



Ansys

©2021 ANSYS, Inc.
All Rights Reserved.
Unauthorized use, distribution or duplication is prohibited.

Basic Analysis Guide

Ansys

ANSYS, Inc.
Southpointe
2600 ANSYS Drive
Canonsburg, PA 15317
ansysinfo@ansys.com
<http://www.ansys.com>
(T) 724-746-3304
(F) 724-514-9494

Release 2021 R1
January 2021

ANSYS, Inc. and
ANSYS Europe,
Ltd. are UL
registered ISO
9001:2015
companies.

Copyright and Trademark Information

© 2021 ANSYS, Inc. Unauthorized use, distribution or duplication is prohibited.

ANSYS, ANSYS Workbench, AUTODYN, CFX, FLUENT and any and all ANSYS, Inc. brand, product, service and feature names, logos and slogans are registered trademarks or trademarks of ANSYS, Inc. or its subsidiaries located in the United States or other countries. ICEM CFD is a trademark used by ANSYS, Inc. under license. CFX is a trademark of Sony Corporation in Japan. All other brand, product, service and feature names or trademarks are the property of their respective owners. FLEXIm and FLEXnet are trademarks of Flexera Software LLC.

Disclaimer Notice

THIS ANSYS SOFTWARE PRODUCT AND PROGRAM DOCUMENTATION INCLUDE TRADE SECRETS AND ARE CONFIDENTIAL AND PROPRIETARY PRODUCTS OF ANSYS, INC., ITS SUBSIDIARIES, OR LICENSORS. The software products and documentation are furnished by ANSYS, Inc., its subsidiaries, or affiliates under a software license agreement that contains provisions concerning non-disclosure, copying, length and nature of use, compliance with exporting laws, warranties, disclaimers, limitations of liability, and remedies, and other provisions. The software products and documentation may be used, disclosed, transferred, or copied only in accordance with the terms and conditions of that software license agreement.

ANSYS, Inc. and ANSYS Europe, Ltd. are UL registered ISO 9001: 2015 companies.

U.S. Government Rights

For U.S. Government users, except as specifically granted by the ANSYS, Inc. software license agreement, the use, duplication, or disclosure by the United States Government is subject to restrictions stated in the ANSYS, Inc. software license agreement and FAR 12.212 (for non-DOD licenses).

Third-Party Software

See the [legal information](#) in the product help files for the complete Legal Notice for ANSYS proprietary software and third-party software. If you are unable to access the Legal Notice, contact ANSYS, Inc.

Published in the U.S.A.

Table of Contents

1. Getting Started	1
1.1. Building the Model	1
1.1.1. Specifying a Jobname and Analysis Title	1
1.1.1.1. Defining the Jobname	1
1.1.1.2. Defining an Analysis Title	2
1.1.1.3. Defining Units	2
1.1.2. Defining Element Types	2
1.1.2.1. Key Options	3
1.1.3. Creating Cross Sections	3
1.1.4. Defining Element Real Constants	3
1.1.5. Defining Material Properties	3
1.1.6. Creating the Model Geometry	4
1.2. Applying Loads and Obtaining the Solution	4
1.2.1. Specifying the Analysis Type and Analysis Options	4
1.2.2. Applying Loads	5
1.2.3. Specifying Load Step Options	6
1.2.4. Initiating the Solution	6
1.3. Reviewing the Results	6
2. Material Properties	9
2.1. Linear Material Properties	9
2.2. Nonlinear Material Properties	12
2.3. Anisotropic Elastic Material Properties	13
2.4. Material Model Interface	13
2.4.1. Accessing the Material Model Interface	13
2.4.2. Choosing Material Behavior	14
2.4.2.1. Material Favorites Folder	14
2.4.3. Entering Material Data	14
2.4.4. Logging/Editing Material Data	17
2.4.5. Example: Defining a Single Material Model	17
2.4.6. Example: Editing Data in a Material Model	18
2.4.7. Example: Defining a Material Model Combination	19
2.4.8. Material Model Interface - Miscellaneous Items	20
2.5. Using Material Library Files	20
2.5.1. Format of Material Library Files	21
2.5.2. Specifying a Default Read/Write Path for Material Library Files	21
2.5.3. Creating (Writing) a Material Library File	21
2.5.4. Reading a Material Library File	22
3. Loading	23
3.1. Understanding Loads	23
3.2. Load Steps, Substeps, and Equilibrium Iterations	24
3.3. The Role of Time in Tracking	26
3.4. Ramped and Stepped Loads	26
3.5. Applying Loads	27
3.5.1. Solid-Model Loads	28
3.5.1.1. Disadvantages of Solid-Model Loads	28
3.5.1.2. Other Considerations for Solid-Model Loads	28
3.5.2. Finite-Element Loads	29
3.5.2.1. Disadvantages of Finite-Element Loads	29
3.5.3. Degree-of-Freedom Constraints	29

3.5.3.1. Velocities and Accelerations	30
3.5.4. Applying Symmetry or Antisymmetry Boundary Conditions	30
3.5.5. Transferring Constraints	31
3.5.5.1. Resetting Constraints	31
3.5.5.2. Scaling Constraint Values	31
3.5.5.3. Resolution of Conflicting Constraint Specifications	32
3.5.6. Forces (Concentrated Loads)	33
3.5.6.1. Repeating a Force	34
3.5.6.2. Scaling Force Values	34
3.5.6.3. Transferring Forces	34
3.5.7. Surface Loads	35
3.5.7.1. Applying Pressure Loads on Beams	36
3.5.7.2. Specifying Node Number Vs. Surface Load	36
3.5.7.3. Specifying a Gradient Slope	37
3.5.7.3.1. Specifying the Gradient in a Cylindrical Coordinate System	38
3.5.7.4. Repeating a Surface Load	40
3.5.7.5. Transferring Surface Loads	40
3.5.7.6. Using Surface Effect Elements to Apply Loads	41
3.5.8. Applying Body Loads	41
3.5.8.1. Specifying Body Loads for Elements	42
3.5.8.2. Specifying Body Loads for Keypoints	43
3.5.8.3. Specifying Body Loads on Lines, Areas and Volumes	44
3.5.8.4. Specifying a Uniform Body Load	44
3.5.8.5. Repeating a Body Load Specification	44
3.5.8.6. Transferring Body Loads	44
3.5.8.7. Scaling Body Load Values	45
3.5.8.8. Resolving Conflicting Body Load Specifications	45
3.5.9. Applying Inertia Loads	46
3.5.10. Applying Ocean Loads	48
3.5.11. Applying Coupled-Field Loads	49
3.5.12. Axisymmetric Loads and Reactions	49
3.5.12.1. Hints and Restrictions	50
3.5.13. Loads to Which the Degree of Freedom Offers No Resistance	50
3.5.14. Initial State Loading	50
3.5.15. Applying Loads Using Tabular Input	51
3.5.15.1. Defining Primary Variables	51
3.5.15.2. Defining Independent Variables	54
3.5.15.3. Operating on Table Parameters	55
3.5.15.4. Verifying Boundary Conditions	55
3.5.15.5. Example Analysis Using 1-D Table Array	56
3.5.15.6. Example Analysis Using 5-D Table Array	56
3.5.16. Applying Loads to Components and Assemblies	58
3.5.16.1. Applying a Load to a Component	58
3.5.16.2. Applying a Load to an Assembly	58
3.6. Specifying Load Step Options	59
3.6.1. Setting General Options	59
3.6.1.1. Time Option	59
3.6.1.2. Number of Substeps and Time Step Size	59
3.6.1.3. Automatic Time Stepping	59
3.6.1.4. Stepping or Ramping Loads	60
3.6.1.5. Other General Options	61

3.6.2. Setting Dynamics Options	62
3.6.3. Setting Nonlinear Options	63
3.6.4. Setting Output Controls	63
3.6.4.1. Element Integration Point Values	64
3.6.5. Setting Biot-Savart Options	64
3.6.6. Setting Spectrum Options	65
3.7. Creating Multiple Load Step Files	65
3.8. Defining Pretension in a Joint Fastener	66
3.8.1. Applying Pretension to a Fastener Meshed as a Single Piece	66
3.8.2. Applying Pretension to a Fastener Meshed as Two Pieces	67
3.8.3. Example: Pretension Analysis	67
3.9. Defining Preload in a Joint Fastener Undergoing Large Rotation	70
3.9.1. Generating a Preload Section in a Fastener Meshed as a Single Piece	71
3.9.2. Applying a Preload to the Joint Element	74
3.9.3. Example: Bolt Sleeve Model Undergoing Large Rotation	75
4. Using the Function Tool	81
4.1. Function Tool Terminology	82
4.2. Using the Function Editor	82
4.2.1. How the Function Editor Works	82
4.2.1.1. Selecting Primary Variables in the Function Editor	83
4.2.2. Creating a Function with the Function Editor	84
4.2.3. Using Your Function	85
4.3. Using the Function Loader	85
4.4. Applying Boundary Conditions Using the Function Tool	86
4.5. Function Tool Example	86
4.6. Graphing or Listing a Function	91
4.6.1. Graphing a Function	92
4.6.2. Listing a Function	92
5. Solution	95
5.1. Selecting a Solver	95
5.2. Types of Solvers	97
5.2.1. The Sparse Direct Solver	97
5.2.1.1. Distributed Sparse Direct Solver	98
5.2.2. The Preconditioned Conjugate Gradient (PCG) Solver	99
5.2.3. The Jacobi Conjugate Gradient (JCG) Solver	101
5.2.4. The Incomplete Cholesky Conjugate Gradient (ICCG) Solver	101
5.2.5. The Quasi-Minimal Residual (QMR) Solver	102
5.3. Solver Memory and Performance	102
5.3.1. Running Solvers Using Parallel Processing	102
5.3.2. Using Large Memory Capabilities with the Sparse Solver	102
5.3.3. Disk Space (I/O) and Postprocessing Performance for Large Memory Problems	103
5.4. Using Special Solution Controls for Certain Types of Structural Analyses	103
5.4.1. Using Abridged Solution Menus	103
5.4.2. Using the Solution Controls Dialog Box	104
5.4.3. Accessing More Information	106
5.5. Obtaining the Solution	106
5.6. Solving Multiple Load Steps	107
5.6.1. Using the Multiple SOLVE Method	107
5.6.2. Using the Load Step File Method	107
5.6.3. Using the Array Parameter Method	108
5.7. Terminating a Running Job	110

5.8. Restarting an Analysis	110
5.8.1. Multiframe Restart	111
5.8.1.1. Multiframe File Restart Requirements	116
5.8.1.1.1. Multiframe Restart Limitations	117
5.8.1.2. Multiframe Restart Procedure	118
5.8.2. Modal Analysis Restart	120
5.9. Singular Matrices	123
5.10. Stopping Solution After Matrix Assembly	124
6. An Overview of Postprocessing	127
6.1. Postprocessors Available	127
6.2. The Results Files	129
6.3. Types of Data Available for Postprocessing	129
7. The General Postprocessor (POST1)	131
7.1. Reading Results Data into the Database	131
7.1.1. Reading Results Data	132
7.1.2. Other Options for Retrieving Results Data	133
7.1.2.1. Defining Data to be Retrieved	133
7.1.2.2. Reading Selected Results Information	133
7.1.2.3. Appending Data to the Database	133
7.1.2.3.1. Avoiding Data Mismatch	134
7.1.3. Creating an Element Table	135
7.1.3.1. Filling the Element Table for Variables Identified By Name	135
7.1.3.2. Filling the Element Table for Variables Identified By Sequence Number	135
7.1.3.3. Considerations for Defining Element Tables	136
7.1.4. Special Considerations for Principal Stresses	136
7.1.5. Resetting the Database	137
7.2. Reviewing Results in POST1	137
7.2.1. Displaying Results Graphically	137
7.2.1.1. Contour Displays	137
7.2.1.1.1. Hints and Recommendations	140
7.2.1.2. Deformed Shape Displays	141
7.2.1.3. Vector Displays	142
7.2.1.4. Path Plots	143
7.2.1.5. Reaction Force Displays	143
7.2.1.6. Charged Particle Traces	143
7.2.1.6.1. Hints and Recommendations	145
7.2.2. Surface Operations	146
7.2.2.1. Defining the Surface	146
7.2.2.2. Mapping Results Data Onto a Surface	147
7.2.2.3. Reviewing Surface Results	148
7.2.2.4. Performing Operations on Mapped Surface Result Sets	148
7.2.2.5. Archiving and Retrieving Surface Data to a File	148
7.2.2.6. Archiving and Retrieving Surface Data to an Array Parameter	149
7.2.2.7. Deleting a Surface	149
7.2.3. Listing Results in Tabular Form	149
7.2.3.1. Listing Nodal and Element Solution Data	150
7.2.3.2. Listing Reaction Loads and Applied Loads	150
7.2.3.3. Listing Element Table Data	152
7.2.3.4. Other Listings	153
7.2.3.5. Sorting Nodes and Elements	153
7.2.3.6. Custom Tabular Listings	154

7.2.4. Mapping Results onto a Path	154
7.2.4.1. Defining the Path	155
7.2.4.2. Using Multiple Paths	156
7.2.4.3. Interpolating Data Along the Path	156
7.2.4.4. Mapping Path Data	157
7.2.4.5. Reviewing Path Items	157
7.2.4.6. Performing Mathematical Operations among Path Items	157
7.2.4.7. Archiving and Retrieving Path Data to a File	158
7.2.4.8. Archiving and Retrieving Path Data to an Array Parameter	158
7.2.4.9. Deleting a Path	159
7.2.5. Estimating Solution Error	160
7.2.6. Using the Results Viewer to Access Results File Data	160
7.2.6.1. The Results Viewer Layout	161
7.2.6.1.1. Main Menu	161
7.2.6.1.2. Toolbar	162
7.2.6.1.3. Step/Sequence Data Access Controls	163
7.2.6.2. Results Viewer Context-Sensitive Menus	164
7.3. Additional POST1 Postprocessing	166
7.3.1. Rotating Results to a Different Coordinate System	167
7.3.2. Performing Arithmetic Operations Among Results Data	169
7.3.2.1. Hints and Recommendations	170
7.3.3. Creating and Combining Load Cases	171
7.3.3.1. Saving a Combined Load Case	173
7.3.3.1.1. Hints and Recommendations	174
7.3.3.2. Combining Load Cases in Harmonic Element Models	174
7.3.3.3. Summable, Non-Summable, and Constant Data	175
7.3.4. Mapping Results onto a Different Mesh or to a Cut Boundary	177
7.3.5. Creating or Modifying Results Data in the Database	177
7.3.6. Splitting Large Results Files	177
7.3.7. Magnetics Command Macros	178
7.3.8. Comparing Nodal Solutions From Two Models or From One Model and Experimental Data (RSTMAC)	179
7.3.8.1. Matching the Nodes of Model 1 and Model 2 Based on Node Location	181
7.3.8.2. Mapping the Nodes of Model 2 Into Elements of Model 1	181
7.3.8.3. Matching the Nodes of Model 1 and Model 2 Based on Their Node Numbers	182
7.3.8.4. Evaluate MAC Between Solutions at Matched/Mapped Nodes	183
7.3.8.5. Evaluate MAC per node pair	184
7.3.8.6. Match the Solutions	184
7.3.8.7. Universal Format File Records	184
7.3.8.8. Examples Using RSTMAC with Different MACOPT Options	185
7.3.8.8.1. Comparing Two Plates Modeled Using Different Elements	185
7.3.8.8.2. Comparing Two Cantilever Beams Meshed Along Different Directions	189
8. The Time-History Postprocessor (POST26)	193
8.1. The Time-History Variable Viewer	194
8.2. Defining Variables	196
8.2.1. Defining the Variable	197
8.2.1.1. Special Considerations for Defining Variables	198
8.2.2. Storing the Variable	198
8.2.2.1. Variable Storage Options	198
8.3. Processing Variables to Develop Calculated Data	199
8.4. Importing Data	200

8.5. Exporting Data	201
8.6. Reviewing the Variables	202
8.6.1. Plotting Result Graphs	202
8.6.2. Listing Results in Tabular Form	203
8.7. Additional Time-History Postprocessing	204
8.7.1. Generating a Response Spectrum	204
8.7.2. Data Smoothing	205
9. Selecting and Components	207
9.1. Selecting Entities	207
9.1.1. Selecting Entities Using Commands	209
9.1.2. Selecting Entities Using the GUI	210
9.1.3. Selecting Lines to Repair CAD Geometry	210
9.1.4. Other Commands for Selecting	211
9.2. Selecting for Meaningful Postprocessing	211
9.3. Using Components and Assemblies	212
9.3.1. Component Types	213
9.3.2. Using the Component Manager	213
9.3.3. Creating Components	213
9.3.4. Nesting Assemblies	214
9.3.5. Selecting Entities by Component or Assembly	214
9.3.6. Adding or Removing Components	215
9.3.7. Modifying Components or Assemblies	215
9.3.8. Viewing Hidden Element Components	215
10. Getting Started with Graphics	217
10.1. Interactive and External Graphics	217
10.2. Identifying the Graphics Device Name for Linux	217
10.2.1. Graphics Device Names Available	218
10.3. Specifying the Graphics Display Device Type (for Windows)	218
10.4. System-Dependent Graphics Information	219
10.4.1. Adjusting Input Focus	219
10.4.2. Displaying X11 Graphics Over Networks	219
10.5. Creating Graphics Displays	220
10.5.1. GUI-Driven Graphics Functions	220
10.5.2. Command-Driven Graphics Functions	221
10.5.3. Immediate Mode Graphics	221
10.5.4. Replotting the Current Display	221
10.5.5. Erasing the Current Display	221
10.5.6. Aborting a Display in Progress	222
10.6. Multi-Plotting Techniques	222
10.6.1. Defining the Window Layout	222
10.6.2. Controlling the Entities That Each Window Displays	222
10.6.3. Controlling the Plot Display	223
10.6.4. Displaying Selected Entities	223
11. General Graphics Specifications	225
11.1. Multiple Windows and Superimposed Displays	225
11.1.1. Superimposing (Overlaying) Multiple Displays	226
11.1.2. Removing Frame Borders	226
11.2. Changing the Viewing Angle, Zooming, and Panning	226
11.2.1. Changing the Viewing Direction	226
11.2.2. Rotating the Display About a Specified Axis	227
11.2.3. Determining the Model Coordinate System Reference Orientation	227

11.2.4. Translating (or Panning) the Display	227
11.2.5. Magnifying (Zooming in on) the Image	227
11.2.6. Resetting Automatic Scaling and Focus	227
11.2.7. Freezing Scale (Distance) and Focus	227
11.3. Controlling Miscellaneous Text and Symbols	227
11.3.1. Using Legends in Your Displays	228
11.3.1.1. Controlling the Content of Your Legends	228
11.3.1.2. Controlling the Placement of Your Contour Legend	229
11.3.2. Controlling Entity Fonts	229
11.3.3. Controlling the Location of the Global XYZ Triad	229
11.3.4. Displaying and Hiding Triad Symbols	230
11.3.5. Changing the Style of the Working Plane Grid	230
11.3.6. Displaying and Hiding the Program Logo	230
11.4. Miscellaneous Graphics Specifications	230
11.4.1. Reviewing Graphics Control Specifications	230
11.4.2. Restoring Defaults for Graphics Slash Commands	230
11.4.3. Saving the Display Specifications in a File	231
11.4.4. Recalling Display Specifications from a File	231
11.4.5. Pausing the Program	231
11.5. 3-D Input Device Support	231
12. PowerGraphics	233
12.1. Characteristics of PowerGraphics	233
12.2. When to Use PowerGraphics	234
12.3. Activating and Deactivating PowerGraphics	234
12.4. Using PowerGraphics	234
12.5. What to Expect from a PowerGraphics Plot	234
12.5.1. Viewing Your Element Model	234
12.5.2. Printing and Plotting Node and Element Results	235
13. Creating Geometry Displays	237
13.1. Creating Displays of Solid-Model Entities	237
13.2. Controlling the Geometry Display Specifications	238
13.2.1. Controlling the Style of Your Display	238
13.2.1.1. Displaying Line and Shell Elements as Solids	239
13.2.1.2. Displaying Only the Edges of an Object	239
13.2.1.3. Displaying the Interior Element Edges of an Object	239
13.2.1.4. Using Dashed Element Outlines	239
13.2.1.5. Shrinking Entities for Clarity	239
13.2.1.6. Changing the Display Aspect Ratio	239
13.2.1.7. Changing the Number of Facets	240
13.2.1.8. Changing Facets for PowerGraphics Displays	240
13.2.1.9. Changing Hidden-Line Options	240
13.2.1.10. Section, Slice, or Capped Displays	240
13.2.1.11. Specifying the Cutting Plane	240
13.2.1.12. Vector Vs. Raster Mode	241
13.2.1.13. Perspective Displays	241
13.2.2. Applying Styles to Enhance the Model Appearance	241
13.2.2.1. Applying Textures to Selected Items	241
13.2.2.2. Creating Translucent Displays	242
13.2.2.3. Changing Light-Source Shading	242
13.2.2.4. Adding Background Shading and Textures	242
13.2.2.5. Using the Create Best Quality Image Capability	242

13.2.3. Controlling Numbers and Colors	244
13.2.3.1. Turning Item Numbers On and Off	245
13.2.3.2. Specifying a Format for the Graphical Display of Numbers	245
13.2.3.3. Controlling Number and Color Options	245
13.2.3.4. Controlling Color Values	245
13.2.4. Displaying Loads and Other Special Symbols	246
13.2.4.1. Enaling and Disabling Load Symbols and Contours	246
13.2.4.2. Displaying Boundary Condition Values Next to a Symbol	246
13.2.4.3. Displaying Boundary Condition Symbols for Hidden Surfaces	247
13.2.4.4. Scaling Vector Load Symbols	247
13.2.4.5. Enabling and Disabling Other Symbols	247
14. Creating Geometric Results Displays	249
14.1. Using the GUI to Display Geometric Results	249
14.2. Options for Creating Geometric Results Displays	250
14.3. Changing the Specifications for POST1 Results Displays	251
14.3.1. Controlling Displaced Shape Displays	251
14.3.2. Controlling Vector Symbols in Your Results Display	252
14.3.3. Controlling Contour Displays	252
14.3.4. Changing the Number of Contours	253
14.4. Q-Slice Techniques	254
14.5. Isosurface Techniques	254
14.6. Controlling Charged Particle Trace Displays	255
15. Creating Graphs	257
15.1. Graph Display Actions	257
15.2. Changing the Specifications for Graph Displays	258
15.2.1. Changing the Type, Style, and Color of Your Graph Display	258
15.2.2. Labeling Your Graph	259
15.2.3. Defining X and Y Variables and Their Ranges	259
15.2.3.1. Defining the X Variable	260
15.2.3.2. Defining the Part of the Complex Variable to Be Displayed	260
15.2.3.3. Defining the Y Variable	260
15.2.3.4. Setting the X Range	260
15.2.3.5. Defining the TIME (or FREQ for Harmonic Analysis) Range	260
15.2.3.6. Setting the Y Range	260
16. Annotation	261
16.1. 2-D Annotation	261
16.2. Creating Annotations for Models	262
16.3. 3-D Annotation	263
16.4. 3-D Query Annotation	264
17. Animation	265
17.1. Creating Animated Displays	265
17.2. Using the Basic Animation Commands	265
17.3. Using One-Step Animation Macros	266
17.4. Animation in the Windows Environment	267
17.4.1. Understanding AVI File Support	267
17.4.2. Other Uses for AVI Files	267
18. External Graphics	269
18.1. External Graphics Options	269
18.1.1. Printing Graphics in Windows	269
18.1.2. Exporting Graphics in Windows	269
18.1.2.1. PNG Format	270

18.1.2.2. ClearType and Reverse Video	270
18.1.2.3. Create Exportable Graphics	270
18.1.2.4. Export Windows Metafiles	270
18.1.3. Printing Graphics in Linux	271
18.1.4. Exporting Graphics in Linux	271
19. The Report Generator	273
19.1. Starting the Report Generator	273
19.1.1. Specifying a Location for Captured Data and Reports	274
19.1.2. Understanding the Behavior of the Graphics Window	274
19.1.3. A Note About the Graphics File Format	274
19.2. Capturing an Image	274
19.2.1. Interactive	274
19.2.2. Batch	275
19.3. Capturing Animation	275
19.3.1. Interactive	275
19.3.2. Batch	276
19.4. Capturing a Data Table	276
19.4.1. Interactive	276
19.4.1.1. Creating a Custom Table	277
19.4.2. Batch	278
19.5. Capturing a Listing	279
19.5.1. Interactive	280
19.5.2. Batch	280
19.6. Assembling a Report	280
19.6.1. Interactive Report Assembly	280
19.6.2. Batch Report Assembly	283
19.6.3. Report Assembly Using the JavaScript Interface	283
19.6.3.1. Inserting an Image	283
19.6.3.2. Inserting an Animation	284
19.6.3.3. Inserting a Data Table	285
19.6.3.4. Inserting a Listing	285
19.7. Setting Report Generator Defaults	286
20. File Management and Files	289
20.1. File Management Overview	289
20.2. Changing the Default File Name	290
20.3. Sending Output to Screens, Files, or Both	290
20.4. Text and Binary Files	290
20.4.1. Program-Generated Files	291
20.4.2. File Compression	294
20.5. Reading Your Own Files into the Program	295
20.6. Writing Your Own Files from the Program	296
20.7. Assigning File Names	296
20.8. Reviewing Contents of Binary Files (AUX2)	296
20.9. Operating on Results Files (AUX3)	297
20.10. Other File Management Commands	297
21. Memory Management and Configuration	299
21.1. Work and Swap Space Requirements	299
21.2. How the Program Uses Work Space	300
21.3. How and When to Perform Memory Management	301
21.3.1. Determining When to Change the Work Space	301
21.3.2. Changing the Amount of Work Space	301

21.3.3. Changing the Amount of Database Space	302
21.4. Using the Configuration File	303
21.5. Understanding Memory Error Messages	307

List of Figures

1.1. Sample Finite Element Models	4
2.1. Sample MPPLLOT Display	11
2.2. Sample TBPLLOT Display	12
2.3. Material Model Interface Initial Screen	13
2.4. Material Model Interface Tree Structure	14
2.5. A Data Input Dialog Box	15
2.6. Data Input Dialog Box - Added Column	15
2.7. Data Input Dialog Box - Added Row	16
3.1. Transient Load History Curve	25
3.2. Load Steps, Substeps, and Equilibrium Iterations	25
3.3. Ramped vs. Stepped Loads	27
3.4. Quadratic Interpolation of Rotational Velocities	27
3.5. Symmetry and Antisymmetry Boundary Conditions	30
3.6. Examples of Boundary Conditions	31
3.7. Scaling Temperature Constraints with DSCALE	32
3.8. Example of Beam Surface Loads	36
3.9. Surface Load Gradient	38
3.10. Tapered Load on a Cylindrical Shell	39
3.11. Violation of Guideline 2 (Left) and Guideline 1 (Right)	40
3.12. BFE Load Locations	42
3.13. BFE Load Locations for Shell Elements	43
3.14. BFE Load Locations for Beam and Pipe Elements	43
3.15. Transfers to BFK Loads to Nodes	44
3.16. Inertia Loads Commands	47
3.17. Concentrated Axisymmetric Loads	50
3.18. Central Constraint for Solid Axisymmetric Structure	50
3.19. Pressure Distribution for Load Case 1	57
3.20. Pressure Distribution for Load Case 2	58
3.21. Pretension Definition	66
3.22. Initial Meshed Structure	68
3.23. Pretension Section	68
3.24. Pretension Stress	69
3.25. Bolt Pretension, Torque, and Rotation Around the Bolt Axis	71
3.26. Force Distribution on Two Surfaces After Cutting with PSMESH	71
3.27. Two Force-Distributed Constraints with Joint Element at Cutting Location	76
3.28. UZ Displacement	77
3.29. Normal Stress Plot in the Z-direction	78
3.30. Deformed Shape at Last Load Step	79
3.31. von-Mises Stress at Last Load Step	79
3.32. Animation of von-Mises Stress	80
5.1. Solution Controls Dialog Box	104
5.2. Examples of Time-Varying Loads	108
6.1. A Typical POST1 Contour Display	128
6.2. A Typical POST26 Graph	128
7.1. Contouring Primary Data with PLNSOL	138
7.2. Contouring Derived Data with PLNSOL	138
7.3. A Sample PLESOL Plot Showing Discontinuous Contours	139
7.4. Averaged PLETAB Contours	139
7.5. Nonaveraged PLETAB Contours	140

7.6. A Sample PLDISP Plot	142
7.7. PLVECT Vector Plot of Magnetic Field Intensity	142
7.8. Charge Particle Trace in Electric and/or Magnetic Fields	144
7.9. A Node Plot Showing the Path	156
7.10. PLPATH Display Showing Stress Discontinuity at a Material Interface	159
7.11. PLPAGM Display	159
7.12. The Results Viewer	160
7.13. Results Viewer File Menu	161
7.14. Results Viewer View Menu	162
7.15. Results Viewer Toolbar	162
7.16. Results Viewer Step/Sequence Data Access Controls	163
7.17. Graphics Window Context Menu	165
7.18. Rotation of Results by RSYS	167
7.19. SY in Global Cartesian and Cylindrical Systems	168
7.20. An Overview of MACOPT Command Options	180
7.21. SHELL181 Model vs BEAM189 Model	186
7.22. Two Simply Supported Beams.	189
8.1. Example Time-History Plot (XVAR = 1 (time))	202
8.2. Example Time-History Plot (XVAR ≠ 1)	203
9.1. Shell Model with Different Thicknesses	212
9.2. Layered Shell (SHELL281) with Nodes Located at Midplane	212
9.3. Layered Shell (SHELL281) with Nodes Located at Bottom Surface	212
9.4. Nested Assembly Schematic	214
11.1. Focus Point, Viewpoint, and Viewing Distance	226
11.2. The Window Options Dialog Box	228
11.3. The Multi Legend Text Legend	229
11.4. The Multi Legend Contour Legend	229
13.1. Create Best Quality Image Function Box	243
14.1. Contour Results Plot	249
14.2. A Typical Results Plot	251
15.1. Typical ANSYS Graphs	257
16.1. Stroke Text Annotation Dialog Box	262
19.1. Report Generator GUI	273
19.2. Custom Table Definition	277
19.3. HTML Report Assembler Window	281
19.4. Report Generator Settings Dialog	286
21.1. Comparing Available Memory	299
21.2. Work Space	300
21.3. Changing Work Space	302
21.4. Dividing Work Space	303
21.5. Memory Diagram in Terms of Configuration Keywords	305

List of Tables

3.1. DOF Constraints Available in Each Discipline	29
3.2. Commands for DOF Constraints	30
3.3. Forces Available in Each Discipline	33
3.4. Commands for Applying Force Loads	34
3.5. Surface Loads Available in Each Discipline	35
3.6. Commands for Applying Surface Loads	36
3.7. Body Loads Available in Each Discipline	41
3.8. Commands for Applying Body Loads	42
3.9. Ways of Specifying Density	48
3.10. Boundary Condition Type and Corresponding Primary Variable	51
3.11. Real Constants and Corresponding Primary Variable for SURF151, SURF152, and FLUID116	54
3.12. Handling of Ramped Loads (KBC = 0) Under Different Conditions	61
3.13. Dynamic and Other Transient Analyses Commands	62
3.14. Nonlinear Analyses Commands	63
3.15. Output Controls Commands	63
3.16. Biot-Savart Commands	64
3.17. Surface-Based Constraints Generated by PSMESH	72
5.1. Shared Memory Solver Selection Guidelines	96
5.2. Distributed Memory Solver Selection Guidelines	96
5.3. Relationships Between Tabs of the Solution Controls Dialog Box and Commands	105
6.1. Primary and Derived Data for Different Disciplines	129
7.1. Saving the Database Properly for Postprocessing	131
7.2. Examples of Summable POST1 Results	176
7.3. Examples of Non-Summable POST1 Results	176
7.4. Examples of Constant POST1 Results	176
7.5. MACOPT Commands and Results of RSTMAC Command for All Cases	188
7.6. RSTMAC Command Results for All Cases	191
9.1. Selection Functions	207
9.2. Select Commands	209
11.1. Basic Window Operations	225
13.1. Commands for Displaying Solid-Model Entities	237
14.1. Commands for Creating Geometric Results Displays	250
17.1. Animation Macros	266
20.1. Program-Generated Temporary Files	291
20.2. Program-Generated Permanent Files	292
20.3. Commands for Reading in Special-Purpose Text Files	295
20.4. Commands for Reading in Binary Files	295
20.5. Other Commands for Writing Files	296
20.6. Special-Purpose File Management Commands	297

Chapter 1: Getting Started

The program has many finite-element analysis capabilities, ranging from a simple, linear, static analysis to a complex, nonlinear, transient dynamic analysis. The analysis guides in the documentation set describe specific procedures for performing analyses for different engineering disciplines.

The process for a typical analysis involves three general tasks:

- 1.1. Building the Model
- 1.2. Applying Loads and Obtaining the Solution
- 1.3. Reviewing the Results

1.1. Building the Model

Building a finite element model requires more of your time than any other part of the analysis. First, you specify a jobname and analysis title. Then, you use the PREP7 preprocessor to define the element types, element real constants, material properties, and the model geometry.

1.1.1. Specifying a Jobname and Analysis Title

A jobname is not required for an analysis but is *recommended*.

- 1.1.1.1. Defining the Jobname
- 1.1.1.2. Defining an Analysis Title
- 1.1.1.3. Defining Units

1.1.1.1. Defining the Jobname

The *jobname* is a name that identifies an analysis job. When you define a jobname for an analysis, the jobname becomes the first part of the name of all files the analysis creates. (The extension or suffix for these files' names is a file identifier such as .DB.) Assigning a jobname for each analysis ensures that no files are overwritten.

If no jobname is specified, all files are named FILE. You can change the default jobname by using the initial jobname entry option when you start the Mechanical APDL program (via the [launcher](#) or [execution command](#)) or via the **/FILNAME** command.

The **/FILNAME** command is valid only at the Begin level. It lets you change the jobname even if you had specified an initial jobname after starting the program. The jobname applies only to files you open after using **/FILNAME** and not to files that were already open. To start new files (such as the log file, Jobname.LOG, and error file Jobname.ERR) via **/FILNAME**, set the *Key* argument to 1; otherwise, the files that were already open will still have the initial jobname.

1.1.1.2. Defining an Analysis Title

The **/TITLE** command defines a title for the analysis. The program includes the title on all graphics displays and on the solution output. Issue the **/STITLE** command to add subtitles, which appear in the output but not in graphic displays.

1.1.1.3. Defining Units

The program does not assume a system of units for your analysis. Except in magnetic field analyses, you can use any system of units so long as you make sure that you use that system for all the data you enter. (Units must be consistent for all input data.)

For micro-electromechanical systems (MEMS), where dimensions are on the order of microns, see the conversion factors in *System of Units in the Coupled-Field Analysis Guide*.

Using the **/UNITS** command, you can set a marker in the database indicating the system of units that you are using. This command *does not* convert data from one system of units to another; it simply serves as a record for subsequent reviews of the analysis.

1.1.2. Defining Element Types

The element library offers [many element types](#). Each element type has a prefix identifying the element category and a unique ID number: **PLANE182**, **SOLID185**, **BEAM188**, **ELBOW290**, and so on.

The element type determines:

- The degree-of-freedom set (which in turn implies the discipline - structural, thermal, magnetic, electric, quadrilateral, brick, etc.)
- Whether the element lies in 2-D or 3-D space.

BEAM188, for example, has six structural degrees of freedom (UX, UY, UZ, ROTX, ROTY, ROTZ), is a line element, and can be modeled in 3-D space. **PLANE77** has a thermal degree of freedom (TEMP), is an 8-node quadrilateral element, and can be modeled only in 2-D space.

You must be in PREP7, the general preprocessor, to define element types. To do so, use the **ET** family of commands (**ET**, **ETCHG**, etc.). Define the element type by name and give the element a type reference number.

Example 1.1: Defining Element Types

The following commands define two element types, **BEAM188** and **SHELL181**, and assign them type reference numbers 1 and 2, respectively:

```
ET,1,BEAM188  
ET,2,SHELL181
```

This table of type reference number versus element name is called the *element type table*. While defining the actual elements, point to the appropriate type reference number via the **TYPE** command.

1.1.2.1. Key Options

Many element types have *key options*, or KEYOPTs, and are referred to as KEYOPT(1), KEYOPT(2), etc. For example, KEYOPT(3) for **BEAM188** sets the shape function along the length of the beam, and KEYOPT(8) for **SHELL181** specifies how layer data should be stored.

Specify KEYOPTs via the **ET** or **KEYOPT** command.

1.1.3. Creating Cross Sections

If you are building a model using beams or shells, issue the section commands (**SECTYPE**, **SECDATA**, etc.) to define and use cross sections in your models. See [Beam and Pipe Cross Sections in the Structural Analysis Guide](#) for information about using BeamTool to create cross sections.

For line and area elements that require geometry data (cross-sectional area, thickness, diameter, etc.) to be specified as sections, you can verify the input graphically via the **/ESHAPe** and **EPlot** commands (in that order).

The program displays the elements as solid elements, determining the geometry from the section shape and dimension. A rectangular cross-section is used for links.

1.1.4. Defining Element Real Constants

Element real constants are properties that depend on the element type, such as contact stiffness and penetration. Not all element types require real constants, and different elements of the same type may have different real constant values.

Specify real constants via the **R** family of commands (**R**, **RMODIF**, etc.) or their equivalent menu paths; see the [Command Reference](#) for further information. As with element types, each set of real constants has a reference number, and the table of reference number versus real constant set is called the *real constant table*. While defining the elements, point to the appropriate real constant reference number via the **REAL** command.

Following are hints for defining real constants:

- For models using multiple element types, use a separate real constant set (that is, a different **REAL** reference number) for each element type. The program issues a warning message if multiple element types reference the same real constant set. However, a single element type may reference several real constant sets.
- To verify your real constant input, issue the **RLIST** and **ELIST** commands, with *RKEY* = 1. **RLIST** lists real constant values for all sets. The command **ELIST**...1 displays an easier-to-read list that shows, for each element, the real constant labels and their values.

1.1.5. Defining Material Properties

Most element types require material properties. Depending on the application, material properties can be linear (see [Linear Material Properties \(p. 9\)](#)) or nonlinear (see [Nonlinear Material Properties \(p. 12\)](#)).

As with element types and real constants, each set of material properties has a material reference number. The table of material reference numbers versus material property sets is called the *material*

table. Within one analysis, you may have multiple material property sets (to correspond with multiple materials used in the model). The program identifies each set with a unique reference number.

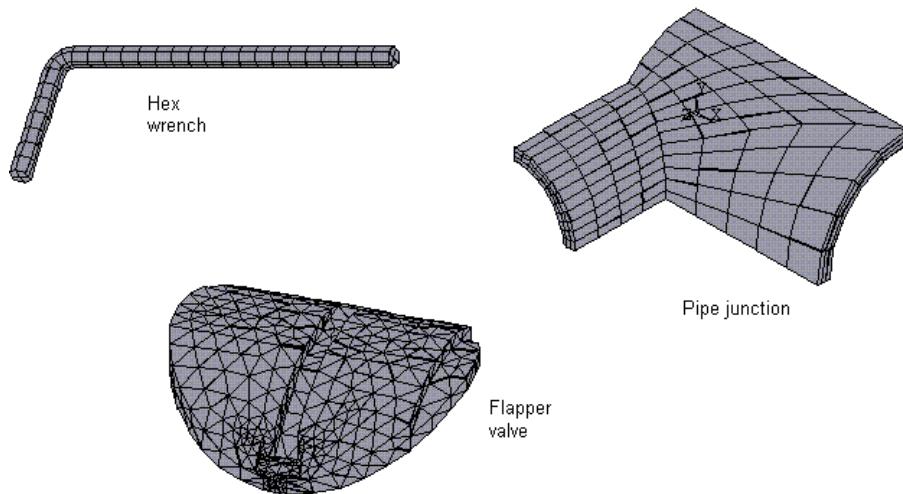
While defining the elements, point to the appropriate material reference number via **MAT**.

See [Material Properties \(p. 9\)](#) for more information about the types of material properties and how to define them.

1.1.6. Creating the Model Geometry

Once you have defined material properties, the next step in an analysis is generating a finite element model - nodes and elements - that adequately describes the model geometry. The graphic below shows some sample finite element models:

Figure 1.1: Sample Finite Element Models



There are two methods to create the finite element model: solid modeling and direct generation. With *solid modeling*, you describe the geometric shape of your model, then instruct the program to automatically *mesh* the geometry with nodes and elements. You can control the size and shape in the elements that the program creates. With *direct generation*, you "manually" define the location of each node and the connectivity of each element. Several convenience operations, such as copying patterns of existing nodes and elements, symmetry reflection, etc. are available.

Details of the two methods and many other aspects related to model generation - coordinate systems, working planes, coupling, constraint equations, etc. - are described in the [Modeling and Meshing Guide](#).

1.2. Applying Loads and Obtaining the Solution

In this step, the SOLUTION processor defines the analysis type and analysis options, apply loads, specify load step options, and initiate the finite element solution. You can also apply loads via the PREP7 pre-processor.

1.2.1. Specifying the Analysis Type and Analysis Options

Specify the analysis type based on the loading conditions and the response you wish to calculate. For example, if natural frequencies and mode shapes are to be calculated, you would choose a

modal analysis. You can perform the following analysis types in the program: static (or steady-state), transient, harmonic, modal, spectrum, buckling, and substructuring.

Not all analysis types are valid for all disciplines. Modal analysis, for example, is not valid for a thermal model. The analysis guides in the documentation set describe the analysis types available for each discipline and the procedures to do those analyses.

Analysis options allow you to customize the analysis type. Typical analysis options are the method of solution, stress stiffening on or off, and Newton-Raphson options.

To define the analysis type and analysis options, issue the **ANTYPE** command (**Main Menu> Preprocessor> Loads> Analysis Type> New Analysis** or **Main Menu> Preprocessor> Loads> Analysis Type> Restart**) and the appropriate analysis option commands (**TRNOPT**, **HROPT**, **MODOPT**, **NROPT**, etc.). For GUI equivalents for the other commands, see the documentation for the given command in the *Command Reference*.

If you are performing a static or full transient analysis, you can take advantage of the Solution Controls dialog box to define many options for the analysis. For details about the Solution Controls dialog box, see [Solution \(p. 95\)](#).

You can specify either a new analysis or a restart, but a new analysis is the norm in most cases. A multiframe restart that enables you to restart an analysis at any point is available for static and transient (full or mode-superposition method) analyses. For more information, see [Restarting an Analysis \(p. 110\)](#). The various analysis guides provide more specific information about restart requirements. You cannot change the analysis type and analysis options after the first solution.

An example input listing for a structural transient analysis is shown below. Remember that the discipline (structural, thermal, magnetic, etc.) is implied by the *element types* used in the model.

```
ANTYPE, TRANS
TRNOPT, FULL
NLGEOM, ON
```

After you have defined the analysis type and analysis options, the next step is to apply loads. Some structural analysis types require other items to be defined first, such as master degrees of freedom and gap conditions. The *Structural Analysis Guide* describes these items where necessary.

1.2.2. Applying Loads

The word *loads* as used in the documentation includes boundary conditions (constraints, supports, or boundary field specifications) as well as other externally and internally applied loads. Loads are divided into these categories:

- DOF Constraints
- Forces
- Surface Loads
- Body Loads
- Inertia Loads
- Coupled-field Loads

You can apply most of these loads either on the solid model (keypoints, lines, and areas) or the finite element model (nodes and elements). For details about the load categories and how they can be applied on your model, see [Loading \(p. 23\)](#) in this manual.

Two important load-related terms you need to know are load step and substep. A *load step* is simply a configuration of loads for which you obtain a solution. In a structural analysis, for example, you may apply wind loads in one load step and gravity in a second load step. Load steps are also useful in dividing a transient load history curve into several segments.

Substeps are incremental steps taken within a load step. You use them primarily for accuracy and convergence purposes in transient and nonlinear analyses. Substeps are also known as *time steps* - steps taken over a period of time.

Note:

The program uses the concept of *time* in transient analyses as well as static (or steady-state) analyses. In a transient analysis, time represents actual time, in seconds, minutes, or hours. In a static or steady-state analysis, time simply acts as a counter to identify load steps and substeps.

1.2.3. Specifying Load Step Options

Load step options are options that you can change from load step to load step, such as number of substeps, time at the end of a load step, and output controls. Depending on the type of analysis you are doing, load step options may or may not be required. The analysis procedures in the analysis guide manuals describe the appropriate load step options as necessary. See [Loading \(p. 23\)](#) for a general description of load step options.

1.2.4. Initiating the Solution

Issue the **SOLVE** command to initiate solution calculations. When you issue the command, the program takes model and loading information from the database and calculates the results.

The program writes the results to the results file (`Jobname.RST`, `Jobname.RTH`, or `Jobname.RMG`) and also to the database. The only difference is that only one set of results can reside in the database at one time, while you can write all sets of results (for all substeps) to the results file.

You can conveniently solve multiple load steps (**LSSOLVE**). [Solution \(p. 95\)](#) discusses this and other solution-related topics.

1.3. Reviewing the Results

After the solution has been calculated, use the postprocessors to review the results. Two postprocessors are available: POST1 and POST26.

- Use [POST1 \(p. 131\)](#), the general postprocessor, to review results at one substep (time step) over the entire model or selected portion of the model. The command for entering POST1 is **/POST1**, valid only at the Begin level. You can obtain contour displays, deformed shapes, and tabular listings to review and interpret the results of the analysis. POST1 offers many other capabilities, including error estimation, load case combinations, calculations among results data, and path operations.

- Use **POST26** (p. 193), the time-history postprocessor, to review results at specific points in the model over all time steps. The command for entering POST26 is **/POST26**, valid only at the Begin level. You can obtain graph plots of results data versus time (or frequency) and tabular listings. Other POST26 capabilities include arithmetic calculations and complex algebra.

Chapter 2: Material Properties

Linear material properties are typically defined via the **MP** family of commands, and nonlinear material properties require the **TB** family of commands. (The **TB** command can also be used to input *some* linear material properties, such as **anisotropic elasticity**, **material structural damping**, and **piezoelectric matrix**.)

Material properties can be temperature-dependent, and some can be defined as a function of primary variables. The material library file capability enables you to store frequently used materials for future use.

The following material property topics are available:

- 2.1. Linear Material Properties
- 2.2. Nonlinear Material Properties
- 2.3. Anisotropic Elastic Material Properties
- 2.4. Material Model Interface
- 2.5. Using Material Library Files

Also see the *Material Reference* for detailed information about specific material models.

2.1. Linear Material Properties

Linear material properties can be constant or temperature-dependent, and isotropic or orthotropic. Issue the **MP** command to define *constant* material properties (either isotropic or orthotropic).

Specify the appropriate property label on the **MP** command (for example, EX, EY, EZ for Young's modulus, and KXX, KYY, KZZ for thermal conductivity). For isotropic material, define only the X direction property; the other directions default to the X direction value. For example:

```
MP,EX,1,2E11 ! Young's modulus for material ref. no. 1 is 2E11
MP,DENS,1,7800 ! Density for material ref. no. 1 is 7800
MP,KXX,1,43   ! Thermal conductivity for material ref. no 1 is 43
```

Some material property defaults are built-in to reduce the amount of input. For example, Poisson's ratio (NUXY) defaults to 0.3, shear modulus (GXY) defaults to EX/2(1+NUXY)), and emissivity (EMIS) defaults to 1.0. See [Linear Material Properties in the Material Reference](#) for details.

An example material library which contains constant, isotropic, linear material properties is provided (see [Reading a Material Library File \(p. 22\)](#)). Young's modulus, density, coefficient of thermal expansion,

Poisson's ratio, thermal conductivity and specific heat are available for 10 materials in four unit systems. Electromagnetic examples are available for 6 materials in SI units.

Caution:

The property values in the example material library are provided for your convenience. They are typical values for the materials you can use for preliminary analyses and noncritical applications. As always, *you* are responsible for all data input to the program.

To define *temperature-dependent* material properties, issue the **MP** command in combination with the **MPTEMP** or **MPTGEN** command. You also can issue the **MPTEMP** and **MPDATA** commands. The **MP** command allows you to define a property-versus-temperature function in the form of a polynomial. The polynomial may be linear, quadratic, cubic, or quartic:

$$\text{Property} = C_0 + C_1 T + C_2 T^2 + C_3 T^3 + C_4 T^4$$

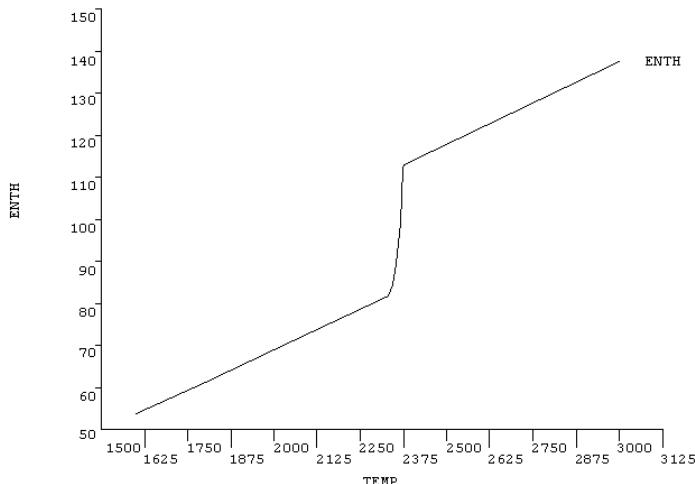
C_n are the coefficients and T is the temperature. You enter the coefficients using the $C0$, $C1$, $C2$, $C3$, and $C4$ arguments on the **MP** command. If you specify just $C0$, the material property is constant; if you specify $C0$ and $C1$, the material property varies linearly with temperature; and so on. When you specify a temperature-dependent property in this manner, the program internally evaluates the polynomial at discrete temperature points with linear interpolation between points (that is, piecewise linear representation) and a constant-valued extrapolation beyond the extreme points. You *must* use the **MPTEMP** or **MPTGEN** command *before* the **MP** command for second and higher-order properties to define appropriate temperature steps.

The second way to define temperature-dependent material properties is to issue a combination of **MPTEMP** and **MPDATA** commands. **MPTEMP** (or **MPTGEN**) defines a series of temperatures, and **MPDATA** defines corresponding material property values. For example, the following commands define a temperature-dependent enthalpy for material 4:

```
MPTEMP,1,1600,1800,2000,2325,2326,2335      ! 6 temperatures (temps 1-6)
MPTEMP,7,2345,2355,2365,2374,2375,3000      ! 6 more temps (temps 7-12)
MPDATA,ENTH,4,1,53.81,61.23,68.83,81.51,81.55,82.31 ! Corresponding
MPDATA,ENTH,4,7,84.48,89.53,99.05,112.12,113.00,137.40 ! enthalpy values
```

If an unequal number of property data points and temperature data points are defined, the program uses only those locations having both points defined for the property function table. To define a different set of temperatures for the next material property, you should first erase the *current* temperature table by issuing **MPTEMP** (without any arguments) and then define new temperatures (using additional **MPTEMP** or **MPTGEN** commands).

The **MPPLLOT** command displays a graph of material property versus temperature. [Figure 2.1: Sample MPPLLOT Display \(p. 11\)](#) shows a plot of the enthalpy-temperature curve defined in the example above. The **MPLIST** command lists material properties.

Figure 2.1: Sample MPLOT Display

Consider the following hints for temperature-dependent material properties:

- To modify a property data point on an existing curve, simply redefine the desired data point by issuing **MPDATA** with the appropriate location number. For example, to change the ENTH value in location 6 of the above enthalpy-temperature curve from 82.31 to 83.09, the command would be:

```
MPDATA, ENTH, 4, 6, 83.09
```

- To modify a temperature data point on an existing curve, you need two commands: **MPTEMP** with the appropriate location number to specify the new temperature value, and **MPDRES** to associate the new temperature table with the material property. For example, to change the temperature in location 7 of the above enthalpy-temperature curve from 2345 to 2340, the commands would be:

```
MPTEMP, 7, 2340      ! Modifies location 7, retains other locations
MPDRES, ENTH, 4      ! Associates ENTH for material 4 with new temps
```

To modify stored properties, issue the **MPDRES** command. Whenever you define a temperature-dependent property, the temperature-property data pairs are immediately stored in the database. Modifying the temperature data points affects only material properties that are subsequently defined, not what is already stored. The **MPDRES** command forces modification of what is already stored in the database. Two additional fields on **MPDRES** allow you to modify a stored property and store it under a new label or a new material reference number.

The **MPTRES** command enables you to replace the current temperature table with that of a previously defined material property in the database. You can then use the previous temperature data points for another property.

For temperature-dependent secant coefficients of thermal expansion (ALPX, ALPY, ALPZ), if the base temperature for which they are defined (the *definition* temperature) differs from the reference temperature (the temperature at which zero thermal strains exist, defined by **MP,REFT** or **TREF**), then issue the **MPAMOD** command to convert the data to the reference temperature. This conversion is not necessary when you input the thermal strains (THSX, THSY, THSZ) or the instantaneous coefficients of thermal expansion (CTEX, CTEY, CTEZ).

The program accounts for temperature-dependent material properties during solution when element matrices are formulated. The materials are evaluated at once (at or near the centroid of the element) or at each of the integration points. For more information about how the program evaluates temperature-dependent material properties, see the [Material Reference](#).

You can save linear material properties (whether they are temperature-dependent or constant) to a file or restore them from a text file. (See [Using Material Library Files \(p. 20\)](#) for a discussion of material library files.) You also can issue **CDWRITE,MAT** to write both linear and nonlinear material properties to a file.

If using the **CDWRITE** command in any related product (ANSYS Mechanical Pro, ANSYS Mechanical Premium, etc.), edit the Jobname.CDB file that **CDWRITE** creates to remove commands which are not available in the derived product. You must do this before reading the Jobname.CDB file.

2.2. Nonlinear Material Properties

Nonlinear material properties are usually tabular data, such as plasticity data (stress-strain curves for different hardening laws), magnetic field data (B-H curves), creep data, swelling data, hyperelastic material data, etc. The first step in defining a nonlinear material property is to activate a data table using the **TB** command. (See [Material Model Interface \(p. 13\)](#) for the GUI equivalent). For example, **TB,BH,2** activates the B-H table for material reference number 2.

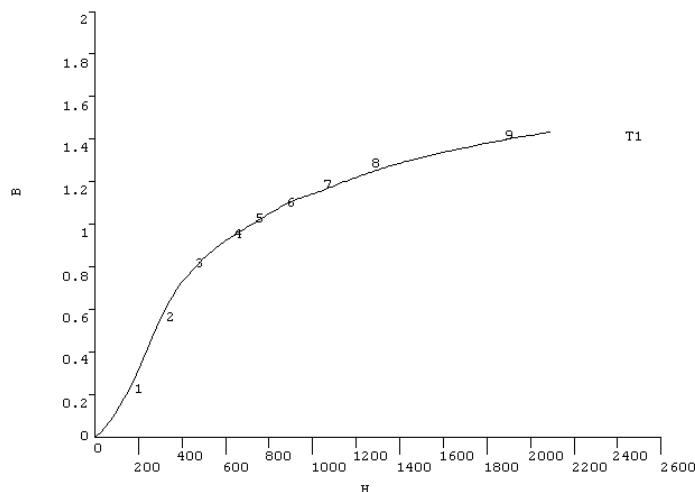
To enter the tabular data, issue the **TBPT** command. For example, the following commands define a B-H curve:

```
TBPT,DEFI,150,.21
TBPT,DEFI,300,.55
TBPT,DEFI,460,.80
TBPT,DEFI,640,.95
TBPT,DEFI,720,1.0
TBPT,DEFI,890,1.1
TBPT,DEFI,1020,1.15
TBPT,DEFI,1280,1.25
TBPT,DEFI,1900,1.4
```

You can verify the data table through displays and listings using the **TBPLOT** or **TBLIST** commands.

Figure 2.2: Sample TBPLOT Display (p. 12) shows a sample **TBPLOT** (of the B-H curve defined above):

Figure 2.2: Sample TBPLOT Display



For a more detailed discussion of nonlinear materials, see [Nonlinear Material Properties in the *Material Reference*](#).

2.3. Anisotropic Elastic Material Properties

Some element types accept anisotropic elastic material properties, which are usually input in the form of a matrix. (These properties are different from anisotropic plasticity, which requires different stress-strain curves in different directions.) Among the element types that allow elastic anisotropy are **PLANE223** (the 2-D coupled-field solid), **SOLID226**, and **SOLID227** (the 3-D coupled-field solids).

The procedure to specify anisotropic elastic material properties resembles that for nonlinear properties. You first activate a data table using the **TB** command (with *Lab* = ANEL) and then define the terms of the elastic coefficient matrix using the **TBDATA** command. Be sure to verify your input with the **TBLIST** command.

For a more information, see [Anisotropic Elasticity in the *Material Reference*](#).

2.4. Material Model Interface

The program GUI includes an intuitive hierarchical tree structure interface for defining many material models. A logical top-down arrangement of material categories guides you in defining the appropriate model for your analysis.

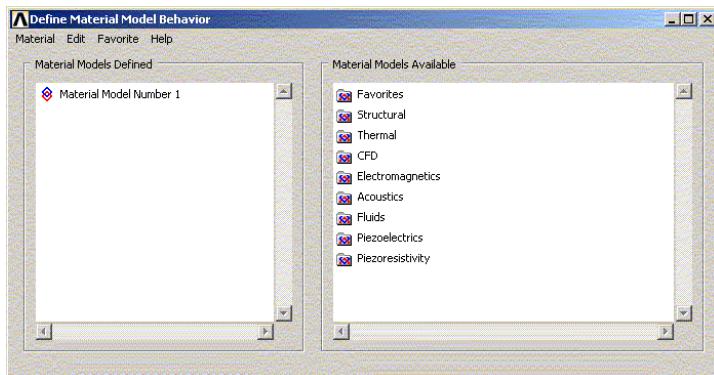
Not all material models are available via the GUI. For more information, see [GUI-Inaccessible Material Properties in the *Material Reference*](#).

For more information on individual material models, see the [Material Reference](#).

2.4.1. Accessing the Material Model Interface

Access the material model interface via **Main Menu> Preprocessor> Material Props> Material Models**. The **Define Material Model Behavior** dialog box appears, which originally displays the top level of the tree structure:

Figure 2.3: Material Model Interface Initial Screen

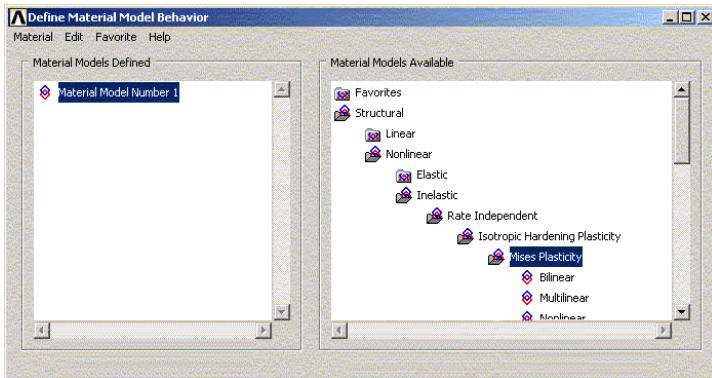


2.4.2. Choosing Material Behavior

The **Material Models Available** window on the right displays a list of material categories (for example, Favorites, Structural, Thermal, Electromagnetics).

If a category is preceded by a folder icon, there are subcategories available under the main category. When you double-click the category, the subcategories appear indented and below the category:

Figure 2.4: Material Model Interface Tree Structure



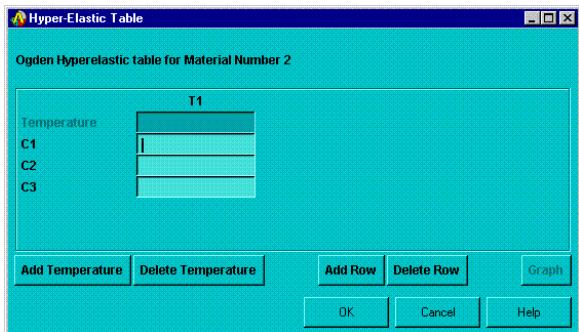
For example, under Structural are categories Linear, Nonlinear, and others. The models are further categorized so that you will eventually see a vertical list of material property sets or material models that are included under that category (for example, under von Mises Plasticity are: Bilinear, Multilinear, and Nonlinear). When you have decided which material property set or model to use, choose it by double-clicking on the item. A dialog box appears that prompts you for the required input data for that particular model or property set. Details of a data input dialog box are presented in [Entering Material Data \(p. 14\)](#).

2.4.2.1. Material Favorites Folder

A Material Favorite is a template of material properties. It is used as a short cut to frequently used properties, instead of navigating through the detailed tree structure each session. You can create a named template based on a currently defined material model through **Favorite>New Favorite**. You can also delete a named template through the **Favorite** menu. For any consecutive sessions, you will then be able to access this named template in the Favorites folder shown in the **Material Models Available** window.

2.4.3. Entering Material Data

Included in a data input dialog box is a table whose rows and columns you can alter depending on the requirements of the specific material property or model you have chosen. A typical data input dialog box is shown below:

Figure 2.5: A Data Input Dialog Box

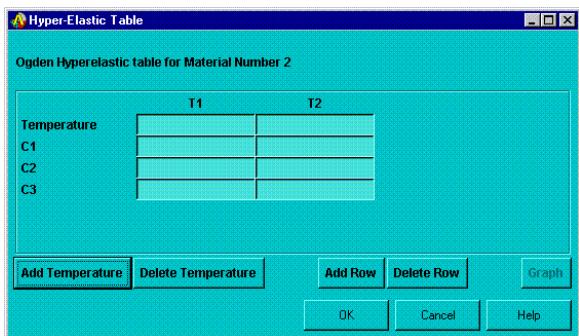
There are two interaction areas within a material data input dialog box: the data input table, and a series of action buttons that appear at the bottom. Depending on the material item you are defining, the labels in the table vary, as do the number of rows and columns that appear initially. The material item also dictates the number of rows and columns that you are allowed to add or delete. In most cases, the columns represent temperatures, and the rows represent data values (for example, density as a linear isotropic property, or constants for a particular nonlinear model).

Temperature Dependent Data

Initially, the table is set up for temperature independent data so the temperature field is grayed out. At this point, should you decide to enter data for various temperatures, you can quickly add columns of text fields for the data representing each temperature. You can add or delete the temperature dependent data at any time. You do not need to *predetermine* if the data should be temperature dependent.

Adding and Deleting Columns

To add a column, position the text cursor in any field in the existing column, then click the **Add Temperature** button. A new column appears to the right of the existing column, and both temperature fields become active, as shown below:

Figure 2.6: Data Input Dialog Box - Added Column

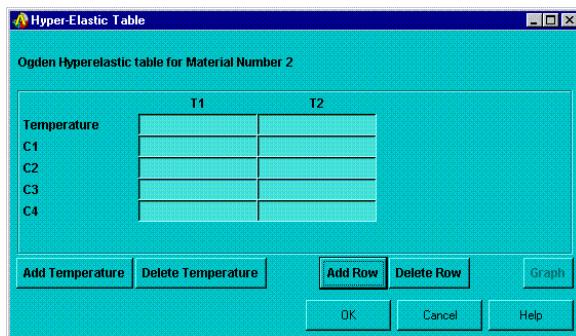
You then enter the two temperatures and the associated data in the rows. You can add more temperature columns, as needed, by following the same procedure. You can insert columns *between* existing columns by clicking the text cursor in a field within a column that is to the left of where you want to insert the new column, then clicking on the **Add Temperature** button.

You can delete a temperature column by positioning the text cursor in any field within the column, and clicking on the **Delete Temperature** button.

Adding and Deleting Rows

You may have the need to add another row of constants or other data for a specific temperature. You add or delete rows in a similar way as is described above for adding or deleting columns. To add a row, click the text cursor in any field in an existing row, then click the **Add Row** (or **Add Point**) button. A new row appears beneath the existing row, as shown below:

Figure 2.7: Data Input Dialog Box - Added Row



You can insert rows *between* existing rows by positioning the text cursor in a field in the top row, then clicking on the **Add Row** (or **Add Point**) button.

You can delete a row by positioning the text cursor in any field within the row, and clicking on the **Delete Row** (or **Delete Point**) button.

Entering/Editing Data in Text Fields

When a data dialog box first appears, one of the text fields is highlighted, meaning that the field is ready to accept data as you type. Use the arrow keys and the tab key to move from field to field.

You can use Ctrl-C and Ctrl-V to copy and paste data from one field to another. You can select multiple *adjacent* text fields to copy and paste entire or partial rows and columns.

Typical **Action Buttons** found in the material interface include:

- **Add Temperature:** Adds a new column of data entry fields to the right of the column where the text cursor is currently positioned. If the button does not appear, the material item has no temperature dependency.
- **Delete Temperature:** Deletes the column of data entry fields where the text cursor is currently positioned. If the button does not appear, the material item has no temperature dependency.
- **Add Row (or Add Point):** Adds a new row of data entry fields beneath the row where the text cursor is currently positioned. If the button does not appear, the material item has no provision for adding more data.
- **Delete Row (or Delete Point):** Deletes the row of data entry fields where the text cursor is currently positioned. If the button does not appear, the material item requires that all data entry fields must be completed.
- **Graph:** Displays a graph of the current data in the Graphics window. If required, you can change the data in the table and click on the **Graph** button again before clicking the **OK** button.
- **OK:** Commits all data that you have entered to the database and removes this dialog box. Material Model Number # appears in the **Material Models Defined** tree structure window,

where # = 1 for the first model, or the number that you specified in the **Define Material ID** dialog box.

- **Cancel:** Cancels all data entered, and removes the dialog box.
- **Help:** Displays help information that is specific to the particular material property or material constant.

Some material data input dialog boxes may include other buttons or interaction components that are necessary for completely defining a material property or model.

Considerations for a Structural Analysis

When performing a structural analysis, several inelastic material models require you to input values for *elastic* material properties (elastic modulus and/or Poisson's ratio) in addition to the inelastic constants that are specific to the model (for example, Yield Stress and Tangent Modulus for the Bilinear Isotropic Hardening model). In these instances, you must enter the elastic material properties *before* you enter the inelastic constants. A data input dialog box appears that prompts you for the elastic material properties.

2.4.4. Logging/Editing Material Data

The **Material Models Defined** window (the left window in the **Define Material Model Behavior** dialog box) displays a log of each material model you have specified. After you have chosen OK in the data input dialog box, this window displays a folder icon, and **Material Model Number #** (the first # is 1 by default), followed by the properties defined for this model. You can define additional models with unique numbers by choosing **Material> New Model**, then typing a new number in the **Define Material ID** dialog box. If you double-click on any material model or property (furthest to the right in the tree), the associated data input dialog box appears where you can edit the data, if you choose.

2.4.5. Example: Defining a Single Material Model

This example and the following two examples show typical uses of the material model interface for use in structural analyses. If your specialty or interest is in performing analyses other than structural analyses, you should still read and perform these examples to become familiar with maneuvering within the material model interface. You are then encouraged to try one of your own problems in your particular discipline, or try one of the many sample problems presented throughout the various analysis guides. Here is a sampling of these problems:

- Performing a Steady-State Thermal Analysis (GUI Method) in the *Thermal Analysis Guide*.
- Example: Current-Carrying Conductor in the *Low-Frequency Electromagnetic Analysis Guide*.
- Example: Thermoelastic Damping in a Silicon Beam in the *Coupled-Field Analysis Guide*.

The first example below is intended to show you how to completely define a single material model. It steps you through a procedure that uses the material model interface to define a model for simulating nonlinear isotropic hardening, using the Voce law, in a large strain structural analysis at two temperatures.

1. From the Main Menu, click the following menu path: **Preprocessor> Material Props> Material Models**. The **Define Material Model Behavior** dialog box appears.

2. In the **Material Models Available** window, double-click on the following options: **Structural, Linear, Elastic, Isotropic**. A dialog box appears.
 3. Enter values for material properties, as required (EX for elastic modulus, and PRXY for Poisson's ratio). Click **OK**. **Material Model Number 1** properties appear listed in the **Material Models Defined** window.
 4. In the **Material Models Available** window, double-click on the following options: **Nonlinear, Inelastic, Rate Independent, Isotropic Hardening Plasticity, von Mises Plasticity, Nonlinear**. A dialog box appears that includes a table where you can add temperature columns or add rows for material data, as needed for your application. Note that the temperature field is grayed out. This is because the program assumes a temperature independent application, by default, so you would not need to enter a temperature value. Because this example is temperature *dependent* (involving two temperatures), you must first add another temperature column, as described in the next step.
 5. Click the **Add Temperature** button. A second column appears.
 6. Enter the first temperature in the **Temperature** row and the **T1** column.
 7. Enter the Voce constants required for the first temperature in the rows under the **T1** column (see *Nonlinear Isotropic Hardening* in the *Material Reference*).
 8. Enter the second temperature in the **Temperature** row, and the **T2** column.
 9. Enter the Voce constants required for the second temperature in the rows under the **T2** column.
- Note that if you needed to input constants for a third temperature, you would position the cursor in the **Temperature** row of the **T2** column, then click the **Add Temperature** button again. This would cause a third column to appear.
- This material model only requires four constants per temperature. If you were using another model that allowed more constants, the **Add Row** button would be active. For those models, the same functionality is included for adding or inserting rows by using the **Add Row** (or **Add Point**) button.
10. Click **OK**. The dialog box closes. The properties defined for that material are listed under **Material Model Number 1**.

2.4.6. Example: Editing Data in a Material Model

This example shows you how to use some of the basic editing features within the material model interface. It assumes that you have completed the previous example (see [Example: Defining a Single Material Model \(p. 17\)](#)), and that the completed material model is listed in the **Material Models Defined** window.

Editing data typically falls into two general categories: changing data within an existing material property, and copying an entire material property set to form another model with slightly different properties.

Consider a case where you need to change the constants that you assigned to the Nonlinear Isotropic model. To perform this task:

1. Double-click **Nonlinear Isotropic**. The associated dialog box appears with the existing data displayed in the fields.
2. Edit the constants in the appropriate fields, and click on **OK**.

Note that if you needed to change any of the other material properties, you would double-click **Linear Isotropic** in the previous step. This would cause the dialog box associated with linear isotropic properties to appear. You could then edit those properties.

