was found that utilization of the software could improve comprehension of the analysis results and promote formulation of "user-intuitive" design modifications to improve bearing performance. The BDAA code was used to review several analysis data sets for evaluation purposes. It was found that observing the animation provided an excellent qualitative understanding of the dynamic operating characteristics of the bearing. However, more quantitative data are required to fully evaluate the suitability of a particular bearing for operating in a specified environment. During the study it was found that auxiliary informational displays could be incorporated onto the data display screen to present quantitative data along with the animation. The BDAA software includes displays developed to display cage forces, contact angles, and Hertzian stresses. These informational displays make it possible to quantitatively evaluate bearing performance. Future versions of the BDAA code should include additional informational displays to present different types of bearing performance data. The various displays should be developed as user definable options to allow the user to select specific data of interest for presentation. Data types that could be displayed include contact force, contact stress, contact geometry, film thickness, slip velocity, heat generation, wear rate, velocity vectors, and other data.

The BDAA code was developed on an Apollo workstation. The workstation provides an excellent environment for code development. The Apollo has advanced debugging routines and program development tools which expedite code development. However, one of the objectives of this study was to create the software to be as machine independent as possible. This objective was achieved by using a high level programming language (FORTRAN) to create the code. Screen drawing functions are performed with FORTRAN calls to a graphics library. It has been recommended that future versions of this code be developed for compatibility with IBM compatible personal computers. The availability of these machines would make utilization of

the code available to the maximum number of users. The feasibility of this option was demonstrated by converting portions of the BDAA code to run on a personal computer. Figure 5.1 illustrates the resolution achieved with the PC version. Based on the experience gained during this study and the success in converting the code to PC compatibility, it is recommended that this code development approach be followed for future versions of BDAA.

Throughout development of the BDAA code the objective of creating a full 3-D version of the software with user definable viewing perspectives was considered. The coordinate transformations used to describe the points that define the surfaces of the bearing components for display are capable of 3-D rotations and transformations. Therefore, no major technical obstacles are anticipated for developing a complete 3-D version of BDAA. However, the 3-D conversion will require an extensive code development effort to accomplish the "bookkeeping" duties required to track and manipulate the greatly increased number of points and surfaces required to define the bearing in 3-D. The fact that many more surface points are required for a complete 3-D graphical bearing model will result in a significant increase in the amount of computational time required to generate each frame of the animation. However, the bit map animation routine developed for this study allows the software to generate all of the screens prior to displaying the animation. Therefore, computational speed of the computer should not be a limiting factor for creating a 3-D version. The capability for 3-D viewing will allow the user to investigate the complex six-degree-of-freedom motions of the bearing in much more detail than is possible with the axial view only version of the code. Therefore, it is recommended that development of a full 3-D version of the code be pursued in the future.

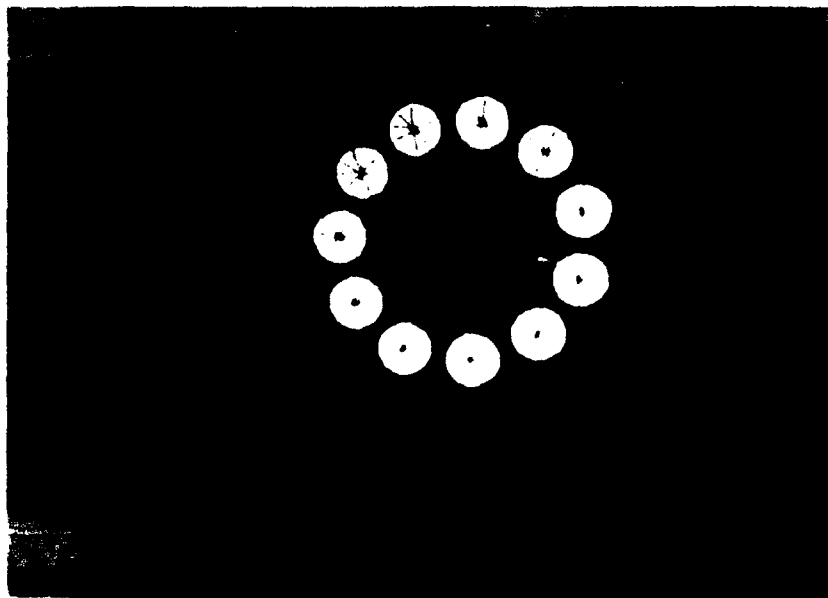


FIGURE 5.1 BEARING GRAPHICS SCREEN ON A COMPAQ PC WITH VGA GRAPHICS

Utilization of bit mapping graphics techniques in the BDAA code provides an additional benefit which may be exploited in future versions of the code. The display rate for displaying a bit map depends on the size of the portion of the screen defined not on the level of detail of the picture within that area. Therefore, it is possible to generate high levels of graphical detail on the bearing components without substantially slowing the maximum display rate. High levels of detail make it possible to display color contours or other shading techniques to present additional data during the animation. The capability to display bearing characteristics such as temperatures and stresses would strongly complement the current capabilities of the code. It is anticipated that displaying this type of analysis results could be accomplished using bit map graphics. The major effort required to develop this capability involves the integration of additional analysis results (such as temperature or stress gradients) with the results from a dynamic analysis. Integrating results from different types of analysis will be difficult; however, it is felt that the benefits of presenting the maximum amount of bearing performance data in a clear and concise graphical format justify the effort and should be pursued through further code development.

6.0 BIBLIOGRAPHY

1. Gupta, P. K., Advanced Dynamics of Rolling Elements, Springer-Verlay, 1984.
2. Harris, T. A., Rolling Bearing Analysis, John Wiley and Sons, 1981.
3. Myers, R. E., Microcomputer Graphics for the IBM PC, Addison-Wesley Publishing Co., 1984.
4. Shigley, J. E., Mischke, C. R., Standard Handbook of Machine Design, McGraw-Hill Book Company, 1986.