Consider another case where you have the requirement for two material models, where the second model is the same as the first except that it needs to include constants for one more temperature. To perform this task:

1. In the **Define Material Model Behavior** dialog box, click the following menu path: **Edit> Copy**, then choose 1 for **from Material number**, and enter 2 for **to Material number**. Click **OK**. The **Material Models Defined** window now includes **Material Model Number 2** in its list. If you double-click on **Material Model Number 2**, the identical material properties appear below **Material Model Number 2** as those listed for **Material Model Number 1**.
2. Double-click **Nonlinear Isotropic** under **Material Model Number 2**. The associated dialog box appears.
3. Move the text cursor to the **Temperature** row in the column furthest to the right, and click the **Add Temperature** button. A **T3** column appears.
4. In the new column, enter the new temperature and the four constants associated with this temperature.
5. Click on **OK**. The dialog box closes. If you double-click **Nonlinear Isotropic** under **Material Model Number 2**, the associated dialog box appears and reflects the new temperature data that you added for **Material Model Number 2**.

2.4.7. Example: Defining a Material Model Combination

This example is intended to show you how to define a material based on a combination of two material models. It steps you through a procedure that uses the material model interface to define a material for simulating cyclic softening at one temperature. This is accomplished by using the Nonlinear Isotropic model combined with the Chaboche model.

If you performed either of the previous examples in this section, start a new session before beginning the following example.

1. From the Main Menu, click the following menu path: **Preprocessor> Material Props> Material Models**. The **Define Material Model Behavior** dialog box appears.
2. In the **Material Models Available** window, double-click on the following options: **Structural, Linear, Elastic, Isotropic**. A dialog box appears.
3. Enter values for material properties, as required (EX for elastic modulus, and PRXY for Poisson's ratio). Click **OK**. **Material Model Number 1** and **Linear Isotropic** appear in the **Material Models Defined** window.

4. In the **Material Models Available** window, double-click on the following options: **Nonlinear, Inelastic, Rate Independent, Combined Kinematic and Isotropic Hardening Plasticity, von Mises Plasticity**.
5. Double-click **Chaboche and Nonlinear Isotropic**. A dialog box appears for defining the constants for the Chaboche model.
6. Enter the first three constants associated with the Chaboche model (click on the Help button for information on these constants).
7. The Chaboche model enables you to specify more constants. If you choose to specify more constants, click the **Add Row** button, and enter the next constant.
8. Repeat the previous step for all the remaining Chaboche constants that you want to define.
9. Click **OK**. The dialog box closes and another dialog box appears for defining the constants for the Nonlinear Isotropic model.
10. Enter the constants associated with the Nonlinear Isotropic model (click on the Help button for information on these constants).
11. Click **OK**. The dialog box closes. Under **Material Model Number 1**, the following are listed: **Linear Isotropic, Chaboche, and Nonlinear Isotropic**. You can then edit any of the data (see [Example: Editing Data in a Material Model \(p. 18\)](#)).

2.4.8. Material Model Interface - Miscellaneous Items

Other characteristics of the material model interface are the following:

- Any batch files you use to enter material data will be converted to material models and will appear listed in the **Material Models Defined** window of the **Define Material Model Behavior** dialog box.
- The material model interface does not import data from the material library discussed in [Using Material Library Files \(p. 20\)](#).

2.5. Using Material Library Files

Although you can define material properties separately for each finite element analysis, you can also store a material property set in an archival material library file, then retrieve the set and reuse it in multiple analyses. (Each material property set has its own library file.) The material library files also enable several users to share commonly used material property data.

The material library feature offers you other advantages:

- Because the archived contents of material library files are reusable, you can use them to define other, similar material property sets quickly and with fewer errors. For example, suppose that you have defined material properties for one grade of steel and want to create a material property set for another grade of steel that is slightly different. You can write the existing steel material property set to a material library file, read it back in under a different material number, and then make the minor changes needed to define properties for the second type of steel.

- The **/MPLIB** command defines a material library read and write path. Doing so enables you to protect your material data resources in a read-only archive, while giving users the ability to write their material data locally without switching paths.
- You can give your material library files meaningful names that reflect the characteristics of the data they contain. For example, the name of a material library file describing properties of a steel casting might be STEELCST.SI_MPL. (See [Creating \(Writing\) a Material Library File \(p. 21\)](#) for an explanation of file naming conventions.)
- You can design your own directory hierarchy for material library files. This enables you to classify and catalog the files by material type (plastic, aluminum, etc.), by units, or by any category you choose.

The next few sections describe how to create and read material library files. For additional information, see the descriptions of the **/MPLIB**, **MPREAD**, and **MPWRITE** commands in the [Command Reference](#).

2.5.1. Format of Material Library Files

Material library files are command files. The file format supports both linear and nonlinear properties. You can reuse material library files because the commands in them are written so that, once you read a material property set into the database, you can associate that set with any material number you wish.

2.5.2. Specifying a Default Read/Write Path for Material Library Files

The material library is located at <drive:>\Program Files\Ansys Inc\v211\ANSYS\matlib.

Before creating any material library files, issue the **/MPLIB** command to define a default read path and write path for those files:

```
/MPLIB,R-W_opt,PATH
```

In place of *R-W_opt*, specify READ (to set the read path), WRITE (to set the write path), or STAT to see what read and write paths currently are in use. In place of *PATH*, specify the path to be used for material library files.

2.5.3. Creating (Writing) a Material Library File

To create an archival material library file, perform these steps:

1. To tell the program what system of units you are using, issue the **/UNITS** command. For example, to specify the international system of units, you would issue the command **/UNITS,SI**. You cannot access the **/UNITS** command directly from the GUI.
2. Define a material property using the **MP** command (**Main Menu> Preprocessor> Material Props> Isotropic**). To do so, you must specify a material number and at least one material property value (for example, magnetic permeability or MURX).
3. From the PREP7 preprocessor, issue the **MPWRITE** command (**Main Menu> Preprocessor> Material Props> Material Library> Export Library**) and specify the file name (*Fname* argument) for the material library file:

```
MPWRITE ,Fname , ,LIB ,MAT
```

Issuing **MPWRITE** writes the material data specified by material number *MAT* into the named file in the current working directory. (If you previously specified a material library write path by issuing the **/MPLIB** command (**Main Menu> Preprocessor> Material Props> Material Library> Library Path**), the program writes the file to that location instead.)

Naming conventions for a material library file are as follows:

- The name of the file is the name you specify on the **MPWRITE** command. If you do not specify a file name, the default name is *JOBNAME*.
- The extension of a material library file name follows the pattern *.xxx_MPL*, where *xxx* identifies the system of units for this material property sets. For example, if the system of units is the CGS system, the file extension is *.CGS_MPL*. The default extension, used if you do not specify a units system before creating the material library file, is *.USER_MPL*. (This indicates a user-defined system of units.)

2.5.4. Reading a Material Library File

To read a material library file into the database:

1. Issue the **/UNITS** command to specify the desired system of units.

Note:

The default system of units is SI. The GUI lists only material library files with the currently active units.

2. Specify a new material reference number or an existing number that you want to overwrite (**MAT**).

Caution:

Overwriting an existing material in the database deletes all data associated with it.

3. Read the material library file into the database (**MPREAD**).

The *LIB* argument on **MPREAD** supports a file search hierarchy. The program searches for the named material library file first in the current working directory, then in your home directory, then in the read path directory specified by the **/MPLIB** command, and finally in the program directory */ansys211/matlib*. If you omit the *LIB* argument, the programs searches only in the current working directory.

Chapter 3: Loading

The primary objective of a finite element analysis is to examine how a structure or component responds to the given loading conditions. Specifying the proper loading conditions is, therefore, a key component of the analysis.

You can apply loads on the model in several ways. With the help of load step options, you can control how the loads are actually used during solution.

The following topics related to loading are available:

- 3.1. Understanding Loads
- 3.2. Load Steps, Substeps, and Equilibrium Iterations
- 3.3. The Role of Time in Tracking
- 3.4. Ramped and Stepped Loads
- 3.5. Applying Loads
- 3.6. Specifying Load Step Options
- 3.7. Creating Multiple Load Step Files
- 3.8. Defining Pretension in a Joint Fastener
- 3.9. Defining Preload in a Joint Fastener Undergoing Large Rotation

3.1. Understanding Loads

The term *loads* includes boundary conditions *and* externally or internally applied forcing functions. Following are examples of loading types in various disciplines:

- *Structural*: Displacements, velocities, accelerations, forces, pressures, temperatures (for thermal strain), gravity
- *Thermal*: Temperatures, heat flow rates, convections, internal heat generation, infinite surface
- *Magnetic*: Magnetic potentials, magnetic flux, magnetic current segments, source current density, infinite surface
- *Electric*: Electric potentials (voltage), electric current, electric charges, charge densities, infinite surface
- *Acoustic*: Pressures, displacements
- *Diffusion*: Concentration, diffusion flow rate

Loads are divided into six categories: Degree-of-freedom (DOF) constraints, forces (concentrated loads), surface loads, body loads, inertia loads, and coupled-field loads.

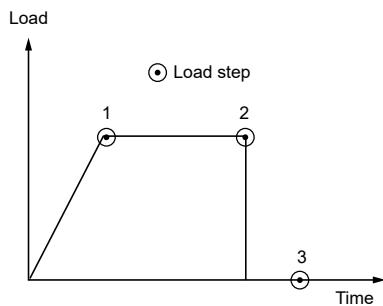
Load Category	Description
DOF constraint	<p>Fixes a degree of freedom (DOF) to a known value.</p> <p><i>Examples:</i> Specified displacements and symmetry boundary conditions in a structural analysis, prescribed temperatures in a thermal analysis, and flux-parallel boundary conditions.</p> <p>In a structural analysis, a DOF constraint can be replaced by its differentiation form, which is a velocity constraint. In a structural transient analysis, an acceleration can also be applied, which is the second order differentiation form of the corresponding DOF constraint.</p>
Force	<p>A concentrated load applied at a node in the model.</p> <p><i>Examples:</i> Forces and moments in a structural analysis, heat flow rates in a thermal analysis, and current segments in a magnetic field analysis.</p>
Surface load	<p>A distributed load applied over a surface.</p> <p><i>Examples:</i> Pressures in a structural analysis, and convections and heat fluxes in a thermal analysis.</p>
Body load	<p>A volumetric or field load.</p> <p><i>Examples:</i> Temperatures and fluences in a structural analysis, heat generation rates in a thermal analysis, and current densities in a magnetic field analysis.</p>
Inertia load	<p>A load attributable to the inertia (mass matrix) of a body, used primarily in structural analyses.</p> <p><i>Examples:</i> Gravitational acceleration, angular velocity, and angular acceleration.</p>
Coupled-field load	<p>A special case of one of the loads listed above, where results from one analysis are used as loads in another.</p> <p><i>Example:</i> Applying magnetic forces calculated in a magnetic field analysis as force loads in a structural analysis.</p>

3.2. Load Steps, Substeps, and Equilibrium Iterations

A *load step* is a configuration of loads for which a solution is obtained. In a linear static or steady-state analysis, you can use different load steps to apply different sets of loads - wind load in the first load step, gravity load in the second load step, both loads and a different support condition in the third load step, and so on. In a transient analysis, multiple load steps apply different segments of the load history curve.

The program uses the set of elements which you select (**ESEL**) for the first load step for all subsequent load steps, no matter which element sets you specify for the later steps.

This figure shows a load history curve requiring three load steps:

Figure 3.1: Transient Load History Curve

The first load step is for the ramped load, the second load step is for the constant portion of the load, and the third load step is for load removal.

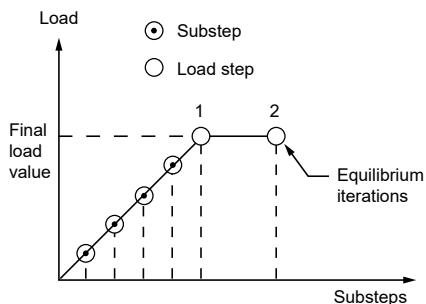
Substeps are points within a load step at which solutions are calculated. You use them for different reasons:

- In a nonlinear static or steady-state analysis, use substeps to apply the loads gradually so that an accurate solution can be obtained.
- In a linear or nonlinear transient analysis, use substeps to satisfy transient time integration rules (which usually dictate a minimum integration time step for an accurate solution).
- In a harmonic analysis, use substeps to obtain solutions at several frequencies within the harmonic frequency range.

Equilibrium iterations are additional solutions calculated at a given substep for convergence purposes. They are iterative corrections used only in nonlinear analyses (static or transient), where convergence plays an important role.

Consider, for example, a 2-D, nonlinear static magnetic analysis. To obtain an accurate solution, two load steps are commonly used:

- The first load step applies the loads gradually over five to 10 substeps, each with just one equilibrium iteration.
- The second load step obtains a final, converged solution with just one substep that uses 15 to 25 equilibrium iterations.

Figure 3.2: Load Steps, Substeps, and Equilibrium Iterations

3.3. The Role of Time in Tracking

The program uses time as a tracking parameter in *all* static and transient analyses, regardless of whether they are actually time-dependent. You can therefore use one consistent counter or tracker in all cases, eliminating the need for analysis-dependent terminology. Time always increases monotonically, and most occurrences in nature happen over a period of time, however brief the period may be.

In a transient analysis or rate-dependent static analysis (creep or viscoplasticity), *time* represents actual, chronological time in seconds, minutes, or hours. You assign the time (**TIME**) at the end of each load step while specifying the load history curve.

In a rate-independent analysis, however, *time* becomes a counter that identifies load steps and substeps. By default, the program automatically assigns time = 1.0 at the end of load step 1, time = 2.0 at the end of load step 2, and so on. Any substeps within a load step are assigned the appropriate, linearly interpolated time value. By assigning your own time values in such analyses, you can establish your own tracking parameter. For example, if a load of 100 units is to be applied incrementally over one load step, you can specify time at the end of that load step to be 100, so that the load and time values are synchronous.

If you obtain a deflection-vs.-time graph in the postprocessor, it means the same as deflection vs. load. An example of the usefulness of this technique is in a large-deflection buckling analysis, where the objective may be to track the deflection of the structure as it is loaded incrementally.

Time adopts yet another meaning when using the arc-length method in the solution. In this case, *time* equals the value of time at the beginning of a load step, plus the value of the arc-length load factor (the multiplier on the currently applied loads). ALLF does not have to be monotonically increasing (that is, it can increase, decrease, or even become negative), and it is reset to zero at the beginning of each load step. As a result, *time* is not considered a "counter" in arc-length solutions.

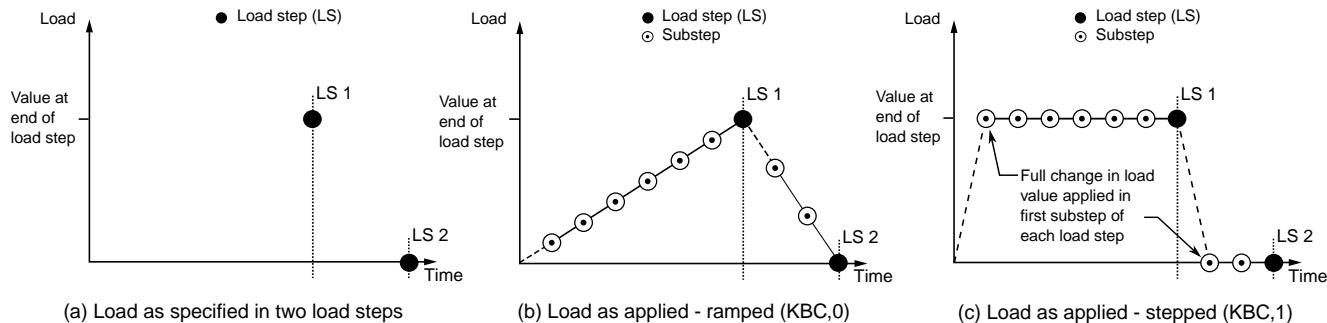
The arc-length method is an advanced solution technique. For more information about using it, see [Nonlinear Structural Analysis in the Structural Analysis Guide](#).

A load step is a set of loads applied over a given *time span*. Substeps are *time points* within a load step at which intermediate solutions are calculated. The difference in time between two successive substeps can be called a *time step* or *time increment*. Equilibrium iterations are iterative solutions calculated at a given time point purely for convergence purposes.

3.4. Ramped and Stepped Loads

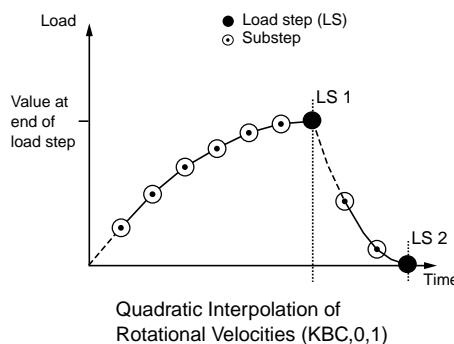
When you specify more than one substep in a load step, you may choose to *ramp* apply or *step* apply the loads:

- If you *ramp* apply the loads, their values are incrementally applied at each substep in a linearly interpolated fashion, reaching the full values at the end of the load step.
- If you *step* apply the loads, their values are fully applied at the first substep and remain constant for the remainder of the load step.

Figure 3.3: Ramped vs. Stepped Loads

Use the **KEY** argument on the **KBC** command to specify whether loads are ramped (**KBC,0**) or stepped (**KBC,1**) for the load step. The default setting of **KEY** depends on the discipline and type of analysis.

If you are applying rotational velocity loads (**OMEGA**, **CMOMEGA**, and **CMROTATE**) in a ramped (**KBC,0**) fashion, there is a third option: you can choose quadratic interpolation instead of linear interpolation by setting the **OMGSQRDKEY** argument on the **KBC** command to 1.

Figure 3.4: Quadratic Interpolation of Rotational Velocities

Various *load step options* (p. 59) control load application, such as **time** (p. 59), **number of substeps** (p. 59), the **time step** (p. 59), and load ramping or stepping. Other types of load step options include **convergence tolerances** (p. 63) (used in nonlinear analyses), **damping specifications** (p. 62) in a structural analysis, and **output controls** (p. 63).

3.5. Applying Loads

You can apply most loads either on the solid model (on keypoints, lines, and areas) or on the finite element model (on nodes and elements). For example, you can specify forces at a keypoint or a node. Similarly, you can specify convective (and other surface loads) on lines and areas or on nodes and element faces. No matter how you specify the loads, the solver expects all loads to be in terms of the finite element model. Therefore, if you specify loads on the solid model, the program automatically transfers them to the nodes and elements at the beginning of solution.

The following topics related to applying loads are available:

- 3.5.1. Solid-Model Loads
- 3.5.2. Finite-Element Loads
- 3.5.3. Degree-of-Freedom Constraints

-
- 3.5.4. Applying Symmetry or Antisymmetry Boundary Conditions
 - 3.5.5. Transferring Constraints
 - 3.5.6. Forces (Concentrated Loads)
 - 3.5.7. Surface Loads
 - 3.5.8. Applying Body Loads
 - 3.5.9. Applying Inertia Loads
 - 3.5.10. Applying Ocean Loads
 - 3.5.11. Applying Coupled-Field Loads
 - 3.5.12. Axisymmetric Loads and Reactions
 - 3.5.13. Loads to Which the Degree of Freedom Offers No Resistance
 - 3.5.14. Initial State Loading
 - 3.5.15. Applying Loads Using Tabular Input
 - 3.5.16. Applying Loads to Components and Assemblies

3.5.1. Solid-Model Loads

Solid-model loads are independent of the finite element mesh; that is, you can change the element mesh without affecting the applied loads. This capability enables you to make mesh modifications and conduct mesh sensitivity studies without having to reapply loads each time.

The solid model usually involves fewer entities than the finite element model. Therefore, selecting solid model entities and applying loads on them is much easier, especially with graphical picking.

3.5.1.1. Disadvantages of Solid-Model Loads

- Elements generated by meshing commands are in the currently active element coordinate system. Nodes generated by meshing commands use the global Cartesian coordinate system. Therefore, the solid model and the finite element model may have different coordinate systems and loading directions.
- Applying keypoint constraints requires care, especially when the constraint expansion option is used. (The expansion option enables you to expand a constraint specification to all nodes between two keypoints that are connected by a line.)
- You cannot display all solid-model loads.

3.5.1.2. Other Considerations for Solid-Model Loads

Solid-model loads are transferred automatically to the finite element model at the beginning of the solution. If you mix solid model loads with finite-element model loads, couplings, or constraint equations, be aware of the following possible conflicts:

- Transferred solid loads replace nodal or element loads already present, regardless of the order in which the loads were input. For example, **DL,,UX** on a line overwrites any **D,,UX** loads on the nodes of that line at transfer time. (**DL,,UX** also overwrites **D,,VELX** velocity loads and **D,,ACCX** acceleration loads.)

- Deleting solid model loads also deletes any corresponding finite element loads. For example, **SFADELE,,PRES** on an area immediately deletes any **SFE,,PRES** loads on the elements in that area.
- Line or area symmetry or antisymmetry conditions (**DL,,SYMM**, **DL,,ASYM**, **DA,,SYMM**, or **DA,,ASYM**) often introduce nodal rotations that could effect nodal constraints, nodal forces, couplings, or constraint equations on nodes belonging to constrained lines or areas.

3.5.2. Finite-Element Loads

With finite element loads, constraint expansion is not a concern. You can select all desired nodes and specify the appropriate constraints.

3.5.2.1. Disadvantages of Finite-Element Loads

Any modification of the finite element mesh invalidates the loads, requiring you to delete the previous loads and re-apply them on the new mesh.

Applying loads by graphical picking is inconvenient, unless only a few nodes or elements are involved.

3.5.3. Degree-of-Freedom Constraints

This table shows the degrees of freedom (DOFs) that can be constrained in each discipline and the corresponding labels:

Table 3.1: DOF Constraints Available in Each Discipline

Discipline	Degree of Freedom	Label
Structural	Translations Rotations Hydrostatic Pressure	UX, UY, UZ ROTX, ROTY, ROTZ HDSP
Thermal	Temperature	TEMP, TBOT, TE2, ... TTOP
Magnetic	Vector Potential Scalar Potential	AZ MAG
Electric	Voltage	VOLT
Acoustic	Pressure Displacement Energy Density Velocity Temperature	PRES UX, UY, UZ ENKE VX, VY, VZ TEMP
Diffusion	Concentration	CONC

Any directions implied by the labels (such as UX, ROTZ, AY, etc.) are in the nodal coordinate system.

This table shows the commands for applying, listing, and deleting DOF constraints:

Table 3.2: Commands for DOF Constraints

Location	Basic Commands	Additional Commands
Nodes	D, DLIST, DDELETE	DSYM, DSSCALE, DCUM
Keypoints	DK, DKLIST, DKDELETE	-
Lines	DL, DLLIST, DLDELETE	-
Areas	DA, DALIST, DADELETE	-
Transfer	SBCTRAN	DTRAN

3.5.3.1. Velocities and Accelerations

For structural static and transient analyses, velocities and accelerations can be applied as finite element loads on nodes (**D**). Velocities can be applied in static or transient analyses; accelerations can only be applied in transient analyses. The labels for these loads are:

VELX, VELY, VELZ - Translational velocities

OMGX, OMGY, OMGZ - Rotational velocities

ACCX, ACCY, ACCZ - Translational accelerations

DMGX, DMGY, DMGZ - Rotational accelerations

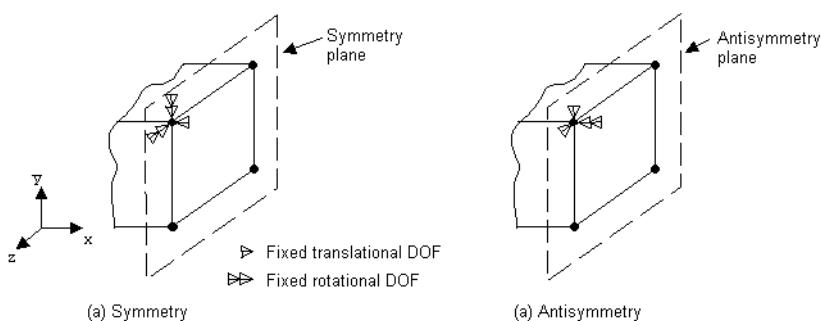
Although these are not strictly degree-of-freedom constraints, they are boundary conditions that act upon the translation and rotation degrees of freedom. See the **D** command for more information.

3.5.4. Applying Symmetry or Antisymmetry Boundary Conditions

To apply symmetry or antisymmetry boundary conditions on a plane of nodes, issue the **DSYM** command. The command generates the appropriate DOF constraints.

In a structural analysis, for example, a symmetry boundary condition means that out-of-plane translations and in-plane rotations are set to zero, and an antisymmetry condition means that in-plane translations and out-of-plane rotations are set to zero.

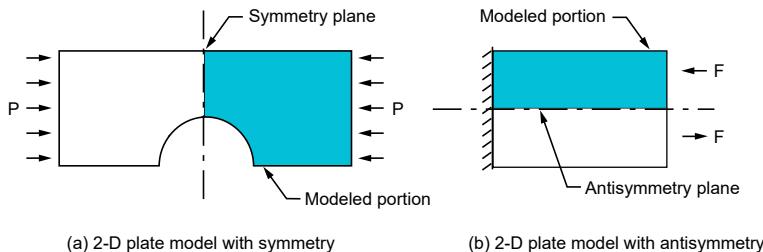
Figure 3.5: Symmetry and Antisymmetry Boundary Conditions



All nodes on the symmetry plane are rotated into the coordinate system via the *KCN* argument on the **DSYM** command.

The following figure illustrates the use of symmetry and antisymmetry boundary conditions:

Figure 3.6: Examples of Boundary Conditions



The **DL** and **DA** commands work in a similar fashion when you apply symmetry or antisymmetry conditions on lines and areas.

If the node rotation angles that exist in the database are different from those used in the solution being postprocessed, POST1 may display incorrect results. This condition typically occurs if you introduce node rotations in a second or later load step by applying symmetry or antisymmetry boundary conditions.

3.5.5. Transferring Constraints

To transfer constraints that have been applied to the solid model to the corresponding finite element model, issue the **DTRAN** command.

To transfer all solid model boundary conditions, issue the **SBCTRAN** command.

The following related topics are available:

- 3.5.5.1. Resetting Constraints
- 3.5.5.2. Scaling Constraint Values
- 3.5.5.3. Resolution of Conflicting Constraint Specifications

3.5.5.1. Resetting Constraints

By default, if you repeat a DOF constraint on the same degree of freedom, the new specification replaces the previous one. You can change this default to *add* (for accumulation) or *ignore* with the **DCUM** command. For example:

```

NSEL,...           ! Selects a set of nodes
D,ALL,UX,40       ! Sets UX = 40 at all selected nodes
D,ALL,UX,50       ! Changes UX value to 50 (replacement)
DCUM,ADD          ! Subsequent D's to be added
D,ALL,UX,25       ! UX = 50+25 = 75 at all selected nodes
DCUM,IGNORE        ! Subsequent D's to be ignored
D,ALL,UX,1325      ! These UX values are ignored!
DCUM              ! Resets DCUM to default (replacement)

```

Any DOF constraints set via **DCUM** remain until another **DCUM** command is issued. To reset the default setting (replacement), issue the command with no arguments.

3.5.5.2. Scaling Constraint Values

You can scale existing DOF constraint values via the **DSCALE** command.

Both the **DSCALE** and **DCUM** commands work on all selected nodes and also *on all selected DOF labels*. By default, DOF labels that are active are those associated with the element types in the model.

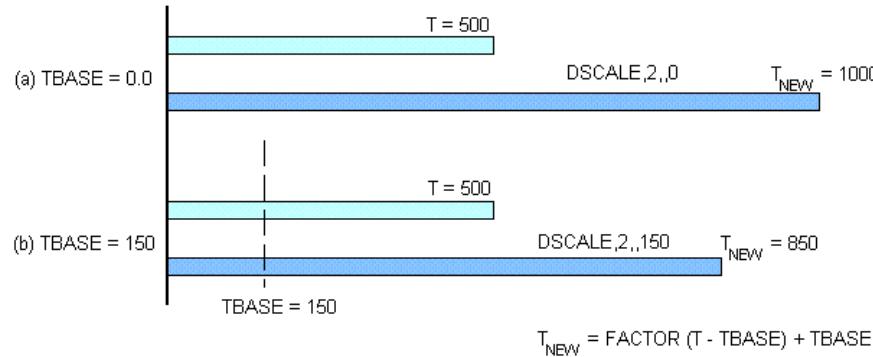
For example, if you want to scale only UX values and not any other DOF label, you can use the following commands:

```
DOFSEL,S,UX      ! Selects UX label
DSCALE,0.5       ! Scales UX at all selected nodes by 0.5
DOFSEL,ALL        ! Reactivates all DOF labels
```

DSCALE and **DCUM** also affect velocity and acceleration loads applied in a structural analysis.

When scaling temperature constraints (TEMP) in a thermal analysis, you can use the *TBASE* field on the **DSCALE** command to scale the temperature offset from a base temperature (that is, to scale $|TEMP-TBASE|$) rather than the actual temperature values.

Figure 3.7: Scaling Temperature Constraints with DSCALE



3.5.5.3. Resolution of Conflicting Constraint Specifications

Be aware of the possibility of conflicting **DK**, **DL**, and **DA** constraint specifications and how the program handles them. The following conflicts can arise:

- A **DL** specification can conflict with a **DL** specification on an adjacent line (shared keypoint).
- A **DL** specification can conflict with a **DK** specification at either keypoint.
- A **DA** specification can conflict with a **DA** specification on an adjacent area (shared lines/keypoints).
- A **DA** specification can conflict with a **DL** specification on any of its lines.
- A **DA** specification can conflict with a **DK** specification on any of its keypoints.

The program transfers constraints that have been applied to the solid model to the corresponding finite element model in the following sequence:

1. In ascending area number order, DOF **DA** constraints transfer to nodes on areas (and bounding lines and keypoints).

2. In ascending area number order, SYMM and ASYM **DA** constraints transfer to nodes on areas (and bounding lines and keypoints).
3. In ascending line number order, DOF **DL** constraints transfer to nodes on lines (and bounding keypoints).
4. In ascending line number order, SYMM and ASYM **DL** constraints transfer to nodes on lines (and bounding keypoints).
5. **DK** constraints transfer to nodes on keypoints (and on attached lines, areas, and volumes if expansion conditions are met).

Accordingly, for conflicting constraints, **DK** commands overwrite **DL** commands and **DL** commands overwrite **DA** commands. For conflicting constraints, constraints specified for a higher line number or area number overwrite the constraints specified for a lower line number or area number, respectively. The constraint specification issue order does not matter.

Changing the value of **DK**, **DL**, or **DA** constraints between solutions may produce many of these warnings at the second or later solid BC transfer. These can be prevented by deleting the nodal **D** constraints between solutions using **DADELE**, **DLDELETE**, and/or **DDELETE**.

3.5.6. Forces (Concentrated Loads)

The following table shows the forces available in each discipline and the corresponding labels:

Table 3.3: Forces Available in Each Discipline

Discipline	Force	Label
Structural	Forces	FX, FY, FZ
	Moments	MX, MY, MZ
	Fluid Mass Flow Rate	DVOL
Thermal	Heat Flow Rate	HEAT, HBOT, HE2, ... HTOP
Magnetic	Current Segment Magnetic Flux Electrical Charge	CSGZ FLUX CHRG
Electric	Current Charge	AMPS CHRG
Fluid	Fluid Flow Rate	FLOW
Diffusion	Diffusion Flow Rate	RATE
Acoustic	Volumetric body force	FX, FY, FZ

Any directions implied by the labels (such as FX, MZ, CSGY, etc.) are in the nodal coordinate system.

This table show the commands to apply, list, and delete forces:

Table 3.4: Commands for Applying Force Loads

Location	Basic Commands	Additional Commands
Nodes	F, FLIST, FDELE	FSCALE, FCUM
Keypoints	FK, FKLIST, FKDELETE	-
Transfer	SBCTRAN	FTRAN

3.5.6.1. Repeating a Force

By default, if you repeat a force at the same degree of freedom, the new specification *replaces* the previous one. You can change this default to *add* (for accumulation) or *ignore* via the **FCUM** command.

Example 3.1: FCUM Command Usage

For example:

```

F,447,FY,3000      ! Applies FY = 3000 at node 447
F,447,FY,2500      ! Changes FY value to 2500 (replacement)
FCUM,ADD           ! Subsequent F's to be added
F,447,FY,-1000     ! FY = 2500-1000 = 1500 at node 447
FCUM,IGNORE         ! Subsequent F's to be ignored
F,25,FZ,350         ! This force is ignored!
FCUM               ! Resets FCUM to default (replacement)

```

Any force set via **FCUM** stays set until another **FCUM** command is issued. To reset the default setting (replacement), issue the command with no arguments.

3.5.6.2. Scaling Force Values

Issue the **FSCALE** command to scale existing force values.

FSCALE and **FCUM** work on all selected nodes and also on *all selected* force labels. By default, force labels that are active are those associated with the element types in the model. You can select a subset of force labels via the **DOFSEL** command.

Example 3.2: DOFSEL Command Usage

```

DOFSEL,S,FX        ! Selects FX label
FSCALE,0.5          ! Scales FX at all selected nodes by 0.5
DOFSEL,ALL          ! Reactivates all DOF labels

```

3.5.6.3. Transferring Forces

To transfer forces that have been applied to the solid model to the corresponding finite element model, issue the **FTRAN** command.

To transfer all solid model boundary conditions, issue the **SBCTRAN** command.

3.5.7. Surface Loads

The following table shows surface loads available in each discipline and their corresponding labels:

Table 3.5: Surface Loads Available in Each Discipline

Discipline	Surface Load	Label
Structural	Pressure	PRES [1]
Thermal	Convection	CONV
	Heat Flux	HFLUX
	Infinite Surface	INF
Magnetic	Maxwell Surface	MXWF
	Infinite Surface	INF
Electric	Maxwell Surface	MXWF
	Surface Charge Density	CHRGS
	Infinite Surface	INF
Fluid	Fluid-Structure Interface	FSI
	Impedance	IMPD
All	Superelement Load Vector	SELV
Diffusion	Diffusion Load	DFLUX
Acoustic	Fluid-structure interaction (FSI) flag	FSI
	Impedance or admittance coefficient	IMPD
	Surface normal velocity or acceleration	SHLD
	Maxwell surface flag or equivalent source surface	MXWF
	Free surface flag	FREE
	Exterior Robin radiation boundary flag	INF
	Port number	PORT
	Absorption coefficient or transmission loss	ATTN
	Viscous-thermal boundary layer surface flag	BLI
	Rigid wall flag (Neumann boundary)	RIGW
	One way structure-to-acoustic coupling interface number	FSIN
	Pressure	PRES
	Heat flux	CONV
	Viscous impedance	VIMP
	Thermal impedance	TIMP

1. Do not confuse this body load with the PRES degree of freedom

The following table shows the commands to apply, list, and delete surface loads:

Table 3.6: Commands for Applying Surface Loads

Location	Basic Commands	Additional Commands
Nodes	SF, SFLIST, SFDELE	SFSCALE, SFCUM, SFFUN, SFGRAD
Elements	SFE, SFE LIST, SFEDELE	SFBEAM, SFFUN, SFGRAD
Lines	SFL, SFL LIST, SFLDELE	SFGRAD
Areas	SFA, SFALIST, SFADELE	SFGRAD
Transfer	SFTRAN	--

The program stores surface loads specified on nodes internally in terms of elements and element faces; therefore, if you use both nodal and element surface load commands for the same surface, only the last specification is used.

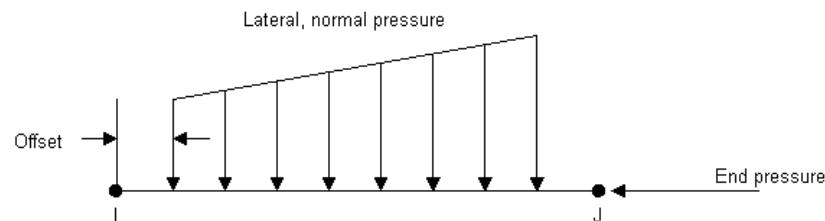
The program applies pressures on axisymmetric shell elements or beam elements on their inner or outer surfaces, as appropriate. In-plane pressure load vectors for layered shells (such as **SHELL281**) are applied on the nodal plane. The element's KEYOPT(11) determines the location of the nodal plane within the shell. When using flat elements to represent doubly curved surfaces, values which should be a function of the active radius of the meridian be inaccurate.

3.5.7.1. Applying Pressure Loads on Beams

To apply pressure loads on the lateral faces and the two ends of beam elements, issue the **SFBEAM** command.

You can apply lateral pressures, which have units of force per unit length, both in the normal and tangential directions. The pressures may vary linearly along the element length, and can be specified on a portion of the element, as shown in this figure:

Figure 3.8: Example of Beam Surface Loads



You can also reduce the pressure down to a force (point load) at any location on a beam element by setting the *JOFFSET* value to -1. End pressures have units of force.

3.5.7.2. Specifying Node Number Vs. Surface Load

The **SFFUN** command specifies a function of node number vs. surface load to be used when applying surface loads on nodes or elements.

Issue the command when you want to apply nodal surface loads calculated elsewhere (by another software package, for instance). First define the function in the form of an array parameter containing the load values. The location of the value in the array parameter implies the node number.

Example 3.3: SFFUN Command Usage

This array parameter specifies four surface load values at nodes 1, 2, 3, and 4, respectively:

$$ABC = \begin{bmatrix} 400.0 \\ 587.2 \\ 965.6 \\ 740.0 \end{bmatrix}$$

Assuming that these are heat flux values, you would apply them as follows:

```
*DIM,ABC,ARRAY,4           ! Declares dimensions of array parameter ABC
ABC(1)=400,587.2,965.6,740 ! Defines values for ABC
SFFUN,HFLUX,ABC(1)         ! ABC to be used as heat flux function
SF,ALL,HFLUX,100            ! Heat flux of 100 on all selected nodes,
                            ! 100 + ABC(i) at node i.
```

The **SF** command in the example above specifies a heat flux of 100 on all selected nodes. If nodes 1 through 4 are part of the selected set, those nodes are assigned heat fluxes of $100 + ABC(i)$: $100 + 400 = 500$ at node 1, $100 + 587.2 = 687.2$ at node 2, and so on.

What you specify with the **SFFUN** command stays active for all subsequent **SF** and **SFE** commands. To remove the specification, issue the command with no arguments.

3.5.7.3. Specifying a Gradient Slope

Issue the **SFGRAD** command to specify that a gradient (slope) is to be used for subsequently applied surface loads. You can also use the command to apply a linearly varying surface load, such as hydrostatic pressure on a structure immersed in water.

To create the gradient specification, specify the following:

- The type of load to be controlled (the *Lab* argument)
- The coordinate system and coordinate direction in which the slope is defined (*SLKCN* and *Sldir*, respectively)
- The coordinate location where the value of the load (as specified on a subsequent surface load command) is in effect (*SLZER*)
- The slope (*SLOPE*).

Example 3.4: SFGRAD Command Usage

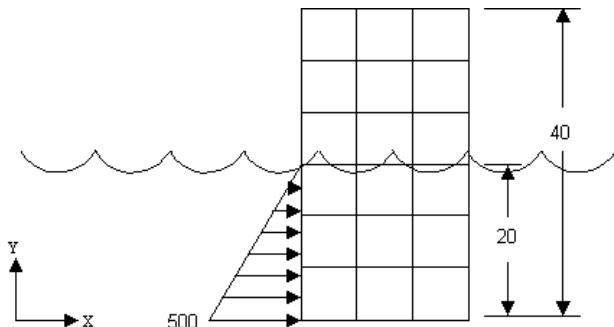
The hydrostatic pressure (*Lab* = PRES) shown in [Figure 3.9: Surface Load Gradient \(p. 38\)](#) is to be applied.

Its slope is specified in the global Cartesian system (*SLKCN* = 0) in the Y direction (*Sldir* = Y).

The pressure (specified on a subsequent **SF** command) is 500 at $Y = 0$ (*SLZER* = 0), and decreases by 25 units per length in the positive Y direction (*SLOPE* = -25).

```
SFGRAD,PRES,0,Y,0,-25      ! Y slope of -25 in global Cartesian
NSEL,...                   ! Select nodes for pressure application
SF,ALL,PRES,500            ! Pressure at all selected nodes:
                           ! 500 at Y=0, 250 at Y=10, 0 at Y=20
```

Figure 3.9: Surface Load Gradient



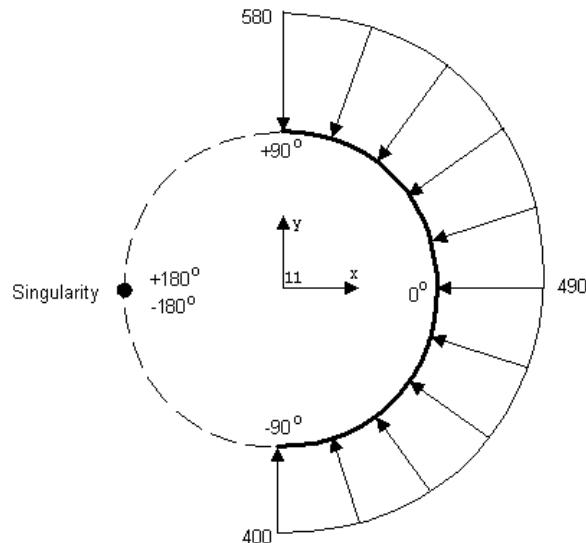
The **SFGRAD** specification stays active for all subsequent load application commands. To remove the specification, issue **SFGRAD** without any arguments. Also, if an **SFGRAD** specification is active when a load step file is read, the program erases the specification before reading the file.

Large-deflection effects can change the node locations significantly. The **SFGRAD** slope and load value calculations, which are based on node locations, are not updated to account for these changes. If you need this capability, use **SURF153** with face 3 loading or **SURF154** with face 4 loading.

3.5.7.3.1. Specifying the Gradient in a Cylindrical Coordinate System

When specifying the gradient in a cylindrical coordinate system (*SLKCN* = 1, for example), *SLZER* is in degrees, and *SLOPE* is in units of load/degree. Set **CSCIR** (for controlling the coordinate system singularity location) such that the surface to be loaded does *not* cross the coordinate system singularity. Choose *SLZER* to be consistent with the **CSCIR** setting (that is, *SLZER* should be between +180° if the singularity is at 180° [**CSCIR,KCN,0**], and *SLZER* should be between 0° and 360° if the singularity is at 0° [**CSCIR,KCN,1**]).

Consider a semicircle shell as shown in this figure, located in a local cylindrical system 11:

Figure 3.10: Tapered Load on a Cylindrical Shell

The shell is to be loaded with an external tapered pressure, tapering from 400 at -90° to 580 at $+90^\circ$. By default, the singularity in the cylindrical system is located at 180° , therefore the θ coordinates of the shell range from -90° to $+90^\circ$. The following commands apply the desired pressure load:

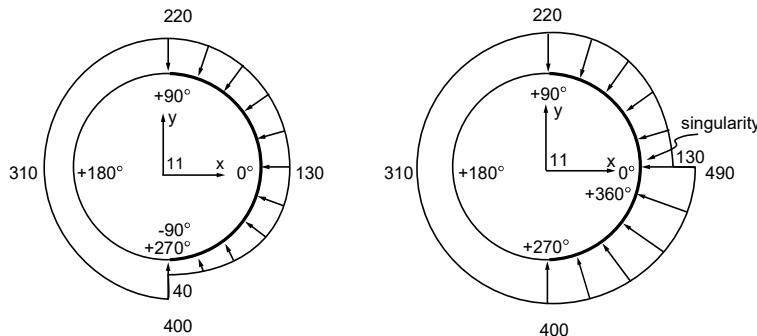
```
SFGRAD,PRES,11,Y,-90,1 ! Slope the pressure in the theta direction
!   of C.S. 11. Specified pressure in effect
!   at  $-90^\circ$ , tapering at 1 unit per degree
SF,ALL,PRES,400          ! Pressure at all selected nodes:
!   400 at  $-90^\circ$ , 490 at  $0^\circ$ , 580 at  $+90^\circ$ .
```

At -90° , the pressure value is 400 (as specified), increasing as θ increases by a slope of 1 unit per degree, to 490 at 0° and 580 at $+90^\circ$.

You might think to specify 270° rather than -90° for *SLZER*, as follows:

```
SFGRAD,PRES,11,Y,270,1 ! Slope the pressure in the theta direction
!   of C.S. 11. Specified pressure in effect
!   at  $270^\circ$ , tapering at 1 unit per degree
SF,ALL,PRES,400          ! Pressure at all selected nodes:
!   400 at  $-90^\circ$ , 490 at  $0^\circ$ , 580 at  $+90^\circ$ 
```

As shown on the left in this figure, however, specifying 270° results in a tapered load much different than the one intended:

Figure 3.11: Violation of Guideline 2 (Left) and Guideline 1 (Right)

This behavior occurs because the singularity is still located at 180° (the θ coordinates still range from -90° to $+90^\circ$), but *SLZER* is not between -180° and $+180^\circ$. As a result, the program uses a load value of 400 at 270° , and a slope of 1 unit per degree to calculate the applied load values of 220 at $+90^\circ$, 130 at 0° , and 40 at -90° . Avoid this behavior by following the second guideline, that is, specifying *SLZER* to be a value between $\pm 180^\circ$ when the singularity is at 180° , and between 0° and 360° when the singularity is at 0° .

Suppose that you change the singularity location to 0° , thereby satisfying the second guideline (270° is then between 0° and 360°). But then the θ coordinates of the nodes range from 0° to $+90^\circ$ for the upper half of the shell, and 270° to 360° for the lower half. The surface to be loaded crosses the singularity:

```
CSCIR,11,1      ! Change singularity to 0°
SFGRAD,PRES,11,Y,270,1 ! Slope the pressure in the theta direction
                        !   of C.S. 11. Specified pressure in effect
                        !   at 270°, tapering at 1 unit per degree
SF,ALL,PRES,400 ! Pressure at all selected nodes:
                  ! 400 at 270°, 490 at 360°, 220 at +90°
                  ! and 130 at 0°
```

Again, the program uses a load value of 400 at 270° and a slope of 1 unit per degree to calculate the applied load values of 400 at 270° , 490 at 360° , 220 at 90° , and 130 at 0° . Due to node discretization, the actual load applied does not change as abruptly at the singularity as it is shown in the figure. Instead, the node at 0° has the load value of, in the case shown, 130, while the next node clockwise (say, at 358°) has a value of 488.

3.5.7.4. Repeating a Surface Load

By default, if you repeat a surface load at the same surface, the new specification *replaces* the previous one. You can change the default to *add* (for accumulation) or *ignore* via the **SFCUM** command.

Any surface load you set stays set until you issue another **SFCUM** command. To reset the default setting (replacement), issue **SFCUM** without any arguments. The **SFSCALE** command enables you to scale existing surface load values. Both **SFCUM** and **SFSCALE** act only on the selected set of elements. The *Lab* field enables you to choose the surface load label.

3.5.7.5. Transferring Surface Loads

To transfer surface loads that have been applied to the solid model to the corresponding finite element model, issue the **SFTRAN** command.

To transfer all solid model boundary conditions, use the **SBCTRAN** command. (See [Degree-of-Freedom Constraints \(p. 29\)](#) for a description of DOF constraints.)

3.5.7.6. Using Surface Effect Elements to Apply Loads

Occasionally, you may need to apply a surface load that the element type you are using does not accept. For example, you may need to apply uniform tangential (or any non-normal or directed) pressures on structural solid elements, radiation specifications on thermal solid elements, etc. In such cases, you can overlay the surface where you want to apply the load with *surface effect* elements and use them as a "conduit" to apply the desired loads. Currently, the following surface effect elements are available: **SURF151** and **SURF153** for 2-D models and **SURF152**, **SURF154**, **SURF156**, and **SURF159** for 3-D models.

3.5.8. Applying Body Loads

The following table shows all body loads available in each discipline and their corresponding labels:

Table 3.7: Body Loads Available in Each Discipline

Discipline	Body Load	Label
Structural	Temperature	TEMP [1]
	Frequency	FREQ [2]
	Fluence	FLUE
Thermal	Heat Generation Rate	HGEN
Magnetic	Temperature	TEMP [1]
	Current Density	JS
	Virtual Displacement	MVDI
Electric	Temperature	TEMP [1]
	Volume Charge Density	CHRGD
Diffusion	Diffusing Substance Generation Rate	DGEN
	Temperature	TEMP
	Transport velocity	VELO
Acoustic	Mass source; mass source rate; or power source in an energy diffusion solution for room acoustics	MASS
	Impedance sheet	IMPD
	Static pressure	SPRE
	Temperature	TEMP
	Velocity or acceleration	VELO
	Interior acoustics port	PORT
	Floquet periodic boundary condition	FPBC
	Mean flow velocity	VMEN
	Force potential	UFOR
	Shear force	SFOR
	Volumetric heat source	HFLW

1. Do not confuse this body load with the TEMP degree of freedom.
2. Frequency (FREQ) is available for harmonic analyses only.

Table 3.8: Commands for Applying Body Loads

Location	Basic Commands	Additional Commands
Nodes	BF, BFLIST, BFDELE	BFSCALE, BFCUM, BFUNIF
Elements	BFE, BFELIST, BFEDELE	BFESCAL, BFECUM
Keypoints	BFK, BFKLIST, BFKDELE	-
Lines	BFL, BFLLIST, BFLDELE	-
Areas	BFA, BFALIST, BFADELE	-
Volumes	BFV, BFVLIST, BFVDELE	-
Transfer	BFTRAN	-

Body loads specified on nodes are independent of those specified on elements. For a given element, the program determines which loads to use as follows:

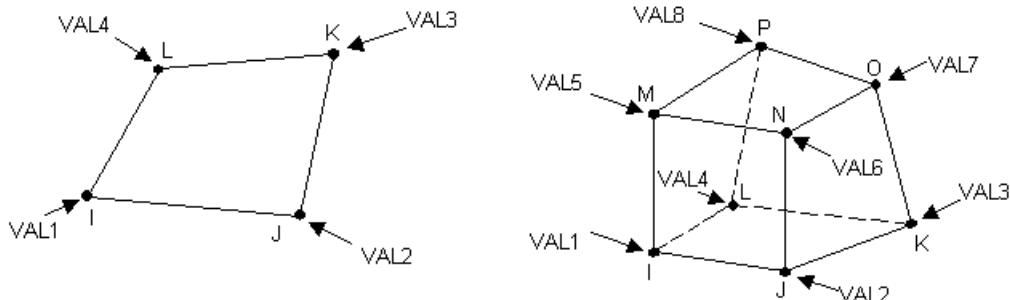
- It checks to see if you have specified elements for body loads.
- If not, it uses body loads specified for nodes.
- If no body loads exist for elements or nodes, the body loads specified via the **BFUNIF** command take effect.

3.5.8.1. Specifying Body Loads for Elements

The **BFE** command specifies body loads on an element-by-element basis. However, you can specify body loads at several locations on an element, requiring multiple load values for one element. The locations used vary from element type to element type. The defaults (for locations where no body loads are specified) also vary between element types; therefore, refer to the documentation for a given element before specifying body loads on elements.

- For 2-D and 3-D solid elements (PLANE nnn and SOLID nnn), the locations for body loads are usually the corner nodes.

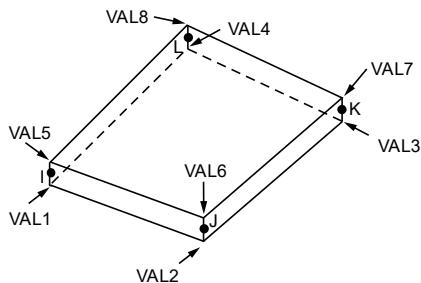
Figure 3.12: BFE Load Locations



For 2-D and 3-D Solids

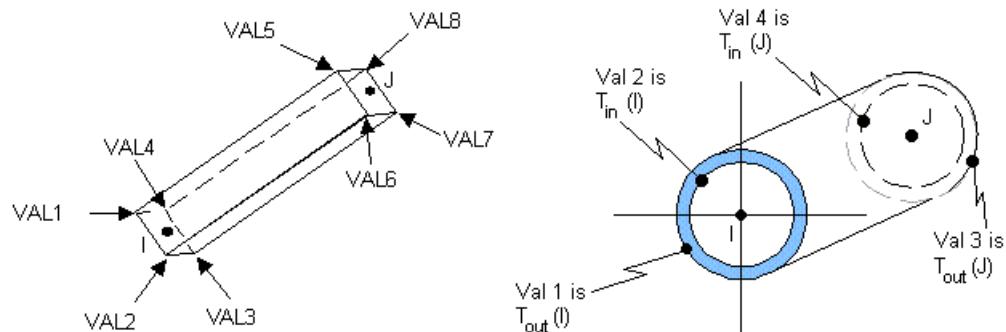
- For shell elements (**SHELLnnn**), the locations for body loads are usually the "pseudo-nodes" at the top and bottom planes, as shown below.

Figure 3.13: BFE Load Locations for Shell Elements



- Line elements (**BEAMnnn**, **LINKnnn**, **PIPEnnn**, etc.) are similar to shell elements; the locations for body loads are usually the pseudo-nodes at each end of the element.

Figure 3.14: BFE Load Locations for Beam and Pipe Elements

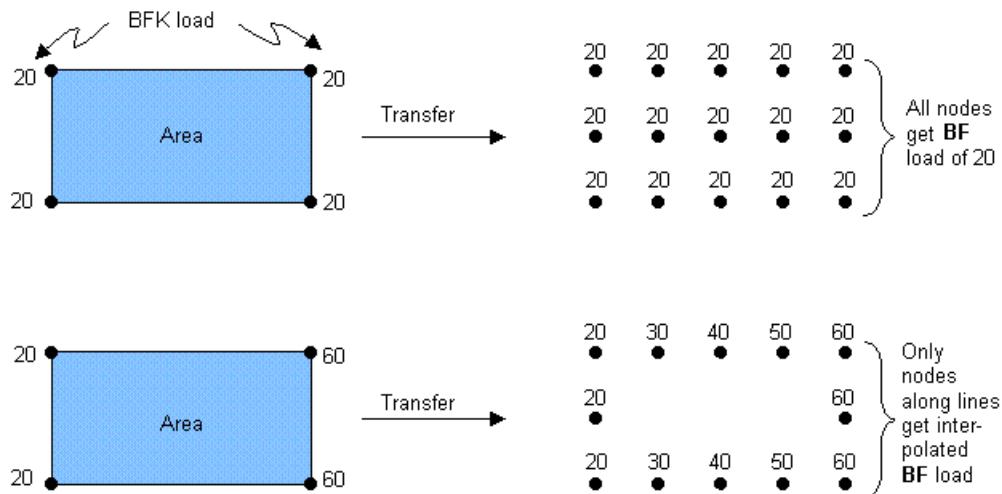


- In all cases, if degenerate (collapsed) elements are involved, you must specify element loads at *all* of its locations, including duplicate values at the duplicate (collapsed) nodes. A simple alternative is to specify body loads directly at the nodes, using the **BF** command.

3.5.8.2. Specifying Body Loads for Keypoints

The **BFK** command applies body loads at keypoints. When specifying loads at the corner keypoints of an area or a volume, all load values must be equal for the loads to be transferred to the *interior* nodes of the area or volume. Unequal load values are transferred (with linear interpolation) only to the nodes along the lines that connect the keypoints. [Figure 3.15: Transfers to BFK Loads to Nodes \(p. 44\)](#) illustrates this:

The **BFK** command can also specify table names at keypoints. When specifying table names at corner keypoints of an area or a volume, all table names must be equal for the loads to be transferred to the *interior* nodes of the area or volume.

Figure 3.15: Transfers to BFK Loads to Nodes

3.5.8.3. Specifying Body Loads on Lines, Areas and Volumes

The **BFL**, **BFA**, and **BFV** commands specify body loads on lines, areas, and volumes of a solid model, respectively. Body loads on lines of a solid model are transferred to the corresponding nodes of the finite element model. Body loads on areas or volumes of a solid model are transferred to the corresponding elements of the finite element model.

3.5.8.4. Specifying a Uniform Body Load

The **BFUNIF** command specifies a uniform body load at all nodes in the model. Most often, you use this command or path to specify a uniform temperature field; that is, a uniform temperature body load in a structural analysis or a uniform starting temperature in a transient or nonlinear thermal analysis. This is also the default temperature at which the program evaluates temperature-dependent material properties.

3.5.8.5. Repeating a Body Load Specification

By default, if you repeat a body load at the same node or same element, the new specification *replaces* the previous one. You can change this default to *ignore* via the **BFCUM** and **BFECUM** commands.

The settings you specify with either command or its equivalent GUI paths stay set until they are reused. To reset the default setting (replacement), issue the commands or choose the paths without any arguments.

3.5.8.6. Transferring Body Loads

To transfer body loads that have been applied to the solid model to the corresponding finite element model, issue the **BFTRAN** command.

To transfer all solid model boundary conditions, issue the **SBCTRA**N command. (For more information, see [Degree-of-Freedom Constraints \(p. 29\)](#).)

3.5.8.7. Scaling Body Load Values

You can scale existing body load values via the **BFSCALE** and **BFESCAL** commands

BFCUM and **BFSCALE** act on the selected set of nodes. **BFECUM** and **BFESCAL** act on the selected set of elements.

3.5.8.8. Resolving Conflicting Body Load Specifications

Be aware of the possibility of conflicting **BFK**, **BFL**, **BFA**, and **BFV** body load specifications and how the program handles them.

BFV, **BFA**, and **BFL** specifications transfer to associated volume, area, and line elements, respectively, where they exist. Where elements do not exist, they transfer to the nodes on the volumes, areas, and lines, including nodes on the region boundaries. The possibility of conflicting specifications depends upon how **BFV**, **BFA**, **BFL** and **BFK** are used as described by the following cases.

CASE A: There are elements for every **BFV**, **BFA**, or **BFL**, and every element belongs to a volume, area or line having a **BFV**, **BFA**, or **BFL**, respectively.

Every element have its body loads determined by the corresponding solid body load. Any **BFK**'s present have no effect. No conflict is possible.

CASE B: There are elements for every **BFV**, **BFA**, or **BFL**, but some elements do not belong to a volume area or line having a **BFV**, **BFA**, or **BFL**.

Elements not getting a direct **BFE** transfer from a **BFV**, **BFA**, or **BFL** are unaffected by them, but have their body loads determined by the following: (1 - highest priority) directly defined **BFE** loads, (2) **BFK** loads, (3) directly defined **BF** loads, or (4) **BFUNIF** loads. No conflict among solid model body loads is possible.

CASE C: At least one **BFV**, **BFA**, or **BFL** cannot transfer to elements.

Elements not getting a direct **BFE** transfer from a **BFV**, **BFA**, or **BFL** have their body loads determined by the following (in order of priority, with 1 being the highest):

1. **BFE** loads
2. **BFK** loads
3. **BFL** loads on an attached line that did NOT transfer to line elements
4. **BFA** loads on an attached area that did NOT transfer to area elements
5. **BFV** loads on an attached volume that did NOT transfer to volume elements
6. directly defined **BF** loads
7. **BFUNIF** loads

In "Case C" situations, the following conflicts can arise:

- A **BFL** specification can conflict with a **BFL** specification on an adjacent line (shared keypoint).

- A **BFL** specification can conflict with a **BFK** specification at either keypoint.
- A **BFA** specification can conflict with a **BFA** specification on an adjacent area (shared lines/keypoints).
- A **BFA** specification can conflict with a **BFL** specification on any of its lines.
- A **BFA** specification can conflict with a **BFK** specification on any of its keypoints.
- A **BFV** specification can conflict with a **BFV** specification on an adjacent volume (shared areas/lines/keypoints).
- A **BFV** specification can conflict with a **BFA** specification on any of its areas.
- A **BFV** specification can conflict with a **BFL** specification on any of its lines.
- A **BFV** specification can conflict with a **BFK** specification on any of its keypoints.

The program transfers body loads that have been applied to the solid model to the corresponding finite element model in the following sequence:

1. In ascending volume number order, **BFV** loads transfer to **BFE** loads on volume elements, or, if there are none, to **BF** loads on nodes on volumes (and bounding areas, lines, and keypoints).
2. In ascending area number order, **BFA** loads transfer to **BFE** loads on area elements, or, if there are none, to **BF** loads on nodes on areas (and bounding lines and keypoints).
3. In ascending line number order, **BFL** loads transfer to **BFE** loads on line elements, or, if there are none, to **BF** loads on nodes on lines (and bounding keypoints).
4. **BFK** loads transfer to **BF** loads on nodes on keypoints (and on attached lines, areas, and volumes if expansion conditions are met).

Accordingly, for conflicting solid model body loads in "Case C" situations, **BFK** commands overwrite **BFL** commands, **BFL** commands overwrite **BFA** commands, and **BFA** commands overwrite **BFV** commands. For conflicting body loads, a body load specified for a higher line number, area number, or volume number overwrites the body load specified for a lower line number, area number, or volume number, respectively. The body load specification issue order does not matter.

Any conflict detected during solid model body load transfer generates a warning.

Changing the value of **BFK**, **BFL**, **BFA**, or **BFV** constraints between solutions may produce many of these warnings at the second or later solid BC transfer. These can be prevented by deleting the nodal **BF** loads between solutions using **BFVDELE**, **BFADELE**, **BFLDELE**, and/or **BFDELE**.

3.5.9. Applying Inertia Loads

The following commands are available for applying inertia loads:

Figure 3.16: Inertia Loads Commands

Translational Acceleration	Application		Definition
	Whole structure	Component-based	Vector in the global (X,Y,Z)
ACEL	x		x
CMACEL		x	x

Rotational Motion		Application			Definition	
Velocity	Acceleration	Whole structure	Component based	Global Origin	Vector in the global (X,Y,Z)	User-defined rotational axis
OMEGA	DOMEGA	x			x	
CMOMEGA	CMDOMEGA		x			x
CGOMGA	DCGOMG			x	x	CGLOC

There are no specific commands to list or delete inertia loads. To list them, issue **STAT, INRTIA**. To remove an inertia load, set the load value to zero. You can set an inertia load to zero, but you cannot delete it. For ramped load steps, inertia loads are ramped to zero. (This is also true when you apply inertia loads.)

The **ACEL**, **OMEGA**, and **DOMEGA** commands specify acceleration, angular velocity, and angular acceleration, respectively, in global Cartesian directions. The **CMACEL** command is similar to the **ACEL** command, except that the translational acceleration applies to a component and not the whole structure. (The **ACEL** command applies an acceleration field [not gravity] to a body; therefore, to apply gravity to act in the negative Y direction, specify a positive Y acceleration.)

Use the **CGOMGA** and **DCGOMG** commands to specify angular velocity and angular acceleration of a spinning body which is itself revolving about another reference coordinate system. The **CGLOC** command specifies the location of the reference system with respect to the global Cartesian origin. You can use these commands, for example, to include Coriolis effects in a static analysis.

You can also issue **CMOMEGA** and **CMDOMEGA** commands to specify the rotational velocity and acceleration effects for element components you define. You either specify an axis and the scalar vector quantity, or define the three components of the rotational value and the point in space you are considering. You can use these commands for element components only.

Inertia loads are effective only if your model has some mass, which is usually supplied by a density specification. (You can also supply mass to the model by using mass elements, such as **MASS21**, but density is more commonly used and is more convenient.) As with all other data, the program requires you to use consistent units for mass. If you are accustomed to the U. S. Customary system of units, you might sometimes wish to use *weight density* (lb/in^3) instead of *mass density* ($\text{lb}\cdot\text{sec}^2/\text{in}/\text{in}^3$), for convenience.

Use weight density in place of mass density only under these conditions:

- The model only be used in a static analysis.

- No angular velocity or angular acceleration is applied.
- Gravitational acceleration is unity ($g = 1.0$).

A method for specifying density so that you can use it readily in either a "convenient," weight-density form or "consistent," mass-density form is to define a parameter g for gravitational acceleration:

Table 3.9: Ways of Specifying Density

Convenient Form	Consistent Form	Description
$g = 1.0$	$g = 386.0$	Parameter definition
MP,DENS,1,0.283/g	MP,DENS,1,0.283/g	Density of steel
ACEL,,g	ACEL,,g	Gravity load

You can also use weight density along with system mass matrix scaling to perform your analysis using consistent units. For limitations, see the **MASCALE** command documentation.

3.5.10. Applying Ocean Loads

Ocean loading includes the effects of waves, current, drag, and buoyancy. Loading is input globally via the **ocean family of commands** (**OC** xxxxxxx). Ocean loading is supported in static, full transient (**TRNOPT**,**FULL**), and full harmonic (**HROPT**,**FULL**) analyses.

The following ocean-loading input groups are available:

- Basic (required for any ocean loading)
- Current (optional, for applying drift current)
- Wave (optional, for applying a wave state)
- Zone (optional, for applying *local* ocean effects)

Basic, *wave*, and *zone* inputs use the **OCTYPE**, **OCDATA**, and **OCTABLE** commands. *Current* inputs use the **OCTYPE** and **OCTABLE** commands.

The **OCLIST** and the **OCDELETE** commands assist in data handling and perform the tasks that their names imply.

All ocean loading requires specifying the linear acceleration of the global Cartesian reference frame (**ACEL**, where $ACEL_X = ACEL_Y = 0.0$, and $ACEL_Z$ = acceleration due to gravity).

Ocean-loading support is available for the following current-technology elements:

Element	Description
SURF154	3-D Structural Surface Effect
LINK180	3-D Spar (or Truss)
BEAM188	3-D 2-Node Beam
BEAM189	3-D 3-Node Beam
PIPE288	3-D 2-Node Pipe

Element	Description
PIPE289	3-D 3-Node Pipe

For more information, see the following related topics:

- The [ocean](#) family of commands.
- [Hydrostatic Loads](#) in the *Theory Reference*.
- [Hydrodynamic Loads](#) in the *Theory Reference*.
- Applying Ocean Loading from a Hydrodynamic Analysis in the *Advanced Analysis Guide*.
- Harmonic Ocean Wave Procedure (HOWP) in the *Theory Reference*.
- Subroutine `userPanelHydFor` (Calculating Panel Loads Caused by Ocean Loading) in the *Programmer's Reference*

3.5.11. Applying Coupled-Field Loads

A coupled-field analysis usually involves applying results data from one analysis as loads in a second analysis. For example, you can apply the nodal temperatures calculated in a thermal analysis as body loads in a structural analysis (for thermal strain). Similarly, you can apply magnetic forces calculated in a magnetic field analysis as nodal forces in a structural analysis. To apply such coupled-field loads, issue the **LDREAD** command.

For more information, see the *Coupled-Field Analysis Guide*.

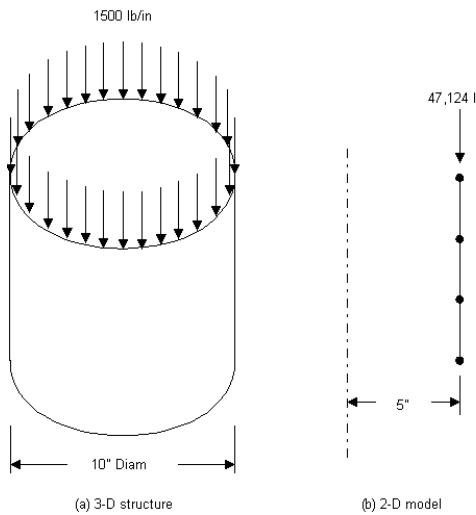
3.5.12. Axisymmetric Loads and Reactions

For constraints, surface loads, body loads, and Y-direction accelerations, you define loads exactly as they would be for any nonaxisymmetric model. However, for concentrated forces the procedure is a little different. For these quantities, input load values of force, moment, etc. are on a "360° basis." That is, the load value is entered in terms of *total load around the circumference*. For example, if an axisymmetric axial load of 1500 pounds per inch of circumference were applied to a 10" diameter pipe ([Figure 3.17: Concentrated Axisymmetric Loads \(p. 50\)](#)), the total load of 47,124 lb. ($1500 * 2 \pi * 5 = 47,124$) would be applied to node N as follows:

```
F,N,FY,-47124
```

Axisymmetric results are interpreted in the same fashion as their corresponding input loads. That is, reaction forces, moments, etc. are reported on a total load (360°) basis.

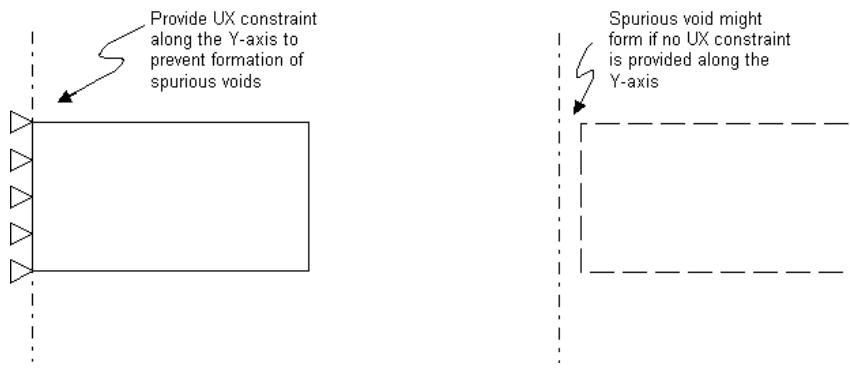
Axisymmetric harmonic elements require that their loads be provided in a form that the program can interpret as a Fourier series. The **MODE** command, together with other load commands (**D**, **F**, **SF**, etc.), are required for these elements.

Figure 3.17: Concentrated Axisymmetric Loads

Defined on a 360° basis

3.5.12.1. Hints and Restrictions

Specify a sufficient number of constraints to prevent unwanted rigid-body motions, discontinuities, or singularities. For example, for an axisymmetric model of a solid structure such as a solid bar, a lack of UX constraint along the axis of symmetry can potentially allow spurious "voids" to form in a structural analysis.

Figure 3.18: Central Constraint for Solid Axisymmetric Structure

3.5.13. Loads to Which the Degree of Freedom Offers No Resistance

If an applied load acts on a degree of freedom which offers no resistance to it (that is, perfectly zero stiffness), the program ignores the load.

3.5.14. Initial State Loading

You can specify **initial state** as a loading parameter for a structural analysis. Initial state loading is valid for static or full transient analyses (either linear or nonlinear), and for modal, buckling and harmonic analyses. Initial state must be applied in the first load step of an analysis.

Initial state is also available in Distributed ANSYS.

For more information, see [Initial State in the Advanced Analysis Guide](#).

3.5.15. Applying Loads Using Tabular Input

To apply loads using tabular input, issue the appropriate loading commands for your analysis and input the name of a table array parameter instead of a load value. Not all boundary conditions support tabular loads; refer to the command documentation for the specific loads you are working with to determine if tabular loads are supported.

When defining loads, enclose the table name within % characters: %tabname%. For example, to specify a table of convection values, issue a command similar to the following:

```
SF,all,conv,%sycnv%,tblk
```

If your data cannot be conveniently expressed as a table, you may want to use function boundary conditions. See [Using the Function Tool \(p. 81\)](#).

Define a table before applying loads (*DIM).

Tabular loads can be defined in the global Cartesian (default) coordinate system, or in a local coordinate system (**LOCAL**). Only Cartesian, spherical, and cylindrical coordinate systems are valid.

The following topics are available concerning applying loads via table array parameters

- [3.5.15.1. Defining Primary Variables](#)
- [3.5.15.2. Defining Independent Variables](#)
- [3.5.15.3. Operating on Table Parameters](#)
- [3.5.15.4. Verifying Boundary Conditions](#)
- [3.5.15.5. Example Analysis Using 1-D Table Array](#)
- [3.5.15.6. Example Analysis Using 5-D Table Array](#)

For more information, see [Tabular Input via Table Array Parameters in the ANSYS Parametric Design Language Guide](#) and [Defining Linear Material Properties Using Tabular Input in the Material Reference](#).

3.5.15.1. Defining Primary Variables

When defining a table array parameter for loading, you can use various primary variables depending on the engineering discipline of your analysis. For a coupled-field analysis, there might be loads from more than one discipline (for example, structural and thermal).

Table 3.10: Boundary Condition Type and Corresponding Primary Variable

Boundary Condition	Primary Variable	Command [1 (p. 54)]
Thermal		
Fixed Temperature	TIME, X, Y, Z, NODE	D ,,(TEMP,TBOT,TE2,TE3,...,TTOP)
Heat Flow	TIME, X, Y, Z, TEMP, NODE	F ,,(HEAT,HBOT,HE2,HE3,...,HTOP)

Boundary Condition	Primary Variable	Command [1 (p. 54)]
Film Coefficient (Convection)	TIME, X, Y, Z, TEMP, VELOCITY, ELEM, NODE	SF,,CONV SFE,,CONV
Bulk Temperature (Convection)	TIME, X, Y, Z, ELEM, NODE	SF,,TBULK SFE,,TBULK
Heat Flux	TIME, X, Y, Z, TEMP, ELEM, NODE	SF,,HFLUX SFE,,HFLUX
Heat Generation	TIME, X, Y, Z, TEMP, ELEM	BFE,,HGEN
Uniform Heat Generation	TIME	BFUNIF,HGEN
Structural		
Displacements	TIME or FREQ, X, Y, Z, TEMP, NODE	D,(UX, UY, UZ, ROTX, ROTY, ROTZ)
Hydrostatic Pressure	TIME or FREQ, X, Y, Z, TEMP	D,HDSP
Forces and Moments	TIME or FREQ, X, Y, Z, TEMP, SECTOR, NODE	F,(FX, FY, FZ, MX, MY, MZ)
Fluid Mass Flow Rate	TIME or FREQ, X, Y, Z, TEMP, SECTOR	F,DVOL
Pressures	TIME or FREQ, X, Y, Z, TEMP, SECTOR, ELEM, NODE	SF,,PRES SFE,,PRES
Temperature	TIME, X, Y, Z, TEMP, ELEM, NODE	BF,,TEMP BFE,,TEMP
Body-force density	TIME or FREQ, ELEM	BFE,,FORC
Linear Acceleration	TIME or FREQ	ACEL,ACEL_X, ACEL_Y,ACEL_Z
Translational Acceleration	TIME or FREQ	CMACEL,CMACEL_X, CMACEL_Y,CMACEL_Z
Superelement Load Vector	TIME	SFE,,SELV
Electromagnetic		
Current Density	TIME, X, Y, Z, ELEM, NODE	BFE,,JS
Velocity	TIME, X, Y, Z, NODE	BF,,VELO
Electric		
Voltage	TIME, X, Y, Z, NODE	D,,VOLT
Current	TIME, X, Y, Z, NODE	F,,AMPS
Fluid		
Pressure	TIME, X, Y, Z, NODE	D,,PRES

Boundary Condition	Primary Variable	Command [1 (p. 54)]
Flow	TIME, X, Y, Z, NODE	F,,FLOW
Diffusion		
Concentration	TIME, X, Y, Z, NODE	D,,CONC
Diffusion Flow Rate	TIME, X, Y, Z, NODE	F,,RATE
Diffusion Flux	TIME, X, Y, Z, TEMP, CONC, ELEM, NODE	SF,,DFLUX SFE,,DFLUX
Diffusion Substance Generation	TIME, X, Y, Z, TEMP, CONC, ELEM, NODE	BF,,DGEN BFE,,DGEN
Temperature	TIME, X, Y, Z	BF,,TEMP BFE,,TEMP
Transport Velocity	TIME, X, Y, Z, NODE	BF,,VELO
Acoustic		
Pressure	TIME or FREQ, X, Y, Z, NODE	D,,PRES
Energy density	TIME or FREQ, X, Y, Z, NODE	D,,ENKE
Velocity	FREQ, X, Y, Z, NODE	D,,(VX, VY, VZ)
Temperature	FREQ, X, Y, Z, NODE	D,,TEMP
Impedance	TIME or FREQ	SF,,IMPD
Surface normal velocity or acceleration	TIME or FREQ	SF,,SHLD
Absorption coefficient	TIME or FREQ	SF,,ATTN
Surface pressure	FREQ	SF,,PRES
Heat flux	FREQ	SF,,CONV
Viscous impedance	FREQ	SF,,VIMP
Thermal impedance	FREQ	SF,,TIMP
Mass source, mass source rate, or power source	TIME or FREQ, X, Y, Z, NODE	BF,,MASS
Static pressure	TIME or FREQ, X, Y, Z, NODE	BF,,SPRE
Temperature	TIME, X, Y, Z, NODE	BF,,TEMP
Velocity or acceleration	TIME or FREQ, X, Y, Z, NODE	BF,,VELO
Mean flow velocity	X, Y, Z, NODE	BF,,VMEN
Shear force	FREQ, X, Y, Z, NODE	BF,,SFOR
Volumetric heat source	FREQ, X, Y, Z, NODE	BF,,HFLW

Additional primary variables are available using function boundary conditions. See [Using the Function Editor \(p. 82\)](#) for more information. Primary variables are shown as the valid labels used by the ***DIM** command. You can apply tabular loads according to a local coordinate system defined via **LOCAL**, and specified in ***DIM**.

When defining the tables, the primary variables must be in ascending order in the table indices (as in any table array).

1. Although not normally used in this manner, the following commands also allow tabular loading: **BFA**, **BFK**, **BFL**, **BFV**, **DA**, **DK**, **DL**, **FK**, **SFA**, and **SFL**.

See the ***DIM** command for more information about defining your labels.

The *VELOCITY* label refers to the magnitude of the velocity degrees of freedom or the computed fluid velocity in **FLUID116** elements.

In addition, some real constants for elements **SURF151**, **SURF152**, and **FLUID116** can have associated primary variables.

Table 3.11: Real Constants and Corresponding Primary Variable for SURF151, SURF152, and FLUID116

Real Constants	Primary Variables
SURF151, SURF152	
Rotational Speed	TIME, X, Y, Z, ELEM
FLUID116	
Rotational Speed	TIME, X, Y, Z, ELEM
Slip Factor	TIME, X, Y, Z, ELEM

Contact elements (**CONTA172**, **CONTA174**, **CONTA175**, **CONTA177**, and **CONTA178**) also support table parameter input for some real constants. For a complete list of these real constants and their associated primary variables, see **Real Constants and Corresponding Primary Variables for CONTA172, CONTA174, CONTA175, CONTA177** in the *Contact Technology Guide* and **Real Constants and Corresponding Primary Variables for CONTA178** in the *Element Reference*.

The following two primary variables are used exclusively with contact element real constants:

- **PRESSURE** -- Contact pressure. The index values associated with **PRESSURE** are positive for compression and negative for tension.
- **GAP** -- Geometrical contact gap/penetration. The index values associated with **GAP** are positive for closed penetration and negative for an open gap.

3.5.15.2. Defining Independent Variables

If you need to specify a variable other than one of the primary variables listed, you can do so by defining an independent parameter. To specify an independent parameter, you define an additional table for the independent parameter. That table must have the same name as the independent parameter, and can be a function of either a primary variable or another independent parameter. You can define as many independent parameters as necessary, but all independent parameters must relate to a primary variable.

Example 3.5: Defining Independent Variables

Consider a convection coefficient (HF) that varies as a function of rotational speed (RPM) and temperature (TEMP).

The primary variable in this case is TEMP. The independent parameter is RPM, which varies with time.

In this scenario, you need two tables: one relating RPM to TIME, and another table relating HF to RPM and TEMP.

```
*DIM,SYCNV,TABLE,3,3,,RPM,TEMP
SYCNV(1,0)=0.0,20.0,40.0
SYCNV(0,1)=0.0,10.0,20.0,40.0
SYCNV(0,2)=0.5,15.0,30.0,60.0
SYCNV(0,3)=1.0,20.0,40.0,80.0
*DIM,RPM,TABLE,4,1,1,TIME
RPM(1,0)=0.0,10.0,40.0,60.0
RPM(1,1)=0.0,5.0,20.0,30.0
SF,ALL,CONV,%SYCNV%
```

When defining the tables, the independent variables must be in ascending order in the table indices (as in any table array).

3.5.15.3. Operating on Table Parameters

For convenience, you can multiply table parameters by constants, add one table to another, and add a constant increment for offset. To do so, issue the ***TOPER** command. The two tables must have the same dimensions and must have the same variable names for the rows and columns. The tables must also have identical index values for rows, columns, etc.

3.5.15.4. Verifying Boundary Conditions

If you use table array parameters to define boundary conditions, you may want to verify that the correct table and the correct values from the table were applied. You can do so in several ways:

- You can look in the Output window. If you apply tabular boundary conditions on finite element or solid model entities, the name of the table, not the numerical value, is echoed in the Output window.
- You can list boundary conditions. If you list the boundary conditions during /PREP7, table names are listed. Longer table names may be truncated. However, if you list boundary conditions during any of the solution or postprocessing phases at a particular entity or time point, the actual numerical value at the location or time is listed.
- You can look at the graphical display. Where tabular boundary conditions were applied, the table name and any appropriate symbols (face outlines, arrows, etc.) can be displayed using the standard graphic display capabilities (**/PBC**, **/PSF**, etc.), provided that table numbering is on (**/PNUM,TABNAM,ON**).
- You can look at the numerically-substituted table of values (**/PNUM,SVAL**) in POST1.
- You can retrieve the values of a table parameter at any given combination of index numbers via the ***STATUS** command. Referring to the above example, ***STATUS,SYNCV,2,2,3,3** [print SYNCV at index position (2,3)] prints 40.0.
- You can retrieve the values of a table parameter at any given combination of its primary variables via the ***SET** command. Referring to the above example, **CON = SYNCV(30.0,0.25)** [evaluate SYNCV at RPM = 30.0 and TEMP = 0.25] gives CON = 37.5.

3.5.15.5. Example Analysis Using 1-D Table Array

An example of how to run a steady-state thermal analysis using tabular boundary conditions is described in [Performing a Thermal Analysis Using Tabular Boundary Conditions in the *Thermal Analysis Guide*](#).

3.5.15.6. Example Analysis Using 5-D Table Array

This example shows how to run an analysis using a 5-D table. Note that 4- and 5-D tables cannot be defined interactively; you must use the command method.

This problem consists of a thermal-stress analysis with a pressure that varies as a function of (x,y,z,time,temp). The table and table values are first defined. The table is applied as a pressure boundary condition to the faces of a rectangular beam. Time and temperature are prescribed for two load steps and solved.

```
/batch,list
/title, Illustrate use of 5D table for SF command (pressure) loading
!!!!
!!!!
      !!!! create 5D table for applied pressure
X1=2          !!! X dimensionality
Y1=2          !!! Y dimensionality
Z1=10         !!! Z dimensionality
D4=5          !!! time dimensionality
D5=5          !!! temperature dimensionality
len=10         !!! cantilever beam length
wid=1          !!! cantilever beam width
hth=2          !!! cantilever beam height

*dim,xval,array,X1          !!! create 1D arrays to load 5D table
xval(1)=0,20                !!! variations per dimension same
*dim,yval,array,Y1          !!! but give different values on each
yval(1)=0,20
*dim,zval,array,10
zval(1)=10,20,30,40,50,60,70,80,90,100
*dim,tval,array,5
tval(1)=1,.90,.80,.70,.60
*dim,tevl,array,5
tevl(1)=1,1.20,1.30,1.60,1.80

*dim,ccc,tab5,X1,Y1,Z1,D4,D5,X,Y,Z,TIME,TEMP
*taxis,ccc(1,1,1,1,1),1,0,wid    !!! X-Dim
*taxis,ccc(1,1,1,1,1),2,0,hth    !!! Y-Dim
*taxis,ccc(1,1,1,1,1),3,1,2,3,4,5,6,7,8,9,10 !!! Z-Dim
*taxis,ccc(1,1,1,1,1),4,0,10,20,30,40   !!! Time
*taxis,ccc(1,1,1,1,1),5,0,50,100,150,200 !!! Temp
*do,ii,1,2
  *do,jj,1,2
    *do,kk,1,10
      *do,ll,1,5
        *do,mm,1,5
          ccc(ii,jj,kk,ll,mm)=(xval(ii)+yval(jj)+zval(kk))*tval(ll)*tevl(mm)
        *enddo
      *enddo
    *enddo
  *enddo
*enddo

/prep7
block,,wid,,hth,,len          !!! create beam volume
et,1,5                         !!! use SOLIDS5
esize,0.5                        !!! element size
```

```

mshkey,1           !!!! mapped mesh
vmesh,all

mp,ex,1,1e7        !!!! material properties
mp,nuxy,1,.3
mp,kxx,1,1

nsel,s,loc,z,0    !!!! fix end of beam
d,all,all
fini
save             !!!! save problem for future restart
/solu
antyp,trans
timint,off

asel,u,loc,z,0
sfa,all,1,pres,%ccc%      !!!! apply pressure to all selected areas
alls
time,le-3          !!!! first solution at time = "0"
nsub,1
outres,all,all     !!!! output everything to results file
d,all,temp,0       !!!! for first problem, temp = 0
solve

time,30      !!!! second solution, time=30
d,all,temp,150    !!!! second solution, temp=150
solve
finish
/post1
/view,1,1,1,1
/psf,press,norm,3,0,1
/pbc,all,0
set,1,1
/title, Pressure distribution; time=0, temp=0
eplot
set,2,1
/title, Pressure distribution; time=30, temp=150
eplot
finish

```

The following plots illustrate the pressure distribution for the two load cases.

Figure 3.19: Pressure Distribution for Load Case 1

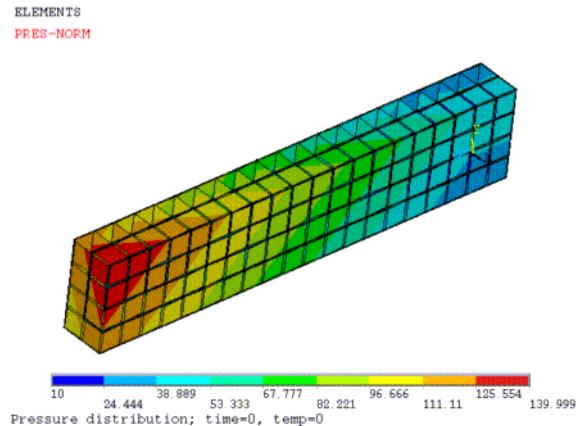
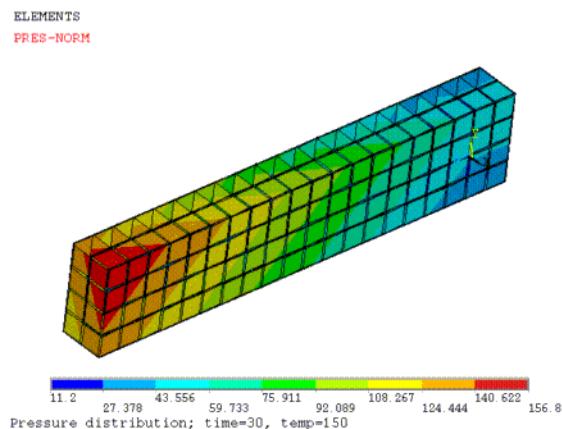


Figure 3.20: Pressure Distribution for Load Case 2

Notice the difference in the pressure load in the second load case.

3.5.16. Applying Loads to Components and Assemblies

You can use components and assemblies to apply loads to portions of the model:

3.5.16.1. Applying a Load to a Component

3.5.16.2. Applying a Load to an Assembly

For related information, see [Using Components and Assemblies \(p. 212\)](#).

3.5.16.1. Applying a Load to a Component

Apply loads to portions of the model using a component as follows:

```
CM,cmp,Entity
SF,cmp,CONV,10
```

The **SF** command specifies the film coefficient of 10 on nodes in the component (*cmp*). All nodes in the component are retrieved regardless of the entity type.

An equivalent method for applying the coefficient uses the **CMSEL**, **ALLSEL**, and **SF** commands, as follows:

```
CMSEL,S,cmp
ALLSEL,BELOW,ALL
SF,ALL,CONV, 10
```

3.5.16.2. Applying a Load to an Assembly

Because an assembly is a group of components, the same principle for loading applies to the assembly, as follows:

```
SF,assm,CONV,10
```

In this case, the coefficient is applied to all nodes in each component within the assembly.

3.6. Specifying Load Step Options

Load step options refer collectively to options controlling how loads are used during solution, and other options such as output controls, damping specifications, and response spectrum data. Load step options can vary from load step to load step.

Several categories of load step options are available:

- 3.6.1. Setting General Options
- 3.6.2. Setting Dynamics Options
- 3.6.3. Setting Nonlinear Options
- 3.6.4. Setting Output Controls
- 3.6.5. Setting Biot-Savart Options
- 3.6.6. Setting Spectrum Options

3.6.1. Setting General Options

These include such options as time at the end of a load step in transient and static analyses, number of substeps or the time step size, stepping or ramping of loads, and reference temperature for thermal strain calculations. A brief description of each option follows.

3.6.1.1. Time Option

The **TIME** command specifies time at the end of a load step in transient and static analyses. In transient and other rate-dependent analyses, **TIME** specifies actual, chronological time, and you are required to specify a time value. In other, rate-independent analyses, time acts as a tracking parameter. You can never set time to zero in an analysis. If you issue **TIME,0** or **TIME,(blank)**, or if you do not issue the **TIME** command at all, the program uses the default time value: 1.0 for the first load step, and 1.0 + previous time for other load steps. To start your analysis at "zero" time, such as in a transient analysis, specify a very small value such as **TIME,1E-6**.

3.6.1.2. Number of Substeps and Time Step Size

For a nonlinear or transient analysis, specify the number of substeps to be taken within a load step via the **DELTIM** and **NSUBST** commands.

NSUBST specifies the number of substeps, and **DELTIM** specifies the time step size. By default, the program uses one substep per load step.

3.6.1.3. Automatic Time Stepping

The **AUTOTS** command activates automatic time stepping.

In automatic time stepping, the program calculates an optimum time step at the end of each substep, based on the response of the structure or component to the applied loads. When used in a nonlinear static (or steady-state) analysis, **AUTOTS** determine the size of load increments between substeps.

3.6.1.4. Stepping or Ramping Loads

When specifying multiple substeps within a load step, indicate whether the loads are to be ramped or stepped by issuing the **KBC** command. **KBC,0** indicates ramped loads, and **KBC,1** indicates stepped loads. The default depends on the discipline and type of analysis.

Note:

There is a third option specific to rotational velocity loads (**OMEGA**, **CMOMEGA**, and **CMROTADE**). The load can be quadratically interpolated (*OMGSQRD* on **KBC**). For more information, see [Ramped and Stepped Loads \(p. 26\)](#).

The concept of stepped versus ramped loading does *not* apply to temperature-dependent film coefficients (input as *-N* on a convection command). These are always applied at the value dictated by their temperature function.

Consider the following when issuing the **KBC** command:

- **Stepped loads:**

- The program handles all loads (constraints, forces, surface loads, body loads, and inertia loads) in the same manner. They are step-applied, step-changed, or step-removed, as the case may be.

- **Ramped loads:**

- All loads applied in the first load step, except film coefficients, are ramped (either from zero or from the value specified via **BFUNIF** or its GUI equivalent, depending on the type of load; see [Table 3.12: Handling of Ramped Loads \(KBC = 0\) Under Different Conditions \(p. 61\)](#)). Film coefficients are step-applied.
 - All loads changed in later load steps are ramped from their previous values. If a film coefficient is specified using the temperature-dependent format (input as *-N*) for one load step and then changed to a constant value for the next step, the new constant value is step-applied. Note that in a full harmonic analysis (**ANTYPE,HARM** with **HROPT,FULL**), surface and body loads ramp as they do in the first load step and *not* from their previous values.
 - For tabular boundary conditions, loads are never ramped but rather evaluated at the current time. If a load is specified using the tabular format for one load step and then changed to a non-tabular for the next, the load is treated as a newly introduced load and ramped from zero or from **BFUNIF** and not from the previous tabular value.
 - All loads newly introduced in later load steps are ramped (either from zero or from **BFUNIF**, depending on the type of load; see [Table 3.12: Handling of Ramped Loads \(KBC = 0\) Under Different Conditions \(p. 61\)](#)).
 - All loads deleted in later load steps are step-removed, except body loads and inertia loads. Body loads are ramped to **BFUNIF**. Inertia loads, which you can delete only by setting them to zero, are ramped to zero.

- Do not delete and respecify loads in the same load step; otherwise, ramping may lead to unpredictable results.

Table 3.12: Handling of Ramped Loads (KBC** = 0) Under Different Conditions**

Load Type	Applied in Load Step 1	Introduced in Later Load Steps
DOF Constraints		
Temperatures	Ramped from TUNIF [2]	Ramped from TUNIF [3]
Others	Ramped from zero	Ramped from zero
Forces		
	Ramped from zero	Ramped from zero
Surface		
TBULK	Ramped from TUNIF [2]	Ramped from TUNIF
HCOEF	Stepped	Ramped from zero[4]
Others	Ramped from zero	Ramped from zero
Body		
Temperatures	Ramped from TUNIF [2]	Ramped from previous TUNIF [3]
Others	Ramped from BFUNIF [5]	Ramped from previous BFUNIF [3]
Inertia [1]	Ramped from zero	Ramped from zero

1. Rotational velocity loads (**OMEGA**, **CMOMEGA**) are ramped linearly; the resulting forces vary quadratically over the load step.
2. The **TUNIF** command specifies a uniform temperature at all nodes. Because **TUNIF** (or **BFUNIF**,**TEMP**) is step-applied in the first iteration, issue a **BF,ALL,TEMP,Value** command to ramp on a uniform temperature load.
3. In this case, the **TUNIF** or **BFUNIF** value from the *previous* load step is used, not the current value.
4. Temperature-dependent film coefficients are always applied at the value dictated by their temperature function, regardless of the **KBC** setting.
5. The **BFUNIF** command is a generic form of **TUNIF**, meant to specify a uniform body load at all nodes.

3.6.1.5. Other General Options

You can also specify the following general options:

- The reference temperature for thermal strain calculations, which defaults to zero degrees (**TREF**).
- Whether a new factorized matrix is required for each solution (that is, each equilibrium iteration) **KUSE**). You can do this only in a static (steady-state) or transient analysis.

By default, the program decides whether a new matrix is required, based on such things as changes in DOF constraints, temperature-dependent material properties, and the Newton-Raphson option. If **KUSE** is set to 1, the program reuses the previous factorized matrix. This

setting cannot be used during a multiframe restart. **KUSE,-1** forces the factorized matrix to be reformulated at every equilibrium iteration. Analyses rarely require this; you use it mainly for debugging purposes.

To generate and keep the sparse solver workspace files (*Jobname.DSPxxxx*), issue the command **EQSLV,SPARSE,,,KEEP** command.

- A mode number (the number of harmonic waves around the circumference) and whether the harmonic component is symmetric or antisymmetric about the global X axis (**MODE**). When you use axisymmetric harmonic elements (axisymmetric elements with nonaxisymmetric loading), the loads are specified as a series of harmonic components (a Fourier series).
- The type of scalar magnetic potential formulation to be used in a 3-D magnetic field analysis (**MAGOPT**).
- The type of solution to be expanded in the expansion pass of a mode-superposition analysis (**NUMEXP, EXPSEL**).

3.6.2. Setting Dynamics Options

These are options used primarily in dynamic and other transient analyses:

Table 3.13: Dynamic and Other Transient Analyses Commands

Command	Purpose
TIMINT	Activates or deactivates time integration effects
HARFRQ	Specifies the frequency range of the loads in a harmonic analysis
ALPHAD	Specifies damping for a structural dynamic analysis
BETAD	Specifies damping for a structural dynamic analysis
DMPRAT	Specifies damping for a structural dynamic analysis
MDAMP	Specifies damping for a structural dynamic analysis
TRNOPT	Specifies transient analysis options

3.6.3. Setting Nonlinear Options

These are options used mainly in nonlinear analyses:

Table 3.14: Nonlinear Analyses Commands

Command	Purpose
NEQIT	Specifies the maximum number of equilibrium iterations per substep
CNVTOL	Specifies convergence tolerances
NCNV	Provides options for terminating analyses

3.6.4. Setting Output Controls

Three primary output controls for specifying the amount and nature of output from an analysis are shown in this table:

Table 3.15: Output Controls Commands

Command	Purpose
OUTRES	Controls what the program writes to the database and results file and how often it is written.
OUTPR	Controls what is printed (written to the solution output file, Job-name.OUT) and how often it is written.
OUTGEOM	Controls the type of geometry data the program writes to the results file.

The example below illustrates using **OUTRES**, **OUTPR**, and **OUTGEOM**:

```
OUTRES,ALL,5      ! Writes all data every 5th substep
OUTPR,NSOL,LAST  ! Prints nodal solution for last substep only
OUTGEOM,MAT,NONE ! Suppresses the writing of the material property data to the results file
```

You can issue a series of **OUTPR**, **OUTRES**, and **OUTGEOM** commands (up to 50 of them combined) to meticulously control the solution output, but be aware that the order in which they are issued is important. For example, the commands shown below write all data to the database and results file every 10th substep and nodal solution data every fifth substep.

```
OUTRES,ALL,10
OUTRES,NSOL,5
```

However, if you reverse the order of the commands (as shown below), the second command essentially overrides the first, resulting in all data being written every 10th substep and nothing every 5th substep.

```
OUTRES,NSOL,5
OUTRES,ALL,10
```

As another example,

```
OUTRES,NSOL,10
OUTRES,NSOL,ALL,TIP
```

writes the solution at all DOFs every 10th substep and the solution at the node component TIP every substep. Again, if you reverse these you only obtain output at all DOF every 10th substep.

The program default for writing out solution data for all elements depends on the analysis type. See the **OUTRES** command description for more information.

To restrict the solution data that is written out, use **OUTRES** to selectively suppress (*Freq* = NONE) the writing of solution data, or first suppress the writing of all solution data (**OUTRES,ALL,NONE**) and then selectively turn on the writing of solution data with subsequent **OUTRES** commands.

The **OUTGEOM** command is used to either write or suppress the geometry data from the results file for all substeps. The only valid frequency specification for this command are ALL and NONE. See the **OUTGEOM** command description for more information.

3.6.4.1. Element Integration Point Values

Another output control command, **ERESX**, enables you to review element integration point values in the postprocessor.

By default, the program extrapolates nodal results that you review in the postprocessor from integration point values for all elements except those with active material nonlinearities (for instance, nonzero plastic strains).

By issuing **ERESX,NO**, you can turn off the extrapolation and instead *copy* integration point values to the nodes, making those values available in the postprocessor.

ERESX,YES forces extrapolation for *all* elements, whether or not they have active material nonlinearities.

3.6.5. Setting Biot-Savart Options

These options are used in a magnetic field analysis:

Table 3.16: Biot-Savart Commands

Command	Purpose
BIOT	Calculates the magnetic source field intensity due to a selected set of current sources.

Command	Purpose
EMSYM	Duplicates current sources that exhibit circular symmetry.

For more information, see the *Low-Frequency Electromagnetic Analysis Guide*.

3.6.6. Setting Spectrum Options

Many commands are available in this category, all meant to specify response spectrum data and power spectral density (PSD) data. Use the commands in spectrum analyses, as described in the *Structural Analysis Guide*.

3.7. Creating Multiple Load Step Files

All loads and load step options put together form a *load step*, for which the program can calculate the solution. If you have multiple load steps, you can store the data for each load step on a file, called the *load step file*, and read it in later for solution.

The **LSPWRITE** command writes the load step file (one file per load step, identified as `Jobname.S01`, `Jobname.S02`, `Jobname.S03`, etc.).

After all load step files are written, you can issue one action command to read in the files sequentially and obtain the solution for each load step.

Example 3.6: Defining Multiple Load Steps

This set of commands defines multiple load steps:

```
/SOLU           ! Enter SOLUTION
0
! Load Step 1:
D, ...          ! Loads
SF, ...
...
NSUBST, ...    ! Load step options
KBC, ...
OUTRES, ...
OUTPR, ...
...
LSPWRITE        ! Writes load step file: Jobname.S01

! Load Step 2:
D, ...          ! Loads
SF, ...
...
NSUBST, ...    ! Load step options
KBC, ...
OUTRES, ...
OUTPR, ...
...
LSPWRITE        ! Writes load step file: Jobname.S02
0
```

The load step data are written to the file in terms of commands.

The **LSPRINT** command does not capture changes to real constants (**R**), material properties (**MP**), couplings (**CP**), or constraint equations (**CE**).

The **LSPRINT** command automatically transfers solid-model loads to the finite element model, so all loads are written in the form of finite-element load commands. In particular, surface loads are always written in terms of **SFE** (or **SFBEAM**) commands, regardless of how they were applied.

To modify data on load step file number n , issue the command **LSREAD**, n to read in the file, make the desired changes, and then issue **LSPRINT**, n (which overwrite the old file n). You can also directly edit the load step file using your system editor, but this is generally not recommended.

The **LSDELETE** command enables you to delete load step files from within the program.

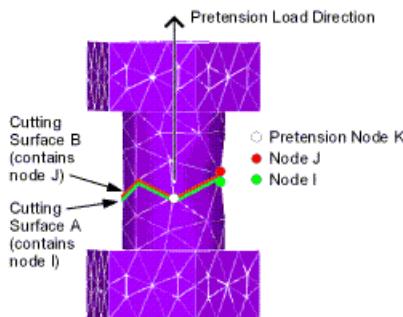
Another useful load-step-related command is **LSCLEAR**, which enables you to delete all loads and reset all load step options to their defaults. For example, you can use it to clean up the load step data before reading in a load step file for modifications.

3.8. Defining Pretension in a Joint Fastener

Preloads in bolts and other structural components often have significant effect on deflections and stresses. Two features, the **PRETS179** pretension element and the **PSMESH** pretension meshing command, can be used for this type of analysis. If the fastener has been meshed in two separate pieces, the pretension elements can be inserted between the pieces using the **EINTF** command.

The pretension load is used to model a pre-assembly load in a joint fastener. The fastener can be made up of any 2-D or 3-D structural, low- or high-order solid, beam, shell, pipe, or link elements. When using the **PSMESH** command, the pretension section, across which the pretension load is applied, must be defined inside the fastener.

Figure 3.21: Pretension Definition



3.8.1. Applying Pretension to a Fastener Meshed as a Single Piece

The easiest way to apply pretension elements to a fastener is via the **PSMESH** command. You can issue the command only if the fastener is *not* meshed in separate pieces. The command defines the pretension section and generates the pretension elements. It automatically cuts the meshed fastener into two parts and inserts the pretension elements. If you want to remove the pretension elements, you can do so automatically by deleting the pretension section. This feature also enables you to undo the cutting operation by merging nodes.

The normal direction is specified via the **PSMESH** command and is part of the section data.

The meshed pretension section does not need to be flat. The elements underlying the pretension section can have almost any shape: line, triangle, quadrilateral, tetrahedron, wedge, or hexahedron; however, there must be coincident nodes on the two sides (A and B) of the pretension section. Sides A and B on the pretension section are connected by one or more pretension elements, one for each coincident node pair.

Pretension node K is for controlling and monitoring the total tension loads. The pretension load direction of the pretension section can be specified relative to side A when the section is created. All pretension elements on a specific pretension section must use the same section, and must have the same pretension node K. Node K is the third position for the pretension element definition.

3.8.2. Applying Pretension to a Fastener Meshed as Two Pieces

If the fastener has been meshed in two separate pieces, the pretension elements (**PRETS179**) can be inserted between the pieces via **EINTF,TOLER,K**.

If *K* is not defined, the program creates it automatically.

You must define the element type ID and section properties before issuing the **EINTF** command. (See the **SECDATA** command for more information about using the PRETENSION section type.)

The connecting surfaces (A and B) must have matching mesh patterns with coincident nodes. If some node pairs between the two surfaces are not connected with pretension elements, the resulting analysis can be inaccurate.

3.8.3. Example: Pretension Analysis

The following example describes the typical procedure used to perform a pretension analysis using the **PSMESH** command.

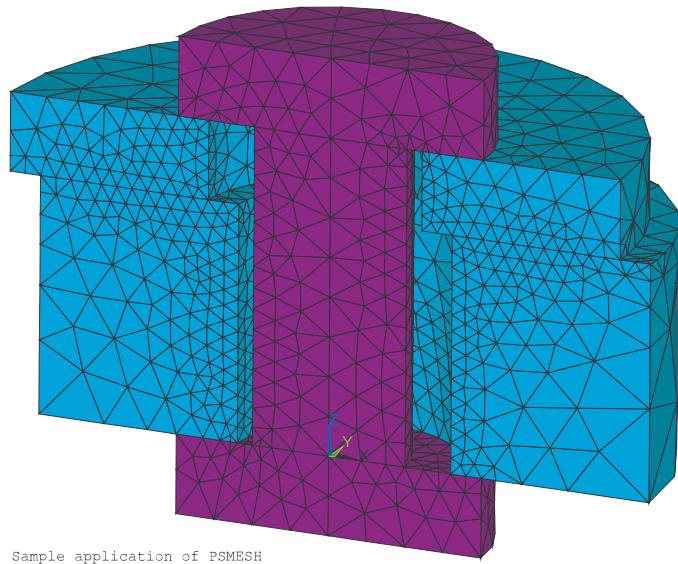
1. Mesh the bolt joint, then cut the mesh and insert the pretension elements to form the pretension section. For example, the following creates a pretension section called "example" by cutting the mesh and inserting the section into volume 1. A component is created as well (npts) that aids in plotting or selecting the pretension elements.

```
psmesh,,example,,volu,1,0,z,0.5,,,npts
```

2. In the first load step, apply a force or displacement to node K. In this case, the load is applied as a force. The force "locks" on the second load step, allowing you to add additional loads. The effect of the initial load is preserved as a displacement after it is locked. This is shown in the following example.

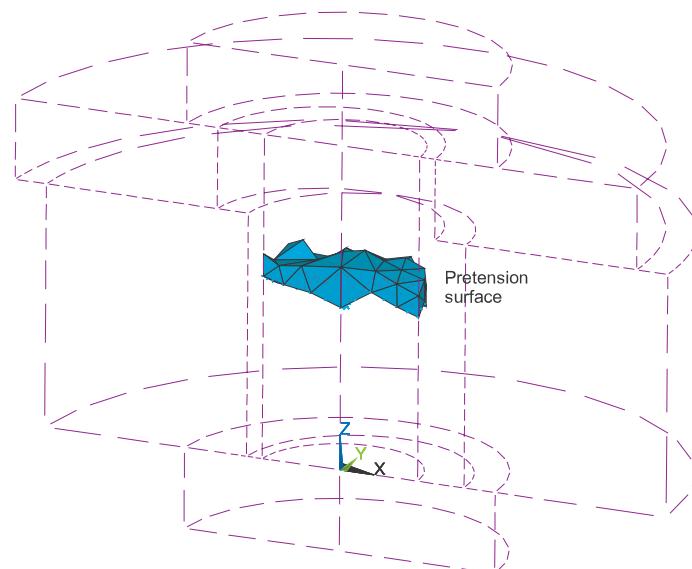
```
sload,1,PL01,tiny,forc,100,1,2
```

3. Apply other external loads as required using the **SLOAD** command.

Figure 3.22: Initial Meshed Structure

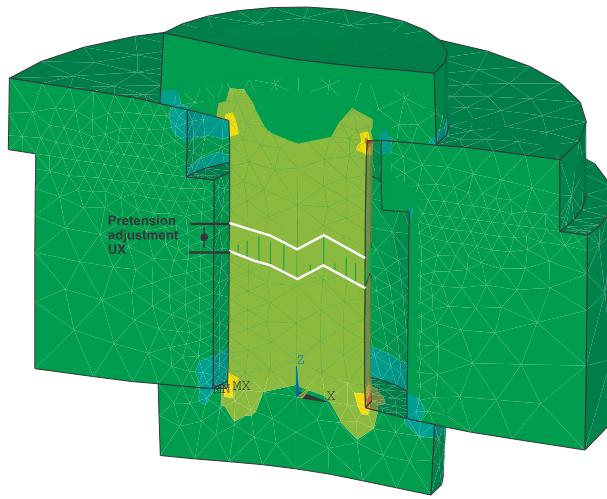
Sample application of PSMESH

The model represents a 180° slice of two annular plates and a single bolt assembled with an offset. The bolt is carbon steel, and the plates are aluminum. (See [Figure 3.22: Initial Meshed Structure \(p. 68\)](#).)

Figure 3.23: Pretension Section

Sample application of PSMESH

Use the **PSMESH** operation to separate the elements of the bolt into two unconnected groups, tied together with **PRETS179** pretension elements. We then plot the element and node components on the pretension interface. (See [Figure 3.23: Pretension Section \(p. 68\)](#).)

Figure 3.24: Pretension Stress

Apply constraints for symmetry and to prevent rigid body motion. The uniform temperature defaults to the reference temperature of 70°F. Apply half of the load (as this is a half model) to the pretension node created by **PSMESH**, solve, and plot the normal stress in the axial direction. As expected, the axial stress is tensile in the bolt, and compressive in the portion of the plates compressed by the bolt heads. (See [Figure 3.24: Pretension Stress \(p. 69\)](#).)

```

/prep7
et,1,187
mp,ex,1,1e7
mp,alpx,1,1.3e-5
mp,prxy,1,0.30
mp,ex,2,3e7
mp,alpx,2,8.4e-6
mp,prxy,2,0.30
tref,70

/foc,,-.09,.34,.42
/dist,..99
/ang,,-55.8
/view,..39,..87,.31
/pnum,volu,1
/num,1
cylind,0.5,, -0.25,0, 0,180
cylind,0.5,, 1,1.25, 0,180
cylind,0.25,, 0,1, 0,180
wpoff,.05
cylind,0.35,1, 0,0.75, 0,180
wpoff,.-1
cylind,0.35,1, 0.75,1, 0,180
wpstyle,,,...,0
vglue,all
numc,all
vplot
mat,1
smrt,off
vmesh,4,5
mat,2
vmesh,1,3
/pnum,mat,1
eplot
psmesh,,example,,volu,1,0,z,0.5,,,elems
cm,lines,LINE

```

```

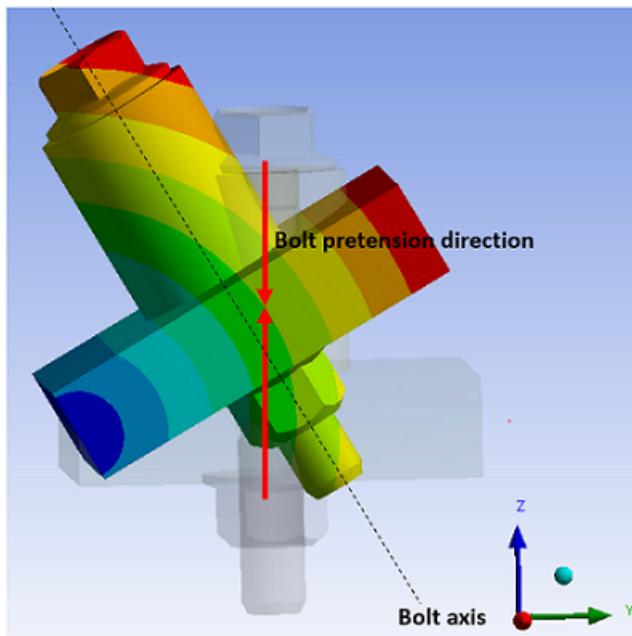
/dist,,1.1
cmplot
/solu
eqslve,pcg,1e-8
asel,s,loc,y
da,all,symm
asel,all
dk,1,ux
dk,12,ux
dk,1,uz
rescontrol,linear,all,1
sload,1,PL01,tiny,forc,100,1,2
/title,Sample application of PSMESH - preload only
solve
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Finally, we construct the actual solution of interest. We want to
! know what happens to the preload in the bolt, and the stress field around
! it, when the assembly temperature rises to 150 degrees F.
! Both the preload and the stresses increase because, for a uniform
! temperature rise, there is greater thermal expansion in the aluminum plates
! than in the steel bolt. Any method for applying preload that did not
! allow the load to change would be unable to predict this result.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
/post1
/show,JPEG
plnsol,s,z
prnsol,s,COMP
/show,CLOSE
/solu
antype,restart
tunif,150
/title,Sample application of PSMESH - uniform 150 degrees
solve
/post1
/show,JPEG
plnsol,s,z
prnsol,s,COMP
/show,CLOSE
/com,*           CHECK HERE          *

```

3.9. Defining Preload in a Joint Fastener Undergoing Large Rotation

Preloads in bolts and other structural components can often be modeled using the [PRETS179](#) pretension element, as described in [Defining Pretension in a Joint Fastener \(p. 66\)](#). With this method, the pretension direction remains in the undeformed axial direction of the bolt. If the bolt experiences a large rotation, the bolt preload direction is not updated with respect to the current bolt axis (see figure below). For a large rotation angle, the error in the preload direction may cause significant error in the results.

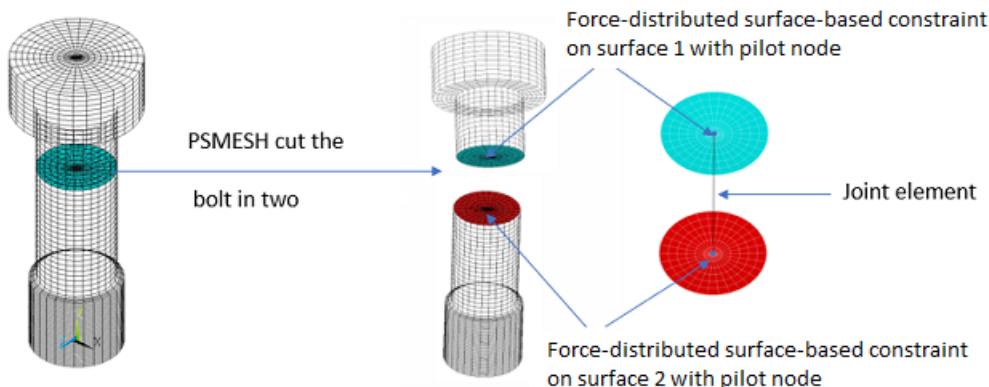
As an alternative to the pretension element, you can use the [MPC184](#) joint element instead to define a preload section in a joint fastener that undergoes large rotation. The [MPC184](#) element follows the current bolt axis during the large rotation; therefore, errors in the results are negligible. Torque and rotation around the bolt axis may be applied to the joint element.

Figure 3.25: Bolt Pretension, Torque, and Rotation Around the Bolt Axis

3.9.1. Generating a Preload Section in a Fastener Meshed as a Single Piece

The easiest way to apply a preload section to a fastener is via **PSMESH**. You can use the command only if the fastener is *not* meshed as separate pieces.

PSMESH automatically cuts the meshed fastener into two parts and generates the necessary components of the preload section. This typically includes two **force-distributed surface-based constraints** (remote points) on the cutting surfaces and an **MPC184** joint element that connects the two pilot nodes of the force-distributed constraints. (If the joint is between beam elements, no force-distributed constraints are generated.) The command also creates a local Cartesian coordinate system at the first node of the joint element to define the normal direction and saves it as part of the section data.

Figure 3.26: Force Distribution on Two Surfaces After Cutting with PSMESH

The elements are created as follows:

1. **Contact Pairs:** Two MPC contact pairs (surface-based constraints) are created on the cutting surfaces resulting from the split performed by **PSMESH**. The program creates **CONTA174** (3-D) or **CONTA172** (2-D) elements on each cutting surface along with a pilot node meshed with either a **TARGE170** (3-D) or a **TARGE169** (2-D) element at the center point of the contact surface. Appropriate contact and target element type IDs are generated, and a unique real constant ID is assigned to each contact pair. See **Table 3.17: Surface-Based Constraints Generated by PSMESH (p. 72)** for possible combinations of elements and the contact element KEYOPT settings.
2. **Joint Element:** A joint element is created to connect surface 1 to surface 2. The two end-nodes of the joint element are the pilot nodes of the two surface-based constraints.

The type of joint element created depends on the *PSTYPE* argument of **PSMESH**:

- If *PSTYPE* is specified as TORQUE, the program creates a **cylindrical joint element** (MPC184 with KEYOPT(1) = 11) to represent the preload section as follows:
 - **For a 2-D model:** An x-axis cylindrical joint element is generated along with two force-distributed surface-based constraints. A local Cartesian coordinate system is created at the first node of the joint element such that the local x- axis is the axis of that element (KEYOPT(4) = 0 for the **MPC184** element).
 - **For a 3-D model:** A z-axis cylindrical joint element is generated along with two force-distributed surface-based constraints. A local Cartesian coordinate system is created at the first node of the joint element such that the local z- axis is the axis of that element (KEYOPT(4) = 1 for the **MPC184** element).
 - **For a 3-D model that contains beam elements:** A z-axis cylindrical joint element is generated between the endpoints of two beam elements. (No force-distributed surface-based constraints are needed.) A local Cartesian coordinate system is created at the first node of the joint element such that the local z- axis is the axis of that element (KEYOPT(4) = 1 for the **MPC184** element).
- If *PSTYPE* is specified as a negative value, the program creates a **screw joint element** (MPC184 with KEYOPT(1) = 17) to represent the preload section. The absolute value of *PSTYPE* is used as the pitch value for the screw joint. This option is only valid for 3-D models. Two force-distributed surface-based constraints are generated at the cutting surfaces, except for the case of a beam model which does not need the force-distributed constraints. A local Cartesian coordinate system is created at the first node of the joint element such that the local z- axis is the axis of that element.

For all cases, a local Cartesian coordinate system is created to define the joint axis. You can change this local coordinate system with the **SECJOINT** command after the **PSMESH** command is issued. The new coordinate system you specify must be Cartesian.

Table 3.17: Surface-Based Constraints Generated by PSMESH

Underlying Elements	Contact Element	Target Element	Contact Element KEYOPT Settings
SOLID185, SOLID186, SOLID187, SOLID285	CONTA174	TARGE170	KEYOPT(2) = 2 (MPC approach) KEYOPT(12) = 5 (bonded always)

Underlying Elements	Contact Element	Target Element	Contact Element KEYOPT Settings
			KEYOPT(4) = 1 (Force-distributed constraint)
PLANE182, PLANE183	CONTA172	TARGE169	KEYOPT(2) = 2 (MPC approach) KEYOPT(12) = 5 (bonded always) KEYOPT(4) = 1 (Force-distributed constraint)
BEAM188, BEAM189	For beam elements, the joint element nodes are connected to the beam end nodes, and no force-distributed constraints are generated for this case.		

Considerations and Restrictions for Preload Sections

The following considerations and restrictions apply to preload sections that use an [MPC184](#) joint element:

- If a preload section is applied on a symmetric geometry model instead of the full geometry model, you must define the symmetry conditions using KEYOPT(6) of the target elements ([TARGE169](#) or [TARGE170](#)) for the force-distributed constraints generated by [PSMESH](#). Otherwise, you might get unexpected results. You must also modify the locations of the pilot nodes of the force-distributed constraints so they are at the symmetry plane/edge. For example:

```
psmesh, ,1000,all,,0,X,6,,,torque
nmodif,273,6,0,0      ! Pilot node moved to symmetry plane
nmodif,274,6,0,0      ! Pilot node moved to symmetry plane
keyopt,3,6,110         ! Symmetry condition with respect to the xy and xz planes
```

- The displacement adjustment for the [MPC184 cylindrical joint element](#) is reported as joint relative displacement and is accessed as an [ETABLE](#) quantity. The output item is JRU1 (SMISC,61) for the x-axis cylindrical joint and JRU3 (SMISC,63) for the z-axis cylindrical joint.
- All restrictions documented in the [Element Reference](#) for the [MPC184 cylindrical joint element](#) and [MPC184 screw joint element](#) also apply when these elements are used as part of a preload section.
- Use of the Lagrange multiplier method (KEYOPT(2) = 3 on the contact element) with the force-distributed constraint is not supported for a preload section.
- Joint element preload sections are not supported for shell element models.
- Joint element preload sections are not supported in [cyclic symmetry](#) analyses.
- Joint element preload sections are not supported in [linear perturbation](#) analyses.

3.9.2. Applying a Preload to the Joint Element

Use the **DJ** and **FJ** commands to apply preloads to the joint element. Load application for the 2-D and 3-D cases are described below.

Note that if a moment load is not applied to the joint element, the rotational degrees of freedom must be suitably constrained.

2-D Case:

A pretension load in the x-direction (FX) can be applied for an x-axis cylindrical joint. ROTX should be constrained as no moment (MX) is applied about the joint axis. The following example shows the typical command sequence.

Load: Apply an FX force as a proload with the **FJ** command. Define the force load in the opposite direction of the joint axis to create pretension in the bolt.

```
fj,all,fx,-12345.67*2
dj,all,rotx,0
```

Adjustment: Apply a displacement as a pre-adjustment with the **DJ** command.

```
dj,all,ux,-.1
```

Lock: Fix all displacements using the %_FIX% option of **DJ**. You may set this state for any load step, except the first one.

```
dj,all,ux,%_FIX%
```

3-D Case:

A pretension load in the z-direction (FZ) and a moment about the joint axis (MZ) can be applied to both the z-axis cylindrical joint and the screw joint. ROTZ should be constrained if an MZ moment is not applied. The following example shows the typical command sequence.

Load: Apply an FZ force as a proload with the **FJ** command. Define the force load in the opposite direction of the joint axis to create pretension in the bolt.

```
fj,all,fz,-12345.67*2
fj,all,mz,10000
```

or

```
fj,all,fz,-12345.67*2
dj,all,rotz,0      ! Constrain ROTZ if no moment is applied
```

Adjustment: Apply a displacement as a pre-adjustment with the **DJ** command.

```
dj,all,uz,-.1
dj,all,rotz,1.04
```

Lock: Fix all displacements using the %_FIX% option of **DJ**. You may set this state for any load step, except the first one.

```
dj,all,uz,%_FIX%
dj,all,rotz,%_FIX%
```

Use **FJDELETE** to delete the previous load step force before applying the incremental adjustment with **DJ**.

Use **DJDELETE** to delete the previous load step constraints before applying the incremental load with **FJ**.

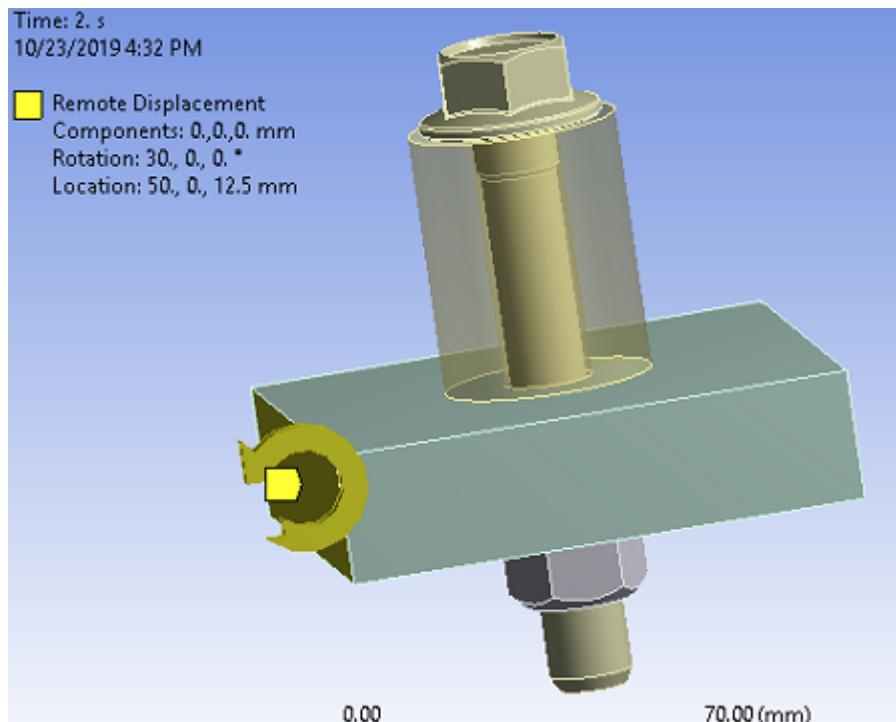
3.9.3. Example: Bolt Sleeve Model Undergoing Large Rotation

The following example describes the procedure to apply preload to a bolt sleeve model undergoing a large rotation.

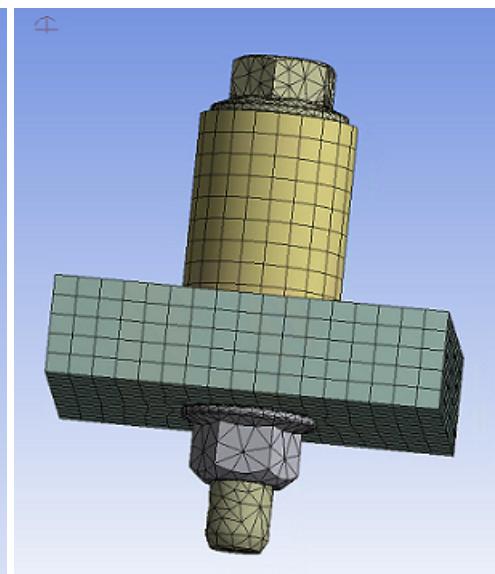
Bolt Sleeve Model

Time: 2. s
10/23/2019 4:32 PM

Remote Displacement
Components: 0,0,0. mm
Rotation: 30, 0, 0.
Location: 50, 0, 12.5 mm

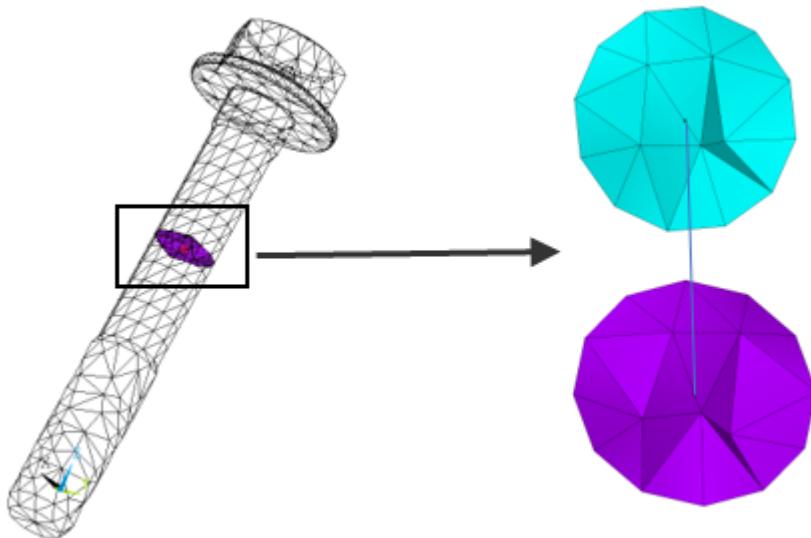


Initial Meshed Structure



- After meshing the bolt sleeve model, **PSMESH** is used to cut the mesh of the selected bolt elements into two parts and generate the required elements. Two force-distributed surface-based constraints are created (one on each surface), and an **MPC184** cylindrical joint element is connected to the two pilot nodes. For this example, a local coordinate system is defined at the cutting location, and the bolt is cut in the z-direction. By default, **KEYOPT(4) = 1** is set for the **MPC184** element to define the z-axis cylindrical joint.

```
local,12,0,50.0026090261661,50.0000009547169,40.,-180.,0.,180.
nsel,s,,,bolt_node
esln
psmash,19,,,all,,12,z,0,,,torque
allsel,all
```

Figure 3.27: Two Force-Distributed Constraints with Joint Element at Cutting Location

2. In the first load step, you typically apply a force or a displacement to the joint element with the **FJ** or **DJ** command. In this case, the load is applied as a force and a moment in the z-direction. A negative FZ value is used to create the effect of pretension. The force "locks" on the second step, allowing additional loads. The effect of the initial load is preserved as a displacement after it is locked. The full model is rotated around the x-axis 30 degrees to the remote point.

```

time,1
esel,s,sec,,19
fj,all,fz,-146100.      ! Preload is applied as FZ
esel,all
esel,s,sec,,19
fj,all,mz,170500.      ! Preload is applied as MZ
allsel,all
solve
time,2
esel,s,sec,,19
fjdele,all,all
dj,all,uz,%_fix%        ! FZ is redefined as UZ
dj,all,rotz,%_fix%      ! MZ is redefined as ROTZ
allsel,all
solve

```

Results

Figure 3.28: UZ Displacement

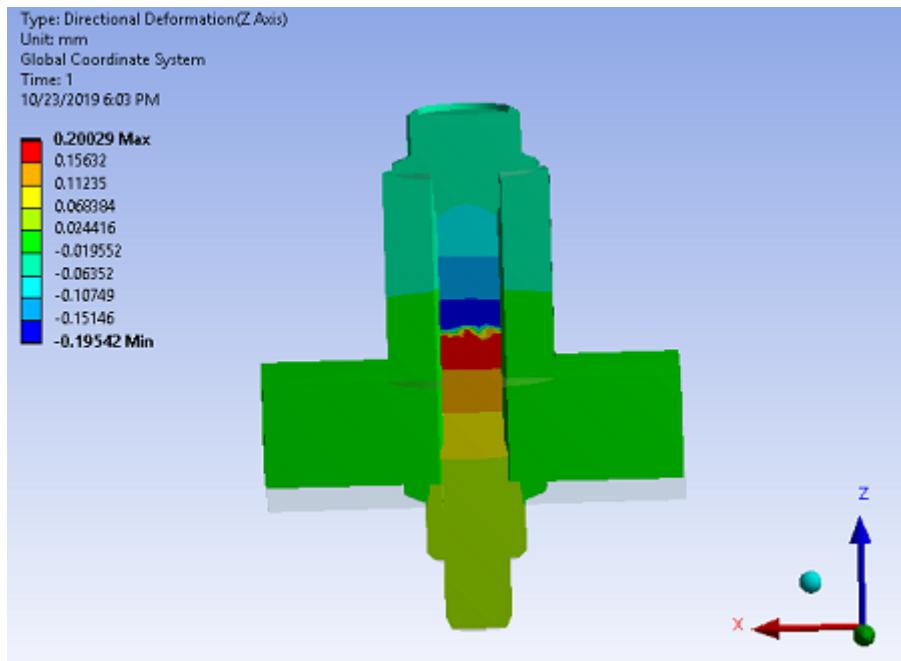
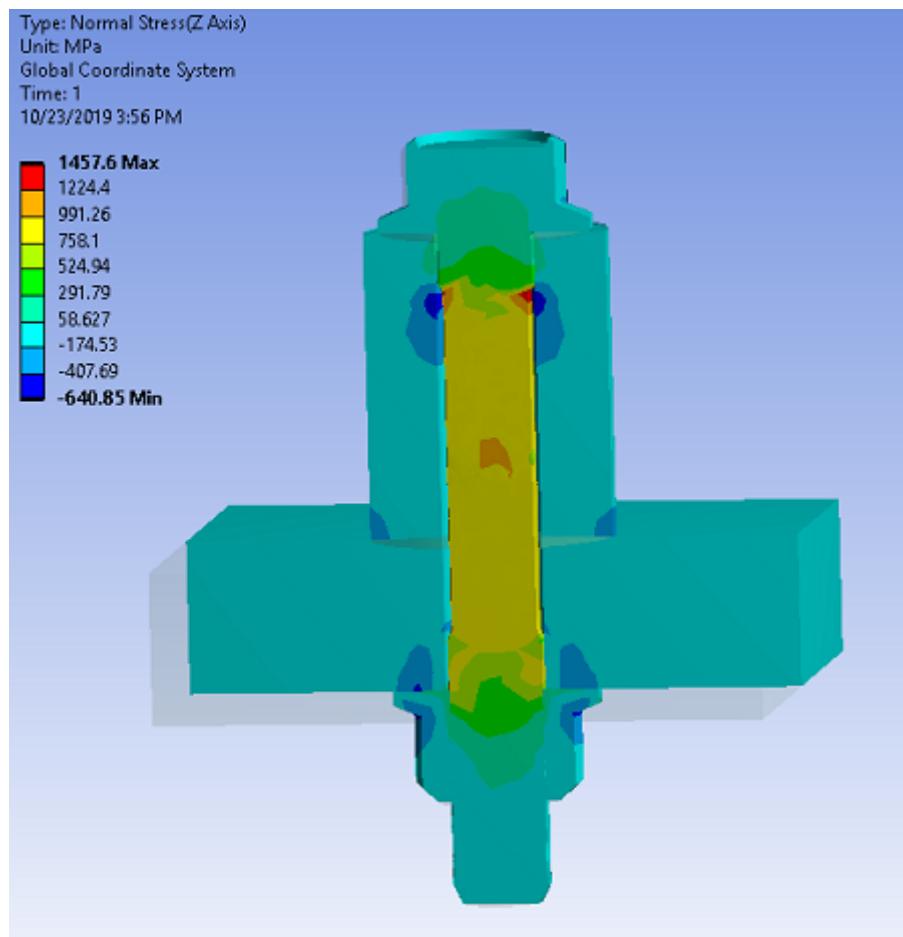
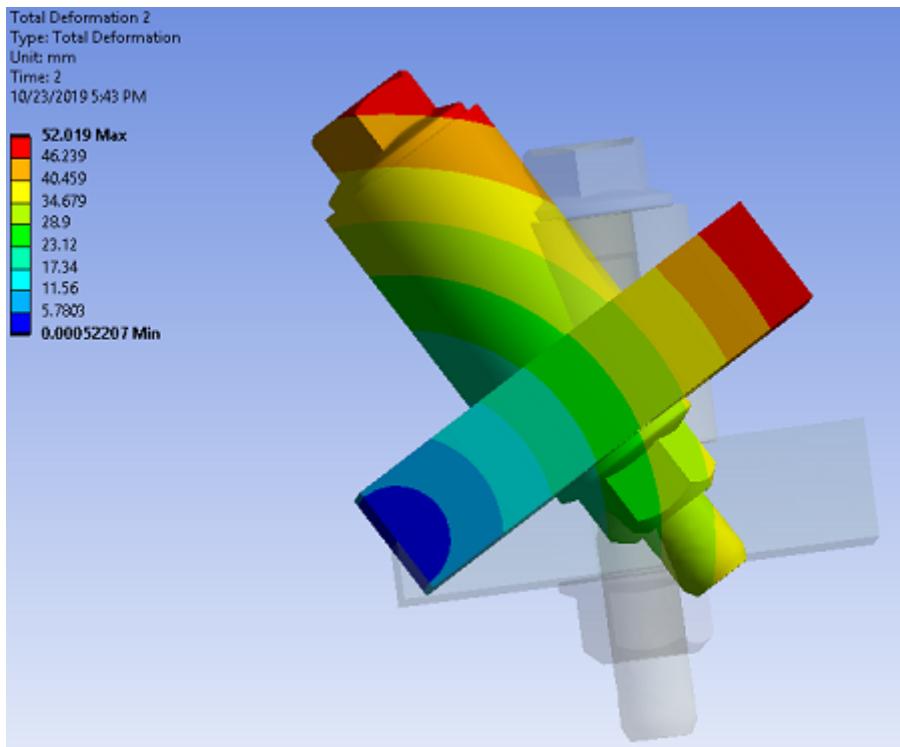
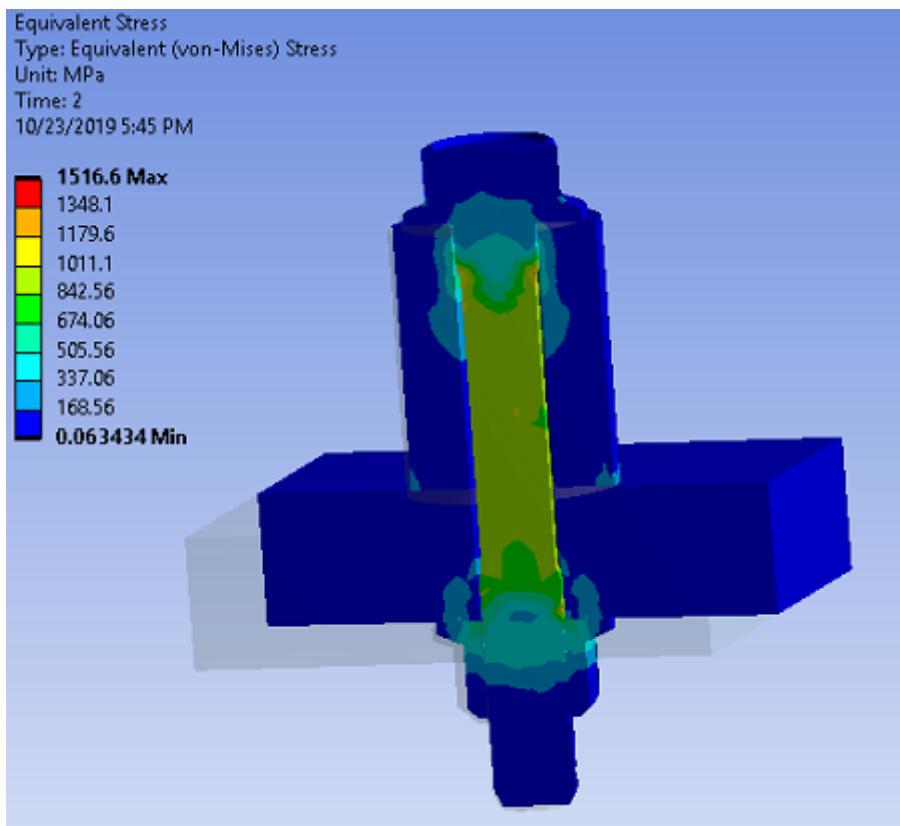


Figure 3.29: Normal Stress Plot in the Z-direction

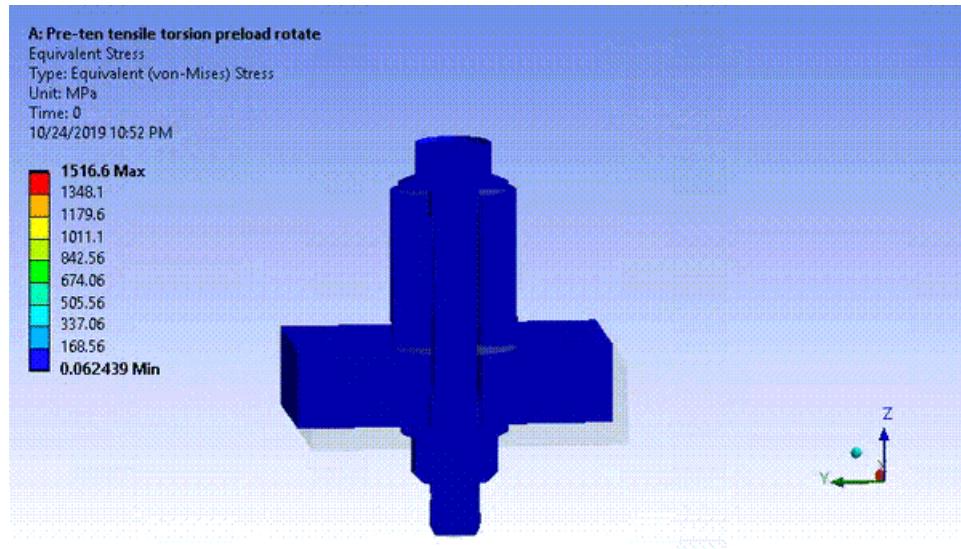


As expected, the axial stress is tensile in the bolt and compressive in the portion of the plates compressed by the bolt heads.

Figure 3.30: Deformed Shape at Last Load Step**Figure 3.31: von-Mises Stress at Last Load Step**

\

Figure 3.32: Animation of von-Mises Stress



Chapter 4: Using the Function Tool

The Function Tool enables you to define a dependent variable as a function of one or more independent variables. Using the Function Tool, you can define complicated boundary conditions on a model, or the nonlinear material behavior for a [joint element](#).

Example 1

Suppose that the applied displacement at a node of the model is a function of temperature and velocity. The function is defined as follows:

$$u = (-0.007 * T + 0.50) V_r$$

where T is the temperature and V_r is the relative velocity.

You can input the function, thereby specifying the boundary condition at that node.

Example 2

Suppose the nonlinear damping force characteristics in a joint element varies quadratically with temperature and linearly with velocity. The function is defined as follows:

$$F = f(T) V_r$$

or

$$F = (C_1 T^2 + C_2 T + C_3) V_r$$

where C_1 , C_2 , and C_3 are constants, T is the temperature, and V_r is the relative velocity.

You can input the function along with the constant values, thereby incorporating the damping characteristics by specifying a nonlinear force that varies with relative velocity and temperature.

The following Function Tool topics are available:

- [4.1. Function Tool Terminology](#)
- [4.2. Using the Function Editor](#)
- [4.3. Using the Function Loader](#)
- [4.4. Applying Boundary Conditions Using the Function Tool](#)
- [4.5. Function Tool Example](#)
- [4.6. Graphing or Listing a Function](#)

For more information, see [Specifying a Function Describing Nonlinear Stiffness Behavior](#) in the [Material Reference](#).

4.1. Function Tool Terminology

The Function Tool has two components:

- **Function Editor (p. 82)** -- Creates functions.
- **Function Loader (p. 85)** -- Retrieves the functions and loads them as table arrays.

The following terms apply when using the Function Tool:

- **Function** -- A set of equations that together define an advanced boundary condition.
- **Primary Variable** -- An independent variable evaluated and used by the program during solution.
- **Regime** -- A portion of an operating range or design space characterized by a single regime variable. Regimes are partitioned according to lower and upper bounds of the regime variable. The regime variable must be continuous across the entire regime. Each regime contains a unique equation to evaluate the function.
- **Regime Variable** -- The defining variable that governs which of the set of equations is used to evaluate the function.
- **Equation Variable** -- A dependent (user-specified) variable, defined when the function is loaded.

4.2. Using the Function Editor

The Function Editor defines an equation or a function (a series of equations). You use a set of [primary variables \(p. 83\)](#), equation variables, and mathematical functions to build the equations. Each equation applies to a particular regime. The equations defined for each regime, taken together, define a function, and the function as a whole is applied (for example, as a boundary condition, or to define the nonlinear material behavior for a joint).

The following topics related to the Function Editor component of the Function Tool are available:

- 4.2.1. [How the Function Editor Works](#)
- 4.2.2. [Creating a Function with the Function Editor](#)
- 4.2.3. [Using Your Function](#)

4.2.1. How the Function Editor Works

Using the Function Editor is similar to using a scientific calculator. For example, when building an equation, you can:

- **Click buttons on the on-screen keypad.**

The keypad includes the numbers 0-9, parentheses, and a set of mathematical operators. In addition to the default set of operators, you can also click the INV key to access an alternate set of operators.

- **Use any variable name.**

The editor interprets any variable name you type as an equation variable. You can use up to 10 user-defined equation variables in a function (up to six regimes). You can use any name you wish, but ANSYS, Inc. recommends against using the same name as one of the primary variables. You define the values for these variables when you *load* the function (described in [Using the Function Loader \(p. 85\)](#)).

- **Select a primary variable (p. 83) from a drop-down list.**

As you build an equation, it appears in standard mathematical syntax in the equation box above the keypad. The various components (primary variables, equation variables, mathematical operators, and numbers) appear in different colors so that you can more easily verify the equation you are entering. You can also graph or list the equation using the **GRAPH/LIST** button in the **Function Editor** dialog box; see [Graphing a Function \(p. 92\)](#) for more information about this feature.

Ensuring the Validity of Your Equation

The Function Editor does not validate the equation construction. (ANSYS generates an error message if you enter an inappropriate equation construction.) You must also ensure the *mathematical* validity of any equation.

Hint: A common error is a divide-by-zero scenario. Another common problem is a negative primary variable; in such a case, multiply the primary variable by -1.

Saving and Retrieving Your Equation

If you intend to use an equation or part of an equation later in the function (such as in another regime), click the **STO** button to store it. The numbers on the keypad change to a series of memory buffers. Click one of them to store the equation in that memory buffer. *Example:* To store your equation in the **Memory1** buffer, click **STO** and then **M1**.)

To retrieve a stored equation, click **INV** and then **INS MEM**, followed by the appropriate memory button. The contents of that memory buffer are then displayed in the equation box. You can also recall an abbreviated form of the contents by clicking **RCL**. If you pause the cursor over a memory button, a tool tip displays the contents of that memory buffer.

4.2.1.1. Selecting Primary Variables in the Function Editor

You can select from among the available primary variables in the Function Editor's drop-down list. Primary variables marked with an asterisk (*) are also available for tabular boundary conditions (BCs). The remaining primary variables are appropriate for use with function BCs only.

- Time* (TIME)
- X location* (X) in local global coordinates
- Y location* (Y) in local global coordinates
- Z location* (Z) in local global coordinates
(coordinate system applicability is determined by the ***DIM** command)
- Temperature* (TEMP degree of freedom)

- Fluid temperature (TFLUID) (computed fluid temperature in **FLUID116** elements for **SURF151** or **SURF152** elements)
- Velocity* (VELOCITY) (magnitude of the Velocity degrees of freedom or the computed fluid velocity in **FLUID116** elements)
- Applied surface pressure* (PRES)
- Tsurf* (TS) (element surface temperature for **SURF151** or **SURF152** elements)
- Density (ρ) (material property DENS)
- Specific heat (material property C)
- Thermal conductivity (material property kxx)
- Thermal conductivity (material property kyy)
- Thermal conductivity (material property kzz)
- Viscosity (material property μ)
- Emissivity (material property ϵ)
- Reference location* (Xr) (ALE formulations only)
- Reference location* (Yr) (ALE formulations only)
- Reference location* (Zr) (ALE formulations only)
- Contact pressure (PRESSURE) (used only to define certain real constants for contact elements **CONTA172**, **CONTA174**, **CONTA175**, **CONTA177**, and **CONTA178**)
- Geometrical contact gap/penetration (GAP) (used only to define certain real constants for contact elements **CONTA172**, **CONTA174**, **CONTA175**, **CONTA177**, and **CONTA178**)
- Rotational speed (OMEGS) (rotational speed for **SURF151**, **SURF152**, and **COMBI214** elements)
- Rotational speed (OMEGF) (rotational speed for **FLUID116** elements)
- Slip factor (SLIP) (slip factor for **FLUID116** elements)
- Tabular data as a function of frequency of excitation (FREQ)
- Relative displacement (DJU)
- Relative velocity (DJV)

4.2.2. Creating a Function with the Function Editor

Access the Function Editor via the GUI in either of the following ways:

- **Main Menu> Solution> Define Loads> Apply> Functions> Define/Edit**
- **Utility Menu> Parameters> Functions> Define/Edit**

Create a function as follows:

1. Select the function type. Select either a single equation or a multivalued function. If you select the latter, you must type in the name of your regime variable. This is the variable that governs the equations in the function. When you select a multivalued function, the six regime tabs become active.
2. Select degrees or radians. This setting determines only how the equation is evaluated and has no effect on ***AFUN** settings.
3. Define the result equation (if a single equation) or the equation describing the regime variable (if a multivalued function) using [primary variables \(p. 83\)](#), equation variables, and the keypad. If you are defining a single-equation function, go to Step 10 to comment and save the equation. If you are defining a multivalued function, continue with Step 5.
4. Select the **Regime 1** tab. Type in the appropriate lower and upper limits for the regime variable you defined under the **Function** tab.
5. Define the equation for this regime.
6. Select the **Regime 2** tab. Notice that the lower limit for the regime variable is already defined and unchangeable. This feature ensures that the regimes remain continuous, with no gaps. Define the upper limit for this regime.
7. Define the equation for this regime.
8. Continue this process for up to six regimes. You do not have to store or save the individual equations in each regime, unless you wish to reuse the equation in another regime.
9. *Optional:* Enter a comment to describe the function. Select **Editor> Comment** and type your comment in the area provided.
10. Save the function. Select **Editor> Save** and type in a name. The filename must have a **.func** extension.

4.2.3. Using Your Function

After you have defined and saved your function, you can use it in any applicable ANSYS analysis, and any other ANSYS user with access to the file can use it. For example, you could create a corporate library of functions and place them in a common directory that all users can access via a network.

To use the function, you must load it, assign values to any equation variables, and provide a table parameter name for use in a given analysis. Functions are stored in a table array in equation format, not as discrete table values. All of these tasks occur via the [Function Loader \(p. 85\)](#).

4.3. Using the Function Loader

When you are ready to apply specific values to the equation variables, specify a table parameter name, and use the function in an analysis, you must load the function into the Function Loader.

Access the Function Loader via the ANSYS GUI in either of the following ways:

- **Main Menu> Solution> Define Loads> Apply> Functions> Read file**
- **Utility Menu> Parameters> Functions> Read from file**
 1. Navigate to the directory where you saved the function, select the appropriate file, and open it.
 2. In the **Function Loader** dialog box, enter a table parameter name. This is the name you will use (%tabname%) when you specify this function as a tabular boundary condition.
 3. On the bottom half of the dialog box, you will see a **Function** tab and a **Regime** tab for each regime defined for the function. Select the **Function** tab. You will see a data entry area for each equation variable you specified. You will also see a data entry area for material IDs if you used any variable that requires a material ID. Enter the appropriate values in these data entry areas.

Note:

Only numeric data is supported for the constant values in the Function Loader dialog box. Character data and expressions are not supported as constant values.

4. Repeat the process for each regime you defined.
5. Click **Save**. You will not be able to save this as a table array parameter until you have provided values for all variables in all regimes in the function.

After you have saved the function as a named table array parameter using the Function Loader, you can apply it as a tabular boundary condition. See [Applying Loads Using Tabular Input \(p. 51\)](#) for detailed information on using tabular boundary conditions in your analysis.

The function is loaded into the table as a coded equation. This coded equation is processed in ANSYS when the table is called for evaluation.

4.4. Applying Boundary Conditions Using the Function Tool

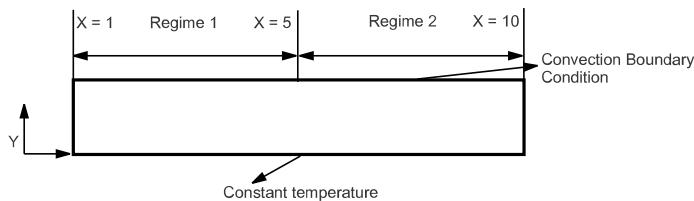
If your data can be conveniently expressed as a table, ANSYS recommends using tabular boundary conditions. ANSYS applies [function boundary conditions \(p. 86\)](#) to a model using the tabular boundary condition process described in [Applying Loads Using Tabular Input \(p. 51\)](#). You must define your function and load it as a table array *before* you try to add it as a load.

You cannot use function boundary conditions to circumvent the restrictions on boundary conditions and their corresponding [primary variables \(p. 83\)](#) as supported by tabular boundary conditions. For example, in a structural analysis, the primary variables supported with a pressure load are TIME, X, Y, Z, and TEMP; therefore, when using a function boundary condition, the only primary variables allowed in the equation are TIME, X, Y, Z, and TEMP. The list in [Using the Function Editor \(p. 82\)](#) shows which primary variables are available for each type of operation.

4.5. Function Tool Example

The following example shows how to create and apply a boundary condition using a function representation.

The convection heat transfer coefficient from a fluid flowing over a flat plate is applied as a function boundary condition, using the correlation for laminar heat transfer coefficient. The figure below shows the flat plate with the applied boundary conditions.



The bottom of the plate is fixed at a constant temperature. The top of the plate, where the convection boundary condition is being applied, is split into two regimes:

Regime 1 is defined for X between $1 \leq X < 5$, and the convection heat transfer coefficient is given by:

$$h(x) = 0.332 * (k_{xx}/x) * Re^{(1/2)} * Pr^{(1/3)}$$

Regime 2 is defined for X between $5 < X \leq 10$, and the convection heat transfer coefficient is given by:

$$h(x) = 0.566 * (k_{xx}/x) * Re^{(1/2)} * Pr^{(1/3)}$$

In the above equations, the Reynolds number Re is given by:

$$Re = (\text{dens} * \text{vel} * x) / \text{visc}$$

and the Prandtl number PR is given by:

$$Pr = (\text{visc} * c) / k_{xx}$$

The properties of the fluid over the flat plate are:

Density (dens) = 1, thermal conductivity (k_{xx}) = 10, specific heat (c) = 10, and viscosity (visc) = 0.01

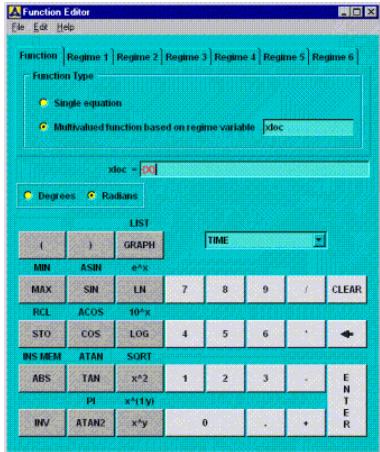
The velocity of the fluid (vel) over the flat plate is equal to 100 for Regime 1 and 50 for Regime 2. Bulk temperature for the fluid for both regimes is 100 degrees.

1. Create a rectangle and assign element type **PLANE55**, define your material properties, and mesh:

```
/prep7
rect,1,10,.,5
et,1,55
!Define Fluid Properties
mp,KXX,1,10 !Thermal conductivity
mp,DENS,1,1 !Density
mp,C,1,10 !Specific heat
mp,VISC,1,0.01 !Viscosity
!Define Plate Properties
mp,kxx,2,10
mp,dens,2,10
mp,c,2,5
mat,2
esize,.25
amesh,all
```

2. Define the convection boundary condition as a function.

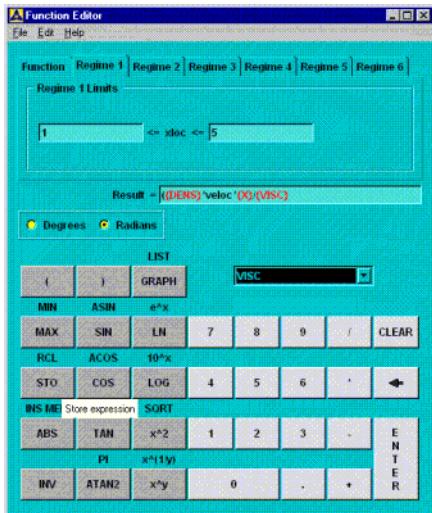
Select **Utility Menu> Parameters> Functions> Define/Edit** to bring up the function editor. The function boundary condition being applied is a multivalued function, its final value being dependent on the X location in the domain. In the **Function Editor** dialog box, click the radio button for "Multivalued function based on regime variable" and type *xloc* as the name of the regime variable in the text entry box. The name *xloc* appears as the name of the regime variable. To define *xloc*, select "X" from the drop-down box on the lower half of the dialog box. Your dialog box should look like this:



3. Define the equations for the heat transfer coefficient in the two regimes. Select the **Regime 1** tab. Under this tab, you will define the equation for the first regime, $1 \leq X \leq 5$. Type "1" and "5" in the Regime 1 Limits text entry boxes.
4. For the sake of convenience, define those expressions in the equations that you will use more than once or that are part of a very long equation, and store them in memory.

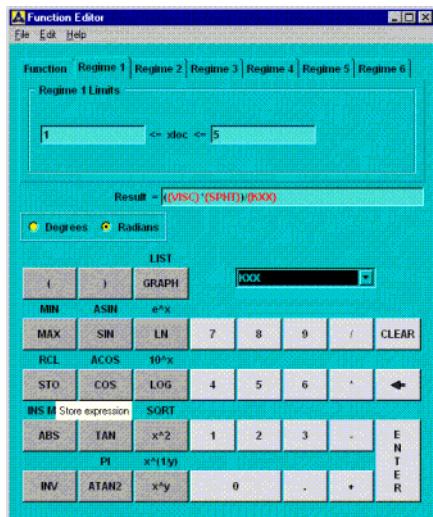
In this example, expressions for the Reynold's number and Prandtl number are used repeatedly in both equations. They are good examples of expressions that can be stored and used throughout the function editor, in all regimes.

To store the Reynold's number, fill in the Result box as shown below. Select the primary variables DENS, X, and VISC (shown in {brackets}) from the drop-down list on the lower half of the dialog box. Use the keypad to insert the math functions such as * and /. Your dialog box should look like this:



Click **STO**, then **M0** on the number pad to store the expression in memory location 0.

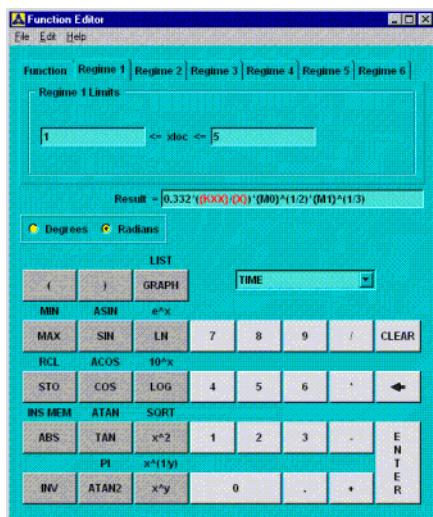
To store the Prandtl number, clear the Results box by clicking the **Clear** button and then fill it again as shown below. Select the terms VISC, SPHT, and KXX from the drop-down list. Your dialog box should look like this:



Click **STO**, then **M1** on the number pad to store the expression in memory location 1.

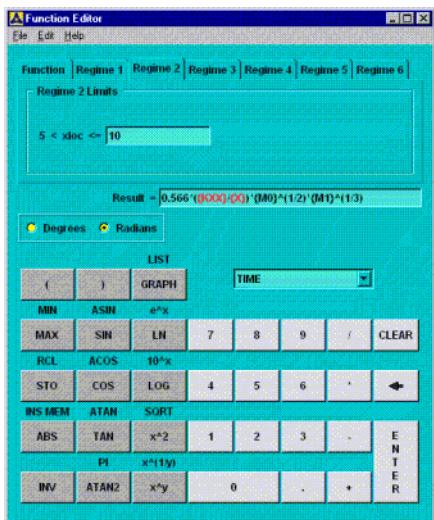
- Define an expression for the heat-transfer coefficient for Regime 1.

Click **Clear** to clear the contents of the text entry box. Type in the expression for the heat transfer coefficient for Regime 1 as shown below. Select the primary variables (**{KXX}**) and (**X**) from the drop-down list. The terms **M0** and **M1** are the terms you stored in memory earlier. To place them in the equation, click the **INV** button, and then **RCL**, then **M0** and **M1** respectively.



- Define the equation for Regime 2.

Select the **Regime2** tab. First, enter "10" as the upper limit for the regime variable for which this equation is valid. Notice that the lower limit for this regime is already set as the upper limit from Regime 1. This feature ensures continuity between the regimes. Type in the expression for the heat transfer coefficient as shown below. You can use the same stored memory locations **M0** and **M1** to replace expressions for Reynold's number and Prandtl number, respectively. Your dialog box should look like this:



7. *Optional:* Enter comments for this function.

Select **File> Comments**.

8. Save the function.

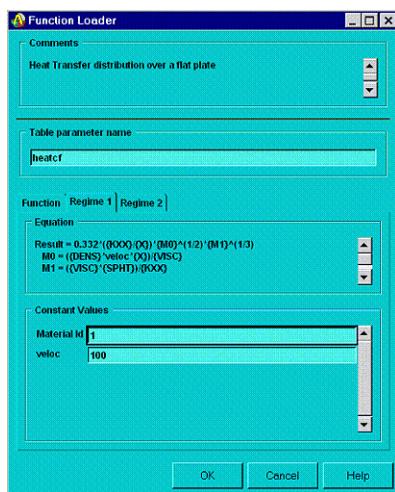
Select **File> Save**. Functions are saved with a .func extension.

You must save the function. After you have saved the function, you can then load it as a table parameter into ANSYS.

9. Load the function. Select **Utility Menu> Parameters> Functions> Read from File**. Select the .func file that you saved earlier. The **Function Loader** dialog box appears.
10. Provide a table parameter name that you will use when applying the function as a boundary condition.

Type "heatcf" for this example. (The parameter name cannot contain more than seven characters.) Provide values for any variables that you defined in the Function Editor.

Select the **Regime 1** tab and enter 1 for the material ID (to obtain the material primary variables) and 100 for the velocity. (The Function Tool prompts you for the material ID only if you have used a material property in your expression.) Your dialog box should look like this:



Note:

Only numeric data is supported for the constant values in the Function Loader dialog box. Character data and expressions are not supported as constant values.

11. Select the **Regime 2** tab and enter "1" for the material ID and enter 50 for the velocity.

Notice that the **OK** button is not active until all required variables have been entered. Click **OK** when the button becomes active.

12. You can now finish the analysis. When you apply this function as a boundary condition, use the table name that you assigned earlier.

```

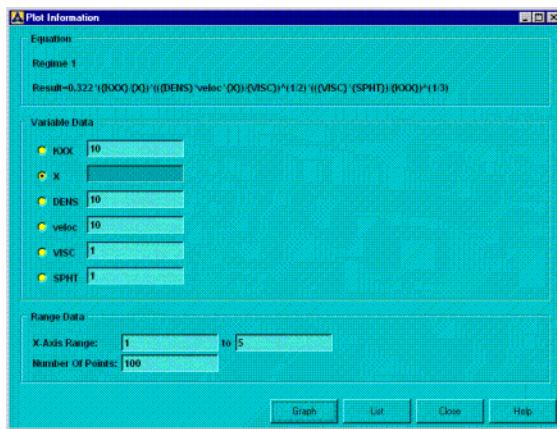
nsel,s,loc,y,0
d,all,temp,25
nsel,s,loc,y,0.5
sf,all,conv,%heatcf%,100    Apply the function as a boundary condition
finish
/solu
time,1
deltim,.1
outres,all,all
allsel
solve
finish
/post1
set,last
/pst,conv,hcoe,2,0.e+00,1
/replot !show surface load symbols
finish

```

4.6. Graphing or Listing a Function

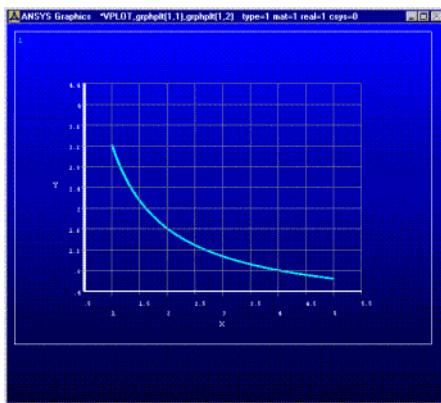
You can graph the function you enter and see a visual representation of the current function, or you can list results of the equation. Graphing and listing allow you to easily verify that your equation is behaving as you expect.

For either graphing or listing, select a variable to graph the result against, and set an x-axis range and the number of points to graph.



4.6.1. Graphing a Function

From the **Plot Information** dialog box, click **Graph** after you set up your plot. An example of a plot is shown below.



You can apply any standard graph functionality. (For example, fill in under curve using the command input window or via the GUI **Utility Menu> PlotCtrls> Style> Graphs**.) You can also save an image for later use.

4.6.2. Listing a Function

To generate a table displaying the plot point values, select the **List** option from the **Plot Information** dialog box. The settings you chose in the Plot Information dialog box are used to generate the values. An example of such a table follows:

List Results	
Equation	
Regime 1	
$\text{Result} = 0.322 \cdot (\text{KXX} \cdot (\text{X})^{\text{DENS}} \cdot \text{veloc}^{\text{VISC}})^{(1/2)} \cdot ((\text{VISC} \cdot (\text{SPHT})) \cdot (\text{KXX})^{(1/3)})$	
Constants:	
KXX = 10	
DENS = 10	
veloc = 10	
VISC = 1	
SPHT = 1	
X	Result
+1.000e+000	+3.220e+000
+1.040e+000	+3.096e+000
+1.080e+000	+2.981e+000
+1.120e+000	+2.875e+000
+1.160e+000	+2.776e+000
+1.200e+000	+2.683e+000
+1.240e+000	+2.597e+000
+1.280e+000	+2.516e+000
+1.320e+000	+2.439e+000
+1.360e+000	+2.368e+000
+1.400e+000	+2.300e+000
+1.440e+000	+2.236e+000
+1.480e+000	+2.176e+000
+1.520e+000	+2.118e+000
+1.560e+000	+2.064e+000

Save Close

You cannot edit the table, but you can copy and past it into a spreadsheet. You can also save the information to a text file; the file will contain all equation data and calculated coordinates.

Chapter 5: Solution

In the solution phase of an analysis, the computer takes over and solves the simultaneous set of equations that the finite element method generates. The results of the solution are:

- Nodal degree of freedom values, which form the primary solution
- Derived values, which form the element solution.

The element solution is usually calculated at the elements' integration points. The program writes the results to the database as well as to the results file (.RST, .RTH, or .RMG files).

The following solution topics are available:

- 5.1. Selecting a Solver
- 5.2. Types of Solvers
- 5.3. Solver Memory and Performance
- 5.4. Using Special Solution Controls for Certain Types of Structural Analyses
- 5.5. Obtaining the Solution
- 5.6. Solving Multiple Load Steps
- 5.7. Terminating a Running Job
- 5.8. Restarting an Analysis
- 5.9. Singular Matrices
- 5.10. Stopping Solution After Matrix Assembly

5.1. Selecting a Solver

Several methods for solving the system of simultaneous equations are available in the program: sparse direct solution, Preconditioned Conjugate Gradient (PCG) solution, Jacobi Conjugate Gradient (JCG) solution, Incomplete Cholesky Conjugate Gradient (ICCG) solution, and Quasi-Minimal Residual (QMR) solution. In addition, distributed versions of the sparse, PCG, and JCG solvers are available for use in Distributed ANSYS (refer to the [Parallel Processing Guide](#)).

To select a solver, issue the **EQSLV** command. See the command description for details about each solver.

The following tables provide general guidelines that you may find useful when selecting an appropriate solver for a given problem. The first table lists solvers that can be run using [shared memory parallelism](#) on shared memory parallel hardware. The second table lists solvers that can be run using [distributed](#)

memory parallelism on shared memory parallel hardware (for example, a desktop or workstation) or distributed memory parallel hardware (for example, a cluster). MDOF indicates million degrees of freedom.

Table 5.1: Shared Memory Solver Selection Guidelines

Solver	Typical Applications	Ideal Model Size	Memory Use	Disk (I/O) Use
Sparse Direct Solver (p. 97) (direct elimination)	When robustness and solution speed are required (nonlinear analysis); for linear analysis where iterative solvers are slow to converge (especially for ill-conditioned matrices, such as poorly shaped elements).	100,000 DOF (and beyond)	Out-of-core: 1 GB/MDOF In-core: 10 GB/MDOF	Out-of-core: 10 GB/MDOF In-core: 1 GB/MDOF
PCG Solver (p. 99) (iterative solver)	Reduces disk I/O requirement relative to sparse solver. Best for large models with solid elements and fine meshes. Most robust iterative solver in ANSYS.	500,000 DOF to 20 MDOF+	0.3 GB/MDOF w/ MSAVE ,ON; 1 GB/MDOF without MSAVE	0.5 GB/MDOF
JCG Solver (p. 101) (iterative solver)	Best for single field problems - (thermal, magnetics, acoustics, and multiphysics). Uses a fast but simple preconditioner with minimal memory requirement. Not as robust as PCG solver.	500,000 DOF to 20 MDOF+	0.5 GB/MDOF	0.5 GB/MDOF
ICCG Solver (p. 101) (iterative solver)	More sophisticated preconditioner than JCG. Best for more difficult problems where JCG fails, such as unsymmetric thermal analyses.	50,000 to 1,000,000+ DOF	1.5 GB/MDOF	0.5 GB/MDOF
QMR Solver (p. 102) (iterative solver)	Used for full harmonic analyses. This solver is appropriate for symmetric, complex, definite, and indefinite matrices. The QMR solver only supports 1 core.	50,000 to 1,000,000+ DOF	1.5 GB/MDOF	0.5 GB/MDOF

Table 5.2: Distributed Memory Solver Selection Guidelines

Solver	Typical Applications	Ideal Model Size	Memory Use	Disk (I/O) Use
Distributed Memory Sparse Direct Solver (p. 98)	Same as sparse solver but can also be run on distributed memory parallel hardware systems.	500,000 DOF to 10 MDOF (works well outside this range)	Out-of-core: 1.5 GB/MDOF on head compute node, 1.0 GB/MDOF on other	Out-of-core: 10 GB/MDOF In-core: 1 GB/MDOF

Solver	Typical Applications	Ideal Model Size	Memory Use	Disk (I/O) Use
			compute nodes In-core: 15 GB/MDOF on head compute node, 10 GB/MDOF on other compute nodes	
Distributed Memory PCG Solver (p. 99)	Same as PCG solver but can also be run on distributed memory parallel hardware systems.	1 MDOF to 100 MDOF	1.5-2.0 GB/MDOF in total*	0.5 GB/MDOF
Distributed Memory JCG Solver (p. 101)	Same as JCG solver but can also be run on distributed memory parallel hardware systems. Not as robust as the distributed memory PCG or shared memory PCG solver.	1 MDOF to 100 MDOF	0.5 GB/MDOF in total*	0.5 GB/MDOF

* In total means the sum of all processors.

To use more than four processors, the shared memory and distributed memory solvers require HPC licenses. For information, see [HPC Licensing](#) in the [Parallel Processing Guide](#).

5.2. Types of Solvers

5.2.1. The Sparse Direct Solver

The sparse direct solver (including the Block Lanczos method for modal and buckling analyses) is based on a direct elimination of equations, as opposed to iterative solvers, where the solution is obtained through an iterative process that successively refines an initial guess to a solution that is within an acceptable tolerance of the exact solution. Direct elimination requires the factorization of an initial very sparse linear system of equations into a lower triangular matrix followed by forward and backward substitution using this triangular system. The space required for the lower triangular matrix factors is typically much more than the initial assembled sparse matrix, hence the large disk or in-core memory requirements for direct methods.

Sparse direct solvers seek to minimize the cost of factorizing the matrix as well as the size of the factor using sophisticated equation reordering strategies. Iterative solvers do not require a matrix factorization and typically iterate towards the solution using a series of very sparse matrix-vector multiplications along with a preconditioning step, both of which require less memory and time per iteration than direct factorization. However, convergence of iterative methods is not guaranteed and

the number of iterations required to reach an acceptable solution may be so large that direct methods are faster in some cases.

Because the sparse direct solver is based on direct elimination, poorly conditioned matrices do not pose any difficulty in producing a solution (although accuracy may be compromised). Direct factorization methods always give an answer if the equation system is not singular. When the system is close to singular, the solver can usually give a solution (although you must verify the accuracy).

The sparse solver can run completely in memory (also known as in-core) if sufficient memory is available. The sparse solver can also run efficiently by using a balance of memory and disk usage (also known as out-of-core). The out-of-core mode typically requires about the same memory usage as the PCG solver (~1 GB per million DOFs) and requires a large disk file to store the factorized matrix (~10 GB per million DOFs). The amount of I/O required for a typical static analysis is three times the size of the matrix factorization. Running the solver factorization in-core (completely in memory) for modal/buckling runs can save significant amounts of wall (elapsed) time because modal/buckling analyses require several factorizations (typically 2 - 4) and repeated forward/backward substitutions (10 - 40+ block solves are typical). The same effect can often be seen with nonlinear or transient runs which also have repeated factor/solve steps.

The **BCSOPTION** command allows you to choose a memory strategy for the sparse solver. The available options for the *Memory_Option* field are DEFAULT, INCORE, OUTOFCORE, and FORCE. Depending on the availability of memory on the system, each memory strategy has its benefits. For systems with a large amount of physical memory, the INCORE memory mode often results in the best performance. Conversely, the OUTOFCORE memory mode often gives the worst solver performance and, therefore, is only recommended if necessary due to limited memory resources. In most cases you should use the DEFAULT memory mode. In this mode, the sparse solver uses sophisticated memory usage heuristics to balance available memory with the specific memory requirements of the sparse solver for each job. By default, most smaller jobs automatically run in the INCORE memory mode, but larger jobs may run in the INCORE memory mode or in the OUTOFCORE memory mode. In some cases you may want to explicitly set the sparse solver memory mode or memory allocation size using the **BC-SOPTION** command. However, doing so is only recommended if you know how much physical memory is on the system and understand the sparse solver memory requirements for the job in question.

When the sparse solver is selected in Distributed ANSYS, the distributed sparse direct solver is automatically used instead. The distributed sparse solver is mathematically identical to the shared-memory parallel sparse solver. It should typically be used for problems with which the PCG and JCG have convergence difficulty and on computer systems where large memory is available.

5.2.1.1. Distributed Sparse Direct Solver

The distributed sparse direct solver decomposes a large sparse matrix into smaller submatrices (instead of decomposing element domains), and then sends these submatrices to multiple cores on either shared-memory (for example, server) or distributed-memory (for example, cluster) hardware. Depending on the number of cores used, HPC licenses may be required. (For more information, see [HPC Licensing](#) in the *Parallel Processing Guide*.)

During the matrix factorization phase, each distributed process factorizes its submatrices simultaneously and communicates the information as necessary. The submatrices are automatically split into pieces (or fronts) by the solver during the factorization step. The non-distributed sparse solver works on one front at a time, while the distributed sparse solver works on n fronts at the same time (where n is the total number of processes used). Each front in the distributed sparse solver is stored in-core by each process while it is factored, even while the distributed sparse solver is running

in out-of-core mode. This is essentially equivalent to the out-of-core mode for the non-distributed sparse solver. Therefore, the total memory usage of the distributed sparse solver when using the out-of-core memory mode is about n times the memory that is needed to hold the largest front. As more cores are used the total memory used by the solver (summed across all processes) actually increases when running in this memory mode.

The **DSOPTION** command allows you to choose a specific memory strategy for the distributed sparse solver. The available options for the *Memory_Option* field are DEFAULT, INCORE, OUTOFCORE, and FORCE. Sophisticated memory usage heuristics, similar to those used by the sparse solver, are used to balance the specific memory requirements of the distributed sparse solver with the available memory on the machine(s) being used. By default, most smaller jobs run in the INCORE memory mode, while larger jobs can run either in the INCORE memory mode or in the OUTOFCORE memory mode. In some cases, you may want to explicitly set the memory mode using the **DSPOPTION** command. However, this is only recommended if you fully understand the solver memory used on each machine and the available memory for each machine.

When the distributed sparse solver runs in the out-of-core memory mode, it does substantial I/O to the disk storage device on the machine. If multiple solver processes write to the same disk, the performance of the solver decreases as more solver processes are used, meaning the total elapsed time of the solver does not decrease as much as expected. The ideal configuration for the distributed sparse solver when running in out-of-core mode is to run using a single process on each machine in a cluster or network, spreading the I/O across the hard drives of each machine, assuming that a high-speed network such as Infiniband is being used. Running the distributed sparse solver in out-of-core mode on a shared disk resource (for example, NAS or SAN disk) is typically not recommended. You can effectively run the distributed sparse solver using multiple processes with one drive (or a shared disk resource) if:

- The problem size is small enough relative to the physical memory on the system that the system buffer cache can hold all of the distributed sparse solver (I/O) files and other files in memory.
- You have a very fast hard drive configuration that can handle multiple I/O requests simultaneously. For a shared disk resource on a cluster, a very fast interconnect is also needed to handle the I/O traffic along with the regular communication of data within the solver.
- You use the **DSOPTION,,INCORE** command to force the distributed sparse solver into an in-core mode.

5.2.2. The Preconditioned Conjugate Gradient (PCG) Solver

The PCG solver starts with element matrix formulation. Instead of factoring the global matrix, the PCG solver assembles the full global stiffness matrix and calculates the DOF solution by iterating to convergence (starting with an initial guess solution for all DOFs). The PCG solver uses a proprietary preconditioner that is material property and element-dependent.

- The PCG solver is usually about 4 to 10 times faster than the JCG solver for structural solid elements and about 10 times faster than JCG for shell elements. Savings increase with the problem size.
- The PCG solver usually requires approximately twice as much memory as the JCG solver because it retains two matrices in memory:
 - The preconditioner, which is almost the same size as the stiffness matrix

- The symmetric, nonzero part of the stiffness matrix

You can use [Table 5.1: Shared Memory Solver Selection Guidelines \(p. 96\)](#) as a general guideline for memory usage.

This solver is available only for static or steady-state analyses and transient analyses, or for PCG Lanczos modal analyses. The PCG solver performs well on most static analyses and certain nonlinear analyses. It is valid for elements with definite or indefinite matrices. Contact analyses that use penalty-based or penalty and augmented Lagrangian-based methods work well with the PCG solver as long as contact does not generate rigid body motions throughout the nonlinear iterations (for example, full loss of contact). The Lagrange multiplier method used by the following [MPC184](#) element types can also be solved by the PCG solver: rigid beam, rigid link, slider, revolute joint, universal joint, weld joint, and general joint. However, for all other [MPC184](#) element types, Lagrange-formulation contact methods, and incompressible u-P formulations, the PCG solver cannot be used and the sparse solver is required. For more information, see the [PCGOPT](#) command.

Because they take fewer iterations to converge, well-conditioned models perform better than ill-conditioned models when using the PCG solver. Ill-conditioning often occurs in models containing elongated elements (that is, elements with high aspect ratios) or contact elements. To determine if your model is ill-conditioned, view the `Jobname . PCS` file to see the number of PCG iterations needed to reach a converged solution. Generally, static or full transient solutions that require more than 1500 PCG iterations are considered to be ill-conditioned for the PCG solver. For such models, the PCG solver may not be the most effective choice, and the program may automatically decide to switch to the sparse direct solver to provide a more efficient solution. Automatic switching is controlled via the *Fallback* option on the [PCGOPT](#) command.

For ill-conditioned models, the [PCGOPT](#) command can sometimes reduce solution times. You can adjust the level of difficulty ([PCGOPT,Lev_Diff](#)) depending on the amount of ill-conditioning in the model. By default, the program automatically adjusts the level of difficulty for the PCG solver based on the model. However, sometimes forcing a higher level of difficulty value for ill-conditioned models can reduce the overall solution time.

The PCG solver primarily solves for displacements/rotations (in structural analysis), temperatures (in thermal analysis), etc. The accuracy of other derived variables (such as strains, stresses, flux, etc.) is dependent upon accurate prediction of primary variables. Therefore, the program uses a very conservative setting for PCG tolerance (defaults to 1.0E-8) The primary solution accuracy is controlled by the PCG. For most applications, setting the PCG tolerance to 1.0E-6 provides a very accurate displacement solution and may save considerable CPU time compared with the default setting. Use the [EQSLV](#) command to change the PCG solver tolerance.

Direct solvers (such as the sparse direct solver) produce very accurate solutions. Iterative solvers, such as the PCG solver, require that a PCG convergence tolerance be specified. Therefore, a large relaxation of the default tolerance may significantly affect the accuracy, especially of derived quantities.

With all iterative solvers you must verify that the model is appropriately constrained. No minimum pivot is calculated, and the solver continues to iterate if any rigid body motion exists. Note that in this situation, if the PCG solver fails to converge, the program may decide to automatically switch to the sparse direct solver to improve the convergence behavior and provide a solution. If the model is truly underconstrained, even the sparse solver may fail with messages indicating there are one or more near-zero pivot terms during the matrix factorization step.

In a modal analysis using the PCG solver (**MODOPT**,**LANPCG**), the number of modes should be limited to 100 or less for efficiency. PCG Lanczos modal solutions can solve for a few hundred modes, but with less efficiency than Block Lanczos (**MODOPT**,**LANB**).

When the PCG solver encounters an indefinite matrix, the program may automatically switch to using the sparse direct solver in order to provide a more efficient solution. If this occurs during a nonlinear analysis, the program continues to use the sparse solver for the duration of the substep. At the completion of the current substep, the program typically reverts to the PCG solver (see the **PCGOPT** command for more details). If the fallback logic and automatic switching is disabled by setting *FALLBACK* = OFF on the **PCGOPT** command, then the solver invokes an algorithm that handles indefinite matrices. If the indefinite matrix algorithm also fails (this happens when the equation system is ill-conditioned; for example, losing contact at a substep or a plastic hinge development), the outer Newton-Raphson loop is triggered to perform a bisection. Normally, the stiffness matrix is better conditioned after bisection, and the PCG solver can eventually solve all the nonlinear steps.

The solution time grows linearly with problems size for iterative methods so huge models can still be solved within very reasonable times. For modal analyses of large models (for example, 10 million DOF or larger), **MODOPT**,**LANPCG** is a viable solution method if the number of modes is limited to approximately 100.

Use **MSAVE**,ON (the default in most cases) for memory savings of up to 70 percent. The **MSAVE** command uses an element-by-element approach (rather than globally assembling the stiffness matrix) for the parts of the structure involving **SOLID185**, **SOLID186**, or **SOLID187** elements with linear material properties. This feature applies only to static analyses or modal analyses using the PCG Lanczos method (**ANTYPE**,STATIC; or **ANTYPE**,MODAL with **MODOPT**,**LANPCG**). Note that the **MSAVE** feature does not apply to any linear perturbation analysis types. The solution time may be affected depending on the hardware (processor speed, memory bandwidth, etc.), as well as the chosen element options.

5.2.3. The Jacobi Conjugate Gradient (JCG) Solver

The JCG solver also starts with element matrix formulation. Instead of factoring the global matrix, the JCG solver assembles the full global stiffness matrix and calculates the DOF solution by iterating to convergence (starting with an initial guess solution for all DOFs). The JCG solver uses the diagonal of the stiffness matrix as a preconditioner. The JCG solver is typically used for thermal analyses and is best suited for 3-D scalar field analyses that involve large, sparse matrices.

For some cases, the tolerance default value (set via the **EQSLV**,**JCG** command) of 1.0E-8 may be too restrictive, and may increase running time needlessly. The value 1.0E-5 may be acceptable in many situations.

The JCG solver is available only for static analyses, full harmonic analyses, or full transient analyses. (You specify these analysis types using the commands **ANTYPE**,STATIC, **HROPT**,**FULL**, or **TRNOPT**,**FULL** respectively.)

With all iterative solvers, be particularly careful to check that the model is appropriately constrained. No minimum pivot is calculated and the solver continues to iterate if any rigid body motion is possible.

5.2.4. The Incomplete Cholesky Conjugate Gradient (ICCG) Solver

The ICCG solver operates similarly to the JCG solver with the following exceptions:

- The ICCG solver is more robust than the JCG solver for matrices that are not well-conditioned. Performance varies with matrix conditioning, but in general ICCG performance compares to that of the JCG solver.
- The ICCG solver uses a more sophisticated preconditioner than the JCG solver. Therefore, the ICCG solver requires approximately twice as much memory as the JCG solver.

The ICCG solver is typically used for unsymmetric thermal analyses and electromagnetic analyses and is available only for static analyses, full harmonic analyses [**HROPT**,**FULL**], or full transient analyses [**TRNOPT**,**FULL**]. (You specify the analysis type using the **ANTYPE** command.) The ICCG solver is useful for structural and multiphysics applications, and for symmetric, unsymmetric, complex, definite, and indefinite matrices.

5.2.5. The Quasi-Minimal Residual (QMR) Solver

The QMR solver is used for acoustic analyses and is available only for full harmonic analyses [**HROPT**,**FULL**]. (You specify the analysis type using the **ANTYPE** command.) This solver is appropriate for symmetric, complex, definite, and indefinite matrices. The QMR solver is more robust than the ICCG solver.

5.3. Solver Memory and Performance

For best performance, understand the individual solvers' memory usage and performance under certain conditions. Each solver uses different methods to obtain memory; understanding how memory is used by each solver can help you to avoid problems (such as running out of memory during solution) and maximize the problem size you can handle on your system.

5.3.1. Running Solvers Using Parallel Processing

One of the easiest ways to improve solver performance is to run the solvers using multiple processors to take advantage of parallel processing. For detailed information about using parallel processing, see the *Parallel Processing Guide*.

All equation solvers, including eigensolvers for modal and buckling analyses, have highly tuned computational kernels that are called in parallel to take advantage of multiple processor cores. The amount of speedup, which can be significant in many cases, depends highly on the particular simulation involved and the characteristics of hardware being used.

5.3.2. Using Large Memory Capabilities with the Sparse Solver

The sparse solver automatically attempts to use the available system memory to run as efficiently as possible. However, you can configure various memory options for the sparse solver explicitly using the **BCSOPTION** command, if necessary.

An important factor in big memory systems is system configuration. The best performance occurs when jobs run comfortably within the physical memory limits of a given system configuration. For example, a sparse solver job that requires 7500 MB on a system with 8 GB does not run as well as the same job on a 12-16 GB system. Large memory systems use their memory to hide I/O costs by keeping files resident in memory automatically, so even jobs too large to run in-core benefit from large memory.

5.3.3. Disk Space (I/O) and Postprocessing Performance for Large Memory Problems

I/O performance with large memory

One of the hidden system benefits of large memory systems is the ability to cache large I/O requests. Even for modest-sized jobs, you can considerably reduce the cost of I/O when the system free memory is larger than the sum of file sizes active in a job. This feature, often called buffer cache, is a system-tunable parameter and can effectively move all I/O traffic to memory copy speeds. The system details are different for various vendors; consult your hardware manufacturer for details on their systems. For most Linux versions and Windows systems, the benefit of the system buffer cache is automatic and does not require tuning.

5.4. Using Special Solution Controls for Certain Types of Structural Analyses

When you are performing certain types of structural analyses, you can take advantage of these special solution tools:

- Abridged **Solution** menus, which are available for static, transient (all solution methods), modal, and buckling analyses. See [Using Abridged Solution Menus \(p. 103\)](#).
- The Solution Controls dialog box, which is available for static and transient (full solution method only) analyses. See [Using the Solution Controls Dialog Box \(p. 104\)](#).

5.4.1. Using Abridged Solution Menus

If you are using the GUI to perform a structural static, transient, modal, or buckling analysis, you can use either abridged or unabridged **Solution** menus:

- Unabridged **Solution** menus list all solution options, regardless of whether it is recommended, or even possible, for you to use them in the current analysis. (If it is not possible for you to use an option in the current analysis, the option is listed but is grayed out.)
- Abridged **Solution** menus are simpler. They list only those options that apply to the type of analysis that you are performing. For example, if you are performing a static analysis, the **Modal Cyclic Sym** option does not appear on the abridged **Solution** menu. Only those options that are valid and/or recommended for the current analysis type appear.

If you are performing a structural analysis, the abridged **Solution** menu appears by default when you enter the solution processor (**Main Menu> Solution**).

If your analysis is either static or full transient, you can use the options on the menu to complete the solution phase of your analysis. However, if you select a different analysis type, the default abridged **Solution** menu that you see above is replaced by a different **Solution** menu. The new menu is appropriate for the analysis type you select.

All variants of the abridged **Solution** menu contain an **Unabridged Menu** option. This option is always available for you to select in case you prefer using the unabridged menu.

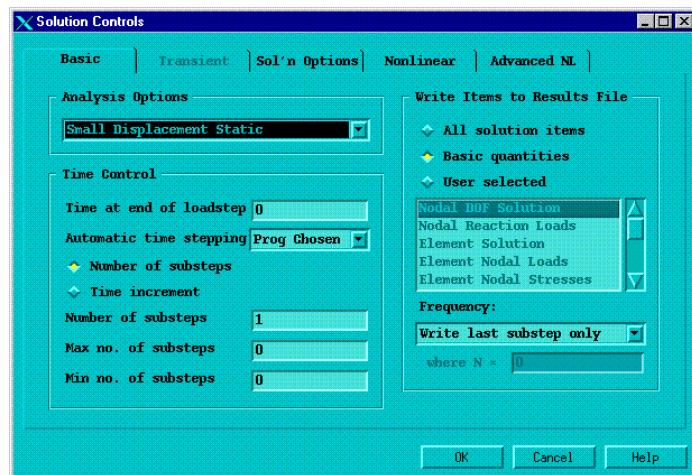
If you perform one analysis and then start a new analysis within the same session, the program (by default) presents you with the same type of **Solution** menu that you used for the first analysis. For example, if you choose to use the unabridged **Solution** menu to perform a static analysis and then select a new buckling analysis, the unabridged **Solution** menu that is appropriate for buckling analyses appears. You can toggle between the unabridged and abridged **Solution** menus at any time during the solution phase of the analysis by selecting the appropriate menu option (**Main Menu> Solution> Unabridged Menu** or **Main Menu> Solution> Abridged Menu**).

5.4.2. Using the Solution Controls Dialog Box

If you are performing a structural static or full transient analysis, you can use a streamlined solution interface (called the Solution Controls dialog box) for setting many of your analysis options. The Solution Controls dialog box consists of five tabbed "pages," each of which contains a set of related solution controls. The dialog box is useful for specifying the settings for each load step of a multiple load step analysis.

As long as you are performing a structural static or full transient analysis, your **Solution** menu contains the **Sol'n Control** option. When you click the **Sol'n Control** menu item, the Solution Controls dialog box appears. This dialog box provides you with a single interface for setting analysis and load step options. See [Figure 5.1: Solution Controls Dialog Box \(p. 104\)](#) for an illustration.

Figure 5.1: Solution Controls Dialog Box



The **Basic** tab, which is shown above, is active when you access the dialog box. The complete list of tabs, in order from left to right, is as follows:

- **Basic**
- **Transient**
- **Sol'n Options**
- **Nonlinear**
- **Advanced NL**

Each set of controls is logically grouped on a tab; the most basic controls appear on the first tab, with each subsequent tab providing more advanced controls. The **Transient** tab contains transient analysis

controls; it is available only if you choose a transient analysis and remains grayed out when you choose a static analysis.

Each of the controls on the Solution Controls dialog box corresponds to a command. The table below illustrates the relationships between the tabs and the command functionality that you can access from each.

Table 5.3: Relationships Between Tabs of the Solution Controls Dialog Box and Commands

Solution Controls Dialog Box Tab	What Does This Tab Enable You to Do?	What Commands Are Related to This Tab?
Basic	<ul style="list-style-type: none"> Specify the type of analysis that you want to perform. Control various time settings. Specify the solution data that you want to write to the database. Specify the prestress effects. 	ANTYPE, NLGEOM, TIME, AUTOTS, NSUBST, DELTIM, OUTRES, PSTRES
Transient	<ul style="list-style-type: none"> Specify transient options, such as transient effects and ramped vs. stepped loading. Specify damping options. Choose time integration method. Define integration parameters. Set midstep residual criterion. 	TIMINT, KBC, ALPHAD, BETAD, TRNOPT, TINTP, MIDTOL
Sol'n Options	<ul style="list-style-type: none"> Specify the type of equation solver that you want to use. Specify parameters for performing a multiframe restart. 	EQSLV, RESCONTROL,
Nonlinear	<ul style="list-style-type: none"> Control nonlinear options, such as line search and solution predictor. Specify the maximum number of iterations that are allowed per substep. Indicate whether you want to include creep calculation in the analysis. Control bisections. Set convergence criteria. Specify the creep strain rate effect. 	LNSRCH, PRED, NEQIT, RATE, CUTCONTROL, CNVTOL, RATE

Solution Controls Dialog Box Tab	What Does This Tab Enable You to Do?	What Commands Are Related to This Tab?
Advanced NL	<p>Specify analysis termination criteria.</p> <p>Control activation and termination of the arc-length method.</p> <p>Active elements that support nonlinear stabilization.</p>	NCNV, ARCLEN, ARCTRM, SSTATE

When you are satisfied with the settings on the **Basic** tab, you do not need to progress through the remaining tabs unless you want to change some of the advanced controls. As soon as you click **OK** on any tab of the dialog box, the settings are applied to the database and the dialog box closes.

Whether you make changes to only one or to multiple tabbed pages, your changes are applied to the database only when you click **OK** to close the dialog box.

5.4.3. Accessing More Information

Discussions of the Solution Controls dialog box are included throughout this documentation as applicable.

For additional information, refer to the following:

- Online help for the Solution Controls dialog box
- Structural Static Analysis in the *Structural Analysis Guide*
- Transient Dynamic Analysis in the *Structural Analysis Guide*
- Nonlinear Structural Analysis in the *Structural Analysis Guide*

5.5. Obtaining the Solution

To initiate the solution, issue the **SOLVE** command.

Because the solution phase generally requires more computer resources than the other phases of an analysis, it is better suited to batch (background) mode than interactive mode.

The solver writes output to the output file (*Jobname.OUT*) and the results file. If you run the solution interactively, your display takes the place of an output file. By issuing the **/OUTPUT** command before **SOLVE**, you can divert the output to a file instead of the screen.

Data written to the output file consist of the following:

- Load summary information
- Mass and moments of inertia of the model
- Solution summary information

- A final closing banner that gives total CPU time and elapsed time.
- Data requested by the **OUTPR** output control command

In interactive mode, much of the output is suppressed. The results file (.RST, .RTH, or .RMG) contains all results data in binary form, which you can then review in the postprocessors.

Another useful file produced during solution is **Jobname.STAT**, which gives the status of the solution. You can use this file to monitor an analysis while it is running. It is particularly useful in iterative analyses such as nonlinear and transient analyses.

The **SOLVE** command calculates the solution for the load step data currently in the database.

5.6. Solving Multiple Load Steps

There are three ways to define and solve multiple load steps:

- Multiple **SOLVE** method
- Load step file method
- Array parameter method.

5.6.1. Using the Multiple SOLVE Method

This method is the most straightforward. It involves issuing the **SOLVE** command after each load step is defined. The main disadvantage, for interactive use, is that you have to wait for the solution to be completed before defining the next load step. A typical command stream for the multiple **SOLVE** method is shown below:

```
/SOLU
...
! Load step 1:
D,....
SF,....
...
SOLVE           ! Solution for load step 1
! Load step 2
F,....
SF,....
...
SOLVE           ! Solution for load step 2
Etc.
```

5.6.2. Using the Load Step File Method

The load step file is a convenient method to use when you want to solve problems while you are away from your terminal or PC (for example, overnight). It involves writing each load step to a load step file (via the **LSSOLVE** command) and, with one command, reading in each file and obtaining the solution. See [Loading \(p. 23\)](#) for details about creating load step files.

To solve multiple load steps, issue the **LSSOLVE** command. The command is actually a macro that reads in the load step files sequentially and initiates the solution for each load step. An example command input for the load step file method is shown here:

```

/SOLU          ! Enter SOLUTION
...
! Load Step 1:
D,...           ! Loads
SF,....
...
NSUBST,...      ! Load step options
KBC,....
OUTRES,....
OUTPR,....
...
LSWRITE         ! Writes load step file: Jobname.S01
! Load Step 2:
D,...           ! Loads
SF,....
...
NSUBST,...      ! Load step options
KBC,....
OUTRES,....
OUTPR,....
...
LSWRITE         ! Writes load step file: Jobname.S02
...
LSSOLVE,1,2     ! Initiates solution for load step files 1 and 2

```

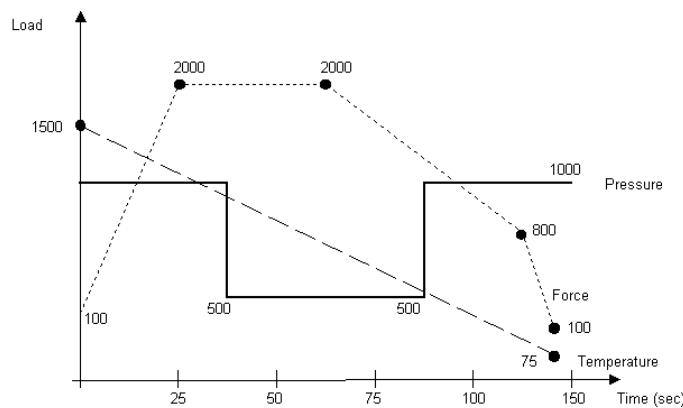
See the [Command Reference](#) for a discussion of the **NSUBST**, **KBC**, **OUTRES**, **OUTPR**, **LSWRITE**, and **LSSOLVE** commands.

5.6.3. Using the Array Parameter Method

This method, mainly intended for transient or nonlinear static (steady-state) analyses, requires knowledge of array parameters and do-loops, which are part of APDL (ANSYS Parametric Design Language). See the [ANSYS Parametric Design Language Guide](#) for information about APDL. The array parameter method involves building tables of *load* versus *time* using array parameters and is best explained by the following example.

Figure 5.2: Examples of Time-Varying Loads

Force		Pressure		Temperature	
Time	Value	Time	Value	Time	Value
0.0	100	0.0	1000	0.0	1500
21.5	2000	35.0	1000	145.0	75
62.5	2000	35.8	500		
125.0	800	82.5	500		
145.0	100	82.6	1000		
		150.0	1000		



Suppose that you have a set of time-varying loads such as the ones shown above. There are three load functions, so you need to define three array parameters. All three array parameters must be of

type TABLE. The force function has five points, so it needs a 5×1 array; the pressure function needs a 6×1 array; and the temperature function needs a 2×1 array. Notice that all three arrays are one-dimensional. The load values are entered in column 1 and the time values are entered in column zero. (The zeroth column and zeroth row, which normally contain index numbers, must be changed and filled with a monotonically increasing set of numbers if you define the array parameter as a TABLE.)

To define the three array parameters, you first need to declare their type and dimensions. To do so, use either of the following:

Command(s): *DIM

GUI: Utility Menu> Parameters> Array Parameters> Define/Edit

For example:

```
*DIM, FORCE, TABLE, 5, 1
*DIM, PRESSURE, TABLE, 6, 1
*DIM, TEMP, TABLE, 2, 1
```

You can now use either the array parameter editor (**Utility Menu> Parameters> Array Parameters> Define/Edit**) or a set of "=" commands to fill these arrays. The latter method is shown below.

```
FORCE(1,1)=100,2000,2000,800,100 ! Force values in column 1
FORCE(1,0)=0,21.5,50.9,98.7,112 ! Corresponding time values in column 0
FORCE(0,1)=1 ! Zeroth row
PRESSURE(1,1)=1000,1000,500,500,1000,1000
PRESSURE(1,0)=0,35,35.8,74.4,76,112
PRESSURE(0,1)=1
TEMP(1,1)=800,75
TEMP(1,0)=0,112
TEMP(0,1)=1
```

You have now defined the load histories. To apply these loads and obtain the solution, you need to construct a do-loop (using the commands ***DO** and ***ENDDO**) such as the one shown below:

```
TM_START=1E-6           ! Starting time (must be > 0)
TM_END=112             ! Ending time of the transient
TM_INCR=1.5            ! Time increment
*DO,TM,TM_START,TM_END,TM_INCR ! Do for TM from TM_START to TM_END in
                                ! steps of TM_INCR
      TIME,TM          ! Time value
      F,272,FY,FORCE(TM) ! Time-varying force (at node 272, FY)
      NSEL,...          ! Select nodes on pressure surface
      SF,ALL,PRES,PRESSURE(TM) ! Time-varying pressure
      NSEL,ALL          ! Activate all nodes
      NSEL,...          ! Select nodes for temperature specification
      BF,ALL,TEMP,TEMP(TM) ! Time-varying temperature
      NSEL,ALL          ! Activate all nodes
      SOLVE             ! Initiate solution calculations
*ENDDO
```

See the [Command Reference](#) for discussions of the ***DO**, **TIME**, **F**, **NSEL**, **SF**, **BF**, and ***ENDDO** commands.

You can change the time increment (TM_INCR parameter) very easily with this method. With other methods, changing the time increment for such complex load histories would be quite cumbersome.

5.7. Terminating a Running Job

You can terminate a running job, if necessary, with the help of system functions such as a system break, issuing a kill signal, or deleting the entry in the batch queue. This is not the preferred method for nonlinear analyses, because a job terminated in this manner cannot be restarted.

To terminate a nonlinear analysis "cleanly" on a multitasking operating system, create an abort file, named `Jobname.ABT` (or, on some case-sensitive systems, `jobname.abt`), containing the word `nonlinear` on the first line, starting in column 1. At the start of an equilibrium iteration, if the program finds such a file in the working directory, the analysis stops and can be restarted at a later time.

Note:

If commands are being read using a file specified via the **/INPUT** command, the abort file terminates the solution, but the program continues to read commands from the specified input file. Any postprocessing commands included in the input file execute.

5.8. Restarting an Analysis

Occasionally, you may need to restart an analysis after the initial run has been completed. Two different restart procedures are available:

5.8.1. Multiframe Restart

5.8.2. Modal Analysis Restart

The following examples illustrate cases where a restart may be necessary:

- More load steps must be added to the analysis (multiframe restart).
- There are additional loading conditions in a linear static analysis or additional portions of a time-history loading curve in a transient analysis (multiframe restart).
- To recover from a convergence failure in a nonlinear analysis (multiframe restart).
- To do a linear perturbation analysis (multiframe restart). See [General Procedure for Linear Perturbation Analysis](#) for details.
- To generate load vectors, residual vectors, or enforced motion pseudo-static shapes for a subsequent mode-superposition analysis or spectrum analysis (modal analysis restart).

The multiframe restart can resume a job at any point in the analysis for which information is saved, allowing you to perform multiple analyses of a model and giving you more options for recovering from an abnormal termination. The program allows a multiframe restart for static and transient (full or mode-superposition method) analyses. Distributed ANSYS supports multiframe restarts for nonlinear static and full transient analyses, and for linear perturbation analyses.

The modal analysis restart can be used to do additional calculations after the eigensolution. Modal extraction typically requires more computing time than element loads generation, residual vector generation, and enforced static modes calculation. Reusing eigenmodes that have already been generated can save significant time in a downstream analysis.

5.8.1. Multiframe Restart

To perform a multiframe restart, the model must meet the following conditions:

- The analysis type must be static (steady-state), harmonic (2-D magnetic only), or transient (full or mode-superposition method only). No other analysis types can be restarted.
- At least one iteration must have been completed in the initial run.
- The initial run should not have stopped abnormally (for example, a system level abort).
- The initial analysis, which generated the restart files, and the restarted analysis must be performed with the same version of the ANSYS, Inc. software product.

For nonlinear static and full transient analyses, the program sets up the parameters for a multiframe restart by default.

Multiframe restart allows you to save analysis information at many substeps during a run, then restart the run at one of those substeps. Before running the initial analysis, use the **RESCONTROL** command to set up the frequency at which restart files are saved within each load step of the run.

When restarting a job, use the **ANTYPE** command to specify the restart point and type of restart. You can continue the job from the restart point (making any corrections necessary), or terminate a load step at the restart point (rescaling all loading) and continue with the next load step.

The following example input shows how to set up the restart file parameters in an analysis then restart the analysis, continuing from a specified load step and substep.

Example 5.1: Setting Up Restart File Parameters and Restarting the Analysis

```
/prep7
et,1,182,,,          !Define nodes and elements
mp,ex,1,10
mp,alpx,1,0.1
mp,alpy,1,0.1
mp,alpx,1,0.1
mp,nuxy,1,0.2
n,1
n,2,1
n,3,1,1
n,4,,1
n,5,2
n,6,2,1
e,1,2,3,4
e,2,5,6,3
finish

/solu
rescontrol,,all,1,5      !For all load steps, write the restart
                          !file .Rnnn at every substep, but allow
                          !a maximum of 5 restart files per load step
nlgeom,on                !Large strain analysis with temperature loadings
nsubst,6,6,6
d,1,all
d,2,uy
outres,all,all
bfe,1,temp,1,1
bfe,2,temp,1,5
solve
rescontrol,file_summary  !List information contained in all the
```

```
!Rnnn files for this job
finish
/postl
set,1,3
presol
set,last
presol
finish

/solu
antyp,,rest,1,3      !Restart the analysis at load step 1,
                      !substep 3
solve
rescontrol,file_summary
finish
/postl
set,last
presol
finish

/solu
antype,,rest,1,3,endstep !End load step 1 at substep 3
                        !when time (load factor) = 0.5
                        !ldstep = 1, substep = 3, load
solve
                        !execute ENDSTEP, loads are
                        !rescaled to time = 0.5
rescontrol,file_summary
bfe,1,temp,1,2          !apply higher loads,
bfe,2,temp,1,6
solve
                        !execute solve to advance load
                        !factor from previous
                        !time = 0.5 to time = 1.5
/postl
set,last
presol
finish
```

The following example input shows how to terminate a load step at a particular substep, then continue with the next load step.

Example 5.2: Terminating a Load Step and Continuing with Next Load Step

```
/solu
antype,,rest,1,3,endstep !End load step 1 at substep 3
                        !when time (load factor) = 0.6125
                        !ldstep = 1, substep = 3, load
solve
                        !execute ENDSTEP, loads are
                        !rescaled to time = 0.6125
rescontrol,file_summary
bfe,1,temp,1,2          !apply higher loads,
bfe,2,temp,1,6
solve
                        !execute solve to advance load
                        !factor from previous
                        !time = 0.6125 to time = 1.6125
/postl
set,last
presol
finish
```

The following example input shows how to restart an analysis with old and new parameters.

Example 5.3: Restarting an Analysis with Old and New Parameters

```
/prep7
et,1,182               ! Build model
```

```

n,1,0.0,0.0
n,2,0.0,0.5
n,3,0.0,1.0
n,4,1.0,0.0
n,5,1.0,0.5
n,6,1.0,1.0
e, 1,4,5,2
e, 2,5,6,3
mp,ex,1,1000.0
mp,nuxy,1,0.3
mp,alpx,1,1.e-4

d,1,all
d,2,ux,0.0
d,3,ux,0.0
d,4,uy,0.0

*dim,ftbl,table,4,,time ! Make tabular point load
ftbl(1,0)=1,2,3,4
ftbl(1,1)=2.5,5.0,7.5,10.0
nsel,all
f,all,fx,%ftbl%           ! Apply it to all nodes
flist

/solu
rescont,,all,all          ! Save all substeps for possible restart
nlgeo,on
time,4
deltim,1
outres,all,all
solve                      ! Solve with point loads and the *.RDB file is saved
                           ! at the moment. The parameterized tabular point load
                           ! FTBL is also saved into *.RDB

*dim,temtbl,table,4,,time ! Define table TEMTBL and use it for body load: temperature
temtbl(1,0)=1,2,3,4
temtbl(1,1)=250,500.0,750,1000.
! bf,all,TEMP,%temtbl%      ! May use this to apply the body load table
! bflist
parsave,all,moreload       ! Save all the APDL parameters and tables to file: moreload
                           ! NOTE: *.RDB does not have information of table TEMTBL.
fini

/clear, nostart
/solu
antype,,restart,1,3,endstep ! Do restart ENDSTEP because we want to apply TEMTBL at
                           ! TIME = 3.5 (LDSTEP=1,SUBSTEP=3) because we want to
                           ! Apply the temperature load from TIME=3.5 onwards.
                           ! Here, RESTART has resumed *.RDB database where the
                           ! Table FTBL is saved.

solve                      ! Activate ENDSTEP

parresu,,moreload          ! For further load step, we want to apply table TEMTBL
                           ! as body force. NOTE: table TEMTBL is not in *.RDB. Therefore,
                           ! we have to use PARRESU command. APDL file "moreload" is
                           ! saved earlier.

*status
bf,all,TEMP,%temtbl%
bflist
time,4                      ! Solve up to TIME = 4.0 because the load step ENDSTEP only
                           ! carries up to TIME = 3.5
solve
fini
/post1
set,last
prdis
prrsol
fini

```

The following example input demonstrates a restart after changing boundary conditions.

Example 5.4: Restarting after Changing Boundary Conditions

```

/prep7
et,1,21
r,1,1,1,1,1,1,1
n,1
e,1
fini

/solu
antyp,trans
timint,off
time,.1
nsub,2
kbc,0
d,1,ux,100           ! to apply initial velocity (IC command is preferred)
solve

timint,on
ddele,1,ux           ! this requires special handling by multi-frame restart
                       ! if a reaction force exists at this dof, replace it with an equal
                       ! force using the endstop option
time,.2
nsub,5
rescontrol,define,all,1 ! request possible restart from any substep
outres,nsol,1
solve
fini

/solu
antyp,,restart,2,3    ! this command resumes the .rdb database created at the start of solution
                       ! (restart from substep 3)
d,1,ux,100             ! re-specify boundary condition deleted during solution
solve
fini

/post26
nsol,2,1,ux
prvar,2                ! results show constant velocity through restart
finish

```

The following example input demonstrates the use of a negative value input for *Ldstep* on the **RESCONTROL** command in order to reduce the size of the **.LDHI** file and reduce the total number of **.Rnnn** files written.

Example 5.5: Using a Negative *Ldstep* Value on RESCONTROL

```

/prep7
mp,ex,1,10
mp,nuxy,1,0.3
mp,dens,1,0.5
n,1,
n,2,1
n,3,1,1
n,4,0,1
n,5,2
n,6,2,1
n,7,3,0
n,8,3,1
nlist
et,1,182
e,1,2,3,4

```

```

e,2,5,6,3
e,5,7,8,6
nsel,all
d,1,all
d,4,ux,0.01
finish
/solu
antype,static
nlgeom,on
nsubst,2,2,2
rescontrol,,,-3,last      ! For the next 11 loadsteps, write load history
                            ! information to .ldhi file every 3rd loadstep
*do,i,1,11
solve                      ! Solve for 11 loadsteps
*enddo
rescontrol,,-100,last      ! For the next 1300 loadsteps, write load history
                            ! information to .ldhi file every 100th loadstep
*do,i,1,1300
solve                      ! Solve for another 1300 loadsteps
*enddo
finish
/solu
antype,,restart,,,pert     ! Use the LAST possible restart point from
                            ! the previous 1311 loadsteps
                            ! to do a linear perturbation modal analysis
perturb,modal
solve,elform
modopt,lanb,1
solve
finish                     ! Completion of linear perturbation modal phase

```

The following example input shows how to delete a previously defined restart specification prior to adding a new one.

Example 5.6: Deleting Restart Specifications

```

/prep7
et,1,182      ! Build the model
mp,ex,1,3e9
mp,nuxy,1,0.3
mp,dense,1,2500
n,1,
n,2,1
n,3,1,1
n,4,0,1
n,5,2
n,6,2,1
n,7,3,0
n,8,3,1
nlist
e,1,2,3,4
e,2,5,6,3
e,5,7,8,6
nall
nsel,s,loc,x,0
d,all,all,0
nsel,all
save
finish

/solu
antype,static      ! Perform a static analysis
nlgeom,on          ! Large deflection is on
nsel,s,loc,x,3
f,all,fx,50
allsel,all
nsubst,5,5,5

rescontrol,define,-3,last    ! For the next 10 loadsteps, write load history

```

```

        ! information to .ldhi file every 3rd loadstep

*do,i,1,10      ! solve for 10 loadsteps
solve
*enddo

rescontrol,delete,-3      ! Delete previous RESCONTROL command
rescontrol,define,last,last ! For the next 10 loadsteps, write load history
                           ! information to .ldhi file for the last loadstep only

*do,i,1,10      ! solve for another 10 loadsteps
solve
*enddo

rescontrol,file_summary    ! Print the substeps and load step information for all .Xnnn files
                           ! The .Xnnn files saved for loadsteps: 3, 6, 9, 20
                           ! The .ldhi file saved for loadsteps: 2,3,5,6,8,9,19,20
finish

```

Note:

If you are using the Solution Controls dialog to perform a static or full transient analysis, you can specify basic multiframe restart options on the dialog's **Sol'n Options** tab. These options include the maximum number of restart files that you want to write for a load step, as well as how frequently you want the files to be written. For an overview of the Solution Controls dialog box, see [Using Special Solution Controls for Certain Types of Structural Analyses \(p. 103\)](#). For details about how to set options on the Solution Controls dialog box, access the dialog box (**Main Menu > Solution > Sol'n Control**), select the tab that you are interested in, and click the **Help** button.

5.8.1.1. Multiframe File Restart Requirements

The following files are necessary for performing a multiframe restart:

- Jobname.RDB

Database file saved automatically at the first iteration of the first load step, first substep of a job. This file provides a complete description of the solution with all initial conditions, and remains unchanged regardless of how many restarts are done for a particular job. When running a job, you should input all information needed for the solution - including parameters (APDL), components, and mandatory solution setup information - before you issue the first **SOLVE**. If you do not specify parameters before issuing the first **SOLVE** command, the parameters are not saved in the .RDB file. In this case, you must use **PARSAV** before you begin the solution and **PARRES** during the restart to save and restore the parameters. If the information stored in the .RDB file is not sufficient to perform the restart, you must input the additional information in the restart session before issuing the **SOLVE** command.

- Jobname.RDnn

Required if remeshing occurs before the restart, where *nn* is the number of remeshings before the current restart. This file has the same content as Jobname.RDB except that it contains the most recent mesh and is saved automatically at the first remeshing iteration. This file remains unchanged regardless of how many restarts occur for the current job.

- Jobname.LDHI

Load-history file for the specified job. It is an ASCII file (similar to files created by **LSPRINT**) that contains all loading and boundary conditions for specified load steps, including parameters that may be required for tabular loads and boundary conditions. The frequency at which load history information is written is determined by the **RESCONTROL** command. In general, you need the loading and boundary conditions for two contiguous load steps because of the ramped load conditions for a restart.

Loading and boundary conditions are stored for the FE mesh. Loading and boundary conditions applied to the solid model are transferred to the FE mesh before being stored in `Jobname.LDHI`. During a multiframe restart, the program reads the loading and boundary conditions for the restart load step from this file (similar to an **LSREAD** command).

By default, load-history information is written to `Jobname.LDHI` only for the last load step. For analyses that involve many load steps, this saves disk space and speeds up overall solution time. Alternatively, you can specify the frequency that load-step information is written to `Jobname.LDHI` by inputting a negative number for the `Ldstep` argument of the **RESCONTROL** command.

The file is modified at the end of each specified load step. You cannot alter the file manually because any modifications may cause an unexpected restart condition.

- `Jobname.Rnnn`

For nonlinear static and full transient analyses. This file contains element saved records similar to the `.ESAV` or `.OSAV` files. It also contains all solution commands and status for a given substep of a load step. All of the `.Rnnn` files are saved at the converged state of a substep so that all element saved records are valid.

If a substep does not converge, no `.Rnnn` file is written for that substep; instead, an `.Rnnn` file from a previously converged substep is written. If the current substep number is 1, however, the `.Rnnn` file is from the last substep of the previous load step.

- `Jobname.Mnnn`

For mode-superposition transient analysis. This file contains the modal displacements, velocities, and accelerations records and solution commands for a single substep of a load step

Distributed ANSYS Considerations

For Distributed ANSYS, the `Jobname.RDB` and `Jobname.LDHI` files contain data for the entire model and are required by the master process only. Depending on the procedure used, either the `Jobname.Rnnn` file or the `Jobnamen.Rnnn` files (where *n* stands for the process rank of 0 to N-1) are required. For more information, see **Restarts in Distributed ANSYS** in the *Parallel Processing Guide*. For more information on file conventions used by Distributed ANSYS, see **Differences in General Behavior** in the *Parallel Processing Guide*.

5.8.1.1.1. Multiframe Restart Limitations

For a nonlinear static analysis, a nonlinear full transient analysis, or a linear full transient analysis, the following limitations to multiframe restart apply:

- Material properties or elements cannot be changed during a restart.

- The factorized matrix cannot be reused (**KUSE**). A new stiffness matrix and the related sparse solver workspace files (Jobname.DSPxxxx) are regenerated.
- The .Rnnn file does not save **EKILL** and **EALIVE** commands. If **EKILL** or **EALIVE** are required in the restarted session, issue them again. Restarting with tabular input (**EALIVE**,%table%) is not supported.
- The restart does not use the database file (Jobname.DB) that you save (**SAVE**). Instead, it uses the Jobname.RDB file (a database file saved automatically at the start of the first substep of the first load step).
- The .RDB file saves only the database information available at the first substep of the first load step. If other information is input after the first load step and that information is needed for the restart, it must be input again in the restart session. This situation often occurs when parameters are used (APDL); issue **PARSAV** to save the parameters during the initial run, and **PARRES** restore them in the restart. The situation also occurs when changing element REAL constants values; in this case, reissue **R** during the restart session.
- A restart cannot occur at the equation solver level (for example, the PCG iteration level). The job can only be restarted at a substep level (either transient or Newton-Raphson loop).
- Multiframe restart does not support the arc-length method (**ARCLEN**), reading and solving multiple load steps (**LSSOLVE**), or nested ***DO** loops.
- All loading and boundary conditions are stored in the Jobname.LDHI file. Upon restart, removing or deleting solid modeling loading and boundary conditions does not remove these conditions from the finite element model. Loading and boundary conditions must be removed directly from nodes and elements.
- To terminate a nonlinear analysis cleanly on a multitasking operating system, create an abort file named **Jobname.ABT** in the working directory (or on case-sensitive systems, **jobname.abt**). In the first column of the first line of the file, the word **nonlinear** should appear. If the program locates this file at the start of an equilibrium iteration, the analysis stops and can be restarted at a later time.
- Changing the number of substeps (or the time step size) upon restarting an analysis during a load step is not good practice. Change the number of substeps (or the time step size) only between load steps.
- The first time step of a restarted transient solution using the HHT algorithm (**TRNOPT**) uses the Newmark algorithm. Subsequent time steps use the HHT algorithm.
- For distributed memory parallel processing (Distributed ANSYS), it is generally recommended you use the same core count in the restart as in the original run. However, you can change the core count with some limitations. For more information, see [Restarts in Distributed ANSYS in the Parallel Processing Guide](#).

5.8.1.2. Multiframe Restart Procedure

Use the following procedure to restart an analysis:

1. Enter the program and specify the same jobname that was used in the initial run. To do so, issue the **/FILNAME** command. Enter the SOLUTION processor (**/SOLU**).

2. Determine the load step and substep at which to restart by issuing **RESCONTROL, FILE_SUMMARY**. This command prints the substep and load step information for all **.Rnnn** files in the current directory.
3. Resume the database file and indicate that this is a restart analysis by issuing **ANTYPE,,REST,LDSTEP,SUBSTEP,Action**.
4. Specify revised or additional loads as needed. Be sure to take whatever corrective action is necessary if you are restarting from a convergence failure.
5. Initiate the restart solution by issuing the **SOLVE** command. (See [Obtaining the Solution \(p. 106\)](#) for details.) You must issue the **SOLVE** command when taking any restart action, including ENDSTEP or RSTCREATE.
6. Postprocess as desired, then exit the program.

If the files **Jobname.LDHI** and **Jobname.RDB** exist, the **ANTYPE,,REST** command:

- Resumes the database **Jobname.RDB**
- Rebuilds the loading and boundary conditions from the **Jobname.LDHI** file
- Rebuilds the solution commands and status from the **.Rnnn** file, or from the **.Mnnn** file in the case of a mode-superposition transient analysis.

At this point, you can enter other commands to overwrite input restored by the **ANTYPE** command.

The loading and boundary conditions restored from the **Jobname.LDHI** are for the FE mesh. The solid model loading and boundary conditions are not stored on the **Jobname.LDHI**.

After the job is restarted, the files are affected in the following ways:

- The **.RDB** file is unchanged.
- All information for load steps and substeps past the restart point is deleted from the **.LDHI** file. Information for each new load step is then appended to the file.
- All of the **.Rnnn** or **.Mnnn** files that have load steps and substeps earlier than the restart point remain unchanged. Those files containing load steps and substeps beyond the restart point are deleted before the restart solution begins in order to prevent file conflicts.
- For nonlinear static and full transient analyses, the results file (**.RST**) is updated according to the restart. All results from load steps and substeps later than the restart point are deleted from the file to prevent conflicts, and new information from the solution is appended to the end of the results file.
- For a mode-superposition transient analysis, the reduced displacements file **.RDSP** is updated according to the restart. All results from load steps and substeps later than the restart point are deleted from the file to prevent conflicts, and new information from the solution is appended to the end of the reduced displacements file.

When a job is started from the beginning again (first substep, first load step), all of the restart files (**.RDB**, **.LDHI**, and **.Rnnn** or **.Mnnn**) in the current directory for the current jobname are deleted before the new solution begins.

You can issue the command **ANTYPE,,REST,*LDSTEP*,*SUBSTEP*,RSTCREATE** to create a results file for a particular load step and substep of an analysis. Use the **ANTYPE** command with the **OUTRES** command to write the results. A RSTCREATE session does not update or delete any of the restart files, allowing you to use RSTCREATE for any number of saved points in a session. The RSTCREATE option is not supported in mode-superposition analysis.

The following example input shows how to create a results file for a particular substep in an analysis.

Example 5.7: Creating a Results File for a Particular Substep

```
! Restart run:
/solu
antype,,rest,1,3,rstcreate !Create a results file from load
  !step 1, substep 3
outres,all,all  !Store everything into the results file
outpr,all,all  !Optional for printed output
solve  !Execute the results file creation
finish
/post1
set,,1,3  !Get results from load step 1,
  !substep 3
prnsol
finish
```

5.8.2. Modal Analysis Restart

After solving a modal analysis to obtain the eigensolution, you can restart the modal analysis to perform the following calculations:

- Expand modes of interest or all of the modes (**MXPAND** command) when the modes were not expanded during the modal analysis step.
- Generate multiple load vectors (**MODCONT** command).
- Generate residual vectors and residual response (**RESVEC** command).
- Generate enforced motion pseudo-modes (**MODCONT** command).

An eigensolution is not calculated in the restart phase.

The symmetric eigensolvers (LANB, LANPCG, SNODE, and SUBSP on the **MODOPT** command) support all of the above calculations during a modal analysis restart. The complex eigensolvers support only some of the above calculations during a restart, as described below:

- The damped eigensolver (**MODOPT,DAMP**) only supports mode expansion.
- The QR Damped eigensolver (**MODOPT,QRDAMP**) only supports mode expansion when complex solutions are requested (*Cpxmod* = ON on the **MODOPT** command). When complex solutions are not requested (*Cpxmod* = OFF), this eigensolver supports mode expansion and multiple load vector generation.
- The unsymmetric eigensolver (**MODOPT,UNSYM**) supports mode expansion and multiple load vector generation and residual response calculations.

For a modal analysis restart, the database must contain the model data as well as the modal solution data. The model in the database must match the initial model used to solve the first modal solution. In addition, the following files must be available: mode file (Jobname . MODE, as well as Jobname . LM-ODE if the unsymmetric eigensolver was used with **MODOPT**,**UNSYM**,**BOTH**), EMAT file (Jobname . EMAT), ESAV file (Jobname . ESAV), and database (Jobname . DB).

New elements can be added in the restart session to define the load vectors. These new elements must have no mass or stiffness characteristics that could affect the eigenvalues obtained from the original modal analysis. Examples of such elements include:

- Surface elements without density (**SURF153**, **SURF154**, **SURF156**).
- Follower elements (**FOLLW201**) with **KEYOPT(1) = 1** (constant direction load).
- Contact elements using the multipoint constraint (MPC) approach to define surface-based constraints. The loads must be applied to the pilot node. Refer to [Surface-Based Constraints in the Contact Technology Guide](#) for more information.

During each restart solution, the load vectors generated will replace those generated during the previous modal solution (which may be the original modal analysis or a modal analysis restart). See [Example 5.8: Modal Analysis Restart \(p. 121\)](#) for a detailed example of this procedure.

The following restrictions apply to modal analysis restart:

- A modal analysis cannot be restarted if residual vectors or residual responses (**RESVEC** command) are calculated during the first analysis.
- Modal analysis restart is not generally supported for cyclic symmetry analysis. However, a modal analysis restart may be performed to generate loads for a downstream cyclic mode-superposition harmonic analysis.

The following example demonstrates a typical command sequence for a modal analysis restart.

Example 5.8: Modal Analysis Restart

```
/filnam,casel
/prep7
et,1,plane182      ! 2D PLANE182 elements
mp,ex,1,2.0e11     ! Material Properties
mp,dens,1,7800
mp,nuxy,1,0.3

rect,0,4,0,2        ! Rectangular area
esize,0.5
type,1
mat,1
amesh,1            ! Mesh area with PLANE182 elements
allsel,all

nsel,s,loc,x,0
d,all,all,0         ! Fix the model at location x=0
nsel,all
finish

/com,
/com,  First Modal solve
/com,

/solu
antype,modal
```

```
modopt,lanb,20,0,20000 ! Use Block Lanczos, extract 20 modes in frequency range of 0 to 20000
nsel,s,loc,x,4
f,all,fx,10e5           ! First load vector (FX)
nsel,all
solve                  ! First modal solve
save,casel,db
finish
/clear,nostart

/filename,casel
resume,,db
finish

/com,
/com, Adding New elements
/com,

/prep7
et,2,surf153      ! 2D Structural effect element
keyopt,2,4,1       ! No midside node
mp,dens,2,0        ! Density set to zero so it won't affect modal analysis results
type,2
mat,2
lmesh,2
allsel,all
finish

/com,
/com, Modal Restart Analysis
/com,

/solu
antype,modal,restart   ! Restart previous modal solve to define new load vectors
fdele,all,fx           ! Delete previously defined load
modcontrol,on          ! Generate multiple load vectors
mxpand,20,,,yes        ! Expand modes
esel,s,type,,2
sfe,all,1,pres,0,20000 ! First load vector (SFE) overwrites the load vector generated
allsel,all              ! in the first modal solve (FX)
solve                  ! First solve in modal analysis restart

sfedele,all,1,pres,0   ! Delete previously defined load
nsel,s,loc,x,2
nsel,r,loc,y,2
f,all,fy,-10e5         ! Second load vector (FY)
allsel,all
solve                  ! Second solve in modal analysis restart
finish

/com,
/com, MSUP harmonic analysis by scaling the loads generated in modal solve
/com,

/solu
antype,harmonic        ! Perform Harmonic analysis
hropt,msup,20
fdele,all,fy            ! Delete loads defined in the modal analysis
fdele,all,fx
sfedele,all,1,pres,0
outres,all,all
hrout,on
harfrq,315,325          ! Excitation frequency
nsubs,20,20,20           ! Number of substeps
kbc,1
lvscale,0.5,1            ! Scale the first load vector (SFE)
lvscale,0.0,2             ! Do not scale the second load vector (FY)
solve
finish

/com,
```

```

/com, Expansion of MSUP harmonic results
/com,

/solu
expass,on
outres,all,all
numexp,all,,,yes           ! Expand results for all substeps and calculate element results
solve
finish

/com,
/com, Time history post processing of displacement and reaction force results
/com,

/post26
n1=node(4,2,0)
n2=node(0,0,0)
nsol,2,n1,u,x,ux1
rforce,3,n2,f,x,fx1
prvar,2,3
finish

```

5.9. Singular Matrices

A singular matrix exists in an analysis whenever an indeterminate or non-unique solution is possible. A negative or zero equation solver pivot value may indicate such a scenario. In some instances, it may be desirable to continue the analysis, even though a negative or zero pivot value is encountered. You can use the **PIVCHECK** command to specify whether or not to stop the analysis when this occurs.

The default behavior is to check for negative and zero pivot values (**PIVCHECK,AUTO**). When a negative or zero pivot value is encountered, the analysis may stop with an error message or may continue with a warning message, depending on the various criteria pertaining to the type of analysis being solved. You can further control this behavior with other options on the **PIVCHECK** command (see the command description for details). If **PIVCHECK,OFF** is issued, the pivots are not checked; use this command if you want your analysis to continue in spite of a negative or zero pivot value.

Currently, the program only checks for negative and zero pivot values when the sparse or PCG solver is used. If a negative or zero pivot value is encountered when using the sparse solver, the appropriate message is displayed indicating the particular node and degree of freedom where the negative or zero pivot value occurred. You can then review that part of the model to determine what caused the negative or zero pivot value (see possible causes listed below).

Note that negative pivots corresponding to Lagrange multiplier based mixed u-P elements are not checked or reported. Negative pivots arising from the u-P element formulation and related analyses can occur and lead to correct solutions.

The following conditions may cause a singular matrix in the solution process:

- **Insufficient constraints**
- **Contact elements in a model** If the contact conditions are not properly defined, a portion of the model may "break loose" or become separated before coming into contact and essentially be partially unconstrained. In this situation, adding weak springs to the unconstrained bodies or activating contact damping usually helps to prevent potential rigid body motions.
- **Nonlinear elements in a model** (such as gaps, sliders, hinges, cables, etc.). A portion of the structure may have collapsed or may have "broken loose" or become "too soft."

- **Hourglass modes** Higher-order elements (such as [SOLID186](#)) that use a reduced integration scheme may produce hourglass modes when used in a coarse mesh. This can result in a zero pivot value.
- **Negative values of material properties**, such as DENS or C, specified in a transient thermal analysis.
- **Unconstrained joints** The element arrangements may cause singularities. For example, two horizontal spar elements have an unconstrained degree of freedom in the vertical direction at the joint. A linear analysis ignores a vertical load applied at that point. Also, consider a shell element with no in-plane rotational stiffness connected perpendicularly to a beam or pipe element. There is no in-plane rotational stiffness at the joint. A linear analysis ignores an in-plane moment applied at that joint.
- **Buckling** When stress stiffening effects are negative (compressive) the structure weakens under load. If the structure weakens enough to effectively reduce the stiffness to zero or less, a singularity exists and the structure has buckled. The "NEGATIVE PIVOT VALUE - " message is generated.
- **Zero Stiffness Matrix** (on row or column). Both linear and nonlinear analyses ignore an applied load if the stiffness is exactly zero.
- **Overconstraint** As an example, overconstraint can happen when a few joint elements are defined on the same node if the joint elements are not orthogonal to each other. (See [Addressing Overconstraint Issues During Modeling](#) for additional examples.) Overconstraint can also happen when an excessive number of MPC bonded contact elements are defined at a juncture where multiple parts meet. There are ever-increasing cases of overconstraint due to increased usage of automatic model-creation tools.

When overconstraint occurs, the following phenomena often occur as well:

- Negative pivot or zero pivot values are present.
- For a nonlinear solution, the solution may converge to a (slightly) different solution each time the job is executed under the same conditions.

If the above conditions do not apply or do not help to identify the problem area, the following suggestions may help determine which (if any) part of the model is unconstrained:

- Solve the system as a modal analysis, if applicable, and look for the presence of any eigenvectors associated with zero-value eigenvalues (an indication of rigid body motion). Plotting such eigenvectors may help determine the unconstrained portions of the model.
- Review the boundary conditions in the model (including any contact pair definitions) and add arbitrary boundary conditions until any such zero pivot value messages are eliminated.

5.10. Stopping Solution After Matrix Assembly

You can terminate the solution process after the assembled global matrix file (.FULL file) has been written by using [WRFULL](#). By doing so, the equation solution process and the process of writing data to the results file are skipped. This feature can then be used in conjunction with the [HBMAT](#) command in /AUX2 to dump any of the assembled global matrices into a new file that is written in Harwell-Boeing

format. You can also use the **PSMAT** command in /AUX2 to copy the matrices to a postscript format that can be viewed graphically.

The **WRFULL** command is only valid for linear static, full harmonic, and full transient analyses when the sparse direct solver is selected. **WRFULL** is also valid for buckling and modal analyses when any mode extraction method is selected. The command is not valid for nonlinear analyses.

Chapter 6: An Overview of Postprocessing

Postprocessing means reviewing the results of an analysis. It is arguably the most important step in the analysis, because you are trying to understand how the applied loads affect your design.

After building the model and obtaining the solution, you want answers to some critical questions:

- Will the design really work when put to use?
- How high are the stresses in this region?
- How does the temperature of this part vary with time?
- What is the heat loss across this face of my model?
- How does the magnetic flux flow through this device?
- How does the placement of this object affect fluid flow?

The postprocessors can help you to answer such questions and others.

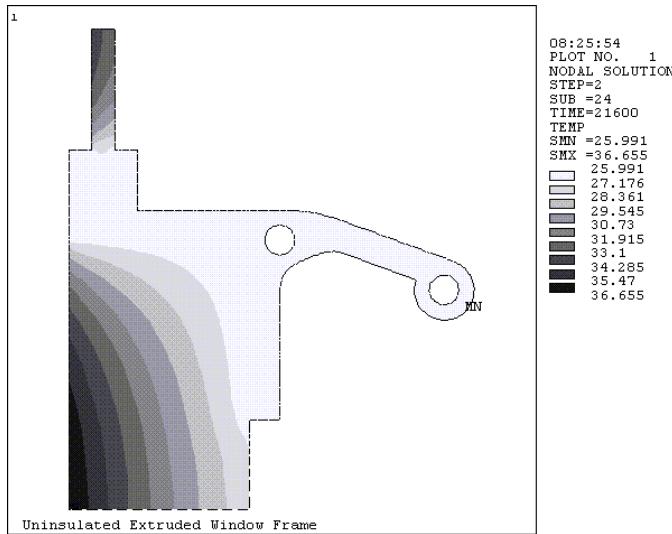
The following postprocessing topics are available:

- [6.1. Postprocessors Available](#)
- [6.2. The Results Files](#)
- [6.3. Types of Data Available for Postprocessing](#)

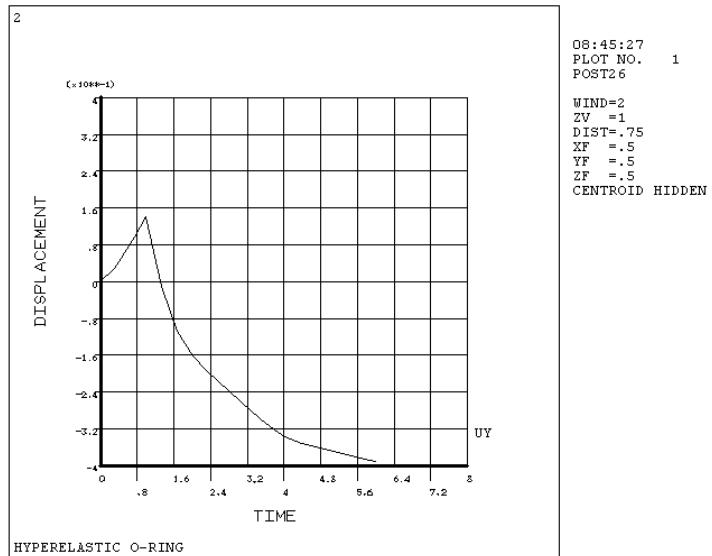
6.1. Postprocessors Available

Two postprocessors are available for reviewing your results: POST1, the general postprocessor, and POST26, the time-history postprocessor.

POST1 allows you to review the results over the entire model at specific load steps and substeps (or at specific time-points or frequencies). In a static structural analysis, for example, you can display the stress distribution for load step 3. Or, in a transient thermal analysis, you can display the temperature distribution at time = 100 seconds. Following is a typical example of a POST1 plot:

Figure 6.1: A Typical POST1 Contour Display

POST26 allows you to review the variation of a particular result item at specific points in the model with respect to time, frequency, or some other result item. In a transient magnetic analysis, for instance, you can graph the eddy current in a particular element versus time. Or, in a nonlinear structural analysis, you can graph the force at a particular node versus its deflection. [Figure 6.2: A Typical POST26 Graph \(p. 128\)](#) is shown below.

Figure 6.2: A Typical POST26 Graph

The postprocessors are *tools* for reviewing analysis results. You must still use your engineering judgment to *interpret* the results. For example, a contour display may show that the highest stress in the model is 37,800 psi. It is your responsibility to determine whether this level of stress is acceptable for your design.

6.2. The Results Files

You can direct the solver to append selected results of an analysis to the *results file* at specified intervals during solution (**OUTRES**). The name of the results file depends on the analysis discipline:

- Jobname.RST for a structural analysis and coupled-field analysis
- Jobname.RTH for a thermal and diffusion analyses
- Jobname.RMG for a magnetic field analysis

For fluid analyses, the file extension is .RST or .RTH, depending on whether structural degrees of freedom are present. (Using different file identifiers for different disciplines helps you in coupled-field analyses where the results from one analysis are used as loads for another. The *Coupled-Field Analysis Guide* presents a complete description of coupled-field analyses.)

6.3. Types of Data Available for Postprocessing

The solution phase calculates two types of results data:

- *Primary data* consist of the degree-of-freedom solution calculated at each node: displacements in a structural analysis, temperatures in a thermal analysis, magnetic potentials in a magnetic analysis, and so on (see [Table 6.1: Primary and Derived Data for Different Disciplines \(p. 129\)](#)). These are also known as nodal solution data.
- *Derived data* are those results calculated from the primary data, such as stresses and strains in a structural analysis, thermal gradients and fluxes in a thermal analysis, magnetic fluxes in a magnetic analysis, and the like. They are typically calculated for each element and may be reported at any of the following locations: at all nodes of each element, at all integration points of each element, or at the centroid of each element. Derived data are also known as element solution data, except when they are averaged at the nodes. In such cases, they become nodal solution data.

Table 6.1: Primary and Derived Data for Different Disciplines

Discipline	Primary Data	Derived Data
Structural	Displacement	Stress, strain, reaction, etc.
Thermal	Temperature	Thermal flux, thermal gradient, etc.
Magnetic	Magnetic Potential	Magnetic flux, current density, etc.
Electric	Electric Scalar Potential	Electric field, flux density, etc.
Fluid	Velocity, Pressure	Pressure gradient, heat flux, etc.
Diffusion	Concentration	Concentration gradient, diffusion flux, etc.

Chapter 7: The General Postprocessor (POST1)

Use the POST1 general postprocessor ([/POST1](#)) to review analysis results over the entire model, or selected portions of the model, for a specifically defined combination of loads at a single time (or frequency). POST1 has many capabilities, ranging from simple graphics displays and tabular listings to more complex data manipulations such as load case combinations.

The following POST1 topics are available:

- [7.1. Reading Results Data into the Database](#)
- [7.2. Reviewing Results in POST1](#)
- [7.3. Additional POST1 Postprocessing](#)

7.1. Reading Results Data into the Database

The first step in POST1 is to read data from the results file into the database. To do so, model data (nodes, elements, etc.) must exist in the database. If the database does not already contain model data, issue the **RESUME** command to read the database file, `Jobname . DB`. The database should contain the *same* model for which the solution was calculated, including the element types, nodes, elements, element real constants, material properties, and nodal coordinate systems.

Caution:

The database should contain the same set of selected nodes and elements that were selected for the solution. Otherwise, a data mismatch can occur. For more information about data mismatches, see [Appending Data to the Database \(p. 133\)](#).

Table 7.1: Saving the Database Properly for Postprocessing

When postprocessing analysis results are obtained using...	Save the database:
Linear perturbation procedures	<p>After solving.</p> <p><i>Reason:</i> The node coordinates are updated with the base analysis displacements during the solution.</p>
Contact analysis	<p>After solving.</p> <p><i>Reason:</i> The nodal connectivity of contact elements is updated and internal nodes are added during the solution.</p>
ELBOW290 elements	After the first solution.

When postprocessing analysis results are obtained using...	Save the database:
	<i>Reason:</i> The nodal connectivity of elbow elements is updated and internal nodes are added during the solution.

After model data are in the database, load the results data from the results file by issuing one of the following commands: **SET**, **SUBSET**, or **APPEND**.

7.1.1. Reading Results Data

SET reads results data over the entire model from the results file into the database for a given loading condition, replacing any data previously stored in the database. **SET** arguments identify the data to be read into the database:

SET, *Lstep*, *Sbstep*, *Fact*, *KIMG*, *TIME*, *ANGLE*, *NSET*, *ORDER*

Boundary condition data (constraints and force loads) is also read, *but only if either element nodal loads or reaction loads are available*. (See **OUTRES** for more information.) If they are not available, no boundary conditions will be available for listing or plotting. Only constraints and forces are read; surface and body loads are not updated and remain at their last specified value. If the surface and body loads were specified via tabular boundary conditions, however, they will reflect the values corresponding to this results set. Loading conditions are identified either by load step and substep or by time (or frequency).

Example 7.1: Specifying Results Data to Read

```
SET,2,5      ! Reads results for load step 2, substep 5
SET,,,3.89   ! Reads in results at TIME = 3.89 (or
!           frequency = 3.89 depending on the analysis type)
```

If you specify a *TIME* value for which no results are available, the program performs linear interpolation to calculate results for the specified time. For a nonlinear analysis, interpolation between time points usually degrades accuracy; therefore, postprocess at a *TIME* value for which a solution is available.

Some convenience operations are also available:

- *Lstep* = FIRST reads in the first substep.
- Lstep* = NEXT reads in the next substep.
- *Lstep* = LAST reads in the last substep.
- *Lstep* = LIST lists the data set number along with its corresponding load step and substep numbers.
- You can specify the data set number *NSET* (retrieved via *Lstep* = LIST) to request a specific set of results.
- *ANGLE* specifies the circumferential location for harmonic elements (structural **PLANE25**, **PLANE83**, **SHELL61**, and thermal **PLANE75**, **PLANE78**).

You can postprocess results without reading in the results data if the solution results were saved to the database file (`Jobname.DB`).

Caution:

Although you can sometimes omit reading results via **SET** if the results from the most recently solved time point are already in the database, the results may not be up to date. For example, if **OUTRES** was issued to write data at a prior time but not for the last time point, data for the prior time point will populate the database. Likewise, if a contact element was in contact at a prior time but not in contact (far-field) at the last time point, contact results for the prior time point will populate the database. In such cases, issuing **SET** is necessary to ensure that the most current results are used for postprocessing.

Distributed ANSYS can only postprocess using the results file (`Jobname.RST`), as no solution results are written to the database. If using Distributed ANSYS, issuing **SET** is necessary before postprocessing.

7.1.2. Other Options for Retrieving Results Data

Other commands also enable you to retrieve results data:

- 7.1.2.1. Defining Data to be Retrieved
- 7.1.2.2. Reading Selected Results Information
- 7.1.2.3. Appending Data to the Database

7.1.2.1. Defining Data to be Retrieved

INRES in POST1 is a companion to **OUTRES** in the PREP7 and SOLUTION processors. Where **OUTRES** controls data written to the database and the results file, **INRES** defines the type of data to be retrieved from the results file (for placement into the database via commands such as **SET**, **SUBSET**, and **APPEND**).

Although not required for postprocessing, **INRES** limits the amount of data retrieved and written to the database. As a result, postprocessing may require less time.

7.1.2.2. Reading Selected Results Information

To read a data set from the results file into the database for the *selected* portions of the model only, issue **SUBSET**. Data not specified for retrieval (**INRES**) is listed as having a zero value.

SUBSET behaves like **SET** except that it retrieves data for the selected portions of the model only. It is convenient to issue **SUBSET** to examine results data for a *portion* of the model. For example, if you are interested only in surface results, you can select the exterior nodes and elements, then issue **SUBSET** to retrieve results data for just those selected items.

7.1.2.3. Appending Data to the Database

Each time you issue **SET** or **SUBSET**, the program writes a new set of data over the data currently in the database. **APPEND** reads a data set from the results file and merges it with the existing data in the database, for the *selected model only*.

You can issue **SET**, **SUBSET**, or **APPEND** to read data from the results file into the database. The only difference between the commands or paths is *how much* or *what type* of data you want to retrieve.

To clear the database of any previous data, issue **LCZERO**. The command gives you a fresh start for further data storage.

If you set the database to zero before appending data to it, the result is the same as issuing **SUBSET**, assuming that the arguments on **SUBSET** and **APPEND** are equivalent.

All options available for **SET** are also available for **SUBSET** and **APPEND**.

By default, **SET**, **SUBSET**, and **APPEND** look for one of these results files: Jobname.RST, Jobname.RTH, or Jobname.RMG. You can specify a different file name (**FILE**) before issuing **SET**, **SUBSET**, or **APPEND**.

7.1.2.3.1. Avoiding Data Mismatch

When appending data (**APPEND**), use care not to generate a data mismatch.

Example 7.2: Data Mismatch

```
/POST1
INRES,NSOL           ! Flag data from nodal DOF solution
NSEL,S,NODE,,1,5      ! Select nodes 1 to 5
SUBSET,1              ! Write data from load step 1 to database
```

At this point, results data for nodes 1 to 5 from load step 1 are in the database.

```
NSEL,S,NODE,,6,10      ! Select nodes 6 to 10
APPEND,2              ! Merge data from load step 2 into database
NSEL,S,NODE,,1,10      ! Select nodes 1 to 10
PRNSOL,DOF            ! Print nodal DOF solution results
```

The database now contains data for *both* load steps 1 and 2. This is a data mismatch.

When printing nodal solution results (**PRNSOL**), the program indicates that you will have data from the second load step, when actually data from two different load steps now exist in the database. The load step listed is merely the one corresponding to the most *recent* load step stored.

Appending data to the database is helpful if you wish to compare results from different load steps; however, if you purposely intend to mix data, it is crucial to keep track of the *source* of the data appended.

To avoid data mismatches when solving a subset of a model that was solved previously using a different set of elements:

- Do not reselect any of the elements that were deselected for the solution currently being postprocessed, or
- Remove the earlier solution from the database (by exiting from the program between solutions or by saving the database between solutions).

7.1.3. Creating an Element Table

The *element table* serves two functions:

- It is a tool for performing arithmetic operations among results data.
- It allows access to certain element results data that are not otherwise directly accessible, such as derived data for structural line elements. (Although the **SET**, **SUBSET**, and **APPEND** commands read all requested results items into the database, not all data are directly accessible via commands such as **PLNSOL**, **PLESOL**, etc.).

Think of the element table as a spreadsheet, where each row represents an element, and each column represents a particular data item for the elements. For example, one column might contain the average SX stress for the elements, while another might contain the element volumes, while yet a third might contain the Y coordinate of the centroid for each element.

To create or erase the element table, issue the **ETABLE** command.

7.1.3.1. Filling the Element Table for Variables Identified By Name

To identify an element table column, assign a label to it via the *Lab* argument on the **ETABLE** command. The label is used as the identifier for all subsequent POST1 commands involving this variable. The data to fill the columns is identified by an *Item* name and a *Comp* (component) name, the other two arguments on the **ETABLE** command. For example, for the SX stresses, SX could be the *Lab*, S would be the *Item*, and X would be the *Comp* argument.

Some items, such as the element volumes, do not require *Comp*; in such cases, *Item* is VOLU and *Comp* is left blank. Identifying data items by an *Item*, and *Comp* if necessary, is called the "Component Name" method of filling the element table. The data which are accessible with the component name method are data generally calculated for most element types or groups of element types.

The **ETABLE** command documentation lists, in general, all the *Item* and *Comp* combinations. See the "Element Output Definitions" table in each element description in the *Element Reference* to see which combinations are valid.

Table 7.188.1: BEAM188 Element Output Definitions is an example of such a table for BEAM188.

You can use any name in the **Name** column of the table that contains a colon (:) to fill the element table via the Component Name method. The portion of the name *before* the colon should be input for the *Item* argument of the **ETABLE** command. The portion (if any) *after* the colon should be input for the *Comp* argument. The **O** and **R** columns indicate the availability of the items in the file Jobname .OUT (**O**) or in the results file (**R**): a **Y** indicates that the item is *always* available, a number refers to a table footnote which describes when the item is *conditionally* available, and a - indicates that the item is *not* available.

7.1.3.2. Filling the Element Table for Variables Identified By Sequence Number

You can load data that is not averaged, or that is not naturally single-valued for each element, into the element table. This type of data includes integration point data, *all* derived data for structural line elements (such as spars, beams, and pipes) and contact elements, *all* derived data for thermal line elements, layer data for layered elements, etc. These data are listed in the "Item and Sequence Numbers for the **ETABLE** and **ESOL** Commands" table with each element type description in the

Command Reference. Table 7.188.2: BEAM188 Item and Sequence Numbers is an example of such a table for BEAM188.

The data in the tables is broken down into *item groups*, such as LS, LEPEL, SMISC, etc. Each item within the item group has an identifying "sequence" number listed. You load these data into the element table by giving the item group as the *Item* argument on the **ETABLE** command, and the sequence number as the *Comp* argument. This is referred to as the "Sequence Number" method of filling the element table.

For some line elements, KEYOPT settings govern the amount of data calculated. This can change the sequence number of a particular data item. Therefore, in these cases a table for each KEYOPT setting is provided.

7.1.3.3. Considerations for Defining Element Tables

The **ETABLE** command works only on the *selected* elements. That is, only data for the elements you have selected are moved to the element table. By changing the selected elements between **ETABLE** commands, you can selectively fill rows of the element table.

The same Sequence Number combination may mean different data for different element types. For example, the combination SMISC,1 means P1 for **SOLID185** (pressure on face 1), and MECHPOWER for **TRANS126** (mechanical power); therefore, if your model has a combination of element types, select elements of one type (**ESEL**) before issuing the **ETABLE** command.

The element table *is not* automatically refilled (updated) when you read in a different set of results (such as for a different load step) or when you alter the results in the database (such as by a load case combination). For example, suppose your model consists of our sample elements, and you issue the following commands in POST1:

```
SET,1      ! Read in results for load step 1
ETABLE,ABC,1S,6 ! Move SDIR at end J (KEYOPT(9)=0) to the element table
                 ! under heading "ABC"
SET,2      ! Read in results for load step 2
```

At this point, the "ABC" column in the element table *still contains data for load step 1*. To refill (update) the column with load step 2 values, issue an **ETABLE,REFL** command.

- You can use the element table as a "worksheet" to do calculations among results data. This feature is described in [Additional POST1 Postprocessing \(p. 166\)](#).
- To save the element table, issue **SAVE,Fname,Ext** in POST1 or issue **/EXIT,ALL** when exiting the program. This saves the table along with the rest of the database onto the database file.
- To erase the entire element table from memory, issue **ETABLE,ERASE**. (Or issue **ETABLE,Lab,ERASE** to erase just the *Lab* column of the element table.) The **RESET** command erases the element table from memory.

7.1.4. Special Considerations for Principal Stresses

Principal stresses for **SHELL61** elements are not readily available for review in POST1. By default, the principal stresses are available for all line elements except in either of the following cases:

- You have requested an interpolated time point or angle specification on the **SET** command.

- You have performed load case operations.

In the above cases (including *all* cases for **SHELL61**), you *must* issue the command **LCOPER,LPRIN** in order to calculate the principal stresses. You can then access this data via **ETABLE** or any appropriate printing or plotting command.

7.1.5. Resetting the Database

The **RESET** command reinitializes the POST1 command defaults portion of the database without exiting POST1. The command has the same effect as leaving and re-entering the program.

7.2. Reviewing Results in POST1

After the desired results data are stored in the database, you can review them via graphics displays and tabular listings. You can also [map the results data onto a path](#) (p. 154).

7.2.1. Displaying Results Graphically

You can display the following types of graphics in POST1:

- [7.2.1.1. Contour Displays](#)
- [7.2.1.2. Deformed Shape Displays](#)
- [7.2.1.3. Vector Displays](#)
- [7.2.1.4. Path Plots](#)
- [7.2.1.5. Reaction Force Displays](#)
- [7.2.1.6. Charged Particle Traces](#)

7.2.1.1. Contour Displays

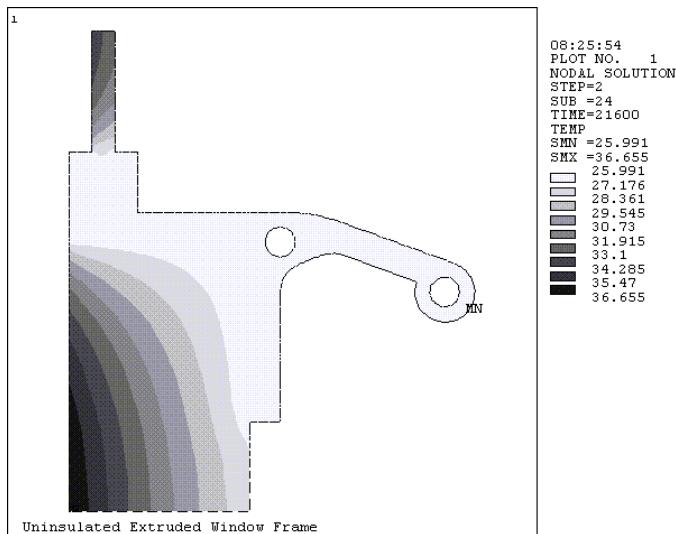
Contour displays show how a result item (such as stress, temperature, magnetic flux density, etc.) varies over the model. Four commands are available for contour displays:

- **PLNSOL** -- Displays solution results as continuous contours
- **PLESOL** -- Displays solution results as discontinuous element contours
- **PLETAB** -- Displays element table items
- **PLLS** -- Displays element table items as contoured areas along elements

The **PLNSOL** command produces contour lines that are continuous across the entire model. Use either for primary as well as derived solution data. Derived solution data, which are typically discontinuous from element to element, are averaged at the nodes so that continuous contour lines can be displayed.

Sample contour displays of primary data (TEMP) and derived data (TGX) are shown below.

```
PLNSOL,TEMP           ! Primary data: degree of freedom TEMP
```

Figure 7.1: Contouring Primary Data with PLNSOL

If PowerGraphics is enabled, you can control averaging of derived data via the **AVRES** command.

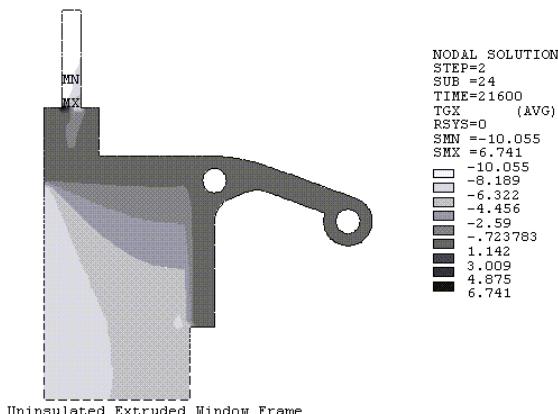
Any of the above allows you to specify whether or not results will be averaged at element boundaries where material and/or real constant discontinuities exist. For more information, see [PowerGraphics \(p. 233\)](#).

Caution:

If PowerGraphics is disabled, you cannot issue the **AVRES** command to control averaging, and the averaging operation is performed at all nodes of the selected elements without regard to the attributes of the elements connected to them. This can be inappropriate in areas of material or geometric discontinuities. When contouring derived data (which are averaged at the nodes), be sure to select elements of the same material, same thickness (for shells), same element coordinate system orientation, etc.

```
PLNSOL,TG,X           ! Derived data: thermal gradient TGX
```

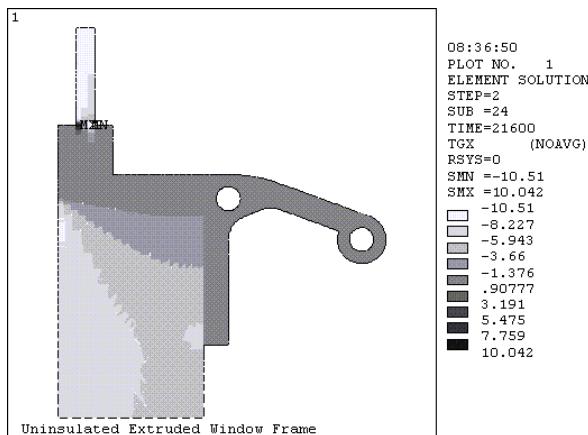
See the **PLNSOL** command description for further information.

Figure 7.2: Contouring Derived Data with PLNSOL

The **PLESOL** command produce contour lines that are discontinuous across element boundaries. Use this type of display mainly for derived solution data. For example:

```
PLESOL,TG,X
```

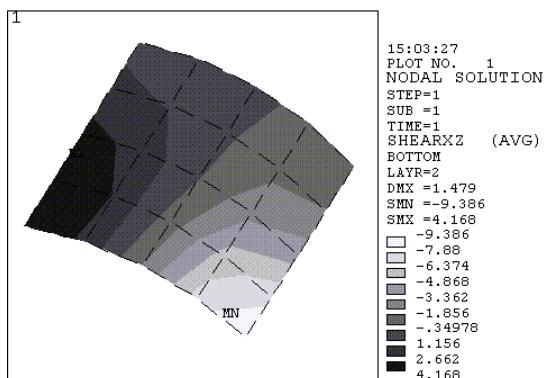
Figure 7.3: A Sample PLESOL Plot Showing Discontinuous Contours



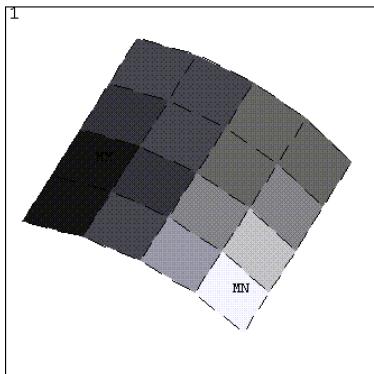
The **PLETAB** command contours data stored in the element table. The *Avglab* field on the **PLETAB** command gives you the option of averaging the data at nodes (for continuous contours) or not averaging (the default, for having discontinuous contours). The example below assumes a layered shell (**SHELL281**) model and shows the difference between averaged and nonaveraged results.

```
ETABLE,SHEARXZ,SMISC,9      ! Interlaminar shear (ILSXZ) at bottom of layer 2
PLETAB,SHEARXZ,AVG          ! Averaged contour plot of SHEARXZ
```

Figure 7.4: Averaged PLETAB Contours



```
PLETAB,SHEARXZ,NOAVG        ! Nonaveraged (default) contour plot of SHEARXZ
```

Figure 7.5: Nonaveraged PLETAB Contours

The **PLLS** command displays line element results in the form of contours. The command also requires data to be stored in the element table. This type of display is commonly used for shear and moment diagrams in beam analyses.

7.2.1.1.1. Hints and Recommendations

*Isosurface
contour
displays*

You can produce *isosurface* contour displays via the *Key* argument on the **/CTYPE** command. See [The Time-History Postprocessor \(POST26\) \(p. 193\)](#) for more information about isosurfaces.

*Averaged
principal
stresses*

By default, principal stresses at each node are calculated from averaged component stresses.

*Vector sum
data*

You can reverse this operation so that principal stresses are first calculated per element, then averaged at the nodes (**AVPRIN**). Averaging operations should *not* be performed at nodal interfaces of differing materials.

These follow the same practice as the principal stresses. By default, the vector sum magnitude (square root of the sum of the squares) at each node is calculated from averaged components.

You can reverse this operation (**AVPRIN**) so that the vector sum magnitudes are first calculated per element, *then* averaged at the nodes.

*Shell
elements or
layered shell
elements*

By default, results for shell or layered elements are assumed to be at the top surface of the shell or layer.

You can display results at the top, middle or bottom surface (**SHELL**), and display the layer number for layered elements (**LAYER**).

*von Mises
equivalent
strains (EQV):*

The effective Poisson's ratio used in computing these quantities can be changed (**AVPRIN**).

Typically, the effective Poisson's ratio is set to the input Poisson's ratio for elastic equivalent strain (item and component EPEL,EQV) and to 0.5 for inelastic strains (item and component EPPL,EQV or EPCR,EQV). For total strains (item and component EPTOT,EQV), you typically use an effective Poisson's ratio between the input Poisson's ratio and 0.5.

Alternatively, you can save the equivalent elastic strains (**ETABLE**) with the effective Poisson's ratio equal to the input Poisson's ratio and save

the equivalent plastic strains in another table using 0.5 as the effective Poisson's ratio, then combine the two table entries via **SADD** to obtain the total equivalent strain.

*Effect of **/EFACET***

When viewing continuous contour plots (**PLNSOL**), you may notice different plots according to the **/EFACET** command argument (*NUM*).

/EFACET,1 -- The contour values for the intermediate locations are interpolated based on the average of the adjacent *averaged* corner node values.

/EFACET,2 -- The midside node values are first calculated within each element, based on the average of the adjacent *nonaveraged* corner node values. The midside node values are then averaged together for a **PLNSOL** contour plot.

/EFACET,4 -- The program uses shape functions to calculate results values at three subgrid points along each element edge. The subgrid values are first calculated within each element and are then averaged together for **PLNSOL** plots; therefore, the contour values at the midside locations will differ based on the **/EFACET** argument.

In most cases, **PLESOL** contours will be the same regardless of the **/EFACET** arguments.

PLESOL contour plots differ if you change **/EFACET** settings along with **RSYS** settings (other than the default *KCN* = 0).

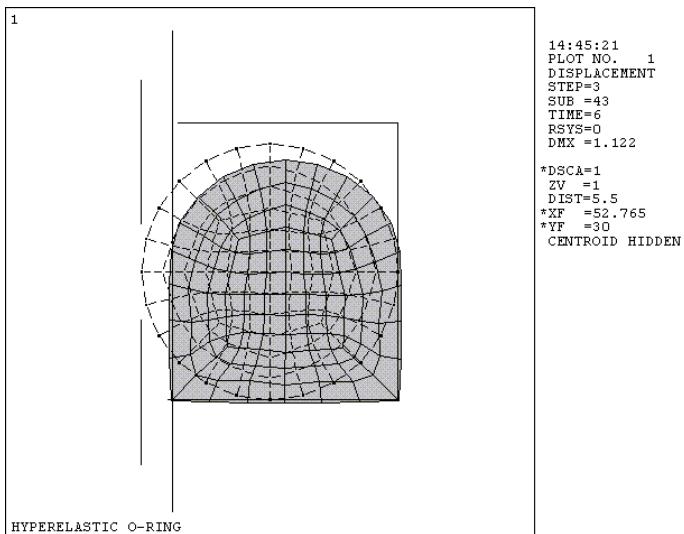
When a coordinate system other than the default global Cartesian (*KCN* = 1, 2, etc.) is selected (**RSYS**), the results are first averaged in the global Cartesian coordinate system, *then* transformed to the specified results coordinate system.

7.2.1.2. Deformed Shape Displays

You can use deformed shape displays in a structural analysis to see how the structure has deformed under the applied loads. To generate a deformed shape display, issue the **PLDISP** command.

For example, you might issue the following **PLDISP** command:

```
PLDISP,1           ! Deformed shape superimposed over undeformed shape
```

Figure 7.6: A Sample PLDISP Plot

You can change the displacement scaling by issuing the **/DSCALE** command.

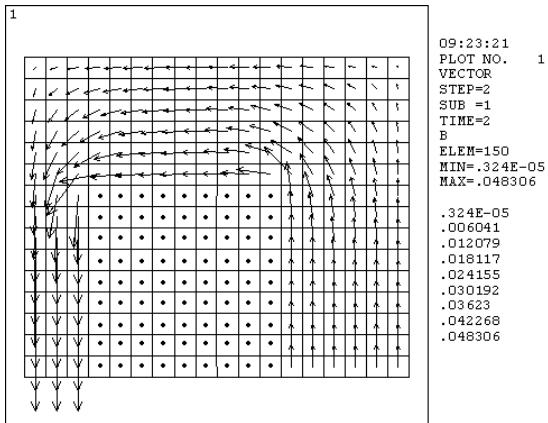
When you enter POST1, all load symbols are deactivated. Load symbols remain off if you subsequently re-enter the PREP7 or SOLUTION processors. If you activate the load symbols in POST1, the resulting display shows the loads on the deformed shape.

7.2.1.3. Vector Displays

Vector displays use arrows to show the variation of both the magnitude and direction of a *vector* quantity in the model. Examples of vector quantities are displacement (U), rotation (ROT), magnetic vector potential (A), magnetic flux density (B), thermal flux (TF), thermal gradient (TG), fluid velocity (V), principal stresses (S), etc.

To produce a vector display, issue the **PLVECT** command. For example:

```
PLVECT,B           ! Vector display of magnetic flux density
```

Figure 7.7: PLVECT Vector Plot of Magnetic Field Intensity

To scale the arrow lengths, issue the **/VSCALE** command.

You can also create your own vector quantity by specifying two or three components on the **PLVECT** command.

7.2.1.4. Path Plots

Path plots graphs that show the variation of a quantity along a predefined path through the model. To produce a path plot, you need to perform these tasks:

1. Define path attributes via the **PATH** command.
2. Define the points of the path via the **PPATH** command.
3. Map the desired quantity on to the path via the **PDEF** command.
4. Issue the **PLPATH** and **PLPAGM** commands to display the results.

For more information, see [Mapping Results onto a Path \(p. 154\)](#).

7.2.1.5. Reaction Force Displays

Reaction force displays are similar to boundary condition displays and are activated via the labels RFOR or RMOM on the **/PBC** command. Any subsequent display (produced by commands such as **NPLOT**, **E PLOT**, or **PLDISP**) will include reaction force symbols at points where DOF constraints were specified. The sum of nodal forces for a DOF belonging to a constraint equation does not include the force passing through that equation.

As with reactions, you can display *nodal* forces via labels NFOR or NMOM on the **/PBC** command. These are forces exerted by an element on its node. The sum of these forces at each node is usually zero except at constrained nodes or at nodes where loads were applied.

By default in a dynamics analysis, the force (or moment) values that are printed and plotted represent the *total* forces (sum of the static, damping, and inertial components). The **FORCE** command enables you to separate the total force into individual components in all dynamics analyses except for modal analysis and spectrum analysis. In this last case, the force type is directly input on the combination command.

7.2.1.6. Charged Particle Traces

A charged particle trace is a graphics display that shows how a charged particle travels in an electric or magnetic field. See [Creating Geometric Results Displays \(p. 249\)](#) for more information on graphic displays and see [Animation \(p. 265\)](#) for information on particle trace animation. See the [Mechanical APDL Theory Reference](#) for simplifying assumptions on electromagnetic particle tracing.

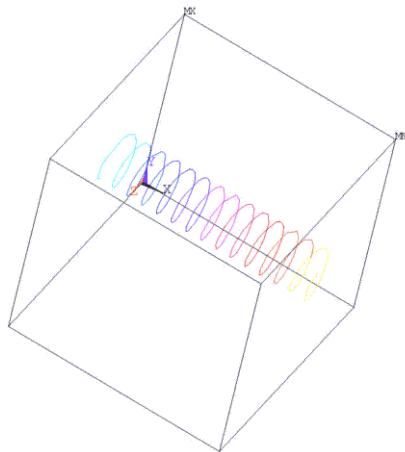
A charged particle trace requires two commands:

1. **TRPOIN** -- Defines a point on the path trajectory (starting point, ending point, or anywhere in between).
2. **PLTRAC** -- Generates the flow trace on an element display. Up to 50 points can be defined and plotted simultaneously.

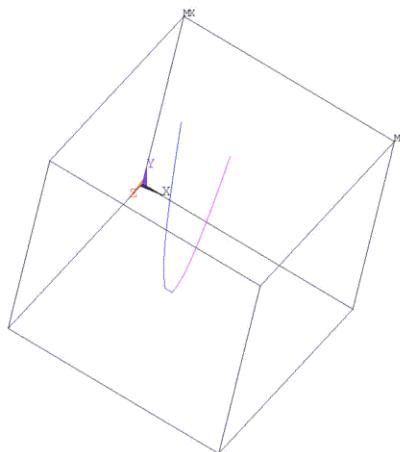
The *Item* and *Comp* fields on the **PLTRAC** command enable you to see the variation of a specified item (such as electric potential for a charged particle trace). The variation of the item is displayed along the path trajectory as a color-contoured ribbon.

Figure 7.8: Charge Particle Trace in Electric and/or Magnetic Fields

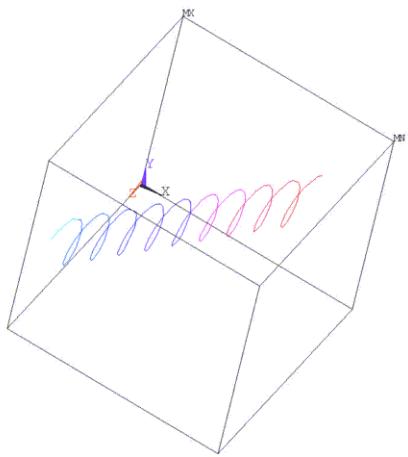
Tracing a particle moving in a pure magnetic field might look like this:



The path of that particle moving through a pure electric field might look like this:



Plotting that same particle in the presence of both the electric and magnetic fields (with E normal to B) would then look like this:



Other useful commands are:

- **TRPLIS** -- Lists trace points.
- **TRPDEL** -- Deletes trace points.
- **TRTIME** -- Defines the particle trace time interval.
- **ANFLOW** -- Generates an animated sequence of particle motion.

7.2.1.6.1. Hints and Recommendations

Three array parameters are created at the time of the particle trace: TRACPOIN, TRACDATA and TRACLABL. The parameters can be used to put the particle velocity and the elapsed time into path form.

The procedure for putting the arrays into a path named PATHNAME is:

```
*get,npts,PARM,TRACPOIN,DIM,x
PATH,PATHNAME,npts,9,1
PAPUT,TRACPOIN,POINTS
PAPUT,TRACDATA,TABLES
PAPUT,TRACLABL,LABELS
PRPATH,S,T_TRACE,VX_TRACE,VY_TRACE,VZ_TRACE,VS_TRACE
```

For charged particle traces, the variables *Chrg* and *Mass* (input via the **TRPOIN** command) have units of coulombs and kilograms, respectively, in the MKS system.

The particle tracing algorithm could lead to an infinite loop. For example, a charged particle trace could lead to an infinite circular loop. To avoid infinite loops, the **PLTRAC** command argument *MXLOOP* sets a limiting value.

Charged particle tracing can be performed after an electrostatic analysis (using only electric field), or after a magnetostatic analysis using only magnetic field or coupled magnetic and electric fields. The latter case could be done using the electric field as a body load applied either with **BFE**,EF command or with **LDREAD**,EF command.

7.2.2. Surface Operations

You can map any nodal results data onto a user defined surface in POST1. You can then perform mathematical operations on these surface results to calculate meaningful quantities, including total force or average stress for a cross section, net charge inside a closed volume, fluid mass flow rate, heat flow for a cross section, and more. You can also plot contours of the mapped results.

A full complement of surface commands is provided to perform surface operations. See [Surface Operations in the Command Reference](#).

You can define surfaces only in models containing 3-D solid elements. Shells, beams and 2-D element types are not supported. Surface creation will operate on selected, valid 3-D solid elements only and ignore other element types if they are present in your model.

The basic steps for surface operations are as follows:

1. Define the surfaces ([SUCR](#)).
2. Map the results data on the selected surfaces ([SUSEL](#), [SUMAP](#)).
3. Operate on the results ([SUEVAL](#), [SUCALC](#), [SUVECT](#)).

After your data is mapped on the surface, you can review the results via the graphical display and the tabular listing capabilities ([SUPL](#), [SUPR](#)).

Additional capabilities include archiving the surface data you create to a file or an array parameter, and recalling stored surface data. The following topics relate primarily to surface definition and usage:

[7.2.2.1. Defining the Surface](#)

[7.2.2.2. Mapping Results Data Onto a Surface](#)

[7.2.2.3. Reviewing Surface Results](#)

[7.2.2.4. Performing Operations on Mapped Surface Result Sets](#)

[7.2.2.5. Archiving and Retrieving Surface Data to a File](#)

[7.2.2.6. Archiving and Retrieving Surface Data to an Array Parameter](#)

[7.2.2.7. Deleting a Surface](#)

7.2.2.1. Defining the Surface

Define your surface via the [SUCR](#) command. The command creates your named surface (containing no more than eight characters), according to a specified category (plane, cylinder, or sphere), at a defined refinement level.

The surfaces you create fall into three categories:

- A cross section based on the current working plane
- A closed surface represented by a sphere at the current working plane origin, with a user-specified radius.
- A cylindrical surface centered at the working plane origin, and extending infinitely in the positive and negative Z directions

For *SurfType* = CPLANE, *nRefine* refers to the number of points that define the surface. If *SurfType* = CPLANE, and *nRefine* = 0, the points reside where the cutting plane section cuts through the element. Increasing *nRefine* to 1 will subdivide each surface facet into 4 subfacets, therefore increasing the number of points at which the results can be interpolated. *nRefine* can vary between 0 and 3. Increasing *nRefine* can have significant effect on memory and speed of surface operations.

/EFACET operations add to this refinement, and values greater than 1 can amplify the effect of *nRefine*. An **/EFACET** setting greater than 1 divides the elements into subelements, and *nRefine* then refines the facets of the subelements.

For *SurfType* = SPHERE, and INFC, *nRefine* is the number of divisions along a 90° arc of the sphere (default is 90, Min = 10, Max = 90).

Each time you create a surface, the following predefined geometric items are calculated and stored:

- GCX, GCY, GCZ -- Global Cartesian coordinates at each point on the surface.
- NORMX, NORMY, NORMZ -- Components of the unit normal at each point on the surface.
- DA -- Contributory area of each point.

The program uses the geometric items to perform mathematical operations with surface data; for example, DA is required to calculate surface integrals). After you create a surface, these quantities (using the predefined labels) are available for all subsequent math operations.

Issue **SUPL**,*SurfName* to display your defined surface. A maximum of 100 surfaces can exist within one model, and all operations (mapping results, math operations, etc.) will be carried out on all selected surfaces. Issue the **SUSEL** command to change the selected surface set.

See the **SUCR** command for more information of creating surfaces. When you define a cylinder (INFC), it is terminated at the geometric limits of your model. Also, any facet lying outside of those limits is discarded.

7.2.2.2. Mapping Results Data Onto a Surface

After defining a surface, issue the **SUMAP** command to map your data onto that surface. Nodal results data in the active results coordinate system is interpolated onto the surface and operated on as a *result set*. Your result sets can be made up of primary data (nodal DOF solution), derived data (stress, flux, gradients, etc.), and other results values.

Define your mapped data by providing a name for the result set, then specifying the type of data and the directional properties.

You can make the results coordinate system match the active coordinate system (used to define the path) by issuing the following pair of commands:

```
*GET,ACTSYS,ACTIVE,,CSYS
RSYS,ACTSYS
```

The first command creates a user-defined parameter named ACTSYS that holds the value defining the currently active coordinate system. The second command sets the results coordinate system to the coordinate system specified by ACTSYS.

Results mapped on to a surface do not account for discontinuities (such as material discontinuities) but are based on the currently selected set of elements. Selecting the proper set of elements is critical to valid surface operations, and improper selection can lead to failed mapping or invalid results.

To clear result sets from the selected surfaces (except GCX, GCY, GCZ, NORMX, NORMY, NORMZ, DA), issue **SUMAP,RSetname,CLEAR**. To form additional labeled result sets by operating on existing surface result sets, issue the **SUEVAL**, **SUVECT** or **SUCALC** commands.

7.2.2.3. Reviewing Surface Results

Issue the **SUPL** command to visually display your surface results, or issue **SUPR** to obtain a tabular listing.

For a single result set item, **SUPL** displays a contour plot on the selected surfaces. You can also obtain a vector plot (such as for fluid velocity vector) by using a special result set naming convention. If *SetName* is a vector prefix (that is, if SetNameX, SetNameY, and SetNameZ exist), the program plots the vectors on the surface as arrows.

Example 7.3: Vector Plot

```
SUCREATE,SURFACE1,CPLANE           ! Create a surface called "SURFACE1"
SUMAP,VELX,V,X                     ! Map x,y,z velocities with VEL as prefix
SUMAP,VELY,V,Y
SUMAP,VELZ,V,Z
SUPLOT,SURFACE1,VEL                ! Results in a vector plot of velocities
```

Display of facet outlines on the surface plots is controlled by the **/EDGE** command, similar to other postprocessing plots.

7.2.2.4. Performing Operations on Mapped Surface Result Sets

These commands are available for mathematical operations among surface result sets:

- **SUCALC** -- Add, multiply, divide, exponentiate and perform trigonometric operations on all selected surfaces.
- **SUVECT** -- Calculate the cross or dot product of two result vectors on all selected surfaces.
- **SUEVAL** -- Calculate surface integral, area weighted average, or sum of a result set on all selected surfaces. The result of this operation is an APDL scalar parameter.

7.2.2.5. Archiving and Retrieving Surface Data to a File

You can store your surface data in a file, so that when you leave POST1, it can be retrieved later. You issue the **SUSAVE** command to store your data. After saving the information for your surface, issue the **SURESU** command to retrieve it.

You can opt to archive all defined surfaces, all selected surfaces or only a specified surface. When you retrieve surface data, it becomes the currently active surface data. Any existing surface data is cleared.

Example 7.4: Archiving and Retrieving Operations

```

/post1
! define spherical surface at WP origin, with a radius of 0.75 and 10 divisions per 90 degree arc
sucreate,surf1,sphere,0.75,10
wpoff,,, -2                                ! offset working plane
! define a plane surface based on the intersection of working plane
! with the currently selected elements
sucreate,surf2,cplane

susel,s,surf1      ! select surface 'surf1'
sumap,psurf1,pres ! map pressure on surf1. Result set name "psurf1"
susel,all          ! select all surfaces
sumap,velx,v,x    ! map VX on both surfaces. Result set name "velx"
sumap,vely,v,y    ! map VY on both surfaces. Result set name "vely"
sumap,velz,v,z    ! map VZ on both surfaces. Result set name "velz"

supr              ! global status of current surface data
supl,surf1,sxsurf1 ! contour plot result set sxsurf1
supl,all,velx,1   ! contour plot result set velx on all surfaces. Plot in context of all elements in
model
supl,surf2,vel    ! vector plot of resultant velocity vector on surface "surf2"

suvect, vdotn,vel,dot,normal     ! dot product of velocity vector and surface normal
                                  ! result is stored in result set "vdotn"
sueval, flowrate, vdotn, INTG   ! integrate "vdotn" over area to get apdl parameter "flow rate"
susave,all,file,surf           ! Store defined surfaces in a file
finish

```

7.2.2.6. Archiving and Retrieving Surface Data to an Array Parameter

Writing surface data to an array allows you to perform APDL operations on your result sets. Issue the **SUGET** command to write either the interpolated results data only (default), or the results data and the geometry data to your defined parameter. The parameter is automatically dimensioned and filled with data.

7.2.2.7. Deleting a Surface

To delete one or more surfaces, along with the mapped results on those surfaces, issue the **SUDEL** command. You can delete all surfaces, or individual surfaces by name. Issue the **SUPR** command to review the current list of surface names.

7.2.3. Listing Results in Tabular Form

An effective method for documenting analysis results (for use in reports or presentations, for example) is to produce tabular listings in POST1. Listing options are available for nodal and element solution data, reaction data, element table data, and more:

- [7.2.3.1. Listing Nodal and Element Solution Data](#)
- [7.2.3.2. Listing Reaction Loads and Applied Loads](#)
- [7.2.3.3. Listing Element Table Data](#)
- [7.2.3.4. Other Listings](#)
- [7.2.3.5. Sorting Nodes and Elements](#)
- [7.2.3.6. Custom Tabular Listings](#)

7.2.3.1. Listing Nodal and Element Solution Data

To list specified nodal solution data (primary as well as derived), issue the **PRNSOL** command.

Example 7.5: PRNSOL,S Listing

```

PRINT S      NODAL SOLUTION PER NODE

***** POST1 NODAL STRESS LISTING *****

LOAD STEP=      5   SUBSTEP=      2
TIME=    1.0000   LOAD CASE=     0

THE FOLLOWING X,Y,Z VALUES ARE IN GLOBAL COORDINATES

NODE      SX        SY        SZ        SXY        SYZ        SXZ
 1    148.01    -294.54    .00000E+00   -56.256    .00000E+00    .00000E+00
 2    144.89    -294.83    .00000E+00    56.841    .00000E+00    .00000E+00
 3    241.84     73.743    .00000E+00   -46.365    .00000E+00    .00000E+00
 4    401.98    -18.212    .00000E+00   -34.299    .00000E+00    .00000E+00
 5    468.15    -27.171    .00000E+00   .48669E-01    .00000E+00    .00000E+00
 6    401.46    -18.183    .00000E+00    34.393    .00000E+00    .00000E+00
 7    239.90     73.614    .00000E+00    46.704    .00000E+00    .00000E+00
 8    -84.741    -39.533    .00000E+00    39.089    .00000E+00    .00000E+00
 9     3.2868    -227.26    .00000E+00    68.563    .00000E+00    .00000E+00
10   -33.232    -99.614    .00000E+00    59.686    .00000E+00    .00000E+00
11   -520.81    -251.12    .00000E+00   .65232E-01    .00000E+00    .00000E+00
12   -160.58    -11.236    .00000E+00    40.463    .00000E+00    .00000E+00
13   -378.55     55.443    .00000E+00    57.741    .00000E+00    .00000E+00
14   -85.022    -39.635    .00000E+00   -39.143    .00000E+00    .00000E+00
15   -378.87     55.460    .00000E+00   -57.637    .00000E+00    .00000E+00
16   -160.91    -11.141    .00000E+00   -40.452    .00000E+00    .00000E+00
17   -33.188    -99.790    .00000E+00   -59.722    .00000E+00    .00000E+00
18     3.1090    -227.24    .00000E+00   -68.279    .00000E+00    .00000E+00
19    41.811     51.777    .00000E+00   -66.760    .00000E+00    .00000E+00
20   -81.004     9.3348    .00000E+00   -63.803    .00000E+00    .00000E+00
21    117.64    -5.8500    .00000E+00   -56.351    .00000E+00    .00000E+00
22   -128.21     30.986    .00000E+00   -68.019    .00000E+00    .00000E+00
23    154.69    -73.136    .00000E+00   .71142E-01    .00000E+00    .00000E+00
24   -127.64    -185.11    .00000E+00   .79422E-01    .00000E+00    .00000E+00
25    117.22    -5.7904    .00000E+00    56.517    .00000E+00    .00000E+00

26   -128.20     31.023    .00000E+00    68.191    .00000E+00    .00000E+00
27    41.558     51.533    .00000E+00    66.997    .00000E+00    .00000E+00
28   -80.975     9.1077    .00000E+00    63.877    .00000E+00    .00000E+00

MINIMUM VALUES
NODE      11          2          1          18          1          1
VALUE   -520.81   -294.83   .00000E+00   -68.279   .00000E+00   .00000E+00

MAXIMUM VALUES
NODE       5          3          1          9          1          1
VALUE    468.15    73.743   .00000E+00    68.563   .00000E+00   .00000E+00

```

To list specified results for selected elements, issue the **PRESOL** command. To obtain line element solution printout, specify the ELEM option. The command returns all applicable element results for the selected elements.

7.2.3.2. Listing Reaction Loads and Applied Loads

You have several options in POST1 for listing reaction loads and applied loads. The **PRRSOL** command lists reactions at constrained nodes in the selected set.

In all dynamics analyses except for spectrum analyses, the **FORCE** command used with **PRRFOR** (not menu accessible) dictates which component of the reaction data is listed: total (default), static, damping, or inertia. In spectrum analyses, the force type is input on the combination command.

The **PRNLD** command lists the summed element nodal loads for the selected nodes, except for any zero values.

The output of summed nodal forces (**PRRFOR**, **NFORCE**, **FSUM** commands, etc.) containing a node that belongs to a constraint equation or part of an MPC bonded contact region (or pilot node) does not include the force redistribution due to the constraints. Issue **PRRSOL**, as it contains the redistribution.

Listing reaction loads and applied loads is a good way to check equilibrium. It is always good practice to check a model's equilibrium after solution. (The sum of the applied loads in a given direction should equal the sum of the reactions in that direction. If the sum of the reaction loads is not what you expect, check your loading to see if it was applied properly.)

The presence of coupling or constraint equations can induce either an actual or apparent loss of equilibrium. Actual loss of load balance can occur for poorly specified couplings or constraint equations (a usually undesirable effect). Coupled sets created by **CPINTF** and constraint equations created by **CEINTF** or **CERIG** will in nearly all cases maintain actual equilibrium. Other cases where you may see an apparent loss of equilibrium are: (a) 4-node shell elements where all 4 nodes do no lie in an exact flat plane, (b) elements with an elastic foundation specified, and (c) unconverged nonlinear solutions. See the *Mechanical APDL Theory Reference*.

The **FSUM** command calculates and lists the force and moment summation for the selected set of nodes.

Example 7.6: FSUM Output

```
*** NOTE ***
Summations based on final geometry and will not agree with solution
reactions.

***** SUMMATION OF TOTAL FORCES AND MOMENTS IN GLOBAL COORDINATES *****
FX = .1147202
FY = .7857315
FZ = .0000000E+00
MX = .0000000E+00
MY = .0000000E+00
MZ = 39.82639

SUMMATION POINT= .00000E+00 .00000E+00 .00000E+00
```

The **NFORCE** command provides the force and moment summation for each selected node, in addition to an overall summation.

Example 7.7: NFORCE Output

```
***** POST1 NODAL TOTAL FORCE SUMMATION *****

LOAD STEP= 3 SUBSTEP= 43

THE FOLLOWING X,Y,Z FORCES ARE IN GLOBAL COORDINATES
```

NODE	FX	FY	FZ
1	-.4281E-01	.4212	.0000E+00
2	.3624E-03	.2349E-01	.0000E+00
3	.6695E-01	.2116	.0000E+00
4	.4522E-01	.3308E-01	.0000E+00
5	.2705E-01	.4722E-01	.0000E+00
6	.1458E-01	.2880E-01	.0000E+00
7	.5507E-02	.2660E-01	.0000E+00
8	-.2080E-02	.1055E-01	.0000E+00
9	-.5551E-03	-.7278E-02	.0000E+00
10	.4906E-03	-.9516E-02	.0000E+00

*** NOTE ***

Summations based on final geometry and will not agree with solution reactions.

***** SUMMATION OF TOTAL FORCES AND MOMENTS IN GLOBAL COORDINATES *****

	FX	FY	FZ
	.1147202	.7857315	.0000000E+00
	.0000000E+00	.0000000E+00	.0000000E+00
	.0000000E+00	.0000000E+00	.0000000E+00
	39.82639		

SUMMATION POINT= .00000E+00 .00000E+00 .00000E+00

The **SPOINT** command defines the point (any point other than the origin) about which moments are summed.

7.2.3.3. Listing Element Table Data

To list specified data stored in the element table, issue the **PRETAB** command

To list the sum of each column in the element table, issue the **SSUM** command.

Example 7.8: PRETAB and SSUM Output

***** POST1 ELEMENT TABLE LISTING *****

STAT	CURRENT	CURRENT	CURRENT
ELEM	SBYTI	SBYBI	MFORYI
1	.95478E-10	-.95478E-10	-2500.0
2	-3750.0	3750.0	-2500.0
3	-7500.0	7500.0	-2500.0
4	-11250.	11250.	-2500.0
5	-15000.	15000.	-2500.0
6	-18750.	18750.	-2500.0
7	-22500.	22500.	-2500.0
8	-26250.	26250.	-2500.0
9	-30000.	30000.	-2500.0
10	-33750.	33750.	-2500.0
11	-37500.	37500.	2500.0
12	-33750.	33750.	2500.0
13	-30000.	30000.	2500.0
14	-26250.	26250.	2500.0
15	-22500.	22500.	2500.0
16	-18750.	18750.	2500.0
17	-15000.	15000.	2500.0
18	-11250.	11250.	2500.0
19	-7500.0	7500.0	2500.0
20	-3750.0	3750.0	2500.0

```

MINIMUM VALUES
ELEM      11          1          8
VALUE    -37500.   -.95478E-10 -2500.0

MAXIMUM VALUES
ELEM      1          11         11
VALUE    .95478E-10 37500.     2500.0

SUM ALL THE ACTIVE ENTRIES IN THE ELEMENT TABLE

TABLE LABEL      TOTAL
SBYTI    -375000.
SBYBI    375000.
MFORYI    .552063E-09

```

7.2.3.4. Other Listings

You can list other types of results via the following commands:

- **PRVECT** -- Lists the magnitude and direction cosines of specified vector quantities for all selected elements.
- **PRPATH** -- Calculates and then lists specified data along a predefined geometry path in the model. You must define the path and map the data onto the path; see [Mapping Results onto a Path \(p. 154\)](#).
- **PRSECT** --Calculates and then lists linearized stresses along a predefined path.
- **PRERR** -- Lists the percent error in energy norm for all selected elements.
- **PRITER** -- Lists iteration summary data.
- **PRENERGY** -- Lists the energies of the entire model or the energies of the specified components.

7.2.3.5. Sorting Nodes and Elements

By default, all tabular listings usually progress in ascending order of node numbers or element numbers. You can change this by first sorting the nodes or elements according to a specified result item. The **NSORT** command sorts nodes based on a specified nodal solution item, and **ESORT** sorts elements based on a specified item stored in the element table.

Example 7.9: Sorting Nodes

```

NSEL,...           ! Selects nodes
NSORT,S,X        ! Sorts nodes based on SX
PRNSOL,S,COMP    ! Lists sorted component stresses

```

Example 7.10: PRNSOL,S and Output After NSORT

```

PRINT S      NODAL SOLUTION PER NODE

***** POST1 NODAL STRESS LISTING *****

LOAD STEP=      3   SUBSTEP=     43
TIME=       6.0000    LOAD CASE=     0

```

THE FOLLOWING X,Y,Z VALUES ARE IN GLOBAL COORDINATES

NODE	SX	SY	SZ	SXY	SYZ	SXZ
111	-.90547	-1.0339	-.96928	-.51186E-01	.00000E+00	.00000E+00
81	-.93657	-1.1249	-.1.0256	-.19898E-01	.00000E+00	.00000E+00
51	-1.0147	-.97795	-.98530	.17839E-01	.00000E+00	.00000E+00
41	-1.0379	-1.0677	-.1.0418	-.50042E-01	.00000E+00	.00000E+00
31	-1.0406	-.99430	-.1.0110	.10425E-01	.00000E+00	.00000E+00
11	-1.0604	-.97167	-.1.0093	-.46465E-03	.00000E+00	.00000E+00
71	-1.0613	-.95595	-.1.0017	.93113E-02	.00000E+00	.00000E+00
21	-1.0652	-.98799	-.1.0267	.31703E-01	.00000E+00	.00000E+00
61	-1.0829	-.94972	-.1.0170	.22630E-03	.00000E+00	.00000E+00
101	-1.0898	-.86700	-.1.0009	-.25154E-01	.00000E+00	.00000E+00
1	-1.1450	-1.0258	-.1.0741	.69372E-01	.00000E+00	.00000E+00

MINIMUM VALUES

NODE	1	81	1	111	111	111
VALUE	-1.1450	-1.1249	-1.0741	-.51186E-01	.00000E+00	.00000E+00

MAXIMUM VALUES

NODE	111	101	111	1	111	111
VALUE	-.90547	-.86700	-.96928	.69372E-01	.00000E+00	.00000E+00

To restore the original order of nodes or elements, issue the **NUSORT** and **EUSORT** commands.

7.2.3.6. Custom Tabular Listings

In some situations, you may need to customize result listings to your specifications. The **/STITLE** command enables you to define up to four subtitles which will be displayed on output listings along with the main title. Other commands available for output customization are: **/FORMAT**, **/HEADER**, and **/PAGE**. They control such things as the number of significant digits, the headers that appear at the top of listings, the number of lines on a printed page, etc. These controls apply only to the **PRRSOL**, **PRNSOL**, **PRESOL**, **PRETAB**, and **PRPATH** commands.

7.2.4. Mapping Results onto a Path

One of the most powerful and useful features of POST1 is its ability to map virtually any results data onto an arbitrary path through your model. This enables you to perform many arithmetic and calculus operations along this path to calculate meaningful results: stress intensity factors and J-integrals around a crack tip, the amount of heat crossing the path, magnetic forces on an object, and so on. A useful side benefit is that you can see, in the form of a graph or a tabular listing, how a result item varies along the path.

You can define paths in models containing solid elements (2-D or 3-D) or shell elements.

Three general steps are necessary for reviewing results along a path:

1. Define the path attributes (**PATH**).
2. Define the path points (**PPATH**).
3. Interpolate (map) results data along the path (**PDEF**).

After the data are interpolated, you can review them via graphics displays (**PLPATH** or **PLPAGM**) and tabular listings or perform mathematical operations such as addition, multiplication, integration, etc.

Advanced mapping techniques to handle material discontinuities and accurate computations are offered in the **PMAP** command (issue this command prior to **PDEF**).

Other path operations you can perform include archiving paths or path data to a file or an array parameter and recalling an existing path with its data. The next few topics discuss path definition and usage.

7.2.4.1. Defining the Path

To define a path, first define the path environment and then the individual path points. Define the path by picking nodes, by picking locations on the working plane, or by filling out a table of specific coordinate locations. Create the path by picking or by issuing the **PATH** and **PPATH** commands.

Provide the following information for the **PATH** command:

- A path name (containing up to eight characters).
- The number of path points (between 2 and 1000). If using graphical picking, the number of path points equals the number of picked points.
- The number of sets of data which may be mapped to this path. (Four is the minimum; default is 30. There is no maximum.)
- The number of divisions between adjacent points. (Default = 20, no maximum.)
- Specify a coordinate system (*CS*) for geometry interpolation (**PPATH**).

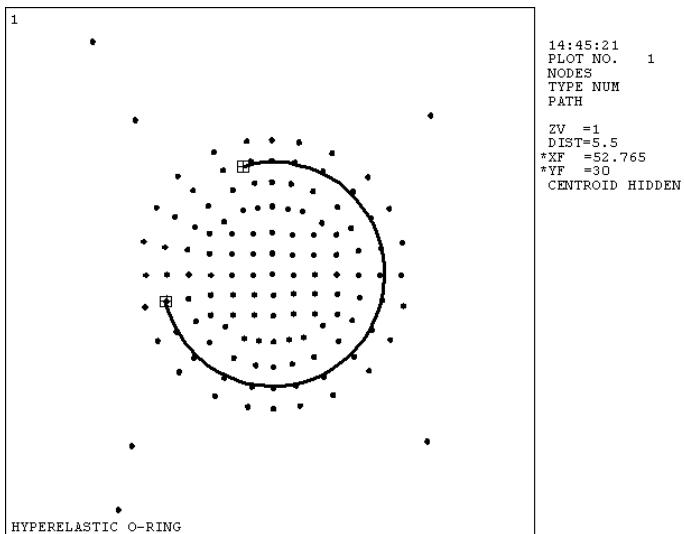
To see the status of path settings, issue a **PATH,STATUS** command.

The **PATH** and **PPATH** commands define the path geometry in the active **CSYS** coordinate system. If the path is a straight line or a circular arc, only the two end nodes are required (unless you want highly accurate interpolation, which may require more path points or divisions).

If necessary, issue the **CSCIR** command to move the coordinate singularity point before defining the path.

To display the path that you have defined, first [interpolate data along the path \(p. 156\)](#), then issue the **/PBC,PATH,1** command, followed by the **NPLOT** or **EPLLOT** command.

The program displays the path as a series of straight line segments. The path shown here was defined in a cylindrical coordinate system:

Figure 7.9: A Node Plot Showing the Path

7.2.4.2. Using Multiple Paths

A maximum of 100 paths can exist within one model. However, only one path at a time can be the current path. To change the current path, choose the **PATH,NAME** command. Do not specify any other arguments on the **PATH** command. The named path will become the new current path.

7.2.4.3. Interpolating Data Along the Path

The path must already be defined. Issue the **PDEF** and **PVECT** commands to interpolate data along the path.

The **PDEF** command enables you to interpolate virtually any results data along the path *in the active results coordinate system*: primary data (nodal DOF solution), derived data (stresses, fluxes, gradients, etc.), element table data, and so on. The rest of this discussion (and in other documentation) refers to an interpolated item as a *path item*. For example, to interpolate thermal flux in the X direction along a path, the command would be as follows:

```
PDEF,XFLUX,TF,X
```

The XFLUX value is an arbitrary user-defined name assigned to the path item. TF and X together identify the item as the thermal flux in the X direction.

You can make the results coordinate system match the active coordinate system (used to define the path) by issuing the following pair of commands:

```
*GET,ACTSYS,ACTIVE,,CSYS
RSYS,ACTSYS
```

The first command creates a user-defined parameter (ACTSYS) that holds the value defining the currently active coordinate system. The second command sets the results coordinate system to the coordinate system specified by ACTSYS.

7.2.4.4. Mapping Path Data

POST1 uses $\{nDiv(nPts-1) + 1\}$ interpolation points to map data onto the path (where $nPts$ is the number of points on the path and $nDiv$ is the number of path divisions between points (**PATH**)). When you create the first path item, the program automatically interpolates the following additional geometry items: XG, YG, ZG, and S. The first three are the global Cartesian coordinates of the interpolation points and S is the path length from the starting node. These items are useful when performing mathematical operations with path items (for instance, S is required to calculate line integrals). To accurately map data across material discontinuities, use the *DISCON = MAT* option on the **PMAP** command.

To clear path items from the path (except XG, YG, ZG, and S), issue **PDEF,CLEAR**. To form additional labeled path items by operating on existing path items, issue the **PCALC** command.

The **PVECT** command defines the normal, tangent, or position vectors along the path. A Cartesian coordinate system must be active for this command. For example, the command shown below defines a unit vector tangent to the path at each interpolation point.

```
PVECT,TANG,TTX,TTY,TTZ
```

TTX, TTY, and TTZ are user-defined names assigned to the X, Y, and Z components of the vector. You can use these vector quantities for fracture mechanics J-integral calculations, dot and cross product operations, etc. For accurate mapping of normal and tangent vectors, use the ACCURATE option on the **PMAP** command. Issue the **PMAP** command prior to mapping data.

7.2.4.5. Reviewing Path Items

To obtain a graph of specified path items versus path distance, issue the **PLPATH** command.

To get a tabular listing of specified path items, issue the **PRPATH** command.

You can control the path distance range (the abscissa) for **PLPATH** and **PRPATH** via the **PRANGE** command. Path-defined variables can also be used in place of the path distance for the abscissa item in the path display.

To calculate and review linearized stresses along a path defined by the first two nodes on the **PPATH** command, you can issue either the **PLSECT** or **PRSECT** command. Typically, you use them in pressure vessel applications to separate stresses into individual components: membrane, membrane plus bending, etc. The path is defined in the active display coordinate system.

You can display a path data item as a color area contour display along the path geometry. The contour display offset from the path may be scaled for clarity. To produce such a display, issue the **PLPAGM** command.

7.2.4.6. Performing Mathematical Operations among Path Items

These commands are available for mathematical operations among path items:

- **PCALC** -- Adds, multiplies, divides, exponentiates, differentiates, and integrates path items.
- **PDOT** -- Calculates the dot product of two path vectors.

- **PCROSS** -- Calculates the cross product or two path vectors.

7.2.4.7. Archiving and Retrieving Path Data to a File

If you wish to retain path data when you leave POST1, you must store it in a file or an array parameter so that you can retrieve it later. Select a path or multiple paths (**PSEL**), then write the current path data to a file (**PASAVE**).

You can then retrieve path information from the saved file and store the data as the currently active path data (**PARESU**).

You can archive or fetch only the path data (data mapped to path [**PDEF**]) or the path points (defined via **PPATH**). When you retrieve path data, it becomes the currently active path data (existing active path data is replaced). If you issue **PARESU** and have multiple paths, the first path from the list becomes the currently active path.

Example 7.11: Archiving and Retrieving Path Data to a File

```
/post1
path,radial,2,30,35      ! Define path name, No. points, No. sets, No. divisions
ppath,1,,.2               ! Define path by location
ppath,2,,.6
pmap,,mat                 ! Map at material discontinuities
pdef,sx,s,x                ! Interpret radial stress
pdef,sz,s,z                ! Interpret hoop stress
plpath,sx,sz                ! Plot stresses
pasave                      ! Store defined paths in a file
finish
/post1
paresu                     ! retrieve path data from file
plpagn,sx,,node             ! plot radial stresses on the path
finish
```

7.2.4.8. Archiving and Retrieving Path Data to an Array Parameter

Writing path data to an array is useful if you want to map a charged particle trace onto a path (**PLTRAC**). If you wish to retain path data in an array parameter, issue the **PAGET** command to write current path data to an array variable.

To retrieve path information from an array variable and store the data as the currently active path data, issue a **PAPUT**, **PARRAY**, **POPT** command.

You can archive or fetch only the path data (data mapped to path [**PDEF**]) or the path points (defined via **PPATH**). The setting for the *POPT* argument on **PAGET** and **PAPUT** determines what is stored or retrieved. You must retrieve path points prior to retrieving path data and labels. When you retrieve path data, it becomes the currently active path data (existing active path data is replaced).

Example 7.12: Archiving and Retrieving Path Data to an Array Parameter

Sample input and output are shown below.

```
/post1
path,radial,2,30,35      ! Define path name, No. points, No. sets, No. divisions
ppath,1,,.2               ! Define path by location
ppath,2,,.6
pmap,,mat                 ! Map at material discontinuities
pdef,sx,s,x                ! Interpret radial stress
```

```

pdef,sz,s,z           ! Interpret hoop stress
plpath,sx,sz          ! Plot stresses
paget,radpts,points   ! Archive path points in array "radpts"
paget,raddat,table    ! Archive path data in array "raddat"
paget,radlab,label    ! Archive path labels in array "radlab"
finish
/post1
*get,npts,parm,radpts,dim,x ! Retrieve number of points from array "radpts"
*get,ndat,parm,raddat,dim,x ! Retrieve number of data points from array "raddat"
*get,nset,parm,radlab,dim,x ! Retrieve number of data labels from array "radlab"
ndiv=(ndat-1)/(npts-1)      ! Calculate number of divisions
path,radial,npts,ns1,ndiv   ! Create path "radial" with number of sets ns1>nset
paput,radpts,points       ! Retrieve path points
paput,raddat,table        ! Retrieve path data
paput,radlab,labels       ! Retrieve path labels
plpagn,sx,,node           ! Plot radial stresses on the path
finish

```

Figure 7.10: PLPATH Display Showing Stress Discontinuity at a Material Interface

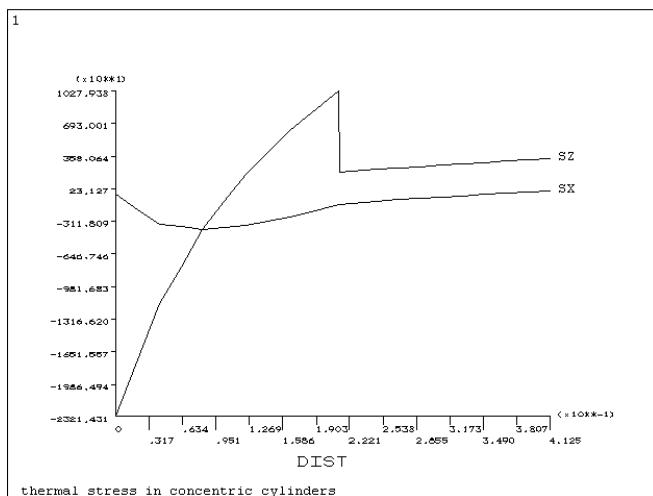
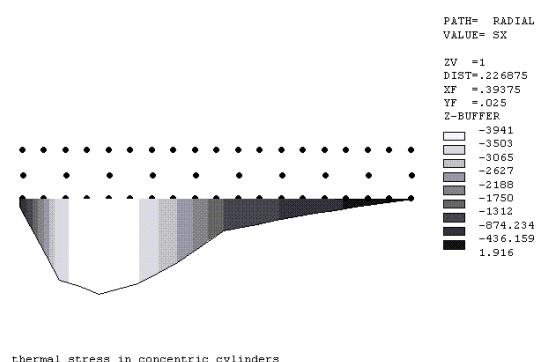


Figure 7.11: PLPAGM Display



7.2.4.9. Deleting a Path

To delete one or more paths, issue a **PADELETE**, **DELOPT** command.

You can delete all paths or a single named path. To review the current list of path names, issue a **PATH,STATUS** command.

7.2.5. Estimating Solution Error

One of the main concerns in a finite element analysis is the adequacy of the finite element mesh. Is the mesh fine enough for good results? If not, what portion of the model should be remeshed? You can get answers to such questions with the error-estimation technique, which estimates the amount of solution error due specifically to mesh discretization. This technique is available only for linear structural and linear/nonlinear thermal analyses using 2-D or 3-D solid elements or shell elements.

In the postprocessor, the program calculates an *energy error* for each element in the model. The energy error is similar in concept to the strain energy. The *structural* energy error (labeled SERR) is a measure of the discontinuity of the stress field from element to element, and the *thermal* energy error (TERR) is a measure of the discontinuity of the heat flux from element to element. Using SERR and TERR, the program calculates a *percent error in energy norm* (SEPC for structural percent error, TEPC for thermal percent error).

Error estimation is based on stiffness and conductivity matrices that are evaluated at the reference temperatures (TREF). Error estimates, therefore, can be incorrect for elements with temperature-dependent material properties if those elements are at a temperature that is significantly different than TREF.

In many cases, you can significantly increase program speed by suppressing error estimation. This improved performance is most evident when error estimation is turned off in a thermal analysis. Therefore, you may want to use error estimation only when needed, such as when you wish to determine if your mesh is adequate for good results.

You may turn error estimation off issuing **ERNORM,OFF**. By default, error estimation is active. Since the value set by the **ERNORM** command is not saved on Jobname.DB, you will need to reissue **ERNORM,OFF** if you wish to again deactivate error estimation after resuming an analysis.

In POST1 then, you can list SEPC and TEPC for all selected elements via the **PRERR** command. The value of SEPC or TEPC indicates the relative error due to a particular mesh discretization. To find out where you should refine the mesh, simply produce a contour display of SERR or TERR and look for high-error regions.

7.2.6. Using the Results Viewer to Access Results File Data

The Results Viewer is a compact toolbar for viewing your analysis results. Selecting the Results Viewer disables much of the standard GUI functionality. Many of these operations are not available because of PowerGraphics limitations. However, a good deal of the POST1 functionality is contained in the Result Viewer menu structure, and in the right and middle mouse button context sensitive menus that are accessible when you use the Results Viewer.

Figure 7.12: The Results Viewer



You can use the Results Viewer to access any data stored in a valid results file (such as *.RST, *.RTH, and *.RMG). Because the viewer can access results data without loading the entire database file, it is an ideal location from which to compare data from many different analyses.

Even if you have loaded other results files, you can return to your original analysis. You can reload the original results file from the current analysis before closing the Results Viewer.

7.2.6.1. The Results Viewer Layout

The Results Viewer has three primary control areas:

7.2.6.1.1. Main Menu

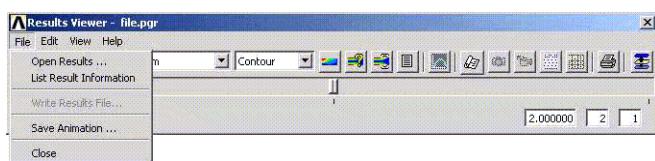
7.2.6.1.2. Toolbar

7.2.6.1.3. Step/Sequence Data Access Controls

7.2.6.1.1. Main Menu

The Main Menu is located along the top of the Results Viewer and provides access to the File, Edit, View and Help menus. The following functions can be accessed from each of these headings.

Figure 7.13: Results Viewer File Menu



File --

Open Results --

Open any results file from any location on your file system.

List Result Information --

Displays a list of all results data included in the current file.

Save Animation --

Saves an animation file (*.anim, *.avi) to a specified location. Animations created from the Results Viewer are not written to the database.

Close --

Closes the Results Viewer and reverts to the standard GUI.

If you have opened the results file from another analysis, return to your original file *before* closing the Results Viewer.

Edit --

Select subsets of the model based on model attributes (material, element type, real ID, and element component). This selection activates the appropriate "Element Select" widget, or picking window.

Figure 7.14: Results Viewer View Menu**View -****Real Data -**

Display the real data from your analysis in the graphics window. This selection is grayed out when only real data exists for your analysis.

Imaginary Data -

Display the imaginary data for your analysis in the graphics window. This selection is grayed out when no valid imaginary data exists.

Expanded Model -

Perform any periodic/cyclic, modal cyclic, and axisymmetric expansions available via the **/EXPAND** command.

Attributes -

Access the attributes of your model (based on the conventions available via the **/PNUM** command).

7.2.6.1.2. Toolbar

The Results Viewer toolbar is located across the middle of the Results Viewer. You can choose the type of results data to plot, and designate how the information should be plotted. You can also query results data from the graphics display, create animations, generate results listings, plot or generate file exports of your screen contents, or open the HTML Report Generator to construct a report on the results data.

Figure 7.15: Results Viewer Toolbar**Element Plot**

The first item on the toolbar is the element plot icon. This is the only model display available.

Result Item Selector

This drop-down menu allows you to choose from the various types of data. The choices displayed may not always be available in your results file.

Plot Type Selector

You left mouse click and hold down on this button and it produces a "fly out" that allows you to access the four types of results plots available - Nodal, Element, Vector and Trace.

Query Results

You use the query tool to retrieve results data directly from selected areas in the graphics window. The picking menu is displayed, allowing you to select multiple items. The information is displayed only for the current view.

Animate Results

You can create animations based on the information you have included in the results file. Because this information is created as a separate file, it is not saved within the results file. You must save the individual animations via the Results Viewer's Main Menu, Save Animation function.

List Results

The list results button creates a text listing of all of the nodal results values for the selected sequence number and result item. You can print this data directly, or save it to a file for use in other applications.

Image Capture

You can send the contents of the graphics window directly to a printer, capture the contents to another window that is created automatically, or port the contents to a graphics file in one of the popular formats (for example, PNG).

Report Generator

This function opens the Report Generator. Use the report generator to capture your screen contents, animations, and result listings, and save them to a report assembly tool. This tool allows you to organize the data and add text in order to assemble a complete report. For more information, see [The Report Generator \(p. 273\)](#).

7.2.6.1.3. Step/Sequence Data Access Controls

When you access a results file, the data is presented according to the sequential data sets of your original analysis. These data sets correspond to a specific time, load step, and substep of your analysis. Use the following controls to access these different result sets.

Figure 7.16: Results Viewer Step/Sequence Data Access Controls



The Data Sequence Slider Bar

The slider bar directly under the Results Viewer Toolbar corresponds to the individual data sequences that are available for the current results file. Each tick mark along the slider represents a data set. The data sequence number is displayed in the text box at the far right of the series of boxes below the slider. You can move to any data set either by moving the slider, or by entering the sequential number of the data set in the box.

The Play and Stop Buttons

You use these buttons to move through the selected data sequence according to the defined load steps or substeps. The play button will step you through each of the data sets and when the final (maximum) set is reached, begin moving incrementally back down.

Time

This text box displays the time for each data set.

Load Step

Each individual load step number is displayed. You can enter a valid load step number here and that load step will be displayed.

Substep

Each individual load substep number is displayed. You can enter a valid substep number here and that substep will be displayed.

Sequence

The result sets are written sequentially to the results file during an analysis. This displays the sequence number from the results file. You also create additional sequences when you append the results file or add new loading data to your original analysis.

7.2.6.2. Results Viewer Context-Sensitive Menus

When you enter the Results Viewer, the remainder of the GUI is disabled, preventing conflicts between the limited data available in the Results Viewer and the functionality that can be accessed from the other GUI areas. Many of the functions needed to deal with the results data have been moved to "context sensitive" menus accessed via the right and middle mouse buttons.

The Results Viewer places your cursor in "picking mode" anytime you place the cursor in the graphics window. This allows you to select data sets and other screen operations dynamically, many times without accessing the Picking Menu. For the Results Viewer, the standard method of using the right mouse button to alternate between picking and unpicking has been moved to the middle mouse button. The middle mouse button allows you to change between picking and unpicking. You can choose the mode of picking desired (selection of points on the working plane, selection of existing entities in the selected set and selection of points on the screen). You can also access entity filters to limit those entities that can be selected. For the two-button PC mouse, the middle button functions are activated via the SHIFT-RIGHT MOUSE combination.

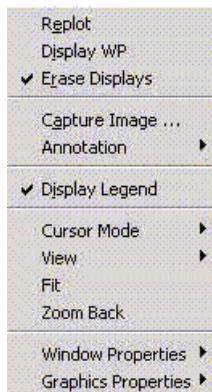
The right mouse button is used for context menus. These menus present choices that are applicable to the current selection. If you select "Screen Picking," you will get a legend data context menu if the cursor is over a legend item, and the graphics window context menu anywhere else.

If you select "WP Picking," and no points have been selected, you will get the legend or graphics context menus. If you have already selected points on the working plane, you will get a context menu that is applicable to the type of analysis you are performing. If you select "Entity Picking," and no entities are selected, you will get the standard Graphics and Legend context menus. If you have selected entities, you will get a context menu that is applicable to the current type of analysis.

Graphics Window Context Menu

The following items are available from the context menu you get when you right mouse click anywhere in the graphics window except over the legend information.

Figure 7.17: Graphics Window Context Menu



Replot -

Replots the screen and integrates any changes you have made.

Display WP -

Toggles the Working Plane display (triad, grid, etc.) on and off.

Erase Displays -

Toggles screen refreshes according to **/ERASE - /NOERASE** command functionality.

Capture Image -

You can send the contents of the graphics window directly to a printer, capture the contents to another window that is created automatically, or port the contents to a graphics file in one of the popular formats (for example, PNG).

Annotation -

You can choose between either static (2-D), or dynamic (3-D) annotations, and you can toggle your annotations on and off.

Display Legend -

Toggles the legend display on and off

Cursor Mode -

When the Results Viewer is selected, the **Cursor Mode** option allows you to switch between modes of picking, that include: **Pointer**, **Working Plane**, and **Entities**.

View -

Provides a drop-down list of view options: **Isometric, Oblique, Front, Right, Top, Back, Left, and Bottom.**

Fit -

Fits the extents of the model in the graphics area.

Zoom Back -

Returns to the previous zoom setting.

Window Properties -

You can access limited legend and display property control, along with a number of viewing angle, rotational setting and magnification controls. The Legend Settings allow you to select which legend items will be displayed, and to specify their location in the graphics window. The Display Settings let you specify Hidden, Capped or Q-Slice displays, and to modify the Z-buffering options, including smoothing and directional light source functions or 3-D options.

Graphics Properties -

This selection refers to those settings which affect all windows specified in a multi-window layout ([/WINDOW](#)) along with access to the Working Plane Settings and Working Plane Offset widgets, and the Window Layout controls dialog boxes.

Legend Area Context Menus

The legend area context menus will vary according to the location of your cursor in the legend, and the content of the legend data already being displayed. Right mouse clicking on the legend data provides control of the legend information, while clicking on the logo provides control of the date display and other miscellaneous functions. Clicking on the contour legend area provides a complete set of menus to customize your contour legend.

The legend setting and font control menus can be accessed from any of the legend context menus.

7.3. Additional POST1 Postprocessing

The following additional POST1 postprocessing techniques are covered in this section:

- 7.3.1. Rotating Results to a Different Coordinate System
- 7.3.2. Performing Arithmetic Operations Among Results Data
- 7.3.3. Creating and Combining Load Cases
- 7.3.4. Mapping Results onto a Different Mesh or to a Cut Boundary
- 7.3.5. Creating or Modifying Results Data in the Database
- 7.3.6. Splitting Large Results Files
- 7.3.7. Magnetics Command Macros
- 7.3.8. Comparing Nodal Solutions From Two Models or From One Model and Experimental Data (RSTMAC)

7.3.1. Rotating Results to a Different Coordinate System

Results data, calculated during solution, consist of displacements (UX, UY, ROTX, etc.), gradients (TGX, TGY, etc.), stresses (SX, SY, SZ, etc.), strains (EPPLX, EPPLXY, etc.), etc. These data are stored in the database and on the results file in either the nodal coordinate system (for the primary, or nodal data) or the element coordinate system (for the derived, or element data). However, results data are *generally* rotated into the active results coordinate system (which is by default the global Cartesian system) for displays, listings, and element table data storage.

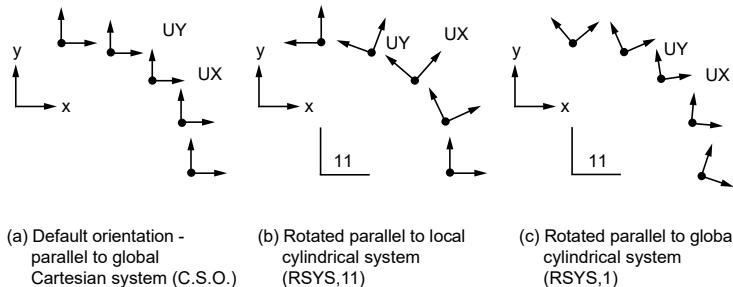
Using the **RSYS** command, you can change the active results coordinate system to global cylindrical (**RSYS,1**), global spherical (**RSYS,2**), any existing local coordinate system (**RSYS,N**, where *N* is the local coordinate system number), or the nodal and element coordinate systems used during solution (**RSYS,SOLU**). If you then list, display, or operate on the results data, they are rotated to this results coordinate system first. You may also set the results system back to global Cartesian (**RSYS,0**).

The default coordinate system for certain elements, notably shells, is not global Cartesian and is frequently not aligned at adjacent elements.

The use of **RSYS,SOLU** with these elements can make nodal averaging of component element results, such as SX, SY, SZ, SXY, SYZ, and SXZ, invalid and is not recommended.

The following figure illustrates how displacements are reported for several different **RSYS** settings:

Figure 7.18: Rotation of Results by RSYS



The displacements are in terms of the nodal coordinate systems (which are always Cartesian systems), but issuing the **RSYS** command causes those nodal systems to be rotated into the specified system. For example, **RSYS,1** causes the results to be rotated parallel to the global cylindrical system such that UX represents a radial displacement and UY represents a tangential displacement. (Similarly, VX and VY in a fluid analysis are reported as radial and tangential values for **RSYS,1**.)

Caution:

Certain element results data are always output in the element coordinate system regardless of the active results coordinate system. These are miscellaneous result items that you would normally expect to be interpreted only in the element coordinate system. They include forces, moments, stresses, and strains for beam, pipe, and spar elements, and member forces and moments for some shell elements.

In most circumstances, such as when working with a single load case or during linear combinations of multiple load cases, rotating results data into the results coordinate system does not affect the final result values. However, most modal combination techniques (PSD, CQC, SRSS, etc.) are performed in the solution coordinate system and involve squaring operations. Since the squaring operation removes

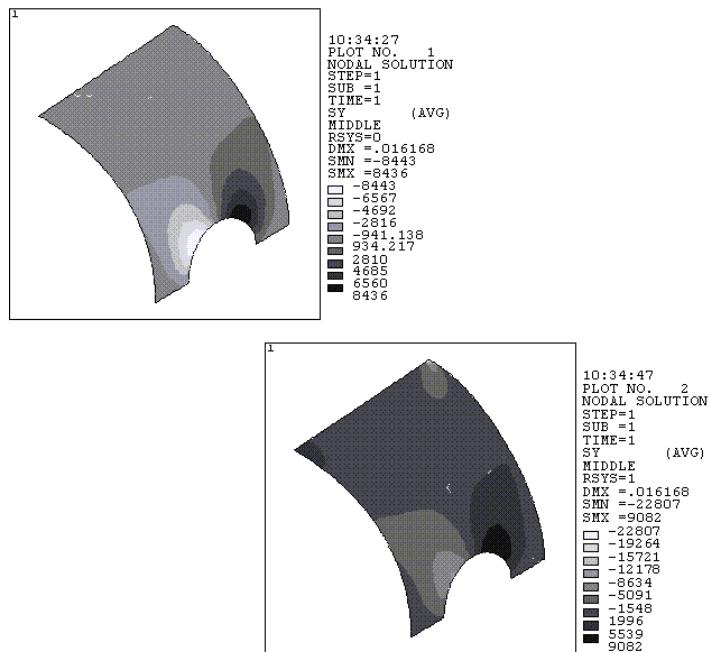
the sign associated with the data, some combined results may not appear as expected after being rotated into the results coordinate system. In these cases, **RSYS**,SOLU is on by default in order to keep the results data in the solution coordinate systems. No other coordinate system may be used.

As an example of when you would need to change the results coordinate system, consider the case of a cylindrical shell model, in which you may be interested in the tangential stress results. The SY stress contours before and after results coordinate system transformation are shown below for such a case.

```
PLNSOL,S,Y           ! Display a: SY is in global Cartesian system (default)
RSYS,1
PLNSOL,S,Y           ! Display b: SY is in global cylindrical system
```

See the **RSYS** and **PLNSOL** command descriptions for further information.

Figure 7.19: SY in Global Cartesian and Cylindrical Systems



Plot 1 (top) illustrates SY in global Cartesian system. Plot 2 (bottom) illustrates SY in global cylindrical system (note that nodes and elements are still in global Cartesian system).

In a large-deformation analysis (**NLGEOM,ON**), the element coordinate system is first rotated by the amount of rigid body rotation of the element. Therefore, the component stresses and strains and other derived element data include the effect of the rigid body rotation. The coordinate system used to display these results is the specified results coordinate system *rotated by the amount of rigid body rotation*.

The exceptions to this are continuum elements such as **PLANE182**, **PLANE183**, **SOLID185**, **SOLID186**, **SOLID187**, **SOLID272**, **SOLID273**, and **SOLID285**. For these elements, the output of the element component results is by default in the initial global coordinate system, and all component result transformations to other coordinate systems will be relative to the initial global coordinate system.

The primary data (for example, displacements) in a large-deformation analysis *do not include the rigid body rotation effect*, because the nodal coordinate systems are not rotated by the amount of rigid body rotation.

7.3.2. Performing Arithmetic Operations Among Results Data

The earlier discussion of operations among path items was limited to items mapped on to a path. Using commands from the POST1 CALC module, you can perform operations among any results data in the database. The only requirement is that you must use the element table. The element table serves as a "worksheet" that allows arithmetic operations among its columns.

The procedure to do calculations among results data requires three simple steps:

1. Issue the **ETABLE** command to bring one or more result items into the element table or "worksheet."
2. Perform the desired arithmetic operations via commands from the CALC module (**SADD**, **SMULT**, **SEXP**, etc.).
3. Review the outcome of the operations via **PRETAB** or **PLETAB**.

The **ETABLE** command moves specified results data for all selected elements into the element table. One value is stored per element. For example, if you select 10 elements and issue the command shown below, an average UX value is calculated for each element from the nodal displacements and stored in the element table under the "ABC" column.

```
ETABLE, ABC, U, X
```

The element table will be ten rows long (because only ten elements were selected). If you now want to double these displacements, the command (**SMULT**) to do so is:

```
SMULT, ABC2, ABC, , 2
```

The element table now has a second column, labeled ABC2, containing twice the values in column ABC. To list the element table, simply choose **PRETAB**.

Example 7.13: PRETAB Output

```
PRINT ELEMENT TABLE ITEMS PER ELEMENT

***** POST1 ELEMENT TABLE LISTING *****

  STAT    CURRENT    CURRENT
  ELEM      ABC        ABC2
    1     .21676     .43351
   11     .27032     .54064
   21     .23686     .47372
   31     .47783     .95565
   41     .36171     .72341
   51     .36693     .73387
   61     .13081     .26162
   71     .50835     1.0167
   81     .35024     .70049
   91     .25630     .51260
```

```
MINIMUM VALUES
ELEM      61          61
VALUE     .13081      .26162
```

```
MAXIMUM VALUES
ELEM      71          71
VALUE     .50835      1.0167
```

Another example of arithmetic operations is to calculate the total volume of selected elements. To do so, you might store all element volumes in the element table, select the desired elements, and sum them via the **SSUM** command:

```
ETABLE,VOLUME,VOLU           ! Store element volumes (VOLU) as VOLUME
ESEL,...                      ! Select desired elements
SSUM                           ! Calculate and print sum of VOLUME column
```

See the **ETABLE**, **ESEL**, and **SSUM** command descriptions for further information.

7.3.2.1. Hints and Recommendations

All operation commands (**SADD**, **SMULT**, **SSUM**, etc.) work only on the selected elements.

The program does not update element table entries automatically when a different set of results is read into the database. An element table output listing displays headers which indicate the status of each column relative to the current database. CURRENT indicates that the column data is from the current database, PREVIOUS indicates that it is from a previous database, and MIXED indicates that it results from an operation between previous and current data. (Once a column is labeled mixed, it will not change status unless you erase or redefine it with all elements selected.) The following commands cause column headers to change from CURRENT to PREVIOUS:

- **SET**
- **LCASE**
- **LCOPER**
- **LCZERO**
- **DESOL**

The **ETABLE,REFL** option, which refills (updates) the element table with values currently in the database, does *not* affect calculated items.

The program does not update the element table automatically after a new **SET** command. You can use that behavior to good advantage, such as comparing element results between two or more load steps, or even between two or more *analyses*.

The following CALC module commands apply to calculations using the element table:

- **SABS** -- Causes absolute values to be used in subsequent element table operations.
- **SADD** -- Adds two specified columns in the element table.
- **SALLOW** -- Defines allowable stresses for safety factor calculations.

- **SEXp** -- Exponentiates and multiplies two columns in the element table.
- **SFACT** -- Defines which safety factor calculations will be performed during subsequent display, select, or sort operations.
- **SFCALC** -- Calculates safety factors (for ETABLE items).
- **SMAX** -- Compares and stores the maximum of two columns.
- **SMIN** -- Compares and stores the minimum of two columns.
- **SMULT** -- Multiplies two specified columns in the element table.
- **SSUM** -- Calculates and prints the sum of each element table column.
- **TALLOW** -- Defines the temperature table for safety factor calculations.
- **VCROSS** -- Calculates the cross product of two vectors stored in the element table.
- **VDOT** -- Calculates the dot product of two vectors stored in the element table.

7.3.3. Creating and Combining Load Cases

In a typical postprocessing session, you read one set of data (load step 1 data, for instance) into the database and process it. Each time you store a new set of data, POST1 clears the results portion of the database and then brings in the new results data. If you want to perform operations between two entire sets of results data (such as comparing and storing the maximum of two sets), you need to create *load cases*.

A *load case* is a set of results data that has been assigned an arbitrary reference number. For instance, you can define the set of results at load step 2, substep 5 as *load case number 1*, the set of results at time = 9.32 as *load case number 2*, and so on. You can define up to 99 load cases, but you can store only one load case in the database at a time.

Note:

You can define a load case at any arbitrary time by via the **SET** command (to specify the time argument) and then via **LCWRITE** to create that load case file. The values will be a linear interpolation of the results already available before and after your specified time.

A *load case combination* is an operation between load cases, typically between the load case currently in the database and a load case on a separate results file (or on the *load case file*, explained later). The outcome of the operation overwrites the results portion of the database, which permits you to display and list the load case combination.

A typical load case combination involves the following steps:

1. Define load cases via the **LCDEF** command.
2. Read *one* of the load cases into the database via the **LCASE** command.
3. Perform the desired operation via the **LCOPER** command.

As an example, suppose the results file contains results for several load steps, and you want to compare load steps 5 and 7 and store the maximum in memory. The commands to do this would look like this:

```
LCDEF,1,5          ! Load case 1 points to load step 5
LCDEF,2,7          ! Load case 2 points to load step 7
LCASE,1            ! Reads load case 1 into memory
LCOPER,MAX,2       ! Compares database with load case 2 and stores the
                  ! maximum in memory
```

The database now contains the maximum of the two load cases, and you can perform any desired postprocessing function.

Note:

Load case operations (**LCOPER**) are performed only on the raw solution results in the solution coordinate system.

The solution results are:

- Element component stresses, strains, and nodal forces in the element coordinate systems
- Nodal degree-of-freedom values, applied forces, and reaction forces in the nodal coordinate systems

To have a load case operation act on the principal/equivalent stresses instead of the component stresses, issue the **SUMTYPE,PRIN** command.

It is important that you know how load case operations are performed. Many load case operations, such as mode combinations, involve squaring, which renders the solution results unsuitable for transformation to the results coordinate system, typically the Global Cartesian, and unsuitable for performing nodal or element averages. A typical postprocessing function such as printing or displaying average nodal stresses (**PRNSOL**, **PLNSOL**), for example, involves both a coordinate system transformation to the results coordinate system and a nodal average. Furthermore, unless **SUMTYPE,PRIN** has been requested, principal/equivalent stresses are not meaningful when computed from squared component values.

To view correct mid-surface results for shells (**SHELL,MID**), use KEYOPT (8) = 2 (for **SHELL181**, **SHELL208**, **SHELL209**, **SHELL281**, or **ELBOW290**). These KEYOPT settings write the mid-surface results directly to the results file, and allow the mid-surface results to be directly operated on during squaring operations, instead of averaging the squared TOP and BOTTOM results.

You should therefore use only untransformed (**RSYS,SOLU**), nonaveraged (**PRESOL**, **PLESOL**) results whenever you perform a squaring operation, such as in a spectrum (SPRS,PSD) analysis.

Issue the **FORCE** command prior to any load case operations to insure that the forces are correctly summed before the operation. Note that once a force type has been combined, the other components are no longer available and will be zero.

7.3.3.1. Saving a Combined Load Case

By default, the results of a load case combination are stored in memory, overwriting the results portion of the database. To save these results (for later review or for subsequent combinations with other load cases), use one of the following methods:

- Write the data to a load case file.
- Append the data to the results file.

Issue the **LCWRITE** command to write the load case currently in memory to a *load case file*. The file is named *Jobname.Lnn*, where *nn* is the load case number you assign. Using *nn* in subsequent load case combinations will refer to the load case stored on the load case file.

Example 7.14: Use of the LCWRITE Command

```
LCDEF,1,5          ! Load case 1 points to load step 5
LCDEF,2,7          ! Load case 2 points to load step 7
LCDEF,3,10         ! Load case 3 points to load step 10
LCASE,1           ! Reads load case 1 into memory
LCOPER,MAX,2       ! Stores max. of database and load case 2 in memory
LCWRITE,12         ! Writes current load case to Jobname.L12
LCASE,3           ! Reads load case 3 into memory
LCOPER,ADD,12      ! Adds database to load case on Jobname.L12
LCDEF,STAT         ! Results in the following output:
```

Example 7.15: Output from LCDEF,STAT

```
LOAD CASE= 1 SELECT= 1 ABS KEY= 0 FACTOR= 1.0000
LOAD STEP=      5 SUBSTEP=      2 CUM. ITER.=      4 TIME/FREQ= .25000
FILE=beam.rst
Simply supported beam

LOAD CASE= 2 SELECT= 1 ABS KEY= 0 FACTOR= 1.0000
LOAD STEP=      7 SUBSTEP=      3 CUM. ITER.=      10 TIME/FREQ= .75000
FILE=beam.rst
Simply supported beam

LOAD CASE= 3 SELECT= 1 ABS KEY= 0 FACTOR= 1.0000
LOAD STEP=     10 SUBSTEP=      2 CUM. ITER.=     12 TIME/FREQ= 1.0000
FILE=beam.rst
Simply supported beam

LOAD CASE= 12 SELECT= 1 ABS KEY= 0 FACTOR= 1.0000
LOAD STEP=      0 SUBSTEP=      0 CUM. ITER.=      0 TIME/FREQ= .00000E+00
FILE=beam.112
Simply supported beam
```

The **RAPPND** command appends the load case currently in memory to the results file. The data are stored on the results file just like any other results data set except that:

- You (not the program) assign the load step and substep numbers used to identify the data.
- Only summable and constant data are available by default; non-summable data are not written to the results file unless requested (**LCSUM** command).

Example 7.16: Use of the RAPPND Command

```
/POST1      ! Following a 2 load step analysis
SET,1          ! Store load step 1
LCDEF,1,2      ! Identify load step 2 as load case 1
LCOPER,ADD,1   ! Add load case 1 to database (ls 1 + ls 2)
RAPPND,3,3     ! Append the combined results to the results file
! as ls 3, time = 3
SET,LIST        ! Observe addition of new load step to results file
```

You can issue the **RAPPND** command to combine results from two results files (created with the same database.) Use the POST1 **FILE** command to toggle between the two results files to alternately store results from one and append to the other.

7.3.3.1.1. Hints and Recommendations

You can define a load case by setting a pointer to a load case via the **LCFILE** command. Later, you can issue the **LCASE** or **LCOPER** commands to read data from the file into memory.

You can erase any load case by issuing **LCDEF,LCNO,ERASE**, where *LCNO* is the load case number. To erase all load cases, issue **LCDEF,ERASE**. These options not only delete all load case pointers, but also delete the appropriate load case files (those with the default file name extensions).

To zero the results portion of the database, issue a **LCOPER,ZERO** command.

The **LCABS** and **LCFACT** commands enable you to specify absolute values and scale factors for specific load cases. The program uses these specifications when you issue either **LCASE** or **LCOPER**. The program applies scale factors *after* it calculates absolute values.

Results data read into the database from the results file (via **SET** or **LCASE** commands) will include boundary condition information (constraints and force loads). However, load cases read in from a load case file will not. Therefore, if boundary conditions appear on graphics displays after you issue an **LCASE** command, they are from a previously processed load case. The **LCASE** command does not reset the boundary condition information in memory.

After a load case combination is performed for structural line elements, the principal stress data are not automatically updated in the database. Issue **LCOPER,LPRIN** to recalculate line element principal stresses based on the current component stress values.

You can select a subset of load cases via the **LCSEL** command. After a subset is selected, you can use the label **ALL** in place of a load case number on load case operations.

Element nodal forces are operated on before summing at the node.

Load case combinations should be performed on load cases created from the same results file.

7.3.3.2. Combining Load Cases in Harmonic Element Models

For models with harmonic elements (axisymmetric elements with nonaxisymmetric loads), the loads are frequently applied in a series of load steps based on a Fourier decomposition (See the *Element Reference*). To get usable results, combine the results of each load step in POST1. You can do so via load case combinations, by saving and summing all results data at a given circumferential angle.

Example 7.17: Combining Load Cases in a Harmonic Element Model

```

/POST1
SET,1,1,,,90          ! Read load step 1 with circumferential
                      ! angle of 90°
LCWRITE,1              ! Write load case 1 to load case file
SET,2,1,,,90          ! Read load case 2, with circumferential
                      ! angle of 90°
LCOPER,ADD,1           ! Use load case operations to add results
                      ! from first load case to second
ESEL,S,ELEM,,1         ! Select element number 1
NSLE,S                 ! Select all nodes on that element
PRNSOL,S               ! Calculate and list component stresses
PRNSOL,S,PRIN          ! Calculate and list principal
                      ! stresses S1, S2, S3; stress intensity
                      ! SINT; and equivalent stress SEQV
FINISH

```

See the **SET**, **LCWRITE**, **LCOPER**, **ESEL**, **NSLE**, and **PRNSOL** command descriptions for further information.

7.3.3.3. Summable, Non-Summable, and Constant Data

By default, when you perform load case combinations in POST1, the program combines only data that are valid for linear superposition, such as displacements and component stresses. Other data, such as plastic strains and element volumes, are not combined, because it is not appropriate or meaningful to combine such data. To determine which data should be combined and which should not, result items are grouped into *summable*, *non-summable*, and *constant* data. This grouping applies to the following POST1 database operations:

- Load case combinations (**LCOPER**).
- Reading in a load case with active scale factors (**LCFACT** or **LCASE**).
- Reading in results data and modifying them via the *FACT* or *ANGLE* field on the **SET** command.

Summable data are those that can participate in the database operations. All primary data (DOF solutions) are considered summable. Among the derived data, component stresses, elastic strains, thermal gradients and fluxes, magnetic flux density, etc. are considered summable (see

[Table 7.2: Examples of Summable POST1 Results \(p. 176\)](#)). (For an inclusive list of summable data, see the description of the **ETABLE** command in the *Command Reference*.)

Sometimes, combining summable data may result in meaningless results. For example, adding nodal temperatures from two load cases of a linear, pure-conduction analysis gives meaningful results, but if convection is involved, the addition of temperatures is not meaningful. Exercise your engineering judgement when reviewing combined load cases.

Non-summable data are those that are not valid for linear superposition, such as nonlinear data (plastic strains, hydrostatic pressures), thermal strains, magnetic forces, Joule heat, etc. (See

[Table 7.3: Examples of Non-Summable POST1 Results \(p. 176\)](#).) These data are simply set to zero when the programs performs a database operation. You can combine non-summable data (**LC-SUM,ALL**) before issuing **LCOPER** commands, but take care to interpret the values appropriately.

Constant data are those that cannot be meaningfully combined, such as element volumes and element centroidal coordinates (see [Table 7.4: Examples of Constant POST1 Results \(p. 176\)](#)). These data are held constant (unchanged) when the program performs a database operation.

Table 7.2: Examples of Summable POST1 Results

"Vector" Data					
Item	Component	Item	Component	Item	Component
U	X, Y, Z	ROT	X, Y, Z	V	X, Y, Z
A	X, Y, Z	S	X, Y, Z, XY, YZ, XZ	EPEL	X, Y, Z, XY, YZ, XZ
TG	X, Y, Z	TF	X, Y, Z	PG	X, Y, Z
EF	X, Y, Z	D	X, Y, Z	H	X, Y, Z
B	X, Y, Z	F	X, Y, Z	M	X, Y, Z
VF	X, Y, Z	CSG	X, Y, Z	JS	X, Y, Z
CG	X, Y, Z	DF	X, Y, Z	EPDI	X, Y, Z, XY, YZ, XZ
LS	(ALL)	LEPEL	(ALL)	SMISC	(ALL)

"Scalar" Data		
TEMP	PRES	MAG
TBOT	FLOW	FLUX
TE2	VOLT	
HEAT	AMPS	

Table 7.3: Examples of Non-Summable POST1 Results

"Vector" Data					
Item	Component	Item	Component	Item	Component
EPPL	X, Y, Z, XY, YZ, XZ	EPTH	X, Y, Z, XY, YZ, XZ	NL	SEPL, SRAT, HPRES, EPEQ, PSV, PLWK
FMAG	X, Y, Z	BFE	TEMP	LEPTH	(ALL)
LEPPL	(ALL)	LEPCR	(ALL)	NLIN	(ALL)
LBFE	(ALL)	NMISC	(ALL)		

"Scalar" Data			
EPSW	SENE	KENE	JHEAT

Table 7.4: Examples of Constant POST1 Results

"Vector" Data	
Item	Component
CENT	X, Y, Z

"Scalar" Data	
VOLU	

7.3.4. Mapping Results onto a Different Mesh or to a Cut Boundary

Just as the **PDEF** command maps results onto an arbitrary path in the model, POST1 also has the ability to map results on to an entirely new mesh or to a portion of a new mesh. This functionality is mainly used in *submodeling*, where you initially analyze a coarse mesh, build a finely meshed submodel of a region of interest, and map results data from the coarse model to the fine submodel.

The **CBDOF** command maps degree-of-freedom results from the coarse model to the cut boundaries of the submodel. **BFINT** maps body force loads (mainly temperatures for a structural analysis) from the coarse model to the submodel. Both commands require a file of nodes to which results are to be mapped, and both commands write a file of appropriate load commands.

For more information, see [Submodeling in the Advanced Analysis Guide](#).

7.3.5. Creating or Modifying Results Data in the Database

You can postprocess without also generating a results file. Create a database containing nodes, elements, and property data, and then put your own results into the database via the following commands:

DESOL -- Defines or modifies solution results at a node of an element.

DETAB -- Modifies element table results in the database.

DNSOL -- Defines or modifies solution results at a node.

After the data are defined, you can use almost any postprocessing function: graphics displays, tabular listings, path operations, etc.

Issuing the **DNSOL** command requires that you have placed the data type (stress/strain) in the element nodal records. To get around this requirement, use the **DESOL** command to add a "dummy" element stress/strain record.

The program performs all load case combinations in the solution coordinate system, and the data resulting from load case combinations are stored in the solution coordinate system. The resultant data are then transformed to the active results coordinate system when listed or displayed; therefore, unless **RSYS,SOLU** is set (no transformation of results data), you may see some unexpected results such as negative values after a square operation or negative values even when you request absolute values.

This feature is intended for reading in data from your own, special-purpose program. By writing output data from that program in the form of the above commands, you can read them into POST1 and process them the same way you would Mechanical APDL results. If you have a Mechanical APDL results file, it is not affected by these commands.

7.3.6. Splitting Large Results Files

If you have a results file that is too large for you to postprocess on your local machine (such as from running a large model on a server or cluster), you can split the results file into smaller files based on

subsets of the model. You can then process these smaller files on your local machine. For example, if your large model is an assembly, you can create individual results files for each part.

You can also use this capability to create a subset of results for efficient postprocessing. For example, you could take the results file from a large model that had all results written, and create a smaller results file containing only stresses on the exterior surface. This smaller file would load and plot quickly, while not losing any of the detailed data written to the full results file.

When you use this feature, the subset geometry is also written to the results file so that you do not need the database file (no **SET** command required). However, do not resume any database when postprocessing one of these results files, and your original results file must have the geometry written to it (that is, do not issue **/CONFIG,NORSTGM,1**).

To use the results file splitting feature, issue the **RSPLIT** command. You can use results file splitting with **INRES** to limit the amount of data written to the results files.

Example 7.18: Splitting Large Results Files

```
/POST1
FILE,jobname,rst           ! Import *.rst file
ESEL,all
INRES,nsol,strs            ! Write out only nodal solution and stresses
RSPLIT,ext,esel,myexterior ! Write the results for the exterior of the whole model to a file
                           ! named myexterior.rst
FINISH
/EXIT
...
/POST1
FILE,myexterior
PLNS,s,eqv                 ! Postprocess the myexterior.rst file as usual
PLNS,u,sum
FINISH
```

7.3.7. Magnetics Command Macros

The following magnetic command macros are also available for calculating and plotting results from a magnetic analysis:

- **CURR2D** -- Calculates current flow in a 2-D conductor.
- **EMAGERR** -- Calculates the relative error in an electrostatic or electromagnetic field analysis.
- **EMF** -- Calculates the electromotive force (emf) or voltage drop along a predefined path.
- **EMFT** -- Summarizes electromagnetic forces and torques on a selected set of nodes.
- **FLUXV** -- Calculates the flux passing through a closed contour.
- **MAGSOLV** -- Specifies magnetic solution options and initiates the solution for a static analysis.
- **MMF** -- Calculates magnetomotive force along a path.
- **PERBC2D** -- Generates periodic constraints for 2-D planar analysis.
- **PLF2D** -- Generates a contour line plot of equipotentials.
- **PMGTRAN** -- Summarizes electromagnetic results from a transient analysis.

- **POWERH** -- Calculates the RMS power loss in a conducting body.
- **RACE** -- Defines a "racetrack" current source.
- **SENERGY** -- Determines the stored magnetic energy or co-energy.

For more information, see [Electric and Magnetic Macros in the Low-Frequency Electromagnetic Analysis Guide](#).

7.3.8. Comparing Nodal Solutions From Two Models or From One Model and Experimental Data (RSTMAC)

In a typical design procedure, you may want to make small changes to your model and compare the solutions you obtain from the new model to solutions from the original model. You may also want to compare numerical solutions with experimental ones.

The **RSTMAC** command compares the nodal solutions from two results files (.rst or .rstp) or from one results file and one Universal Format file (.unv). The details about the Universal Format file supported records can be found in [Universal Format File Records \(p. 184\)](#).

In addition to structural degrees of freedom, some of the non-structural degrees of freedom are considered as well, and nodal solutions (real or complex) from any analysis type are supported. The procedure is based on the Modal Assurance Criterion (MAC) calculations (see [POST1 - Modal Assurance Criterion \(MAC\)](#) for more details).

Because the two files may correspond to different models and/or meshes, the first step is to choose one of the following methods:

- [Matching the Nodes of Model 1 and Model 2 Based on Node Location \(p. 181\)](#). This is the default procedure. It is recommended when meshes are identical or very similar. Note that the default algorithm is based on the first node found, not the nearest one. It is less time consuming when the tolerance is small enough and the element size is homogeneous. Choosing the algorithm based on the nearest node found (KEYALGONOD = YES on **MACOPT**) may take longer, but it is more robust in certain situations; for example, when there are large differences in element size. The default procedure of matching based on node location is also activated when *Option* = ABSTOLN or RELTOLN are set on the **MACOPT** command, which also turns off other previous procedures options (NUMMATCH and NODMAP).
- [Matching the Nodes of Model 1 and Model 2 Based on Their Node Numbers \(p. 182\)](#). This procedure is activated with *Option* = NUMMATCH on the **MACOPT** command. It is recommended when nodes with the same numbers are to be compared, for example when the same model is in a different location. This option turns off other previous procedure options on **MACOPT** (ABSTOLN, RELTOLN, KEYALGONOD and NODMAP).
- [Mapping the Nodes of Model 2 Into Elements of Model 1 \(p. 181\)](#). This procedure is activated with *Option* = NODMAP on the **MACOPT** command. In this case, the solutions of model 1 are interpolated. It is the most general procedure, but it is generally more time consuming. This option turns off other previous procedure options on **MACOPT** (ABSTOLN, RELTOLN, KEYALGONOD, and NUMMATCH).

If you want to compare the nodal solutions of cyclic symmetric sectors (**CYCLIC**), model 1 and model 2 must have the same number of sectors, and the solutions must correspond to the same harmonic

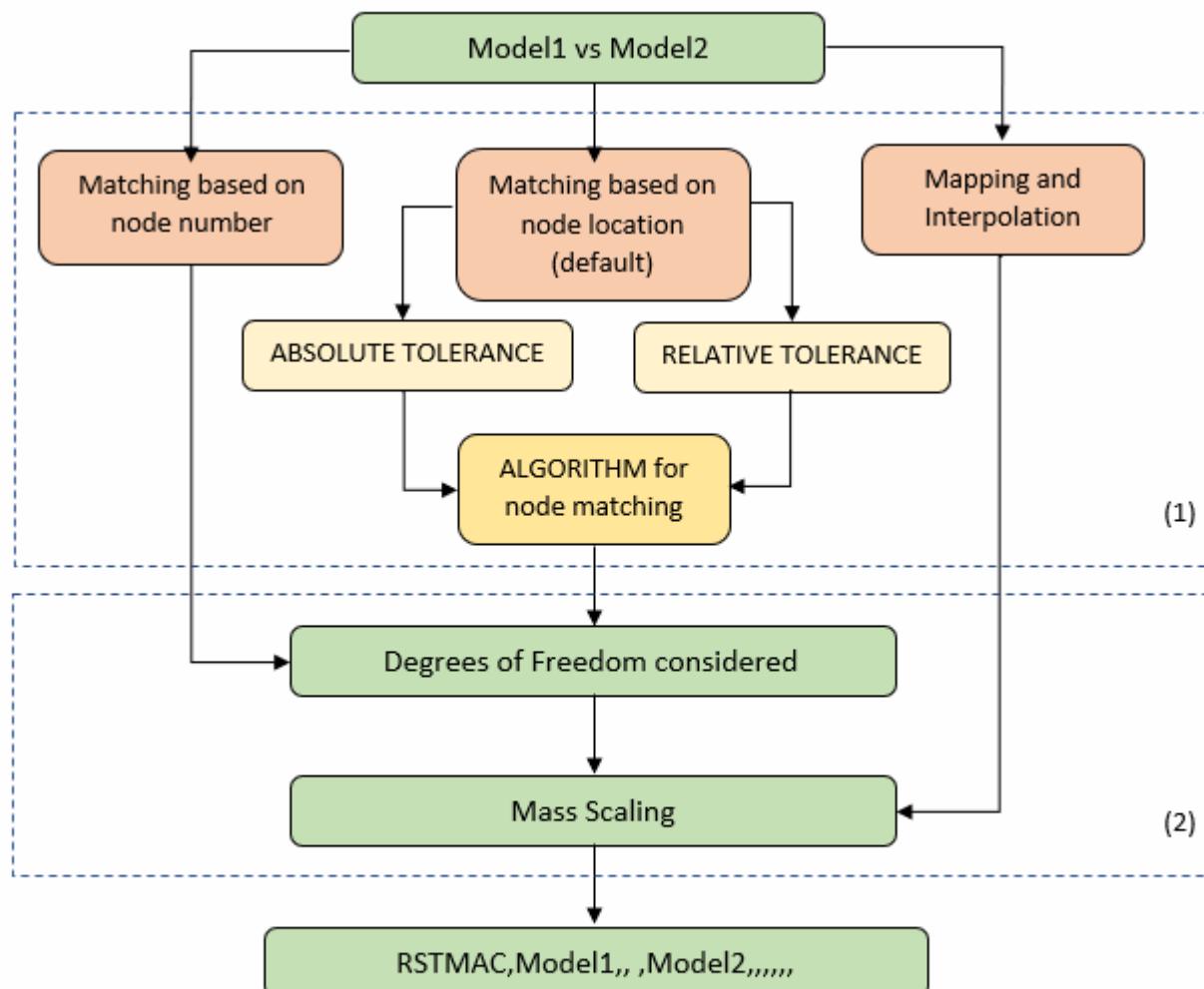
index. If the MSUP option is not activated (**CYCOPT**,MSUP,NO), both the mapping and interpolation method (*Option* = NODMAP on the **MACOPT** command) and the matching based on node number (*Option* = NUMMATCH on the **MACOPT** command) are available. Because the nodes of the base sector and the nodes of the duplicate sector are coincident, node mapping is done separately for each sector.

The next steps consist of:

- Performing the MAC calculations (p. 183)
- Identifying the best solution matches (p. 184)

To perform all of these steps and improve the **RSTMAC** command results, different options are available on the **MACOPT** command. These options can be combined, as shown in [Figure 7.20: An Overview of MACOPT Command Options](#) (p. 180).

Figure 7.20: An Overview of MACOPT Command Options



(1) *Matching and mapping settings*

(2) *MAC calculation settings*

All of these steps are detailed below using two models of a simply supported beam. Model 1, *tsolid*, is a solid element mesh. Model 2, *tbeam*, is a beam element mesh.

All of the first ten eigensolutions are compared, and a full printout is requested (*KeyPrint* = 2 on the **RSTMAC** command).

7.3.8.1. Matching the Nodes of Model 1 and Model 2 Based on Node Location

The input is:

```
rstmac, tsolid,1,all, tbeam,1,all,,, 2
```

The output is:

***** MATCHED NODES *****		
Node in tsolid.rst	Node in tbeam.rst	Distance
33	1	0.0000E+00
521	2	0.0000E+00
526	4	0.0000E+00
531	6	0.2220E-15
536	8	0.4441E-15
541	10	0.8882E-15
546	12	0.1776E-14
551	14	0.2665E-14
556	16	0.3553E-14
5000	50	0.0000E+00

Any pair of nodes with a distance smaller than the tolerance specified (*Option* = ABSTOLN and/or RETOLN on **MACOPT**) is considered matched. If there are no pair of nodes within the tolerance specified , the node matching is failing and the MAC calculations will not be performed.

For specifying the tolerance on the **MACOPT** command, you can choose either an absolute tolerance applied to the distance between nodes (ABSTOLN) or a relative tolerance which is a fraction of the minimum element dimension (RETOLN). The absolute tolerance value specified must be smaller than the element size to obtain relevant matched pairs.

By default, to get matched pairs, the node matching algorithm finds the first node with distance below the tolerance. If you specify the option KEYALGONOD = YES on **MACOPT**, then the node matching algorithm finds the nearest node with distance below the tolerance.

Only the matched nodes will contribute to the MAC calculations.

If you want to match a set of selected nodes only, you may do one of the following:

- Write only the solutions at the desired nodes on the results file via the **OUTRES** command (at the solver level).
- Specify a component name (*Cname*) on the **RSTMAC** command. This component must be based on the desired nodes.

7.3.8.2. Mapping the Nodes of Model 2 Into Elements of Model 1

First, the database corresponding to *tsolid* must be resumed.

When comparing the nodal solutions of cyclic symmetric structures, the database must be saved **AFTER** the solution is finished. The same applies when comparing solutions obtained with the linear perturbation procedures, because the node coordinates are updated with the base analysis displacements during the solution.

The input is:

```
macopt, NODMAP, yes
rstmac, tsolid,1,all, tbeam,1,all,,, 2
```

The output is:

***** MAPPED NODES *****	
Node in tbeam.rst	Element in tsolid.rst
1	1
2	40
3	3
4	5
5	8
6	10
7	13
8	15
9	18
10	20
11	23
12	25
13	28
14	30
15	33
16	35
17	38

Only the mapped nodes will contribute to the MAC calculations.

Nodes that are coincident may lead to erroneous interpolation. For example, this can occur in the case of spring/mass systems attached to solid elements.

If nodes and/or elements are selected (**NSEL** and/or **ESEL**, respectively), the results of the mapping and/or the interpolation show differences. To perform the MAC calculation on a part of the model, you can use the **ESEL** command. For a cyclic analysis where the MSUP option is not activated (**CY-COPT,MSUP,NO**), select the elements of both the base sector and the duplicate sector. For a cyclic analysis where the MSUP option is activated (**CYCOPT,MSUP,YES**) select the elements of the base sector only. All selected element nodes must also be selected (**NSLE**).

Issue the **OUTRES** command only when generating the *second* results file (*File2*). If the **OUTRES** command is issued to reduce the solution data written to the first results file (*File1*), the interpolation may be incorrect. (The problem can occur when an element does not have all of its nodes in the results file, for example.) This option is not supported when comparing cyclic symmetric structure results.

7.3.8.3. Matching the Nodes of Model 1 and Model 2 Based on Their Node Numbers

Model 1, *tbeamx*, is a beam element mesh along X direction. Model 2, *tbeamy*, is a beam element mesh along Y direction with the same node numbering as *tbeamx*.

The input is:

```
macopt,NUMMATCH,yes
rstmac, tbeamx,1,all, tbeamym,1,all,,, 2
```

The output is:

***** MATCHED NODES *****		
Node in FILE1	Node in FILE2	Distance
1	1	0.0000E+000
2	2	0.1414E+001
3	3	0.1414E+000
4	4	0.2828E+000
5	5	0.4243E+000
6	6	0.5657E+000
7	7	0.7071E+000
8	8	0.8485E+000
9	9	0.9899E+000
10	10	0.1131E+001
11	11	0.1273E+001

Any pair of nodes with the same number is considered matched. If there are no matched node pairs, the **RSTMAC** fails, and the MAC calculations will not be performed.

Only the matched nodes will contribute to the MAC calculations.

A set of selected nodes can be studied using the component name (*Cname*) on the **RSTMAC** command.

7.3.8.4. Evaluate MAC Between Solutions at Matched/Mapped Nodes

The MAC output for interpolated solutions (mapped nodes) is:

***** Modal Assurance Criterion (MAC) VALUES *****										
Solutions are real										
Rows:	10 substeps in load step			1 interpolated from file	tsolid.rst					
Columns:	10 substeps in load step			1 from file	tbeam.rst					
	1	2	3	4	5	6	7	8	9	10
1	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
3	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000
5	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000
6	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000
7	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000
10	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000

The modal assurance criterion values can be retrieved as parameters (***GET** with *Entity* = **RSTMAC**).

The modes obtained after a modal analysis of a cyclic symmetric structure are repeated when the harmonic index is not equal to zero and not equal to the number of sectors divided by 2 (for an even number of sectors). In this case, the MAC values table is merged to allow the solutions matching. This merging consists of summing and averaging the MAC values of the repeated frequencies.

7.3.8.5. Evaluate MAC per node pair

MAC per node pair (*Option* = NMAC on **MACOPT**) is computed for matched node pairs for which a solution is available in the results file or .unv file. Node pairs for which MAC is below the minimum acceptable value specified (*MacLim* on the **RSTMAC** command) will be flagged (*). Pairs of nodes for which all degrees of freedom are zero are flagged differently (**).

The MAC per node pair output is as follows (*MacLim* = 0.9):

```
***** Modal Assurance Criterion per node pair *****
      Substep 1 in load step 1 from FILE1
      Substep 1 in load step 1 from FILE2
      Node in          Node in          Nodal MAC
      FILE1            FILE2
      1                1                0.000  **
      2                2                1.000
      3                3                1.000
      4                4                1.000
      5                5                1.000
      6                6                0.752  *
      7                7                1.000
      8                8                1.000
      9                9                1.000
      10               10               1.000
      11               11               1.000
```

In the example above, the first node pair has both nodes fixed, while the 6th pair of nodes has a MAC below the minimum acceptable value specified (*MacLim* on the **RSTMAC** command).

7.3.8.6. Match the Solutions

The Matched Solutions output for interpolated solutions (mapped nodes) is as follows:

```
***** MATCHED SOLUTIONS *****
      Substep in          Substep in          MAC value    Frequency
      tsolid.rst          tbeam.rst        difference (Hz)  Frequency
      (interpolated)                                error (%)
      1                1                1.000       0.10E-01       0.2
      2                2                1.000       -0.47E-02      0.1
      3                3                1.000       0.26E-01       0.2
      4                4                1.000       0.27E-01       0.1
      5                5                1.000       0.40E-01       0.1
      6                6                1.000       0.13E+00       0.2
      7                7                1.000       0.11E+00       0.2
      8                8                1.000       0.82E-01       0.1
      9                9                1.000       -0.12E+00      0.1
      10               10               1.000       -0.96E+00      0.6
```

If no pair of solutions has a MAC value greater than the minimum acceptable value specified (*MacLim* on the **RSTMAC** command), the matching of the solutions is failing. The default limit is set to 0.9.

7.3.8.7. Universal Format File Records

The Universal Format file records supported are the following.

- Dataset Number 2420: Coordinate system

- Dataset Number 15 or 2411: Nodes
- Dataset Number 55: Data at Nodes
 - Record 6
 - Field 1: Model Type = 1 (Structural)
 - Field 2: Analysis Type = 3 (Complex eigenvalue first order) or Analysis Type = 2 (Normal Mode)
 - Field3: Data Characteristic = 2 (3 DOF) or 3 (6 DOF)
 - Field4: Specific Data Type = 8 (Displacement) or 12 (Acceleration)
 - Field5: Data Type = 2 (Real) or 5 (Complex)

The modal assurance criterion is a normalized quantity. Multiplying one solution vector by a constant does not modify the end result. As a consequence, because the modal acceleration is equal to the modal displacement multiplied by the frequency squared, a modal displacement result on the results file can be compared to a modal acceleration vector on the .unv file.

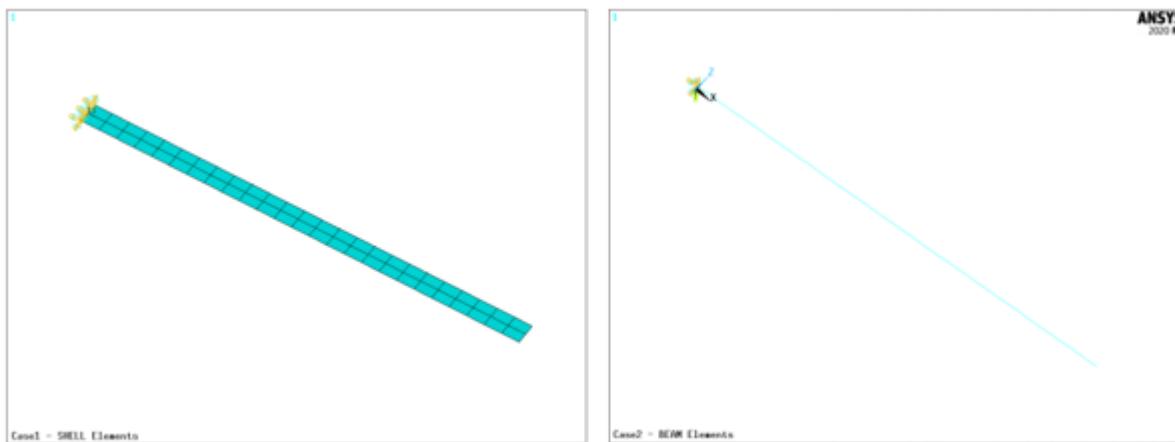
The coordinate system transformation (*Option* = UNVTRAN on **MACOPT**) is based on Dataset Number 2420 and used to transform the solution vector from the local to the global coordinate system. Only Cartesian coordinate systems are supported.

To print out the Universal Format file records, you can use the UNVDEBUG key on **MACOPT**. All information is listed on *File2_unv.txt*.

7.3.8.8. Examples Using **RSTMAC** with Different **MACOPT** Options

7.3.8.8.1. Comparing Two Plates Modeled Using Different Elements

The following example presents some **MACOPT** command options and the corresponding **RSTMAC** results. The studied models are two slender plates clamped at one end and modeled using different element types (**BEAM189** vs **SHELL181**), as shown in [Figure 7.21: SHELL181 Model vs BEAM189 Model \(p. 186\)](#).

Figure 7.21: SHELL181 Model vs BEAM189 Model

```

/com,
/com, -----> Case1 - SHELL elements
/com,
/filename,casel
pi = acos(-1)

! geometry & Material
! -----
L = 1      ! beam length
b = 0.05   ! X-section breadth
h = 0.0025 ! X-section height
E = 2.1e11
nu = 0.3
rho = 7800

parsav,,geom,parm

/prep7

! element type and material prop.
! -----
et,1,181
sectype,1,shell
secdat,h

mp,ex,1,E
mp,prxy,1,nu
mp,dens,1,rho

! create geometry
! -----
local,11,0 ,0,0,0 ,0,90,0
wpcsys,11
blc4,0,-b/2,L,b
csys,0

! meshing
! -----
lsel,s,loc,x,L/2
lesize,all,,, 50

lsel,s,loc,z,0
lesize,all,,, 4

allsel
amesh,all

! boundary conditions

```

```

! -----
nsel,s,loc,x,0
d,all,all
allsel
finish

/solu
antype,modal
modopt,lanb,8,,
mxpand,8
solve
finish
save

/post1
set,list
finish
/clear,nostart

/filename,case2
parres,,geom,parm

/com,
/com, ----- > Case2 - BEAM elements
/com,

/PREP7

! element type and material prop.
! -----
et,1,189
sectype,1,beam,rect
seadata,h,b

mp,ex,1,E
mp,prxy,1,nu
mp,dens,1,rho

! Geometry & Meshing
! -----
k,1,0,0,0.01
k,2,L,0,0.01
l,1,2

Nelt = 30
lesize,all,,,Nelt
lmesh,all

! boundary conditions
! -----
nsel,s,loc,x,0
d,all,all
allsel
/eshape,1
eplot
FINISH

! Modal analysis
! -----
/solu
antype,modal
modopt,lanb,8,,
mxpand,8
solve
finish

/POST1
set,list
FINISH
/clear,nostart

```

```

resume,casel,db
/com,
/com,
/com,      MAC CALCULATION
/com,
/com,
/post1
/com,
/com,
/com, -----> Case 1  Node matching based on node location (DEFAULT)
/com -----> Using an absolute tolerance (0.005)
/com,
MACOPT,ABSTOLN,0.005
rstmac,casel.rst,1,all,case2.rst,1,all,,0.70,,1

/com,
/com,
/com, -----> Case 2  Node matching based on node location (DEFAULT)
/com -----> Using a higher tolerance (0.02)
/com,
MACOPT,ABSTOLT,0.02
rstmac,casel.rst,1,all,case2.rst,1,all,,0.70,,1

/com,
/com,
/com, -----> Case 3  Node mapping
/com,
MACOPT,NODMAP,yes
rstmac,casel.rst,1,all,case2.rst,1,all,,0.70,,1

finish

```

The table below summarizes the results for the three cases. Although the MAC values depend on the method used to match or map the nodes, they are all very close to 1.0. A high number of matched nodes does not necessarily improve the MAC value. The tolerance should be chosen carefully to match relevant nodes.

Table 7.5: MACOPT Commands and Results of RSTMAC Command for All Cases

	Case 1	Case 2	Case 3
Matching / mapping method	<i>Matching based on node location</i>		<i>Mapping</i>
MACOPT command	MACOPT,ABSTOLN,0.005	MACOPT,ABSTOLN,0.02	MACOPT,NODMAP,yes
Resulting matched / mapped nodes	*** NOTE *** Node matching in RSTMAC command succeeded. 31 pairs of nodes are within the tolerance (.005).	*** NOTE *** Node matching in RSTMAC command succeeded. 61 pairs of nodes are within the tolerance (.02).	*** NOTE *** Node mapping in RSTMAC command succeeded. 61 nodes are mapped.
Substep in FILE1	Substep in FILE2	Calculated MAC values	
1	1	1.000	1.000
2	2	1.000	0.996
		1.000	

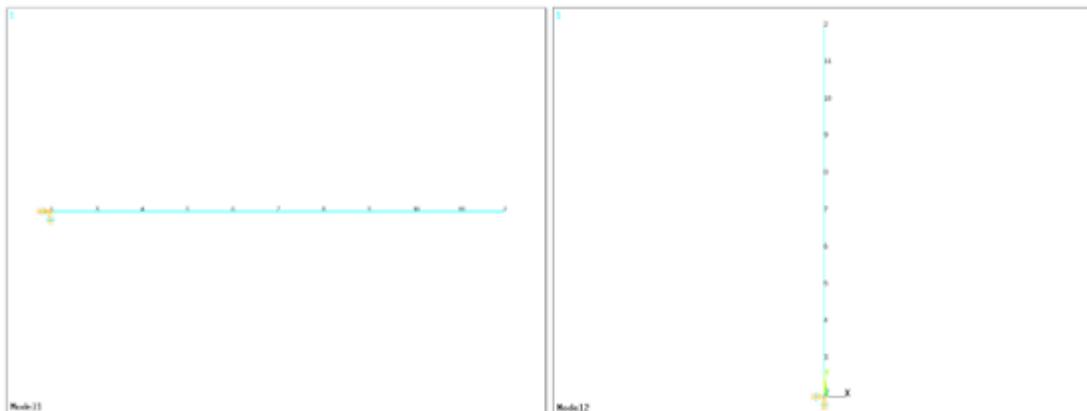
		Case 1	Case 2	Case 3
3	3	0.999	0.990	1.000
4	4	0.999	0.998	1.000
5	5	0.999	0.981	1.000
7	7	0.998	0.969	1.000
8	8	0.997	0.956	1.000

7.3.8.8.2. Comparing Two Cantilever Beams Meshed Along Different Directions

The example shown here demonstrates how **RSTMAC** results can be improved when relevant degrees of freedom and the appropriate matching method are chosen.

Two models of simply supported beam are compared in this example. The first beam is meshed along X direction, and the second one is meshed along Y direction, as shown in [Figure 7.22: Two Simply Supported Beams. \(p. 189\)](#).

Figure 7.22: Two Simply Supported Beams.



```
/filename,Model1
/PREP7
! parameters
! -----
! geometry
    Lb = 1      ! length of beam
    bb = 0.01    ! X-section breadth
    hb = 0.02    ! X-section height
! material
    E   = 1e11
    nu  = 0.3
    rho = 8000
! mesh
    Ndiv = 10
! element types & material
! -----
et,1,188 ! beam element
sectype,1,beam,rect
secdat,bb,hb
mp,ex,1,E
mp,prxy,1,nu
mp,dens,1,rho
```

```

! geometry & meshing
! -----
k,1 ,0 ,0,0
k,2 ,Lb,0,0
l,1,2
allsel

lesize,all,,,Ndiv
seignum,1
lmesh,all

! boundary conditions
! -----
nSEL,s,loc,x,0
d,all,all

allsel
FINISH

/SOLU
antype,modal
modopt,lanb,8
allsel
solve
FINISH

/POST1
set,list
FINISH
/clear,nostart
/filename,Model12
/PREP7

! parameters
! ----

! geometry
Lb = 1      ! length of beam
bb = 0.01 ! X-section breadth
hb = 0.02 ! X-section height
! material
E    = 1e11
nu   = 0.3
rho  = 8000
! mesh
Ndiv = 10

! element types & material
! -----
et,1,188 ! beam element
sectype,1,beam,rect
seCDATA,bb,hb

mp,ex,1,E
mp,prxy,1,nu
mp,dens,1,rho

! geometry & meshing
! -----
k,1 ,0,0 ,0
k,2 ,0,Lb,0
l,1,2
allsel

lesize,all,,,Ndiv
seignum,1
lmesh,all

! boundary conditions
! -----

```

```

nse1,s,loc,y,0
d,all,all
allsel
FINISH

/SOLU
antype,modal
modopt,lanb,8
allsel
solve
FINISH

/POST1
set,list
FINISH

/com =====
/com          MAC COMPARISON
/com =====
/clear,nostart

/POST1
/out
/com -----
/com   Case 1  default matching based on node location
/com       --> One node should be matching
/com -----
rstmac,Modell.rst,,,Model2.rst,,,,,2

/com -----
/com   Case 2  matching based on node number
/com       --> All nodes should be matching
/com       --> Only bending modes in Z direction should match
/com -----
macopt,nummatch,1
rstmac,Modell.rst,,,Model2.rst,,,,,2

/com -----
/com   Case 3  matching based on node number & (UZ + ROTZ) dof only
/com       --> All nodes should be matching
/com       --> All modes should be matching except the 9th
/com -----
macopt,nummatch,1
macopt,dof,rotz
macopt,dof,uz
rstmac,Modell.rst,,,Model2.rst,,,,,2

FINISH

```

The following table summarizes the **RSTMAC** calculations for the three cases above. Matching based on node number is more appropriate in this case. Results can also be improved when specific degrees of freedom are compared: in this case, UZ and ROTZ.

Table 7.6: RSTMAC Command Results for All Cases

	Case 1	Case 2	Case 3
--	---------------	---------------	---------------

Matched nodes	<p>Matching based on node location (DEFAULT)</p> <table border="1" data-bbox="349 240 675 388"> <thead> <tr><th colspan="3">***** MATCHED NODES *****</th></tr> <tr><th>Node in FILE1</th><th>Node in FILE2</th><th>Distance</th></tr> </thead> <tbody> <tr><td>1</td><td>1</td><td>0.0000E+000</td></tr> </tbody> </table>	***** MATCHED NODES *****			Node in FILE1	Node in FILE2	Distance	1	1	0.0000E+000	<p>Matching based on nodes number (MACOPT,NUMMATCH,YES)</p> <table border="1" data-bbox="796 213 1405 620"> <thead> <tr><th colspan="3">***** MATCHED NODES *****</th></tr> <tr><th>Node in FILE1</th><th>Node in FILE2</th><th>Distance</th></tr> </thead> <tbody> <tr><td>1</td><td>1</td><td>0.0000E+000</td></tr> <tr><td>2</td><td>2</td><td>0.1414E+001</td></tr> <tr><td>3</td><td>3</td><td>0.1414E+000</td></tr> <tr><td>4</td><td>4</td><td>0.2828E+000</td></tr> <tr><td>5</td><td>5</td><td>0.4243E+000</td></tr> <tr><td>6</td><td>6</td><td>0.5657E+000</td></tr> <tr><td>7</td><td>7</td><td>0.7071E+000</td></tr> <tr><td>8</td><td>8</td><td>0.8485E+000</td></tr> <tr><td>9</td><td>9</td><td>0.9899E+000</td></tr> <tr><td>10</td><td>10</td><td>0.1131E+001</td></tr> <tr><td>11</td><td>11</td><td>0.1273E+001</td></tr> </tbody> </table>	***** MATCHED NODES *****			Node in FILE1	Node in FILE2	Distance	1	1	0.0000E+000	2	2	0.1414E+001	3	3	0.1414E+000	4	4	0.2828E+000	5	5	0.4243E+000	6	6	0.5657E+000	7	7	0.7071E+000	8	8	0.8485E+000	9	9	0.9899E+000	10	10	0.1131E+001	11	11	0.1273E+001	
***** MATCHED NODES *****																																																			
Node in FILE1	Node in FILE2	Distance																																																	
1	1	0.0000E+000																																																	
***** MATCHED NODES *****																																																			
Node in FILE1	Node in FILE2	Distance																																																	
1	1	0.0000E+000																																																	
2	2	0.1414E+001																																																	
3	3	0.1414E+000																																																	
4	4	0.2828E+000																																																	
5	5	0.4243E+000																																																	
6	6	0.5657E+000																																																	
7	7	0.7071E+000																																																	
8	8	0.8485E+000																																																	
9	9	0.9899E+000																																																	
10	10	0.1131E+001																																																	
11	11	0.1273E+001																																																	
Matched solutions	<p>*** WARNING ***</p> <p>No MAC value is greater than the smallest acceptable value (.9). Solutions matching in RSTMAC failed to identify pairs of solutions.</p>	<p>Only bending modes in Z direction matched (All structural degrees of freedom are considered – DEFAULT)</p> <table border="1" data-bbox="708 798 1090 1009"> <thead> <tr><th colspan="3">*** MATCHED SOLUTIONS ***</th></tr> <tr><th>Substep in FILE1</th><th>Substep in FILE2</th><th>MAC value</th></tr> </thead> <tbody> <tr><td>1</td><td>1</td><td>1.000</td></tr> <tr><td>2</td><td>2</td><td>1.000</td></tr> <tr><td>4</td><td>4</td><td>0.998</td></tr> <tr><td>6</td><td>6</td><td>0.994</td></tr> </tbody> </table>	*** MATCHED SOLUTIONS ***			Substep in FILE1	Substep in FILE2	MAC value	1	1	1.000	2	2	1.000	4	4	0.998	6	6	0.994	<p>All modes matched (Only UZ and ROTZ degrees of freedom are considered using MACOPT,DOF)</p> <table border="1" data-bbox="1111 756 1535 1136"> <thead> <tr><th colspan="3">***** MATCHED SOLUTIONS *****</th></tr> <tr><th>Substep in FILE1</th><th>Substep in FILE2</th><th>MAC value</th></tr> </thead> <tbody> <tr><td>1</td><td>1</td><td>1.000</td></tr> <tr><td>2</td><td>2</td><td>1.000</td></tr> <tr><td>3</td><td>3</td><td>1.000</td></tr> <tr><td>4</td><td>4</td><td>1.000</td></tr> <tr><td>5</td><td>5</td><td>1.000</td></tr> <tr><td>6</td><td>6</td><td>1.000</td></tr> <tr><td>7</td><td>7</td><td>1.000</td></tr> <tr><td>8</td><td>8</td><td>1.000</td></tr> </tbody> </table>	***** MATCHED SOLUTIONS *****			Substep in FILE1	Substep in FILE2	MAC value	1	1	1.000	2	2	1.000	3	3	1.000	4	4	1.000	5	5	1.000	6	6	1.000	7	7	1.000	8	8	1.000
*** MATCHED SOLUTIONS ***																																																			
Substep in FILE1	Substep in FILE2	MAC value																																																	
1	1	1.000																																																	
2	2	1.000																																																	
4	4	0.998																																																	
6	6	0.994																																																	
***** MATCHED SOLUTIONS *****																																																			
Substep in FILE1	Substep in FILE2	MAC value																																																	
1	1	1.000																																																	
2	2	1.000																																																	
3	3	1.000																																																	
4	4	1.000																																																	
5	5	1.000																																																	
6	6	1.000																																																	
7	7	1.000																																																	
8	8	1.000																																																	

Chapter 8: The Time-History Postprocessor (POST26)

Use the POST26 time-history postprocessor ([/POST26](#)) to review analysis results at specific locations in the model as a function of time, frequency, or some other change in the analysis parameters that can be related to time. In this mode, you can process results data in many ways. You can construct graphics displays, chart representations or tabular listings, or you can perform math operations on your data sets. A typical time-history task would be to graph result items versus time in a transient analysis, or to graph force versus deflection in a nonlinear structural analysis.

After you have solved an analysis, the program uses the results data to create a results file. The active results file (* .RST, * .RTH, * .RMG, etc.) is loaded automatically when postprocessing is initiated.

You must have your geometry loaded and a valid results file must be available. If the current analysis contains no results file, you are prompted to provide one.

By default, the time-history processor looks for one of the standard results files described in [The General Postprocessor \(POST1\) \(p. 131\)](#); however, you can specify a different file ([FILE](#)).

Data sets and variable definitions created in the time-history postprocessor are maintained during the current session.

Following is the general process for using the time-history postprocessor:

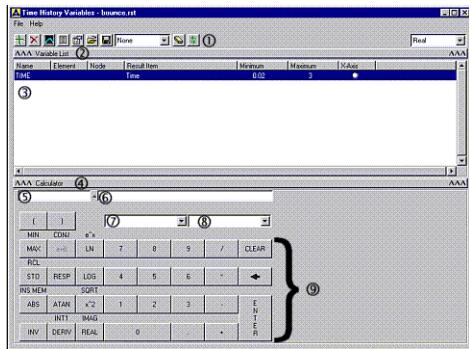
1. Enter the time-history processor ([/POST26](#)).
2. [Define time-history variables \(p. 196\)](#). This involves not only identifying the variables, but also storing the variables.
3. [Process the variables \(p. 199\)](#) to develop calculated data or to extract or generate related variable sets.
4. Prepare output (via graph plots, tabular listings, or file output).

The following POST26 topics are available:

- [8.1. The Time-History Variable Viewer](#)
- [8.2. Defining Variables](#)
- [8.3. Processing Variables to Develop Calculated Data](#)
- [8.4. Importing Data](#)
- [8.5. Exporting Data](#)
- [8.6. Reviewing the Variables](#)
- [8.7. Additional Time-History Postprocessing](#)

8.1. The Time-History Variable Viewer

You can interactively define variables for time-history postprocessing using the variable viewer. A brief description of the variable viewer follows.



1. TOOLBAR

Use the toolbar to control your time-history operations. You can collapse the two expansion bars (2 and 4 below) and retain a compact toolbar that includes these items.

Add Data	Opens the "Add Time-History Variable" dialog. See Defining Variables (p. 196) , later on in this chapter.
Delete Data	Clears selected variable from the Variable List
Graph Data	Graphs up to ten variables according to predefined properties. See Reviewing the Variables (p. 202) , later on in this chapter.
List Data	Generates lists of data, including extremes, for six variables
Properties	You can specify selected variable and global properties
Import Data	Opens dialog for bringing information into the variable space. See Importing Data (p. 200) later on in this chapter
Export Data	Opens dialog for exporting data to a file or an APDL array. See Exporting Data (p. 201) later on in this chapter.
Overlay Data	Drop down list for selecting the data for graph overlay. See Importing Data (p. 200) , later in this chapter
Clear Time-History Data	Clears all variables and returns settings to their default values (RESET).
Refresh Data	Updates variable list. This function is useful if some variables are defined outside of the variable viewer.
Results to View	Drop down list for choosing output form of complex variables (that is, real, imaginary, amplitude, or phase).

2. Hide/Show Variable List

Clicking anywhere on this bar collapses the variable list in order to temporarily reduce the size of the viewer.

3. Variable List

This area will display the defined time-history variables. You can pick from within this list to select and process your variables.

4. Hide/Show Calculator

Clicking anywhere on this bar collapses the calculator to reduce the size of the viewer.

5. Variable Name Input Area

Enter the name (32 character max.) of the variable to be created.

6. Expression Input Area

Enter the expression associated with the variable to be created.

7. APDL Variable Drop Down List

Select a currently-defined APDL variable to use in the expression input.

8. Time-History Variable Drop Down List

Select from previously-stored variables to use in the expression input.

9. Calculator Area

Use the calculator to add standard mathematical operators and functions to the expression input. You click the buttons to enter the function into the expression input area. Clicking the INV button enables the alternate selections shown above the buttons. For examples on how to use the calculator functions, see [Processing Variables to Develop Calculated Data \(p. 199\)](#) in this chapter.

PARENTHESIS	Use the parenthesis to set off the hierarchy of operations, just as you would in any algebraic expression. Many functions will automatically insert parenthesis when needed.
MAX / MIN	Finds the largest of three variables (LARGE) / Finds the smallest of three variables (SMALL)
COMPLEX / CONJUGATE	Forms a complex variable / Forms the complex conjugate of a variable (CONJUG).
LN / e ^X	Forms the natural log of a variable (NLOG) / Forms the exponential of a variable (EXP).
STO / RCL	Stores active information from the expression input area into a memory location / Recalls the memory location for repeated use in an expression.
CVAR	Calculates the covariance between two variables (CVAR). Only available for random vibration (PSD) analyses.
RPSD	Calculates the response power spectral density (RPSD). Only available for random vibration (PSD) analyses.
RESP	Calculates the response power spectrum (RESP) from time-history data. Available for transient analyses.
LOG	Forms the common log of a variable (CLOG).

ABS / INS MEM	Forms the absolute value of a variable. For a complex number, the absolute value is the magnitude (ABS) / Inserts the contents of a memory location into an expression.
ATAN	Forms the arctangent of a complex variable (ATAN).
X ² / SQRT	Forms the square of a variable (PROD) / Forms the square root of a variable (SQRT).
INV	Use this key to make the alternate calculator functions (shown above the buttons) available.
DERIV / INT	Forms the derivative of a variable (DERIV) / Forms the integral of a variable (INT1).
REAL / IMAG	Forms a variable using only the real part of a complex variable (REALVAR) / Forms a variable using only the imaginary part of a complex variable (IMAGIN).
11 KEY NUMBER PAD	Enters real numbers into the expression input area.
/	Calculates the quotient of two variables (QUOT).
*	Calculates the product of two variables (PROD).
-	Calculates the difference between two variables (ADD).
+	Calculates the sum of two variables (ADD).
CLEAR	Clears all data from the variable and expression input area.
BACKSPACE	Backspace from the current cursor location deleting preceding characters.
ENTER	Calculates the expression in the expression input area and stores the result in the variable specified in the variable input area (STORE).

8.2. Defining Variables

Time-history operations deal with *variables*, tables of result item versus time (or versus frequency). The result item may be the UX displacement at a node, the heat flux in an element, the force developed at a node, the stress in an element, or the magnetic flux in an element, for example.

You assign unique identifiers to each of the variables. Up to 200 such variables can be defined.

TIME is reserved for the time value. FREQ is reserved for the frequency value. All other identifiers must be unique, and can be made up of 32 letters and characters.

If you do not provide a unique identifier, the program assigns one.

In addition to the unique identifiers, the program uses numerical indices (reference numbers) to track and manipulate the variables. These numbers can be used interchangeably with the identifiers at the command level, and in some interactive operations. The numerical index is displayed, along with any name you choose in the data properties dialog box.

Defining variables is actually a two-part operation consisting of defining and storing:

8.2.1. Defining the Variable

8.2.2. Storing the Variable

8.2.1. Defining the Variable

Define a variable according to the result item in the results file. To do so, set up pointers to the result item and create labels for the areas where that data will be stored.

You can define up to ten variables by default. The maximum number can be increased up to 200 (**NUMVAR**).

TIME (time) or FREQ (frequency) is always variable 1.

Example 8.1: Defining Time-History Variables Two, Three and Four

```
NSOL,2,358,U,X,UX_at_node_358
ESOL,3,219,47,EPEL,X,Elastic_Estrain
ANSOL,4,101,S,X,Avtg_Stress_101
```

Variable two is a nodal result defined by the **NSOL** command. It is the UX displacement at node 358.

Variable three is an element result defined by the **ESOL** command. It is the X component of elastic strain at node 47 for element 219.

Variable four is an averaged element nodal result defined via the **ANSOL** command. It is the X-component of averaged element nodal stress at node 101.

Any subsequent reference to these results items occurs via either the reference numbers or the labels assigned to them.

Defining a new variable with the same number as an existing variable overwrites the existing variable.

The following commands are available for defining variables:

- **ANSOL** -- Specifies averaged nodal data to be stored from the results file in the solution coordinate system.
- **ESOL** -- Specifies element data to be stored from the results file.
- **GAPF** -- Defines the gap force data to be stored in a variable.
- **NSOL** -- Specifies nodal data to be stored from the results file.
- **RFORCE** -- Specifies the total reaction force data to be stored.
- **SOLU** -- Specifies solution summary data per substep to be stored.

These commands define the location of the results:

- **FORCE** -- Selects the element nodal force type for output.
- **LAYERP26** -- Specifies the element layer for which data are to be stored.
- **SHELL** -- Selects a shell element or shell layer location for results output.

Other useful commands for defining variables are:

- **NSTORE** - Specifies the number of time points or frequency points to be stored.

- **TIMERANGE** - Specifies the time or frequency range in which data are to be stored.
- **TVAR** - Changes the meaning of default variable 1 TIME to a cumulative iteration number.
- **VARNAM** - Assigns a name (up to 32 character) to a variable.
- **RESET** - Removes all variables and resets all specifications to initial defaults.

8.2.1.1. Special Considerations for Defining Variables

The force (or moment) values by default represent the *total* forces (sum of the static, damping, and inertial components). The **FORCE** command allows you to work with the individual components. (The **FORCE** command affects only the output of element nodal forces.)

Results data for shell elements and layered elements are by default assumed to be at the top surface of the shell or layer. The **SHELL** command allows you to specify the top, middle or bottom surface. For layered elements, use the **LAYERP26** and **SHELL** commands to indicate layer number and surface location, respectively.

8.2.2. Storing the Variable

Issue the **STORE** command to store a defined variable. Storing means reading the data from the results file into the database.

The program also stores data automatically when you issue display commands (**PLVAR** and **PRVAR**) or time-history data operation commands (**ADD**, **QUOT**, etc.).

Example 8.2: Using the **STORE** Command

```
/POST26
NSOL,2,23,U,Y      ! Variable 2 = UY at node 23
SHELL, TOP          ! Specify top of shell results
ESOL,3,20,23,S,X   ! Variable 3 = top SX at node 23 of element 20
PRVAR,2,3           ! Store and then print variables 2 and 3
SHELL, BOT          ! Specify bottom of shell results
ESOL,4,20,23,S,X   ! Variable 4 = bottom SX at node 23 of element 20
STORE               ! By command default, place variable 4 in memory with 2 and 3
PLVAR,2,3,4         ! Plot variables 2,3,4
```

In some situations, it is necessary to explicitly request storage. If you issue the **STORE** command after issuing **TIMERANGE** or **NSTORE**, the default is **STORE,NEW**. Otherwise, it is **STORE,MERGE** as listed in the command descriptions below.

This change in the **STORE** command default is required because the **TIMERANGE** and **NSTORE** commands redefine time (or frequency) points and time increment for data storage.

8.2.2.1. Variable Storage Options

The following **STORE** command options are available for storing variable data:

- MERGE -- Adds newly defined variables to previously stored variables for the time points stored in memory, useful if you wish to store data using one specification (**FORCE**, **SHELL**, **LAYERP26**) and store data using another specification, as shown in the prior example.

- NEW -- Replaces previously stored variables, erases previously calculated variables, and stores newly defined variables with current specifications.
- APPEND -- Appends data to previously stored variables. That is, if you think of each variable as a column of data, the APPEND option adds rows to each column. This is useful when you want to "concatenate" the same variable from two files, such as in a transient analysis with results on two separate files. Issue the **FILE** command to specify results file names.
- ALLOC,*N* -- Allocates space for *N* points (*N* rows) for a subsequent storage operation. Previously stored variables, if any, are cleared. Typically, this option is unnecessary because the program determines the number of points required automatically from the results file.

8.3. Processing Variables to Develop Calculated Data

Specific analysis data in the results file can often be processed to yield additional variable sets that provide valuable information. For example, by defining a displacement variable in a transient analysis, you can calculate the velocity and acceleration by taking derivatives with respect to time. Doing so yields an entirely new variable that you may wish to analyze along with other analysis data.

Some commands identify the variable and the format for the output, while others identify the variable data to be used to create the new variable. The actual calculations are performed by specific commands.

Example 8.3: Finding the Derivative of a Variable UYBLOCK with Respect to Variable TIME

```
NSOL,2,100,u,y,UYBLOCK ! Variable 2 is UY of node 100
DERIV, 3,2,1,,VYBLOCK ! Variable 3 is named VYBLOCK It is the
                       ! derivative of variable 2 with respect
                       ! to variable 1 (time)
```

Example 8.4: Finding the Amplitude of Complex Time-History Variable PRESMID

```
NSOL,2,123,PRES,,PRESMID ! Variable 2 is the pressure at node 123
ABS, 3,2,,,AMPL_MID    ! Absolute value of a complex variable
                         ! is its amplitude.
```

Example 8.5: Finding the Phase Angle (in Degrees) of a Complex Time-History Variable UYFANTIP

```
Pi = acos(-1)
ATAN,4,2,,,PHAS_MID,,,180/pi ! ATAN function of a complex
                               ! variable (a + ib) gives atan (b/a)
```

Example 8.6: Multiplying a Complex POST 26 Variable PRESMID with a Factor (2+3i)

```
CFACT,2,3           !Scale factor of 2+3i
ADD,5,2,,,SCAL_MID ! Use ADD command to store variable 2 into
                     ! variable 5 with the scale factor of (2+3i)
```

Example 8.7: Filling a Variable with Ramped Data

```
FILLDATA,6,,,.25,.05,ramp_func !Fill a variable with
                                !ramp function data.
```

The following commands are available to process variables, develop calculated relationships, and store the data:

ABS -- Forms the absolute value of a variable.

ADD -- Adds (sums) variables.

ATAN -- Forms the arctangent of a complex variable.

CLOG -- Forms the common log of a variable.

CONJUG -- Forms the complex conjugate of a variable.

DERIV -- Differentiates a variable.

EXP -- Forms the exponential of a variable.

IMAGIN -- Forms an imaginary variable from a complex variable.

INT1 -- Integrates a variable.

LARGE -- Finds the largest (the envelope) of three variables.

NLOG -- Forms the natural log of a variable.

PROD -- Multiplies variables.

QUOT -- Divides two variables.

REALVAR -- Forms a variable using only the real part of a complex variable.

SMALL -- Finds the smallest of three variables.

SQRT -- Forms the square root of a variable.

RPSD -- Calculates response power spectral density (PSD).

CVAR -- Calculates covariance between two quantities.

RESP -- Generates a response spectrum.

8.4. Importing Data

The ability to import data enables you to read in sets of data from a file into time-history variable(s). For example, you can display and compare test results data against the corresponding program results data.

Import data from a file into a time-history variable via one of the following methods:

- Issue the **DATA** command to read in a pre-formatted file. The file should be in FORTRAN format (**DATA**).
- Read the data from a free-format-, comma-, blank- or tab-delimited file. Store the data as a time-history variable by:
 - Reading the file into a table array (***TREAD**). This step requires that you know the number of data points in the file, as you must pre-specify the table array size (***DIM**).

2. Storing the array into a time-history variable (**VPUT**). You can store one array at a time into a time-history variable
- The following two external commands (issued in the order shown) are available to facilitate importing data into time-history variables.

```
1. ~eui,'ansys::results::timeHist::TREAD directorypath/filename
arrayname'
```

Determines the size of the data file, creates a table array of name 'arrayname', appropriately dimensions the array based on the number of data sets in the file, then reads the data into the array.

```
2. ~eui,'ansys::results::timeHist::vputData arrayname variablenumber'
```

Puts the data stored in the 'arrayname' table into the time-history variables, beginning with variable ID 'variable number'.

For Example:

Example 8.8: Importing Data

```
~eui,'ansys::results::timeHist::TREAD d:\test1\harmonic.prn TESTMID'
~eui,'ansys::results::timeHist::vputData TESTMID 5'
```

The first command reads data sets from the file harmonic.prn in the directory d:\test1 and stores this data in to the table array TESTMID.

The second command imports the data from TESTMID array into time-history variables beginning with variable number 5. If the harmonic.prn file contains multiple data sets, the first data set is stored in variable 5, the next data set in variable 6, and so on. If these variables have already been defined, they are overwritten.

8.5. Exporting Data

The ability to export data enables you to write out selected time-history variables to an ASCII file or to an APDL array/table parameter, useful for passing data on to another program for further processing or to archive data in an easily retrievable format.

Exporting data from a time-history variable into a file is a two-step process:

1. Export a time-history variable data to an array parameter (**VGET**). The size of the array (***DIM**) can be determined via ***GET,SIZE,vari,,nsets**.
2. When the array is filled, the data can be written out to a file (***VWRITE**).

Example 8.9: Exporting Data

```
NSOL,5,55,U,X
STORE,MERGE           ! Store UX at node 55
*GET,SIZE,VARI,,NSETS
*dim,UX55,array,SIZE ! Create array parameter
VGET,UX55(1),5        ! Store time-history data of variable 5 into ux55
*COPEN,disp,dat
```

```
*VWRITE,UX55(1)      ! Write array in given format to file "disp.dat"
(6x,f12.6)
*CFCLOSE
```

8.6. Reviewing the Variables

After the variables have been defined, you can review them via graph plots or tabular listings.

8.6.1. Plotting Result Graphs

The **PLVAR** command graphs up to 10 variables at a time on a graph. By default, the variable used for the X-axis of the graphs is TIME for static and transient analyses or FREQUENCY for harmonic analysis. You can specify a different variable (such as deflection or strain) for the X axis (**XVAR**).

When plotting complex data, such as from a harmonic analysis, **PLVAR** plots amplitude by default. You can switch to plotting Phase angle, or Real part or Imaginary part (**PLCPLX**).

You can display a portion of the stored data by selecting a range for the X axis values (**/XRANGE**).

Figure 8.1: Example Time-History Plot (XVAR = 1 (time))

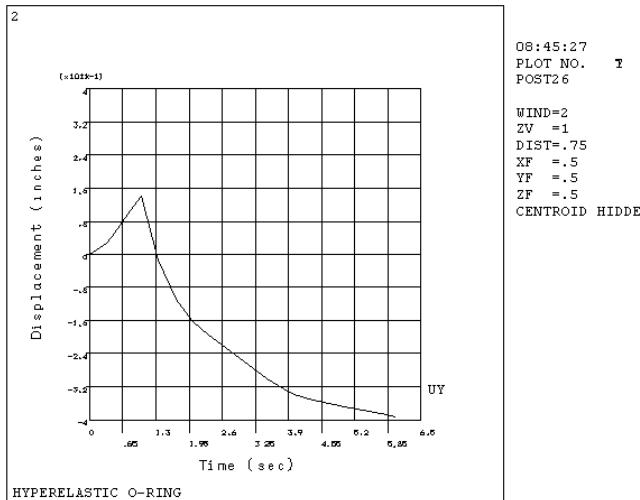
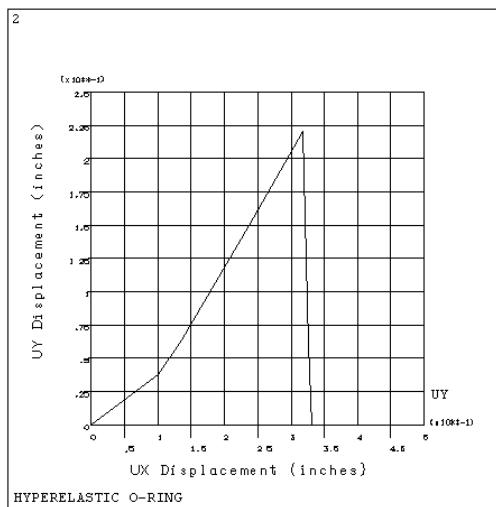


Figure 8.2: Example Time-History Plot (XVAR ≠ 1)

For more information about adjusting the look and feel of your graph plots, see [Creating Graphs \(p. 257\)](#).

8.6.2. Listing Results in Tabular Form

Issue the **PRVAR** command to list up to six variables in tabular form, useful if you want to find the value of a result item at a specific time or frequency.

You can control the times (or frequencies) for which variables are to be printed (**NPRINT** and **PRTIME**).

You can adjust the format of your listing somewhat (**LINES**).

Example 8.10: Example Output from PRVAR

***** Time-History VARIABLE LISTING *****		
TIME	51 UX	30 UY
	UX	UY
.10000E-09	.000000E+00	.000000E+00
.32000	.106832	.371753E-01
.42667	.146785	.620728E-01
.74667	.263833	.144850
.87333	.310339	.178505
1.0000	.356938	.212601
1.3493	.352122	.473230E-01
1.6847	.349681	-.608717E-01

When a complex variable consists of real and imaginary parts, both parts are listed by default. You can work with real and imaginary, or amplitude and phase angle (**PRCPLX**).

Another useful listing command is **EXTREM**, which prints the maximum and minimum Y-variable values within the active X and Y ranges. You can also assign these extreme values to parameters via the ***GET** command.

Example 8.11: Output from EXTREM

Time-History SUMMARY OF VARIABLE EXTREME VALUES					
VARI	TYPE	IDENTIFIERS	NAME	MINIMUM	AT TIME
					MAXIMUM AT TIME

```

1 TIME      1 TIME      TIME      .1000E-09  .1000E-09  6.000    6.000
2 NSOL      50 UX       UX       .0000E+00  .1000E-09  .4170    6.000
3 NSOL      30 UY       UY       -.3930     6.000     .2146    1.000

```

8.7. Additional Time-History Postprocessing

The following are additional time-history postprocessing features:

8.7.1. Generating a Response Spectrum

8.7.2. Data Smoothing

8.7.1. Generating a Response Spectrum

The **RESP** command generates a displacement, velocity, or acceleration response spectrum from a given displacement or acceleration time-history. The response spectrum can then be specified in a single-point or multiple point spectrum analysis (**ANTYPE,SPECTR** with **SPOPT,SRSS** or **MPRS**) to calculate the overall response of a structure. For theory details about the response spectrum calculation, see **POST26 - Response Spectrum Generator (RESP)** in the *Mechanical APDL Theory Reference*.

RESP requires two previously defined variables: one containing frequency values for the response spectrum (field *LFTAB*) and the other containing the time-history (field *LDTAB*). The frequency values in *LFTAB* represent not only the abscissa of the response spectrum curve, but also the frequencies of the one-degree-of-freedom oscillators used to generate the response spectrum. You can create the *LFTAB* variable using either the **FILLDATA** command or the **DATA** command.

The displacement time-history values in *LDTAB* result from a full or mode-superposition transient dynamic analysis. You can create the *LDTAB* variable via the **DATA** command (if the time-history is on a file) or the **NSOL** command. A numerical time-integration scheme is used to calculate the response spectrum.

An example of response spectrum generation based on an acceleration time-history following a full or mode-superposition transient analysis follows. Note that the expansion pass is required to obtain the acceleration time-history in a mode-superposition analysis.

Example 8.12: Response Spectrum Generation Based on Acceleration Time-History

```

/post26

my_node = node(0,0,10)
nsol, 2, my_node, ACC, Z, az10      ! Define variable 2 as acceleration along Z at my_node
plvar, 2

filldata, 3,,, 0, 0.25            ! Define variable 3 as values from 0 with 0.25 increment (default end value)

resp, 4, 3, 2, 3, 0.03          ! Displacement response spectrum generation in variable 4 based on
                                ! frequency values in variable 3 and time-history in variable 2;
                                ! last "3" on resp command specifies acceleration spectrum to be calculated.
plvar, 4

```

An example of response spectrum generation based on a displacement time-history following a mode-superposition transient analysis follows. Note that the expansion pass is not needed in this case.

Example 8.13: Response Spectrum Generation Based on Displacement Time-History

```
/post26

file,,rdsp           ! Time-history is on the reduced displacement (RDSP) file

my_node = node(0,0,10)
nsol, 2, my_node, U, Y, uy10 ! Define variable 2 as displacement along Y at my_node
plvar, 2

filldat, 3,,, 0, 0.25      ! Define variable 3 as values from 0 with 0.25 increment (default end value)

resp, 4, 3, 2, 1,,,,, 1    ! Acceleration response spectrum generation in variable 4 based on
                           ! frequency values in variable 3 and time-history in variable 2;
                           ! first "1" on command specifies displacement spectrum to be calculated,
                           ! and last "1" specifies acceleration time-history input.

plvar, 4
finish
```

8.7.2. Data Smoothing

If you are working with noisy results data, you may want to smooth the response. Smoothing may allow for better understanding or visualization of the response by smoothing out local fluctuations while preserving the global characteristics of the response. The time-history smooth operation allows fitting a n^{th} order polynomial to the actual response.

This operation can be used only on static or transient results. Complex data cannot be fitted.

Four arrays are required for smoothing data. The first two contain the noisy data from the independent and the dependent variables, respectively; the second two will contain the smoothed data (after smoothing takes place) from the independent and dependent variables, respectively. You must always create the first two vectors (***DIM**) and fill these vectors with the noisy data (**VGET**) before smoothing the data. If you are working in interactive mode, the program automatically creates the third and fourth vector, but if you are working in batch mode, you must also create these vectors (***DIM**) before smoothing the data.

When the arrays have been created, issue the **SMOOTH** command. You can opt to smooth all or some of the data points via the command's *DATAP* argument, and you can specify the fitting order for the smoothed curve via the *FITPT* argument. *DATAP* defaults to all points, and *FITPT* defaults to one-half of the data points. To plot the results, you can choose to plot unsmoothed, smoothed, or both sets of data.

Chapter 9: Selecting and Components

If you have a large model, it is helpful to work with just a portion of the model data to apply loads, to speed up graphics displays, to review results selectively, and so on. Because all data reside in a database, you can conveniently *select* subsets of the data.

Selecting enables you to group subsets of nodes, elements, keypoints, lines, etc. so that you can work with just a handful of entities. The program uses a database to store all the data that you define during an analysis. The database design allows you to select only a portion of the data without destroying other data.

Typically, you perform selecting when applying loads. By selecting nodes on a surface, for example, you can conveniently apply a pressure on *all* nodes in the subset instead of applying it to each individual node.

Another useful feature of selecting is that you can select a subset of entities and name that subset. For example, you can select all elements that make up the fin portion of a heat exchanger model and call the resulting subset FIN. Such named subsets are called *components*. You can even group several components into an *assembly*.

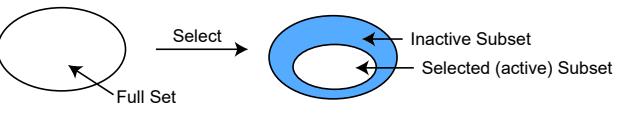
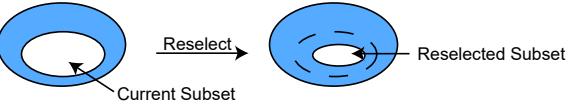
The following topics concerning selecting and components are available:

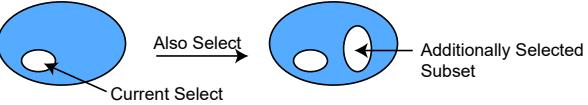
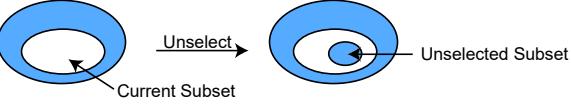
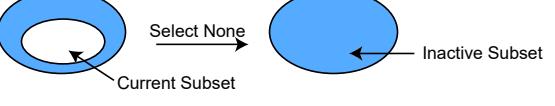
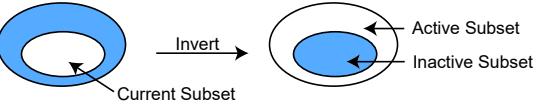
- [9.1. Selecting Entities](#)
- [9.2. Selecting for Meaningful Postprocessing](#)
- [9.3. Using Components and Assemblies](#)

9.1. Selecting Entities

You can select a subset of entities via a combination of the basic select functions shown in the following table:

Table 9.1: Selection Functions

Select Selects items from the full set of data.	
Reselect Selects (again)	

from the selected subset.	
Also Select Adds a different subset to the current subset.	
Unselect Subtracts a portion of the current subset.	
Select All Restores the full set.	
Select None Deactivates the full set (opposite of Select All).	
Invert Switches between the active and inactive portions	

of the set.	
-------------	--

The functions are available for all entities (nodes, elements, keypoints, lines, areas, and volumes) on the **Utility Menu** or via commands.

For additional information about picking, see [Graphical Picking](#) in the *Operations Guide*.

The following additional topics about selecting are available:

- 9.1.1. Selecting Entities Using Commands
- 9.1.2. Selecting Entities Using the GUI
- 9.1.3. Selecting Lines to Repair CAD Geometry
- 9.1.4. Other Commands for Selecting

9.1.1. Selecting Entities Using Commands

Table 9.2: Select Commands (p. 209) shows a summary of commands available to select subsets of entities. Notice the "crossover" commands: commands that allow you to select one entity based on another entity. For example, you can select all keypoints attached to the current subset of lines. Here is a typical sequence of select commands:

```
LSEL,S,LOC,Y,2,6      ! Select lines that have center locations between Y=2 and Y=6
LSEL,A,LOC,Y,9,10     ! Add lines with center locations between Y=9 and Y=10
NSLL,S,1              ! Select all nodes on the selected lines
ESLN                  ! Select all elements attached to selected nodes
```

See the **LSEL**, **NSLL**, and **ESLN** command descriptions in the [Command Reference](#) for further information.

Note:

Crossover commands for selecting finite element model entities (nodes or elements) from solid model entities (keypoints, areas, etc.) are valid only if the finite element entities were generated by a meshing operation on a solid model that contains the associated solid-model entities.

Table 9.2: Select Commands

Entity	Basic Commands	Crossover Command(s)
Nodes	NSEL	NSLE, NSLK, NSLL, NSLA, NSLV
Elements	ESEL	ESLN, ESLL, ESLA, ESLV
Keypoints	KSEL	KSLN, KSLL
Hard Points	KSEL, ASEL, LSEL	None
Lines	LSEL	LSLA, LSLK
Areas	ASEL	ASLL, ASLV
Volumes	VSEL	VSLA
Components	CMSEL	None

9.1.2. Selecting Entities Using the GUI

The GUI path equivalent to issuing most of the commands listed in [Table 9.2: Select Commands \(p. 209\)](#) is **Utility Menu> Select> Entities**.

The GUI option displays the Select Entities dialog box, from which you can choose the type of entities you want to select and the criteria by which you will select them. For example, you can choose **Elements** and **By Num/Pick** to select elements by number or by picking.

Plotting One Entity Type and Selecting Another

It is sometimes useful to plot one entity type and select another. For example, in a model with hidden faces, you may want to obtain a wire-frame view. You can do so by plotting the lines via **Utility Menu> Plot> Lines (Lplot)**, and then selecting areas using graphical picking via **Utility Menu> Select> Entities> Areas> By Num/Pick (ASEL,S,PICK)**. This method is available by default.

Combining Entities Into Components or Assemblies

You will likely want to combine entities into [components or assemblies \(p. 212\)](#) wherever possible for clarity or ease of reference. The following GUI paths provide selection options for defined components or assemblies:

GUI:

Utility Menu> Select> Comp/Assembly> Select All
Utility Menu> Select> Comp/Assembly> Select Comp/Assembly
Utility Menu> Select> Comp/Assembly> Pick Comp/Assembly
Utility Menu> Select> Comp/Assembly> Select None

9.1.3. Selecting Lines to Repair CAD Geometry

When CAD geometry is imported into ANSYS, the transfer may define the display of short line elements, which are difficult to identify on screen.

By choosing the line selection option, you can find and display these short lines:

Command(s): LSEL

GUI: Utility Menu> Select> Entities> Lines> By Length/Radius

Enter the minimum and maximum length or radius in the *VMIN* and *VMAX* fields. These fields, as they are used in this option, represent the range of values which corresponds to the length or radius of the short line elements. You should enter reasonable values in *VMIN* and *VMAX* to assure that the selected set only includes those short lines that you want to display. When the selected set appears on screen, you can pick individual lines within the set and repair the geometry as necessary.

Note:

A line which is not an arc returns a zero radius. RADIUS is only valid for lines that are circular arcs.

9.1.4. Other Commands for Selecting

To restore all entities to their full sets, use one of the following:

Command(s): ALLSEL

GUI: Utility Menu> Select> Everything Below> Selected Areas

Utility Menu> Select> Everything Below> Selected Elements

Utility Menu> Select> Everything Below> Selected Lines

Utility Menu> Select> Everything Below> Selected Keypoints

Utility Menu> Select> Everything Below> Selected Volumes

This one command has the same effect as issuing a series of **NSEL,ALL; ESEL,ALL; KSEL,ALL;** etc. commands.

You also can use **ALLSEL** or its GUI equivalents to select a set of related entities in a hierarchical fashion. For example, given a subset of areas, you can select all lines defining those areas, all keypoints defining those lines, all elements belonging to these areas, lines, and keypoints, and all nodes belonging to these elements, by simply issuing one command: **ALLSEL,BELOW,AREA**

To select a subset of degree of freedom and force labels, use one of the following:

Command(s): DOFSEL

GUI: Main Menu> Preprocessor> Loads> Define Loads> Operate> Scale FE Loads> Constraints

Main Menu> Preprocessor> Loads> Define Loads> Operate> Scale FE Loads> Forces

Main Menu> Preprocessor> Loads> Define Loads> Settings> Replace vs Add> Constraints

Main Menu> Preprocessor> Loads> Define Loads> Settings> Replace vs Add> Forces

Main Menu> Solution> Define Loads> Operate> Scale FE Loads> Constraints

Main Menu> Solution> Define Loads> Operate> Scale FE Loads> Forces

Main Menu> Solution> Define Loads> Settings> Replace vs Add> Constraints

Main Menu> Solution> Define Loads> Settings> Replace vs Add> Forces

By selecting a subset of these labels, you can simply use ALL in the *Label* field of some commands to refer to the entire subset. For instance, the command **DOFSEL,S,UX,UZ** followed by the command **D,ALL,ALL** would put UX and UZ constraints on all selected nodes. **DOFSEL** does not affect the solution degrees of freedom.

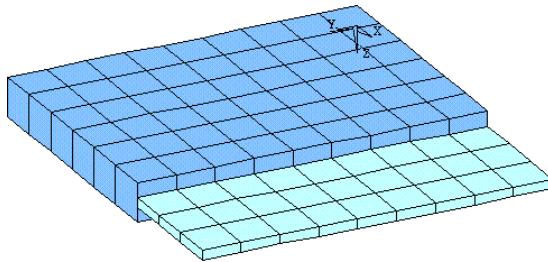
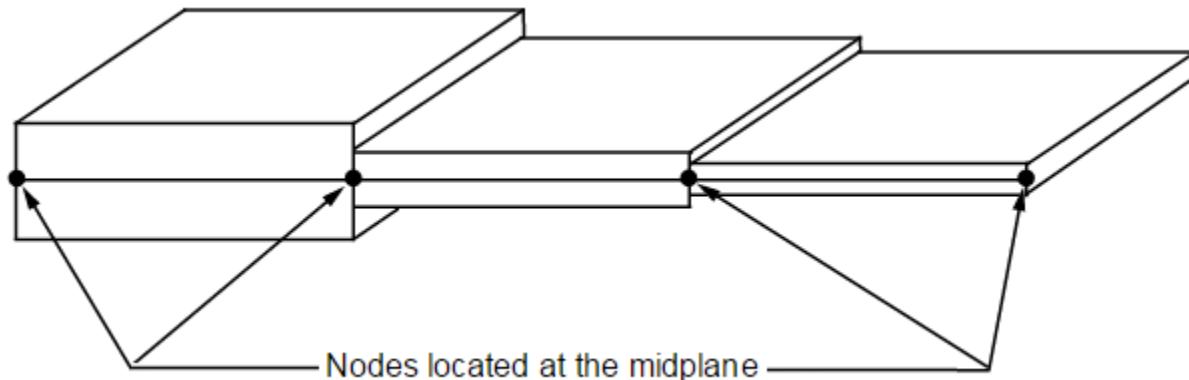
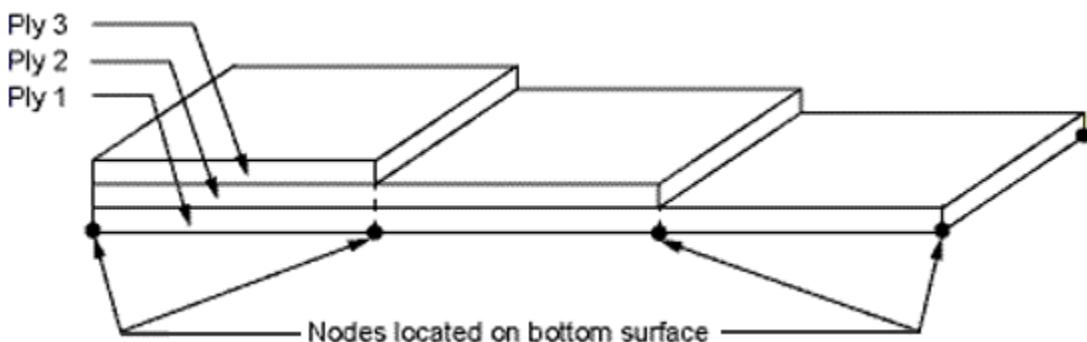
9.2. Selecting for Meaningful Postprocessing

Selecting can help during postprocessing. During POST1 postprocessing, for example, you can select just a portion of your model to display or list the results. You *should always* use selecting to obtain meaningful results in POST1 when the model has discontinuities.

When you request contour displays (**PLNSOL**), the program produces smooth, continuous contours by averaging the data at nodes. Averaging is acceptable so long as the model contains no discontinuities, such as:

- Two different materials modeled adjacent to each other or a model with different thicknesses. (See [Figure 9.1: Shell Model with Different Thicknesses \(p. 212\)](#))
- Adjacent layered shells having a different number of layers. (See [Figure 9.2: Layered Shell \(SHELL281\) with Nodes Located at Midplane \(p. 212\)](#) and [Figure 9.3: Layered Shell \(SHELL281\) with Nodes Located at Bottom Surface \(p. 212\)](#))

When such discontinuities are present, process each side of the discontinuity separately via selecting.

Figure 9.1: Shell Model with Different Thicknesses**Figure 9.2: Layered Shell (**SHELL281**) with Nodes Located at Midplane****Figure 9.3: Layered Shell (**SHELL281**) with Nodes Located at Bottom Surface**

9.3. Using Components and Assemblies

It is often convenient to assign a recognizable name to a given part of your model so that you can work with that part discretely.

For example, you can select entities from a portion of your model and group them into named components, then group those components into an assembly named WHEEL2. You can then apply boundary conditions to WHEEL2, mesh WHEEL2 with nodes and elements, or plot a graphic display of WHEEL2.

A named group of entities is called a *component*. A group of components is called an *assembly*.

The following related topics are available:

[9.3.1. Component Types](#)

- 9.3.2. Using the Component Manager
- 9.3.3. Creating Components
- 9.3.4. Nesting Assemblies
- 9.3.5. Selecting Entities by Component or Assembly
- 9.3.6. Adding or Removing Components
- 9.3.7. Modifying Components or Assemblies
- 9.3.8. Viewing Hidden Element Components

9.3.1. Component Types

A component consists of *one* of the following types of entities: nodes, elements, keypoints, lines, areas, or volumes. A single component cannot have more than one entity type.

Most common are nodal components and element components.

9.3.2. Using the Component Manager

The Component Manager (**Utility Menu> Select> Component Manager**) provides convenient access to your component operations. The Component Manager provides a coordinated and integrated interface to the capabilities of the following commands:

CM	CMDELETE	CMEDIT
CMGRP	CMLIST	CMMOD
CM PLOT	CMSEL	

You can access each command's capability via either the Component Manager or individual GUI paths. The following sections describe the individual component commands and the function that you can perform with each. See the appropriate command documentation for specific capabilities and limitations.

Using the Component Manager toolbar buttons (except **Select Component/Assembly** and **Unselect Component/Assembly**) performs the specified operation on the highlighted component(s), but the select status of the entities in the database are not affected.

9.3.3. Creating Components

Issue the **CM** command to define a component. For example, you can select all elements that constitute the rotor portion of a motor model and group them into a component:

```
ESEL,,MAT,,2 ! Select rotor elements (material 2)
CM,ROTOR,ELEM ! Define component ROTOR using all selected elements
```

The *Command Reference* describes the **ESEL** and **CM** commands in more detail.

An *assembly* may consist of any number of components and other assemblies. Issue the **CMGRP** command to define an assembly. For example, you can group the components ROTOR and WINDINGS (both of which must have been previously defined) into an assembly ROTORASM:

```
NSEL,... ! Select appropriate nodes and
ESLN ! elements that constitute the windings
```

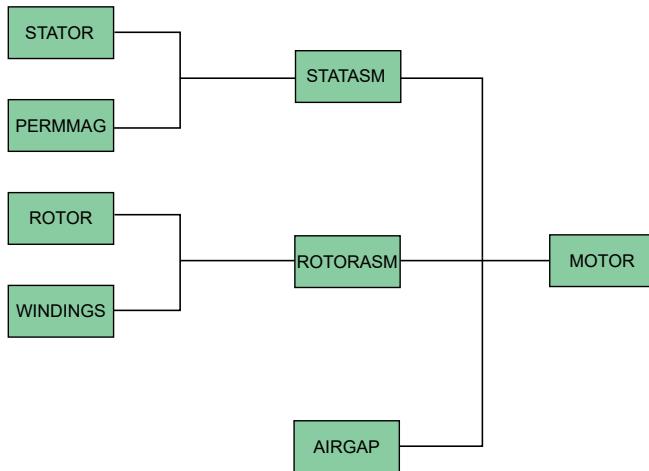
```
CM,WINDINGS,ELEM           ! Define component WINDINGS
CMGRP,ROTORASM,WINDINGS,ROTOR ! Define the assembly ROTORASM
```

The [Command Reference](#) describes the **NSEL**, **ESLN**, **CM**, and **CMGRP** commands in more detail.

9.3.4. Nesting Assemblies

You can nest assemblies up to five levels deep. For example, you can build an assembly named MOTOR from other assemblies and components as shown in the schematic below.

Figure 9.4: Nested Assembly Schematic



Assuming that the assembly ROTORASM and components STATOR, PERMMAG, and AIRGAP have been defined, the commands to define the assembly MOTOR would look like this:

```
CMGRP,STATASM,STATOR,PERMMAG
CMGRP,MOTOR,STATASM,ROTORASM,AIRGAP
```

See the [Command Reference](#) for more information about the **CMGRP** command.

9.3.5. Selecting Entities by Component or Assembly

The main advantage of defining a component or an assembly is that you can conveniently select items that belong to it using a combination of the **CMSEL** and **ALLSEL** commands. The **CMSEL** command selects all entities belonging to a component or assembly by its name. You can then issue **ALLSEL,BELOW** to select all attached lower entities. For example, you can select all elements belonging to the WINDINGS component, apply a current density loading to all of them, and then select all nodes attached to those elements:

```
CMSEL,,WINDINGS
BFE,ALL,JS,,,-1000
ALLSEL,BELOW,ELEM
```

You can also use the picker to select components. By selecting **Utility Menu> Select> Comp/Assembly> Pick Comp/Assembly**, you can select a defined component and all of the items belonging to it. The item is displayed in the prompt window during the select process.

For more information about the **CMSEL**, **BFE** and **ALLSEL** commands, and the **CMEDIT**, **CMDELETE**, and **CMLIST** commands mentioned below, see the *Command Reference*.

9.3.6. Adding or Removing Components

Issuing the **CMEDIT** command enables you to add components to or remove components from an assembly. For example, the following command removes AIRGAP from the assembly MOTOR:

```
CMEDIT,MOTOR,DELE,AIRGAP
```

Delete a component or assembly definition via the **CMDELETE** command. List the entities that make up a particular component via the **CMLIST** command. To generate expanded, detailed listings of the entities that make up specific components, issue **CMLIST** along with **CMSEL**.

The **CMSEL** command also enables you to use components to narrow your selection or increase your selection criteria. Issuing **CMSEL,ALL** will select all defined components *in addition* to any items already selected.

9.3.7. Modifying Components or Assemblies

Modify the specification of a component via the **CMMOD** command.

If an entity is modified (via the **KMODIF** command, for example), that entity may be deleted and then redefined. The deletion may cause the entity to be removed from the component. If all of the entities are removed from the component, the component will also be deleted.

9.3.8. Viewing Hidden Element Components

When plotting **element components** (p. 213) during **postprocessing** (p. 131), a given component may not be visible (or may be only partially visible) because it is obscured by other elements and/or components.

To examine a specific component of interest, a workaround is available via the **/TRLCY**, **/SHRINK**, and **/GLINE** commands.

Example 9.1: Viewing a Hidden Element Component

```
! Select all elements:
ESEL,ALL
!
! Set all elements to be almost completely translucent:
/TRLCY,ELEM,0.9
!
! Set the element component of interest to be completely opaque:
/TRLCY,CM,0.0,ComponentName
!
! Shrink elements so that all elements are visible:
/SHRINK,0.01
!
! Apply a dashed outline to each element for improved visibility:
/GLINE,,1
!
! Display elements:
EPLOT
```

Chapter 10: Getting Started with Graphics

The Mechanical APDL program enables you to view almost any aspect of your model in pictures or graphs. The program has numerous features to help you to customize or enhance your graphics displays to suit your needs.

The following graphics topics are available:

- [10.1. Interactive and External Graphics](#)
- [10.2. Identifying the Graphics Device Name for Linux](#)
- [10.3. Specifying the Graphics Display Device Type \(for Windows\)](#)
- [10.4. System-Dependent Graphics Information](#)
- [10.5. Creating Graphics Displays](#)
- [10.6. Multi-Plotting Techniques](#)

10.1. Interactive and External Graphics

Any discussion of graphics might seem to imply that you are running the program interactively and viewing graphics images on your terminal screen. For the most part, this chapter is written for such a scenario. However, you can run the program in either interactively or batch mode and store graphics images on a file for later viewing and processing. This process is called creating *external graphics*. [External Graphics \(p. 269\)](#) discusses the procedures for external graphics. The following chapters pertain to obtaining graphics displays interactively on your screen.

- [General Graphics Specifications \(p. 225\)](#)
- [PowerGraphics \(p. 233\)](#)
- [Creating Geometry Displays \(p. 237\)](#)
- [Creating Geometric Results Displays \(p. 249\)](#)
- [Creating Graphs \(p. 257\)](#)
- [Annotation \(p. 261\)](#)
- [Animation \(p. 265\)](#)

10.2. Identifying the Graphics Device Name for Linux

One of the first things you must do is specify the graphics device name (sometimes referred to as the graphics *driver*). The program requires this information to properly direct graphics instructions to your display device. The default graphics device name for most systems is **X11**. For example, you can change it from X11 to **3D** if you have a 3-D graphics device.

Specify the graphics device name via the **/SHOW** command after you have entered the program but before you have activated the GUI.

10.2.1. Graphics Device Names Available

X11 (or X11C) and 3D are common graphics device names supported by the program. Each of these are described briefly below.

X11 and X11C

Graphics Device Name = X11: The X11 graphics driver incorporates X - a distributed windowing system developed at Massachusetts Institute of Technology that a variety of platforms support. It provides 2-D graphics capability. The program currently supports Version 11 (therefore, "X11") Release 6 of the X-Window system.

X separates the functionality of traditional graphics systems into two parts: the X server and the X client. The *server* is the part of the system that controls the physical display device. A *client* is a piece of application software. A single server may respond to multiple clients. The server and client may reside on different machines connected to a network. X transparently handles all communication between server and client.

Graphics Device Name = X11C: On 2-D display devices that have more than 16 colors (more than four graphics bit planes; usually eight), the program displays the model using light-source shading. Light-source shading means that when the model is viewed obliquely, the display appears to be 3-D. You can activate the extra colors using the NCPL field on the **/SHOW** command.

These devices also offer a 128-contour color option ("C-option"). This option allows contour displays to use the extra colors by adding *more* colors with a single intensity each. By default, the extra colors are used to display nine contour colors with varying intensities that simulate light-source shading. You activate the 128-contour color option by using X11C for the graphics device name on the **/SHOW** command.

Individual items can also be selected and displayed with varying degrees of translucency on 2-D devices. Translucent items will show black on the initial replot, since the 2-D driver generates only the visible face. The **/SHRINK** command forces the hardware to plot all of the faces and provide the desired translucent effect.

3D

Graphics Device Name = 3D: If you have a 3-D graphics device, you should specify 3D as the graphics device name. A 2-D device contains a "flat" 2-D projection of your model (image manipulation is performed in software), but a 3-D device contains a 3-D model in its local memory (image manipulation is performed by the display hardware). As a result, 3-D devices perform certain graphics functions more efficiently, and 2-D devices do not support certain functions. The 3-D functions include "real-time" dynamic transformation (rotation, translation, etc.) of your *actual* model, translucency, and control of various lighting options, including reflectance, intensity, light direction, and shading. If you are using a 3-D device, you can set certain display option modes using the **/DV3D** command.

10.3. Specifying the Graphics Display Device Type (for Windows)

For Windows users, the program supports these drivers and capabilities:

- A window device

- Hidden line removal
- Light source shading

If you are running the program on Windows platforms, you have three alternatives for specifying the graphics device type:

- Double-click the Interactive icon in the Program Folder. Click the down arrow next to **Graphics device name** and select the appropriate device.
- Within the program, issue the **/SHOW** command.
- Include the device type on the program execution command line. The command option **-d** or **-D** must precede the device type, as shown below:

```
ansys211 -d device_type
```

The device type is one of the following:

- WIN32
- WIN32c
- 3D

Avoid color settings of 256 colors or lower.

Specifying an invalid device type causes the program to divert the graphics to a disk file and inhibits the opening of the menu system, even if you included the **-g** option on the program execution command.

10.4. System-Dependent Graphics Information

This section describes factors affecting how graphics display on different hardware systems. *You should read this information before you activate the graphical user interface.*

10.4.1. Adjusting Input Focus

To enable the display, meshing, and listing interrupts to work correctly, you must set the input focus in the text window from which the program is executing. You can set the focus in either of two ways:

- Position the mouse pointer within the text window. (Use this method only if the window manager sets the focus automatically.)
- Place the mouse pointer on the text window and click the mouse button.

10.4.2. Displaying X11 Graphics Over Networks

You can display X11 graphics within the program over the network if the following conditions exist:

- All computer systems have X11 software installed.
- The program is linked with the X11 driver.

- A **/SHOW** device type of x11 or x11c is used. (You can use either uppercase or lowercase characters to specify device types.)
- The /etc/hosts file on the host machine contains the hostname and the IP address of the remote machine.
- The environment variable **DISPLAY** is set to *Hostname*:0.0, where *Hostname* is either the host name or the IP address of the machine that will display the graphics.

For example, suppose that you want to run the program remotely from another Linux system for local display of X11 graphics on your workstation monitor. You would perform these steps:

1. Open a window on your workstation and issue the following command to authorize remote hosts to access the display:

```
/usr/bin/X11/xhost +
```

2. Log onto a remote host (via Telnet, login, etc.). Type the following command or commands to tell the remote host to display X11 graphics on your workstation.

C Shell:

```
setenv DISPLAY Your_Workstation:0.0
```

Bourne or Korn Shell:

```
DISPLAY=Your_Workstation:0.0  
export DISPLAY
```

Your_Workstation is either the host name or the IP address of your workstation.

3. Execute the program and X11 graphics will be displayed on your workstation monitor:

```
ansys211 -d x11 -g
```

10.5. Creating Graphics Displays

You can create many types of graphics displays: geometry displays (nodes, elements, keypoints, etc.), results displays (temperature or stress contours, etc.), and graphs (stress-strain curves, time-history displays, etc.). Creating any display is a two-step process:

1. You use graphics *specification* functions to establish specifications (such as the viewing direction, number and color controls, etc.) for your display.
2. You use graphics *action* functions to actually produce the display.

You can perform both types of graphics functions either via GUI menu functions or by entering commands.

10.5.1. GUI-Driven Graphics Functions

When running the program interactively, and depending on the type of analysis being performed, you may prefer to use the GUI. The GUI functions execute commands without your seeing or editing

them. (The program records all underlying executed commands in your `Jobname.LOG` file.) You can access graphics-specification functions via **Utility Menu> PlotCtrls**. Graphics action functions reside under **Utility Menu> Plot**.

10.5.2. Command-Driven Graphics Functions

As an alternative to using the GUI functions, you can type commands directly in the Input Window. In general, you enter the graphics specifications using the graphics "slash" commands (for example, **/WINDOW**, **/PNUM**, etc.). Graphics-action commands are typically prefixed with **PL** (**PLNSOL**, **PLVAR**, etc.) or suffixed with **PLOT** (**E PLOT**, **K PLOT**, etc.).

10.5.3. Immediate Mode Graphics

By default in the GUI, your model is displayed immediately as you create new entities (such as areas, keypoints, nodes, elements, local coordinate systems, boundary conditions, etc.). This behavior is called *immediate mode* graphics. Anything drawn immediately in this way, however, is destroyed if you bring up a menu or dialog box on top of it. Or, if you iconify the GUI, the immediate mode graphics image is not displayed when you restore the GUI.

An immediate image is also automatically scaled to fit nicely within the Graphics Window--a feature called *automatic scaling*. Periodically, however, you may need to issue an explicit plot function because you have created new entities which lie "outside" the boundaries of the scaled image already in the Graphics Window and are therefore not captured with immediate mode graphics. The plot function will rescale and redraw the image.

To obtain a more "permanent" image, you need to execute one of the plot functions (such as **Utility Menu> Plot> Volumes**) or a graphics action command (such as **V PLOT**). An image generated in this way will not be destroyed by menu pop-ups or by iconifying the GUI. Also note that symbols (such as keypoint or node numbers, local coordinate systems, boundary conditions, etc.) are also shown immediately but will not be present on a permanent display unless you first activate the appropriate symbol using the functions under **Utility Menu> PlotCtrls** or the appropriate graphics specification command.

If you prefer *not* to see things immediately as you define them, issue the **IMMED** command to disable immediate mode. When you run the program interactively *without* via the GUI, immediate mode is off by default.

10.5.4. Replotting the Current Display

The **/REPLOT** command re-executes the last display action command that was executed. However, the program can execute that command only if it is valid in the current routine. For instance, if you issue a **PLNSOL** command in POST1, then exit that routine and replot while at the Begin level, no contour display will be formed. To save time, you may want to define an abbreviation for the **/REPLOT** command so that it is available on the Toolbar as a "quick pick."

10.5.5. Erasing the Current Display

You can clear the current graphics display via the **ERASE** command. (GUI menus are not erased, however.)

10.5.6. Aborting a Display in Progress

If you have initiated a display and decide to terminate it before it is completed, invoke your system "break." (Typically, this means moving the mouse pointer to the Output Window and typing Ctrl+C. However, the specific procedure varies from system to system.) **You must execute this break while the display is visibly in progress; otherwise, your entire session terminates.**

10.6. Multi-Plotting Techniques

The multi-plotting capabilities within the program enable you to display both multiple entities within a window and multiple windows with varying entity types. Defining each window's composition is a four-step process:

1. Define the window layout.
2. Select the entities that you want each window to display.
3. If displaying elements or graphs, select the type of element or graph display used for plots.
4. Display the selected entities.

10.6.1. Defining the Window Layout

You need to define how many windows you want the program to use for plotting and how those windows appear on your screen. The following layout options are available:

- One window
- Two windows (left and right of the screen, or top and bottom)
- Three windows (two at the top of the screen and one at the bottom, or one window at the top and two windows at the bottom)
- Four windows (two at the top of the screen and two at the bottom)

To define the window layout, issue the **/WINDOW** command. If you use the GUI path, the program displays a dialog box, in which you click the layout you prefer. That dialog box also contains a **Display upon OK/Apply** field, where you also can specify what the program displays next. Choices for this field are **Multi-Plots**, **Replot**, and **No redisplay**. After specifying your layout design, click **Apply** or **OK**.

10.6.2. Controlling the Entities That Each Window Displays

After designing your window layout, choose the entities that each window will display (**/GTYPE,WN,Label,KEY**).

You also can choose the type of plots (**/GCMD**). All entity types *except GRPH* are active by default; to disable an entity type, click on it.

When issuing the **/GTYPE** command, either specify ALL for the *WN* argument to have all windows display the selected entities, or select a specific window number (default is 1). For *Label*, specify any of these entity types:

- NODE (nodes)
- ELEM (elements)
- KEYP (keypoints)
- LINE (lines)
- AREA (areas)
- VOLU (volumes)
- GRPH (graph displays)

When the GRPH entity type is activated, you can display only x-y graphs, and you cannot use the **/GCMD** command to issue other commands (such as **/TYPE**) that affect displays. (For more information, see [Controlling the Plot Display \(p. 223\)](#).) If the GRPH type is off, you can display any combination of the other solid model or finite element entity types, and you can use **/GCMD** to issue other display control commands.

To turn an entity type on via the **/GTYPE** command, use a *KEY* value of 1. To turn an entity type off, specify a *KEY* of 0.

10.6.3. Controlling the Plot Display

When you are displaying either the ELEM or GRPH entity type, you can control the type of element or graph display used for plots (**/GCMD**,*WN*,*Lab1*,...*Lab12*).

You can specify ALL to have all windows use the selected display type, or you can apply that display type only to a specific window (default is window 1). The *Lab1* through *Lab12* values on the **/GCMD** are labels for commands such as **/TYPE** and **PLNSOL**,*S,X*. (For the *Lab* arguments, you can specify only commands that have *WN* (window) arguments.)

Example 10.1: Selecting a Type of Element or Graph Display

Following are two of selecting a type of element or graph display:

- To display a **PLNSOL**,*S,X* command in window 1 when the ELEM entity type is activated, issue the command **/GCMD**,1,**PLNS**,*S,X*.
- To change from an element display to a von Mises display, issue the command **/GCMD**,1,**PLNS**,*S,EQV*.

10.6.4. Displaying Selected Entities

To display your selected entities, issue the **GPOINT** command.

Chapter 11: General Graphics Specifications

Most graphics features apply to any kind of graphics display, whether for geometry, results, or graphs. These general graphics specifications affect such features as multiple windows, viewing directions, zooming and panning your image, etc.

You can create and control your displays via **Utility Menu> Plot** and **Utility Menu> PlotCtrls** or by issuing graphics commands.

The following related topics are available:

- [11.1. Multiple Windows and Superimposed Displays](#)
- [11.2. Changing the Viewing Angle, Zooming, and Panning](#)
- [11.3. Controlling Miscellaneous Text and Symbols](#)
- [11.4. Miscellaneous Graphics Specifications](#)
- [11.5. 3-D Input Device Support](#)

11.1. Multiple Windows and Superimposed Displays

A *window* is a rectangular portion of your display located inside the main Graphics Window. Windows are defined in screen coordinates (X_s, Y_s). You can define up to five different windows, which can be placed anywhere within the Graphics Window, and which can overlap. Each window can have different graphics *specification* settings; however, graphics *action* commands apply to every active window.

The **/WINDOW** command controls basic window operations:

```
/WINDOW, WN, XMIN, XMAX, YMIN, YMAX, NCOPY
```

Table 11.1: Basic Window Operations

Window Operation	/WINDOW Argument(s)	Description
Define size and placement	<i>XMIN, XMAX, YMIN, YMAX</i>	Size and place windows in the top half, bottom half, right top quadrant, etc. of the Graphics Window
Activate / deactivate	<i>XMIN = ON / OFF</i> , respectively	Activates and deactivates a defined window
Delete	<i>XMIN = DELE</i>	Deletes a defined window
Copy display specifications between windows	<i>NCOPY</i>	Copies a set of display specifications (/VIEW , /DIST , etc.) from window <i>NCOPY</i> (1 to 5) to the current window

11.1.1. Superimposing (Overlaying) Multiple Displays

To display dissimilar items in separate windows, a sequence of windows action commands is necessary as you activate and deactivate appropriate windows, while protecting the displays in your deactivated windows from being erased. The key to this operation is the **/NOERASE** command, which prevents the normal screen erase from occurring as new displays are created. After your multiple display has been created, you can return to normal erasing mode via the **/ERASE** command.

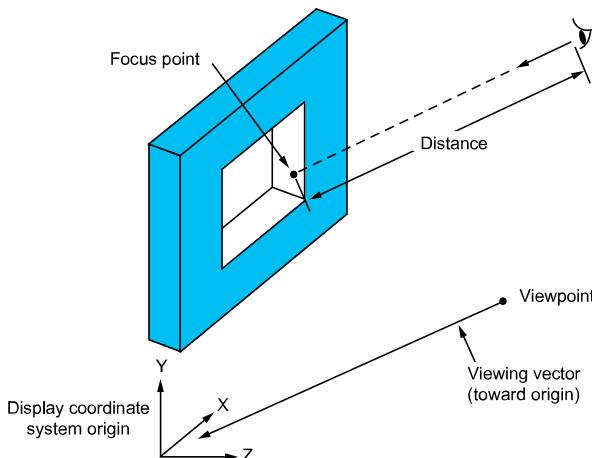
11.1.2. Removing Frame Borders

The FRAME label on the **/PLOPTS** command displays or hides window border lines.

11.2. Changing the Viewing Angle, Zooming, and Panning

Using these display specifications is similar to using a camera. The following sketch illustrates the concepts of *focus point*, *viewpoint*, and *viewing distance*, discussed below.

Figure 11.1: Focus Point, Viewpoint, and Viewing Distance



11.2.1. Changing the Viewing Direction

The viewing direction is established by a vector directed from the *viewpoint* to the display coordinate system origin. Issue the **/VIEW** command to define the position of the viewpoint in the display coordinate system.

Use this shortcut to pan, zoom, and rotate a graphics display:

1. Press the CONTROL key and hold it down. You are now in Dynamic Manipulation Mode. Notice that the cursor assumes a different shape.
2. While holding the CONTROL key down, use the mouse buttons to manipulate your view of the display.
3. Release the CONTROL key to exit Dynamic Manipulation Mode.

You can also remap your mouse buttons to match the operation (in dynamic mode only) of other programs (**/UIS,BORD,LEFT,MIDDLE,RIGHT**).

11.2.2. Rotating the Display About a Specified Axis

To rotate the graphics display about the screen axes or about the global Cartesian axes, issue the **/ANGLE**, **/XFRM** commands. (The right-hand rule defines positive angular rotation about any axis.)

11.2.3. Determining the Model Coordinate System Reference Orientation

The **/VUP** command determines the "starting" orientation of your display. For example, with the viewpoint and rotation at their default settings, **/VUP,WN,X** orients the display such that the positive X axis is vertical pointing upward, Y is horizontal pointing to the left of the screen, and Z points out of the screen.

11.2.4. Translating (or Panning) the Display

The *focus point* is that point on your model that appears at the center of your windows. You can define or redefine the focus point (in terms of the global Cartesian coordinate system) via the **/FOCUS** command.

The command also enables you to translate the focus point along the screen axes or along the global Cartesian axes.

11.2.5. Magnifying (Zooming in on) the Image

The *viewing distance* represents the distance between the observer and the focus point, and determines the magnification of your image. Smaller viewing distances magnify the image (zoom in), and larger distances shrink the image (zoom out). Issue the **/DIST** command to change the viewing distance.

11.2.6. Resetting Automatic Scaling and Focus

Anytime that you change the viewing distance or focus point, your explicitly-defined settings become fixed; that is, automatic scaling or centering of the image are deactivated for subsequent displays. (Fixed parameters are preceded with an asterisk in the legend column of the display.)

To restore automatic scaling and focus, issue the **/AUTO** command.

11.2.7. Freezing Scale (Distance) and Focus

By default, your display is automatically scaled and centered such that the image of your model just fills your windows. To fix these automatically-generated scale and focus settings, issue the **/USER** command.

11.3. Controlling Miscellaneous Text and Symbols

You can control how different symbols and text entries in your graphics window appear:

11.3.1. Using Legends in Your Displays

11.3.2. Controlling Entity Fonts

11.3.3. Controlling the Location of the Global XYZ Triad

11.3.4. Displaying and Hiding Triad Symbols

11.3.5. Changing the Style of the Working Plane Grid

11.3.6. Displaying and Hiding the Program Logo

11.3.1. Using Legends in Your Displays

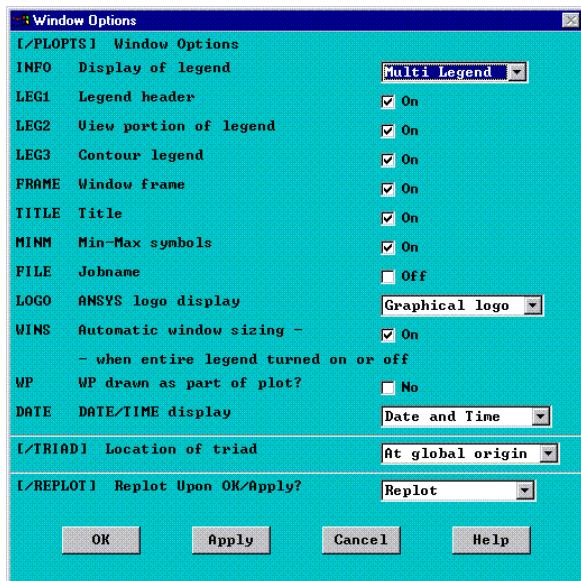
You can use legends to help define and clarify the data in your display. The Window Options Dialog Box is the "master" legend control. It controls whether or not the legend is displayed, the type of legend display, and in some cases, the content of your legend. The position of the Triad is also controlled from this dialog box. See [Figure 11.2: The Window Options Dialog Box \(p. 228\)](#) below.

The **INFO** pull down window provides control for the type of legend. It allows you to turn legend displays on and off, and also to access either the Auto Legend or the Multi-Legend display. The on and off settings control the display of all legend items, for all types of legends.

The Legend On and Auto legend selections control the display of the documentation column. The documentation column display places all of your legend data along the right side of the graphics window and resizes your model area appropriately. Legend On displays the documentation data at all times, while Auto Legend displays the appropriate data, only when it is applicable.

The Multi Legend provides placement options for your text and contour scales within the model area of your graphics window. The Multi Legend options are discussed below.

Figure 11.2: The Window Options Dialog Box



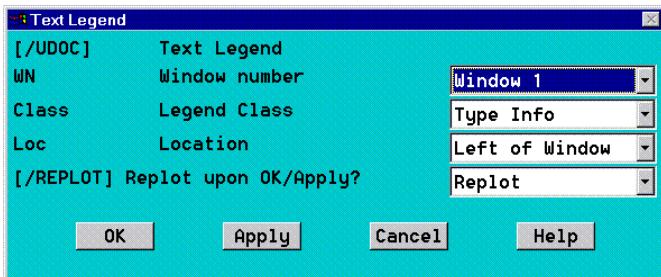
The default, legend setting is the user-defined "Multi-Legend." (**Utility Menu> Plot Ctrls> Window Controls> Window Options> MultiLegend - /PLOPTS,INFO,3**).

11.3.1.1. Controlling the Content of Your Legends

The window options dialog box shown above in [Figure 11.2: The Window Options Dialog Box \(p. 228\)](#) controls the type of legend, along with the content of the documentation column. You control the content of the Multi Legend via dialog boxes found at **Utility Menu> Plot Ctrls> Style> Multi Legend Options**. The Text Legend dialog box shown in [Figure 11.3: The Multi Legend Text Legend \(p. 229\)](#) provides control of the content and placement of the various text items available for

the Multi Legend option. This dialog box corresponds to the controls and priorities listed in the **/UDOC** command.

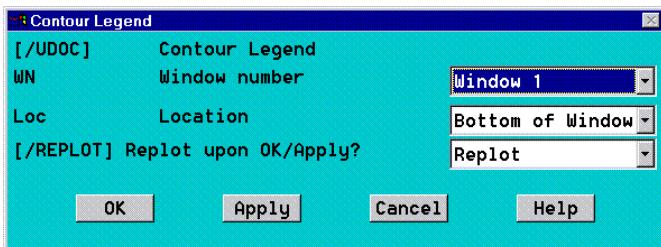
Figure 11.3: The Multi Legend Text Legend



11.3.1.2. Controlling the Placement of Your Contour Legend

The Multi Legend setting allows you to place your contour scales along the four sides of the graphics window. You access this control via **Utility Menu> Plot Ctrls> Style> MultiLegend Options> Contour Legend**. The Contour Legend Dialog box is shown in [Figure 11.4: The Multi Legend Contour Legend \(p. 229\)](#). This dialog box corresponds to the controls and priorities listed in the **/UDOC** command.

Figure 11.4: The Multi Legend Contour Legend



Note:

The settings in the Window Options dialog box will in many cases take precedence over your Multi Legend Options settings. See the command documentation for **/UDOC** and **/PLOPTS** for a complete discussion of these dependencies.

11.3.2. Controlling Entity Fonts

You can change the appearance of the fonts that are used to produce the numbers and characters shown on your displays. In the GUI, select **Utility Menu> PlotCtrls> Font Controls**, or issue the **/DEVICE,FONT,KEY** command. This command requires the *Val1* through *Val6* arguments to specify font information.

11.3.3. Controlling the Location of the Global XYZ Triad

The **/TRIAD** command (**Utility Menu> PlotCtrls> Window Controls> Window Options**) enables you to change the location of the global triad symbol on your display. The actual mathematical position of the global origin does not change.

11.3.4. Displaying and Hiding Triad Symbols

Issue the **/TRIAD** command to display or hide the global triad.

Issue the **/PSYMB** command to control the local, nodal, and element coordinate system triads.

Issue the **WPSTYL** command to control the working plane triad.

11.3.5. Changing the Style of the Working Plane Grid

You can display the working plane grid as a triad only, grid only, or both triad and grid. Issue the **WPSTYL** command to change from one style to another. There are two methods of turning the working plane on for displays:

- **WPSTYL** with no arguments toggles the working plane grid, asterisk, and triad on and off immediately, as an overlay image on the existing display.
- **/PLOPTS,WP,ON** specifies that the working plane be activated for *subsequent* displays.

In this case, the working plane is drawn as *part of* the display (not just an overlaid image). This method is therefore best used in combination with a hidden-line technique for viewing the location of the working plane with respect to a 3-D model.

11.3.6. Displaying and Hiding the Program Logo

Issuing **/PLOPTS,VERS,1** causes the program logo to appear in the upper right corner of the screen (along with the version number).

11.4. Miscellaneous Graphics Specifications

A number of miscellaneous graphics commands enable you to manipulate your graphics environment even further:

- [11.4.1. Reviewing Graphics Control Specifications](#)
- [11.4.2. Restoring Defaults for Graphics Slash Commands](#)
- [11.4.3. Saving the Display Specifications in a File](#)
- [11.4.4. Recalling Display Specifications from a File](#)
- [11.4.5. Pausing the Program](#)

11.4.1. Reviewing Graphics Control Specifications

Issuing the **/PSTATUS** command with no arguments lists the current graphics control specifications. To see the graphics specifications for one window only, specify a specific window number (**WN**).

11.4.2. Restoring Defaults for Graphics Slash Commands

Issue the **/RESET** command to restore the default settings of **/WINDOW**, **/TYPE**, **/VIEW**, and other graphics "/" commands.

11.4.3. Saving the Display Specifications in a File

Issue the **/GSAVE** command to store a copy of your graphics "/" command settings in an ASCII text file (default Jobname . GSAV).

11.4.4. Recalling Display Specifications from a File

Read graphics "/" commands from an ASCII text file via the **/GRESUME** command.

You can also issue an **/INPUT,Filename** command, where *Filename* is the name of the file containing the saved graphics specifications.

11.4.5. Pausing the Program

If you prepare an input file for demonstration or presentation purposes, it may be useful to pause the program after creating a display to allow the display to be viewed for a reasonable length of time. You can do so by adding **/WAIT** commands to your input stream after the display action commands.

11.5. 3-D Input Device Support

The program has been tested with 3Dconnexion SpaceBall and SpacePilot Pro 3-D mouse devices. These devices detect slight fingertip pressures and resolve them into X, Y, and Z translations, rotation components, and movements of your 3-D images. They are intended to provide smooth, dynamic, interactive, simultaneous six-degree-of-freedom control of 3-D graphical images or objects. These devices are designed to be used in conjunction with the mouse, not in place of it.

The requisite developer's kit software had been included in the applicable code, and drivers for the system you are installing to are available at <http://www.3dconnexion.com/downlink.asp>.

Certified devices and drivers can be found here: [Platform Support](#).

Chapter 12: PowerGraphics

Two methods are available for displaying graphics:

- The Full Model display method. Invoke this method via the **/GRAPHICS,FULL** command.
- The PowerGraphics display method. Invoke this method via the **/GRAPHICS,POWER** command.

The PowerGraphics method is the default when the GUI is active and is valid for all element types *except* circuit elements. The Full Model method is valid for all element types.

The display method you choose depends upon the size of your model and the type of elements used in the model. If your model contains circuit elements, for example, select the Full Model method.

The following PowerGraphics topics are available:

- [12.1. Characteristics of PowerGraphics](#)
- [12.2. When to Use PowerGraphics](#)
- [12.3. Activating and Deactivating PowerGraphics](#)
- [12.4. Using PowerGraphics](#)
- [12.5. What to Expect from a PowerGraphics Plot](#)

12.1. Characteristics of PowerGraphics

- Displays for large models are plotted at a much greater speed than with the Full Model method.
- PowerGraphics plots quadratic (curved) surfaces for midside node elements.
- This method can display discontinuous results due to material type and real constant discontinuities.
- Shell element results are displayed at both top and bottom layers, simultaneously.
- You can use the *Query* picking option to query subgrid results for some elements in the Graphical User Interface.
- PowerGraphics is not available for circuit elements.
- When requested results data are not supported by PowerGraphics, the results are output using the Full Model method.
- Results averaging occurs using only the data at the model surface.
- Minimum and maximum values are valid only for data at the model surface.

12.2. When to Use PowerGraphics

Using the PowerGraphics display method has distinct advantages, since graphics displays are plotted at a much faster rate of speed than with the Full Model method. In addition, PowerGraphics produces more realistic results at material type and real constant discontinuities in the model. See the description of the **/GRAPHICS** command for more information.

12.3. Activating and Deactivating PowerGraphics

Activate PowerGraphics by issuing the **/GRAPHICS,POWER** command.

Deactivate PowerGraphics by issuing **/GRAPHICS,FULL**.

12.4. Using PowerGraphics

When the PowerGraphics method for graphics displays is active, it is used for element, area, volume, line, and result displays and result data listings.

PowerGraphics does not support the graphics display or listing for circuit elements; for such cases, the program activates the Full Model graphics method automatically and uses it for that display or listing.

See the **/GRAPHICS** command description for more information.

12.5. What to Expect from a PowerGraphics Plot

Because PowerGraphics plots or listings are given for the exterior surface of the model, expect to see differences in the results compared to those given when using the Full Model method.

The averaging calculations for PowerGraphics include results for only the model surface. The averaging calculations, plots, and listings for the Full Model method include results for the entire model (interior and exterior surfaces). The PowerGraphics and Full Model methods therefore display results values differently for nodal results.

The **E PLOT**, **A PLOT**, **V PLOT**, **L PLOT**, **PLDISP**, **PLNSOL**, and **PRNSOL** commands behave differently for PowerGraphics than for Full Model.

12.5.1. Viewing Your Element Model

The subgrid approach used by PowerGraphics enables you to control the amount of displayed element curvature. You can plot varying degrees of curvature in your model by specifying the number of facets to be used for element display. Facets are piecewise linear approximations of the actual curve represented by the element face or edge. Specify the number of facets per element edge via the **/EFACET** command.

The more facets you specify, the smoother the representation of the element surface for PowerGraphics plots.

The subgrid approach affects both the display of geometric curvature and the display and printout of results quantities (displacements, stresses, etc.). When using PowerGraphics in POST1 for derived

quantities on solid elements, however, the maximum value on the plot and the maximum value in the printout may not agree. PowerGraphics displays do not average at geometric discontinuities. The printouts in PowerGraphics will, however, provide averaging information at geometric discontinuities if the models do not contain shell elements. Carefully inspect the data you obtain at geometric discontinuities.

12.5.2. Printing and Plotting Node and Element Results

List displacements, stresses, and strains at all node locations (both corner and midside nodes), via the **PRNSOL** command.

For shell elements, you can list results and plot them at the top/bottom and middle layer locations. Likewise, these nodal values can be contoured for display purposes. The number of facets per element edge that you specify determines contour resolutions.

Results values for shell elements are displayed simultaneously for the top and bottom layers.

When viewing nodal results via PowerGraphics (**PRNSOL**, or **PLNSOL**), you can average results in various ways. To specify how results are averaged, issue the **AVRES** command. (The command has no effect on the degree-of-freedom (DOF) solution values (UX, UY, TEMP, etc.). You can average results at all boundaries (default), or at all boundaries except where real constant and/or material discontinuities exist. *Results are not averaged at geometric discontinuities.*

In Full Graphics mode, it is possible to deselect an individual node, select all elements (including the element that contains that node), and then perform postprocessing calculations on those elements and have that unselected node not be considered in those calculations. However, if PowerGraphics is active postprocessing always displays based on selected elements.

The minimum and maximum results values reported for your PowerGraphics plot are based on the surface data. For stresses and strains, the values are usually acceptable. Some thermal results, however, may have internal minimum or maximum values, and erroneous values may be reported. In such cases, switch to Full Model graphics.

Plotting and printing element results is similar to the Full Model graphics method. Issue the **PLESOL** or **PRESOL** command.

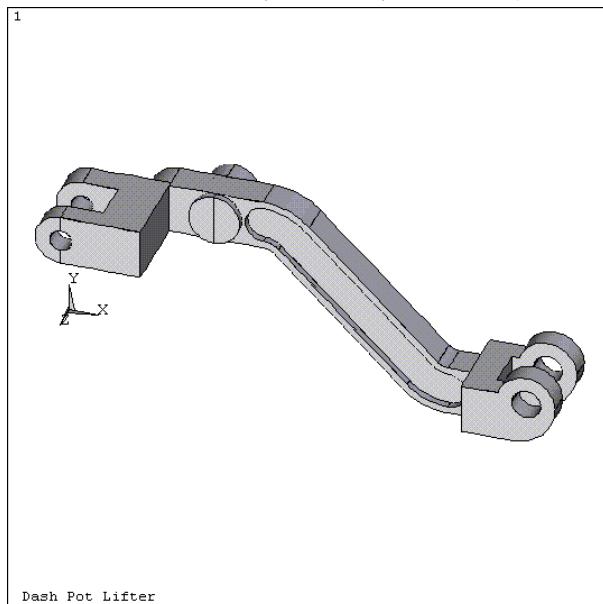
The program unaverages nodal results and sorts them by element number. Averaging results does not affect element results plots. Results are for all nodal locations on the model surface. If you issued the **/EFACET,1** command, the results for the midside nodes are not listed.

PowerGraphics does not support safety factor calculations.

In unusual cases, your model may contain element types having different results data sets. If so, unselect those element types which do not have the data set you are reviewing. Doing so prevents zero values from being averaged with valid results. For example, if your model contains **FLUID30** (Acoustic Fluid) and **SOLID185** (Structural Solid) elements, unselect all **SOLID185** elements before viewing a pressure gradient.

Chapter 13: Creating Geometry Displays

A *geometry display* is a display of your model's geometric features (keypoints, areas, nodes, elements, loads, etc.). This is the kind of display that you might produce during the model-generation and load-definition phases of your analysis. This figure shows a typical geometry display:



Many find that the most convenient way to create and control geometry displays is by using the functions available under **Utility Menu> Plot** and **Utility Menu> PlotCtrls**. You can also issue use graphics action and control commands.

The following geometry display topics are available:

- 13.1. Creating Displays of Solid-Model Entities
- 13.2. Controlling the Geometry Display Specifications

13.1. Creating Displays of Solid-Model Entities

The following commands create displays of solid-model entities:

Table 13.1: Commands for Displaying Solid-Model Entities

Command	Purpose
A PLOT	Displays a plot of areas
E PLOT	Displays a plot of elements
K PLOT	Displays a plot of keypoints
LAYPLOT	Displays the layer stacking sequence and layer angle

Command	Purpose
	orientation of layered element types
LPLOT	Displays a plot of lines
NPLOT	Displays a plot of nodes
/REPLOT	Re-executes the last display action executed
VPLOT	Displays a plot of degenerated volumes

The controls you establish before you invoke these actions can also cause your displays to contain other information, such as lower-order entity numbers (for instance, node numbers associated with selected elements), loads, etc.

13.2. Controlling the Geometry Display Specifications

In addition to the features listed below, also see [Getting Started with Graphics \(p. 217\)](#) for general graphics specifications that apply to any type of display, including geometry displays.

- [13.2.1. Controlling the Style of Your Display](#)
- [13.2.2. Applying Styles to Enhance the Model Appearance](#)
- [13.2.3. Controlling Numbers and Colors](#)
- [13.2.4. Displaying Loads and Other Special Symbols](#)

13.2.1. Controlling the Style of Your Display

Following are several ways to change how the program displays your models:

- [13.2.1.1. Displaying Line and Shell Elements as Solids](#)
- [13.2.1.2. Displaying Only the Edges of an Object](#)
- [13.2.1.3. Displaying the Interior Element Edges of an Object](#)
- [13.2.1.4. Using Dashed Element Outlines](#)
- [13.2.1.5. Shrinking Entities for Clarity](#)
- [13.2.1.6. Changing the Display Aspect Ratio](#)
- [13.2.1.7. Changing the Number of Facets](#)
- [13.2.1.8. Changing Facets for PowerGraphics Displays](#)
- [13.2.1.9. Changing Hidden-Line Options](#)
- [13.2.1.10. Section, Slice, or Capped Displays](#)
- [13.2.1.11. Specifying the Cutting Plane](#)
- [13.2.1.12. Vector Vs. Raster Mode](#)
- [13.2.1.13. Perspective Displays](#)

13.2.1.1. Displaying Line and Shell Elements as Solids

If your model consists of line elements (such as beams and pipes) or shell elements, you can many of them as solids via the **/ESHAPE** command.

The program uses a rectangular cross section for beams and shells, and uses circular cross sections for pipes. The element real constants are used to proportion the cross section.

13.2.1.2. Displaying Only the Edges of an Object

While working with displays, you may want to see only the edges of an object; that is, you may want to remove element outlines from the interior of the object. To see only the edges of non-contour displays (**E PLOT**), issue **/EDGE**, ,1. For contour displays (**PLESOL**, **PLETAB**, **PLNSOL**, **PLTRAC**), edges are displayed by default (**/EDGE**, ,0).

13.2.1.3. Displaying the Interior Element Edges of an Object

While working with displays, you might prefer to see the interior element edges, or detail, of an object. If you are working with non-contour displays (**E PLOT**), the interior element edges are displayed by default (**/EDGE**, ,0). To see the interior element edges of contour displays (**PLESOL**, **PLETAB**, **PLNSOL**, **PLTRAC**), issue **/EDGE**, ,1.

An edge, as used in the above context, is the common line between adjacent faces that are not coplanar. The *ANGLE* field on the **/EDGE** command enables you to specify the "degree of coplanarity" at which an edge should be displayed. That is, if *ANGLE* = 45° (which is the default value), an edge is displayed only if the two adjacent faces deviate from coplanarity by more than 45°. If *ANGLE* = 0°, even the slightest deviation from coplanarity causes the edge to be displayed. The default value of 45° is particularly helpful in displaying a cylindrical shell model as a smooth cylinder rather than as a "faceted" cylinder.

13.2.1.4. Using Dashed Element Outlines

Switch the style of element outlines from solid line to dashed line via the **/GLINE** command. The command enables you to remove element outlines entirely.

13.2.1.5. Shrinking Entities for Clarity

The **/SHRINK** command shrinks displayed elements, lines, areas, and volumes by a specified percentage so that adjacent entities are separated for clarity.

The program ignores a request to shrink the display when the edge option is active.

13.2.1.6. Changing the Display Aspect Ratio

You can artificially distort your display's geometry in a particular direction via the **/RATIO** command. This capability is useful for displaying details within a long, thin object more clearly.

13.2.1.7. Changing the Number of Facets

Area and volume raster displays are made up of numerous small facets (or polygons). Occasionally, you might want to obtain a more precise representation of your areas or volumes by increasing the number of facets used to create these displays.

Issue the **/FACET** command to switch between two different facet densities.

13.2.1.8. Changing Facets for PowerGraphics Displays

When PowerGraphics is enabled, you can display varying degrees of curvature in your model by specifying the number of facets per element edge to be used for element display.

Facets are piecewise linear approximations of the actual curve represented by the element face or edge. The greater the number of facets, the smoother the representation of the element surface for element plots.

Issue the **/EFACET** command to specify the number of facets per edge.

13.2.1.9. Changing Hidden-Line Options

By default, raster displays are created as Z-buffered displays. (See the description of the **/TYPE** command for other hidden-line options.)

All non-Z-buffered hidden-line options produce the same results in vector displays. For area and volume Z-buffered displays, you can further specify the type of surface shading (the smoothness of the object) via the **/SHADE** command. Also, you can issue the **/GFILE** command to set the resolution of Z-buffered displays that are written to graphics files.

13.2.1.10. Section, Slice, or Capped Displays

To view the interior of a 3-D solid element model, you can use *section* displays, *slice* displays, or *capped* displays:

- A section display produces an image of a 2-D planar section that is defined by the intersection between your model and the cutting plane.
- A slice display is similar to a section display except the edge lines of the remaining 3-D model are also shown.
- A capped display produces an image of a 3-D portion of your model with a portion of the model display "cut off" by the cutting plane.

Section, slice, and capped displays are special versions of hidden-line displays controlled via the **/TYPE** command.

13.2.1.11. Specifying the Cutting Plane

Three types of graphics displays (section, slice, and capped) require a cutting plane. Specify the cutting plane via the **/CPLANE** command and define the plane as either:

- Normal to the viewing direction and passing through the focus point (default)

- The working plane

13.2.1.12. Vector Vs. Raster Mode

The **/DEVICE** command (or **/SHOW** command) enables you to toggle between vector and raster mode.

By default, *raster* mode is active; that is, polygons are filled with color when they are displayed. Rastmer mode applies to area, volume and element displays, and the geometry in postprocessing displays.

Vector mode produces wireframe displays, which show only the outlines of entities, and which usually take less time to form than do raster displays. To display wireframe outlines for solid model entities only (areas and volumes) when your graphics session is otherwise in raster mode, specify the **WIRE** option on the **/FACET** command.

13.2.1.13. Perspective Displays

By default, the program creates a non-perspective display of your model. To cause a perspective display to be formed, issue the **/VCONE** command to define a view cone angle. (The larger the view cone angle, the more pronounced the perspective effect will be.)

13.2.2. Applying Styles to Enhance the Model Appearance

You may want to highlight portions of your model in order to provide a clearer representation of its structure or to highlight certain areas. The following techniques are available to enhance and clarify your model.

13.2.2.1. Applying Textures to Selected Items

You can use textures to add realistic effects and differentiate between various items in your model. Textures are controlled via the **/TXTRE** command.

You must be using a 3-D, Open GL display device, with the appropriate graphics driver loaded. You can apply textures to numbered entities by specifying them on the command line, or you can use graphical picking to select the desired items in your graphics window.

Textures can affect the speed of many of your display operations. You can increase the speed by temporarily disabling textures. When textures are off, all texture information is retained so that it can be reapplied when texturing is reenabled.

The **/TXTRE** command can also apply bitmaps on 2-D devices. Other applications of the command require 3-D capability.

Some 3-D effects do not display properly unless the triangle strip display method is disabled. Tri-stripping provides faster resolution of 3-D displays, and is enabled by default. You can control tri-stripping via the **TRIS** option on the **/DV3D** command. Reapply the **TRIS** option after you obtain a satisfactory output.

13.2.2.2. Creating Translucent Displays

On some 2-D and 3-D devices, you can create see-through, translucent images via the **/TRLCY** command. You can specify the entities to be made translucent either by picking, or by entering the appropriate entity numbers in a fill-in box. The level of translucency can range from opaque to fully transparent.

On 2-D devices, the program displays only the visible faces of the selected items. Using a small value for the **/SHRINK** command forces the hardware to plot the hidden faces and produce the desired effect.

13.2.2.3. Changing Light-Source Shading

Light-source shading enhances raster displays on 2-D and 3-D devices having at least eight color planes ($2^8 = 256$ colors). To specify the number of color planes necessary for light-source shading, issue the **/SHOW** command.

On some 3-D devices, you can issue the **/LIGHT** command to adjust the intensity of ambient and directional light, change the light direction, and modify the directional light reflectance factor. You can also change the light direction for 2-D devices when the Z-buffering hidden-line option is used.

13.2.2.4. Adding Background Shading and Textures

Background treatments help to contrast and highlight your model display, while providing a more pleasing output. Background options are available at **Utility Menu> PlotCtrls> Style> Background**. Four options toggle the background on and off and control whether a color, a texture, or a user-specified file is used for the background. The available colors are defined via the **/COLOR** command, and the progression of the gradients (up and down or left and right) can also be specified.

The available textures are defined via the **/TXTRE** command. Depending on the pixel size of your user-specified file, it can be tiled, fill the entire background, or display only a portion. The texture and file backgrounds place a greater load on graphics speed than the color gradients.

External bitmap files can also be used for your background. You can import a sample from another source to create any desired background. The *.bmp, *.png and *.jpg formats are supported for the PC. Linux systems support *.png and *.jpg, along with native XWD format. Your imported bitmaps occupy the number of pixels they were generated at, and cannot be resized in the graphics window. Depending on the size (pixels) of the file, the background will be tiled to produce full coverage. Use an external graphics program to obtain the proper size before you import bitmap files.

The default background for the Graphics window is blue shading with a gradient progression from top to bottom (**/COLOR,PBAK,ON,1,BLUE**). You can modify the background using the methods above, or enable/disable it via the **/UIS,(ON or OFF)** option.

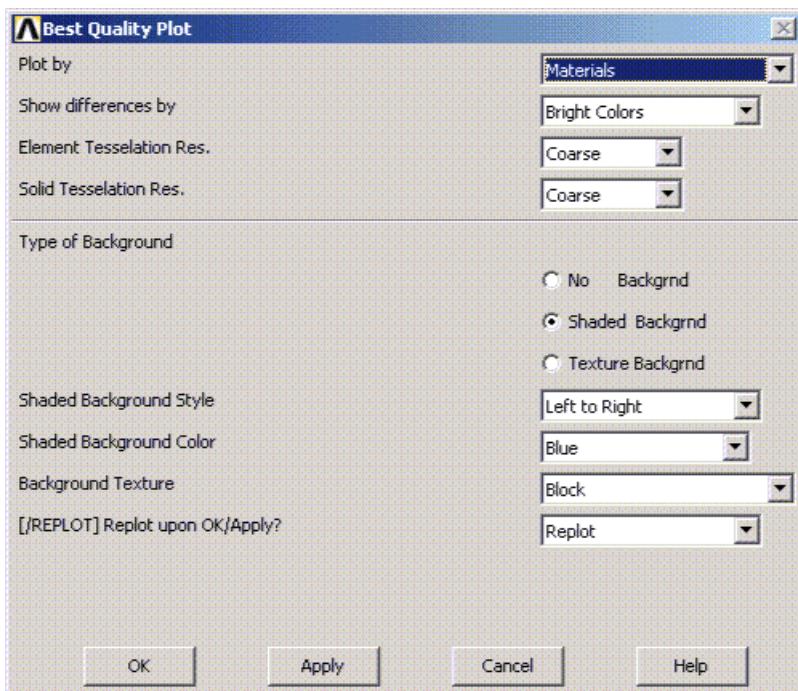
13.2.2.5. Using the Create Best Quality Image Capability

After you have applied various style attributes to your model, you will want to coordinate their presentation to yield an image that not only conveys the information properly, but also provides a realistic representation. Often, this is a trial-and error procedure, where you apply the various attributes, and then replot to see how the model looks. You can use the "Create Best Quality Image"

feature to optimize the use of these effects. Access this feature via **Utility Menu> PlotCtrls> Best Quality Image> Create Best Quality**.

The "Create Best Quality Image" function is a complex macro that takes into consideration the three dimensional nature of your model, the way the various parts of your model are defined, and how the model's attributes should be displayed. It goes through the model's style attributes, and provides the optimal settings. This dialog box, along with a description of each of the controls, is shown in the following description:

Figure 13.1: Create Best Quality Image Function Box



Function Activates the Best Quality Macro. Running this macro optimizes your color palette, light source, translucency and background shading, providing a simple method to arrive at an optimized model representation.

Plot by Selects the model attribute upon which to base the graphical optimization. The choices are:

- **Materials:** This choice will optimize the model representation according to the defined materials in your model.
- **Type:** This choice will optimize the model representation according to the various element types you have defined in your model.
- **Real:** This choice will optimize the model representation according to the various real constants you have defined for your model.
- **Items (Entities):** This choice will use the defined areas, volumes and elements to provide the basis for optimization.

Show differences by	The model colors will be applied according to the Show differences by selections. A pull-down menu shows the choices for two different color schemes. You can also show the differences according to the textures and translucency styles you have already applied to the model. To modify the color scheme go to Utility Menu> PlotCtrls> Best Quality Image> Modify Colors.
Element Tessellation Resolution	Click this pull-down menu to display three choices for the resolution of your element displays. The "Normal" resolution choice provides an optimized mix of speed and quality, while the "Coarse" or "Fine" selections provide maximum speed or quality, respectively.
Solid Tessellation Resolution	Click this pull-down menu to display three choices for the resolution of your solid displays. The "Normal" resolution choice provides an optimized mix of speed and quality, while the "Coarse" or "Fine" selections provide maximum speed or quality, respectively.
Type of Background	The background choices are either blank, shaded, or textured. The shaded or textured backgrounds will correspond to the colors found in the /COLOR command or the textures found in the /TXTRE command. You specify these items in the pull-down menus below the background selections.
Shaded Background Style	This pull-down menu provides four choices for the shading progression of background color.
Shaded Background Color	This pull-down menu enables you to specify the background color according to the color choices listed in the /COLOR command.
Background Texture	This pull-down menu enables you to specify the background texture according to the texture choices listed in the /TXTRE command.
Replot Upon OK/Apply	Controls the application of replot after applying changes.

The Best Quality Image macro modifies the color map, which can affect the color display on subsequent plots. After you have captured or plotted the image, reset the color map by activating either the "Reset to Previous" or "Reset to Global" functions found in the initial Best Quality image menu (**Utility Menu> PlotCtrls> Best Quality Image**). You must issue a replot for those functions.

13.2.3. Controlling Numbers and Colors

Item numbers and colors are usually related. By default, entities are not numbered. Numbering (and associated coloring on appropriate devices) can be enabled and disabled via the following procedures.

13.2.3.1. Turning Item Numbers On and Off

13.2.3.2. Specifying a Format for the Graphical Display of Numbers

13.2.3.3. Controlling Number and Color Options

13.2.3.4. Controlling Color Values

13.2.3.1. Turning Item Numbers On and Off

Issue the **/PNUM** command to toggle numbering on and off for these items:

- Nodes
- Elements
- Element coordinate systems
- Material types
- Real types
- Element types
- Element locations (for reordered elements)
- Contour values (integer only; on element displays)
- Solid-modeling entities (keypoints, lines, areas, and volumes).

Numbers are not shown in face hidden-line or precise hidden-line displays.

13.2.3.2. Specifying a Format for the Graphical Display of Numbers

You can select the format in which you want floating point numbers to be displayed via the **/GFORMAT** command.

The command enables you to indicate the width of the fields in which numbers are displayed and the number of digits that are displayed for a FORTRAN format type. Other commands for tailoring the display appearance include **/PNUM**, **/PBC**, **/PBF**, and **/PSF**.

13.2.3.3. Controlling Number and Color Options

After you have enabled numbering for an item, you can issue the **/NUMBER** command to select one of four possible "on-off" combinations of numbering and coloring: show colors and numbers (default), show colors but not numbers, show numbers but not colors, or suppress both colors and numbers.

13.2.3.4. Controlling Color Values

The **/COLOR** command controls the correspondence between specific items or numbers and their associated colors.

You can also change the overall color map (edit an existing, or store a new, color map on a file) via the **/CMAP** command, which activates the CMAP utility.

The CMAP utility enables you to change the assignment options for the colors you use, and to save different assignment protocols in separate files that you can load later. This capability is especially useful for generating specialized contour plots and intricate component and assembly structures.

When using the CMAP utility, it is good practice to close any other program window, especially the Annotation, Pan Zoom Rotate, Working Plane and Picker windows.

13.2.4. Displaying Loads and Other Special Symbols

The following sections describe how to manipulate loads and other special symbols.

[13.2.4.1. Enaling and Disabling Load Symbols and Contours](#)

[13.2.4.2. Displaying Boundary Condition Values Next to a Symbol](#)

[13.2.4.3. Displaying Boundary Condition Symbols for Hidden Surfaces](#)

[13.2.4.4. Scaling Vector Load Symbols](#)

[13.2.4.5. Enabling and Disabling Other Symbols](#)

13.2.4.1. Enaling and Disabling Load Symbols and Contours

To turn load symbols on or off for degree of freedom constraints and concentrated loads, issue the **/PBC** command. The command controls both solid-model and finite element load symbols.

For surface loads symbols or contours, issue the **/PSF** command. The command activates "immediate" display of surface loads on finite elements, but does not activate "immediate" surface load display on solid model entities.

For body force load contours, issue the **/PBF** command. The command applies to finite-element loads only; body force symbols do not appear in solid model displays. The command does not generate an "immediate" display.

The **/PBF** command can also display your current density magnitude as a vector instead of a contour. **/PBF, JS, 2** displays the current density magnitude as vector arrows along the surface. The length of the arrows is proportional to the current density magnitude.

Typically, you use the above commands to enable load symbols for visual verification when you apply the loads in SOLUTION (or PREP7). The program automatically disables these symbols when entering the POST1 postprocessor. See [Creating Geometric Results Displays \(p. 249\)](#) for more information about controlling postprocessing displays.

13.2.4.2. Displaying Boundary Condition Values Next to a Symbol

Display load symbols via the **/PBC** command. (Also see [Enabling and Disabling Other Symbols \(p. 247\)](#).)

The command also provides an option for displaying the boundary condition values next to the symbols. Some of the boundary condition values that are associated with this command include reaction force (RFOR), reaction moment (RMOM), displacement (U), and current flow (AMPS).

Because your applied forces/momenta can differ by orders of magnitude from your derived forces/momenta, you can specify the FBCS option on the **/PSYMB** command to determine the basis of the Force Boundary Conditions Scaling of your display.

See the [Command Reference](#) for more information about the various boundary values that are supported.

13.2.4.3. Displaying Boundary Condition Symbols for Hidden Surfaces

When hidden surfaces exist, 2-D drivers display your boundary condition symbols, yielding a confusing display. In some cases, however, you may want to see them. Issue use the **/HBC** command to control BC symbol display.

The default setting is *not* to display boundary condition symbols on the hidden surfaces (**/HBC**, WN, OFF). You can set the display ON or OFF individually for each window of your display.

13.2.4.4. Scaling Vector Load Symbols

The **/VSCALE** command enables you to adjust the scale of vector item symbols (such as the arrows representing concentrated forces). The command also offers a uniform scaling option, in which all items' vector symbols have the same length, regardless of their relative magnitudes.

13.2.4.5. Enabling and Disabling Other Symbols

Issue the **/PBC** command to enable or disable symbols for master degrees of freedom (MDOFs), coupled nodes, and nodes in constraint equations.

Issue the **/PSYMB** command to control symbols for local, nodal, and element coordinate systems, line directions, keypoints/nodes, and layer orientation (for layered elements).

Chapter 14: Creating Geometric Results Displays

In a *geometric results display*, you can review your solution results in a postprocessing display of your model's elements.

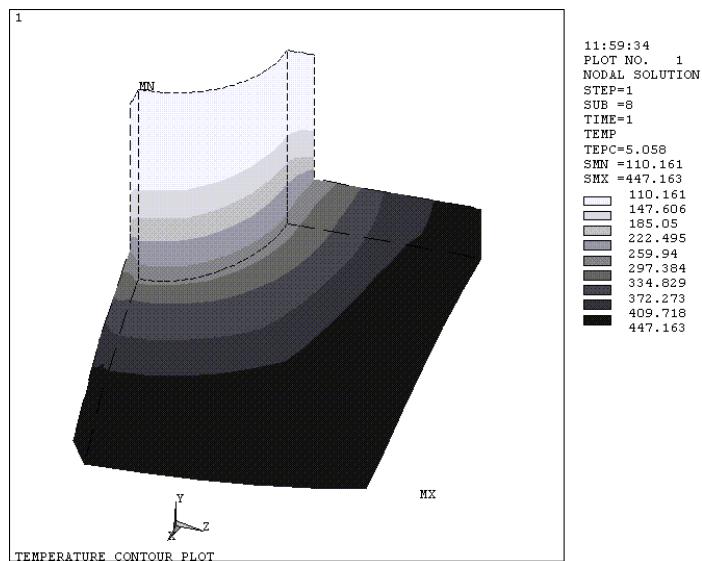
The following geometric results display topics are available:

- 14.1. Using the GUI to Display Geometric Results
- 14.2. Options for Creating Geometric Results Displays
- 14.3. Changing the Specifications for POST1 Results Displays
- 14.4. Q-Slice Techniques
- 14.5. Isosurface Techniques
- 14.6. Controlling Charged Particle Trace Displays

14.1. Using the GUI to Display Geometric Results

The choice of geometric results displays includes displaced shapes, results contours (including line-element "contours," such as moment diagrams), and vector (arrow) results (such as thermal flux vector displays). These displays are available only within POST1, the general postprocessor. The following figure illustrates a typical geometric results display:

Figure 14.1: Contour Results Plot



The most convenient way to create and control geometric results displays is by using the functions available under **Utility Menu> Plot** and **Utility Menu> PlotCtrls**. Alternatively, you can use graphics action and control commands, as described in the following subsections.

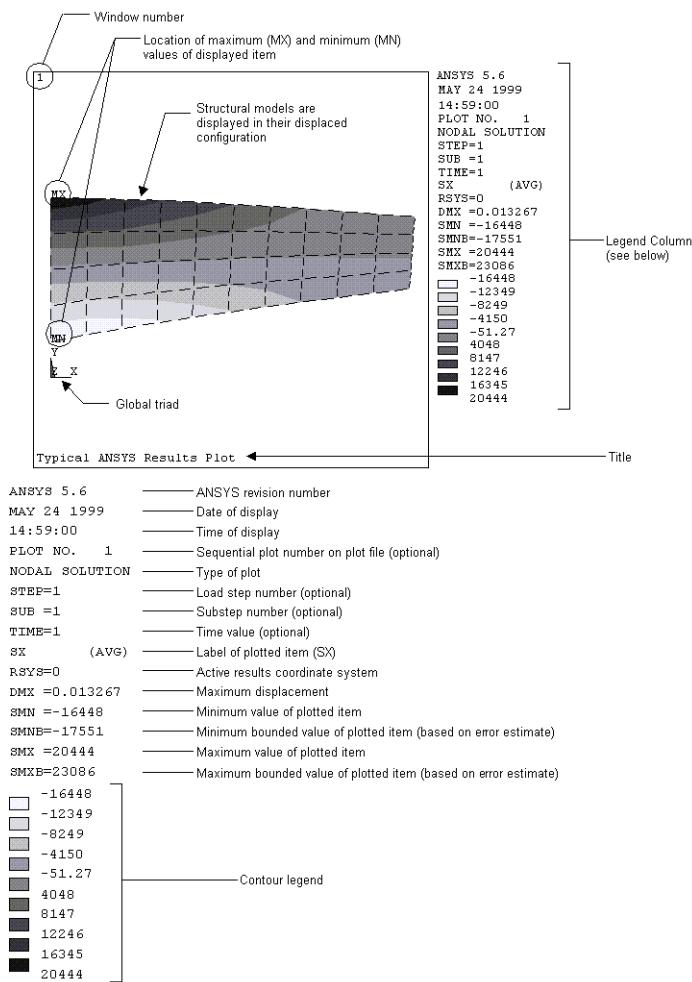
14.2. Options for Creating Geometric Results Displays

The following commands create geometric results displays in POST1:

Table 14.1: Commands for Creating Geometric Results Displays

Command	Purpose
PLDISP	Display displaced shapes
PLESOL	Display contours of results, discontinuous across element boundaries
PLETAB	Display contours of element table data
PLLS	Display element table items along line elements and 2-D axisymmetric shell elements
PLNSOL	Display continuous results contours
PLTRAC	Display charged particle trace
PLVECT	Display solution results as vectors
/REPLOT	Re-executes the last display action that executed

In [Figure 14.2: A Typical Results Plot \(p. 251\)](#), a typical geometric results display (in this example, created with a **PLNSOL** command) illustrates the kinds of information included in such displays.

Figure 14.2: A Typical Results Plot

14.3. Changing the Specifications for POST1 Results Displays

Besides reading about the features listed below, also see [Getting Started with Graphics \(p. 217\)](#) for general graphics specifications that you can apply to any kind of display, including geometric results displays.

14.3.1. Controlling Displaced Shape Displays

You can control displaced shape displays in two ways:

- *By superimposing undisplaced and displaced shapes.* A display of a structure's displaced shape will often be more meaningful if you can compare the displaced configuration against the original configuration. You can do this by using the **KUND** argument on the **PLDISP** command.
- *By multiplying displacements for distortion displays.* In most small-deformation structural analyses, the displaced shape is difficult to distinguish from the undisplaced shape. The program automatically multiplies the displacements in your results display, so that their effect is more readily apparent. You can adjust the multiplication factor via the **/DSCALE** command. The program interprets exactly zero values of this multiplier (**DMULT = 0**) as the default setting,

causing the displacements to be scaled automatically to a readily discernible value; therefore, to obtain "zero" displacements (an undistorted display) set *DMULT* = OFF.

14.3.2. Controlling Vector Symbols in Your Results Display

Two options are available for controlling vector symbols:

- *Displaying nodal or reaction force symbols.* You can add arrow symbols representing nodal and reaction forces (and moments) to your results display (**/PBC**) .
- *Vector length scaling.* You can control the lengths of vector symbols (**/VSCALE**), such as those displayed via **PLVECT** or **/PBC**.

14.3.3. Controlling Contour Displays

When light-source shading is on, the colors shown in the contour legend will not exactly match the contour colors used in the shaded model display. You can manipulate contour displays in the following ways:

- *Labeling contours.* In both vector and raster mode, your contours will always be automatically color-coded. In vector mode, you can add alphabetic contour labels (and a contour legend) via the **/CLABEL** command. In raster mode, **/CLABEL** adds (or removes) the contour legend.
- *Controlling the contour legend.* Sometimes, lengthy text in the legend column can cause part of the contour legend to be truncated. You can make more room available for the contour legend by issuing **/PLOPTS,LEG1,0**. To remove the contour legend from the legend column, issue **/PLOPTS,LEG3,0**.
- *Changing the number of contour labels.* In vector mode, if you apply contour labels, they will, by default, appear in every element crossed by a contour line. You can issue **/CLABEL** to control the number of alphabetic contour labels per element.
- *Changing contour colors.* To change the contour colors used in your display, create a new color-map file and read the new color-map file via the **/CMAP** command.

To restore color to contours that are grayed out, issue the command **/NUMBER,0**.

- *Changing isosurface colors.* Change isosurface colors via the ISURF label on the **/COLOR** command.
- *"Inverting" (or reversing) the contour colors.* By default, the program displays the algebraically greatest results values with a bright red contour color, and the algebraically lowest values with a blue contour color.

In some cases, you may want to invert this order. To create a reversed color-map file, use the CREATE option on the **/CMAP** command. You can then read that reversed color map file into the database (also via the **/CMAP** command).

- *Changing the contour interval.* To change the contour interval on your results display, issue either **/CVAL** or **/CONTOUR**.

These commands change the range of values displayed in contour displays. **/CONTOUR** produces uniform contour intervals, while **/CVAL** produces specified contour values (which need

not be uniform). If you issue both commands, the program uses the last one specified. For related information, see Section [Changing the Number of Contours \(p. 253\)](#).

- *Topographic contour displays.* You can transform flat contour results displays into 3-D topographic displays via the **/SSCALE** command.
- *Displaying numerical results values.* To display results values at each node in a contour display, issue **/PNUM,SVAL,1**.
- *Displaying or suppressing "MN" and "MX" symbols.* The MN and MX symbols identify the locations of the minimum and maximum contour values. Control the symbols via the MINM label on the **/PLOPTS** command.
- *Producing 3-D isosurface, particle gradient, or gradient triad displays.* Isosurfaces, particle clouds, and gradient triads are tools that can help you to visualize the state of response within a 3-D solid body. Change your contour displays to one of those display styles via **/CTYPE**.

14.3.4. Changing the Number of Contours

The program displays nine contours by default. To decrease (but not increase) the number of contours, issue the **/CVAL** command. To change (increase or decrease) the number of contours, issue the **/CONTOUR** command.

One or more of the following factors can prevent the program from displaying more than nine contours:

- The device name.
- Whether the display is directed to the screen or to a file.
- The display mode (vector or raster).
- The number of color planes.

Any of these factors can override the number of contours specified via **/CONTOUR**. Adjust the factors via the **/SHOW** command.

In any case, the maximum number of contours available is 128.

Driver	Contour Display
The X11 driver (screen display) and raster mode	You can display a maximum of nine contours, no matter how many contours the /CONTOUR command specifies.
The X11 driver (screen display) and raster mode	You can display more than nine contours, but the number of contours displayed will be rounded down to the next lowest multiple of nine. For example, if you specify 20 contours, the program displays only 18 contours. In addition, if you specify more than nine contours, contour colors will not be unique (that is, you might have two or more adjacent contour lines with the same color).

Driver	Contour Display
The X11C driver (screen display) in either vector or raster mode	<p>If eight graphic planes are available, you can specify any number of contours, up to 128.</p> <p>If your display device does not support eight graphic planes, you are limited to displaying nine contours. If another process has used some of the colors, making fewer than eight graphic planes available, you cannot display more than nine contours. (To verify how many graphic planes are available, issue the /PSTATUS command after a plot command.)</p> <p>To make more graphic planes available, exit the program, re-enter, and issue the /SHOW,X11C-FORC to force selection of the full set of eight graphic planes.</p>

If the current graphics are *not* displayed as Multi-Plots (**Utility Menu> Plot> Multi-Plots**), then the following is true:

- If the current device is a 3-D device [**/SHOW,3D**], the model contours in all active windows will be the same, even if separate **/CONTOUR** commands are issued for each active window.
- For efficiency, the program's 3-D graphics logic maintains a single data structure (segment), which contains precisely one set of contours. The program displays the same segment in all windows. The view settings of each window constitute the only differences in the contour plots in the active windows.

14.4. Q-Slice Techniques

Q-slicing is a technique for querying the interior of your model via slice planes. To implement Q-slicing, change the hidden surface type to Q-slice by issuing the **/TYPE,1,8** command.

By default, the slice plane is perpendicular to the view and is positioned at the focus point. You can set the slice plane via the GUI path shown above or via the **/CPLANE,1** command.

To position the working plane, use either of these methods:

- **Utility Menu> WorkPlane> Align WP with> Keypoints.**
- Click the dynamic mode button in the Offset WP menu (**Utility Menu> WorkPlane> Offset WP by Increments**).

To animate Q-slices, use either of the following GUI paths:

- **Utility Menu> PlotCtrls> Animate> Q-Slice Contours**
- **Utility Menu> PlotCtrls> Animate> Q-Slice Vectors**

14.5. Isosurface Techniques

Isosurface displays are surfaces of constant values (for example, stress). To obtain an isosurface display of von Mises stress, perform these steps:

1. Issue the command **/CTYPE,1** (**Utility Menu> PlotCtrls> Style> Contours> Contour Style**).
2. Issue the command **PLNSOL,S,EQV** (**Main Menu> General Postproc> Plot Results> Contour Plot> Nodal Solu**).

You can animate isosurfaces. To do so, either invoke the **ANISOS** macro (**Utility Menu> PlotCtrls> Animate> Isosurfaces**).

14.6. Controlling Charged Particle Trace Displays

You can produce graphic displays showing how a charged particle travels in an electric or magnetic field.

Action	Command
Produce charged particle trace displays	PLTRAC
Select trace points by number or by picking	TRPOIN
List trace points	TRPLIS
Delete trace points	TRPDEL
Animate the charged particle trace to a specified elapsed time	TRTIME

See also:

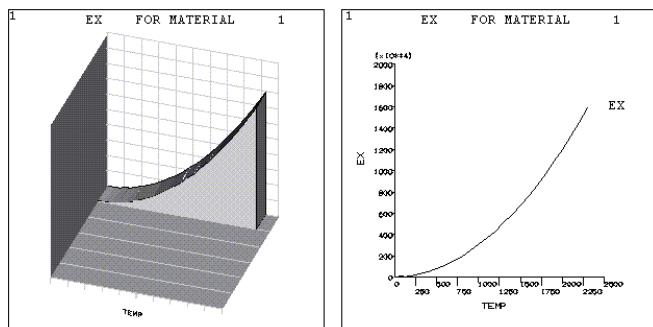
- [The General Postprocessor \(POST1\) \(p. 131\)](#) for more information about graphic displays.
- [Animation \(p. 265\)](#) for information about particle trace animation.

Chapter 15: Creating Graphs

If you want to review your material property curves, trace the time-history response of your system, or examine the relationship between any two items in your analysis, you can often do so most effectively using a graph. Graphs can be either 2-D (X-Y) or 3-D (X-Y-Z, where Z must always be *TIME*).

The following figure shows two typical graphs:

Figure 15.1: Typical ANSYS Graphs



The most convenient way to create and control graph displays is by using the GUI operations available under **Utility Menu> Plot** and **Utility Menu> PlotCtrls**. Alternatively, you can use graphics action and control commands, as described in the following topics:

15.1. Graph Display Actions

15.2. Changing the Specifications for Graph Displays

15.1. Graph Display Actions

The commands described here create graphs anywhere in the program (including at the BEGIN level):

Graph Display Action	Command
Display linear material properties (those defined via the MP family of commands) as a function of temperature	MPPLLOT
Display nonlinear data curves (stress-strain, B-H curve, etc. defined via the TB family of commands)	TBPLLOT
Display column vectors of array parameters	*VPLLOT
The following commands create graphs in POST1 only:	
Calculate and graph path items vs. path length	PLPATH
Calculate and graph the membrane and membrane plus linearized stresses along a path	PLSECT
Other graphic display actions:	

Graph Display Action	Command
Graph any predefined variable as a function of <i>TIME</i> (<i>FREQ</i> for harmonic analyses) or some other variable that you define. Available in POST26 only.	PLVAR
Reissue the last display action command issued	/REPLOT

15.2. Changing the Specifications for Graph Displays

In addition to the features described here, see [Getting Started with Graphics \(p. 217\)](#) for general graphics specifications that apply to any type of display, including graphs.

15.2.1. Changing the Type, Style, and Color of Your Graph Display

Modify the appearance of your graph display as follows:

Graph Display Action	Command(s)
Display or hide axis divisions (tick marks)	/GROPT,AXDV
Display or hide axis scale numbers	/GROPT,AXNIM
Change the size of axis scale numbers	/GROPT,AXNSC
Change the number of significant digits used in axis scale numbers	/GROPT,DIG1 and DIG2
Toggle between log and linear scales	/GROPT,LOGX and LOGY
Set separate Y-axis scales for different curves (useful for graphing two or more different items on one display)	/GRTYP,2 or 3 /GROPT,ASCAL,ON
Set uniform scaling of separate Y axes	/GROPT,ASCAL,OFF
Create data slice graph curves (curves with Z-direction thickness) [1]	/GRTYP,3 /VIEW /DIST /GROPT,FILL,ON
Set the line thickness for axes, grid lines or graph curve lines	/GTHK,AXIS or GRID or CURVE
Display or hide a grid (in the XY plane) [2]	/GRID
Generate a dashed tolerance curve about the displayed curve	SPREAD
Color-fill areas under curves (to enhance the visual effect of the curves)	/GROPT,FILL
Specify the color of curves (and color-filled areas under curves)	/COLOR,CURVE

Graph Display Action	Command(s)
Fill the areas under curves with grids	/GROPT,CGRID,ON
Color the XY, XZ, and/or YZ grid planes	/COLOR,GRBAK
Color the window background	/COLOR,WBAK

1. To see this effect, change your display's viewing angle and distance (for example, via **/VIEW,1,2,2,3** and **/DIST,1,88**) and set the color-fill option (**/GROPT,FILL,ON**).
2. A grid can be either full (horizontal *and* vertical grid lines) or partial (horizontal *or* vertical grid lines).

15.2.2. Labeling Your Graph

Graph Display Action	Command(s)
Label the X and Y axes	/AXLAB
Label the curves (for POST26 plotted-variable graphs) [1]	NSOL, ESOL
Add annotation	/ANNOT [2]

1. For all other types of curves, including array parameter (***VPOINT**) curves, the default label is the item or parameter name specified via the display action command. For these curves, issue the **/GCOLUMN** command to specify curve labels. The command can set any text or character string as a curve label.
2. The **/ANNOT** command is *not* intended to be typed directly into the command line (although it can be included in an input file for batch input or for use with the **/INPUT** command). The command is generated via a GUI selection (**Utility Menu> PlotCtrls> Annotation**) and appears in the log file (*Jobname.LOG*) if annotation is used.

Besides **/ANNOT**, the annotation commands are **/ANUM**, **/TLABEL**, **/LINE**, **/LARC**, **/LSYMBOL**, **/POLYGON**, **/PMORE**, **/PCIRCLE**, **/PWEDGE**, **/TSPEC**, **/LSPEC**, and **/PSPEC**.

For more information, see [Annotation \(p. 261\)](#).

15.2.3. Defining X and Y Variables and Their Ranges

The following topics related to defining X and Y variables and their ranges are available:

- 15.2.3.1. Defining the X Variable
- 15.2.3.2. Defining the Part of the Complex Variable to Be Displayed
- 15.2.3.3. Defining the Y Variable
- 15.2.3.4. Setting the X Range
- 15.2.3.5. Defining the TIME (or FREQ for Harmonic Analysis) Range
- 15.2.3.6. Setting the Y Range

15.2.3.1. Defining the X Variable

In POST26 plotted-variable graphs, the program uses *TIME* (or, for harmonic analyses, *FREQ*) for the X variable.

TIME can represent something other than chronological time. In setting up a time-independent analysis, you can arbitrarily define *TIME* to be equal to the value of some other item of interest (such as input pressure).

To define a parameter other than *TIME* against which the Y variable is to be displayed, issue the **NSOL**, **ESOL**, and **XVAR** commands.

15.2.3.2. Defining the Part of the Complex Variable to Be Displayed

When plotting harmonic-response results in POST26, select what part of the complex variable (amplitude, phase angle, real part, or imaginary part) to display in your graph via the **PLCPLX** command.

15.2.3.3. Defining the Y Variable

The various graphics action commands define the Y variable. Sometimes, the commands refer to labels that have been defined in other commands.

Example 15.1: Commands Referring to Labels Defined in Other Commands.

- **PLPATH** uses labels defined via the **PDEF**, **PVECT**, **PCALC**, **PDOT**, and **PCROSS** commands.
- **PLVAR** also uses labels defined via the **NSOL** and **ESOL** commands.
- **PLSECT**, **FSPLOT** and ***VPLOT**, however, identify the Y variable directly.

15.2.3.4. Setting the X Range

The **/XRANGE** command enables you to graph only a portion of the full range of X-variable data. You to zoom in on (or out of) a specific segment of the curve.

15.2.3.5. Defining the **TIME** (or **FREQ** for Harmonic Analysis) Range

The **PLTIME** command enables you to establish a range of *TIME* for graph displays. The program always displays *TIME* in the Z-axis direction.

If **XVAR** = 1, *TIME* is also displayed in the X-axis direction. **PLTIME** or its equivalent then also sets the abscissa scale range. (A range established by **/XRANGE** takes precedence over one defined by **PLTIME**.)

15.2.3.6. Setting the Y Range

Your graph contains the full range of available Y-variable data by default. To define a smaller or larger range, issue the **/YRANGE** command. By specifying a value for the command's *NUM* argument, you can selectively define different ranges for different curves (provided that separate Y-axis scales have been established).

Chapter 16: Annotation

A common task during the analysis process is presenting model and results data with additional notations applied, such as dimensions, comments, highlights, or other text or artwork. You can enhance the program's standard display with a variety of annotation primitives including text, dimensions, polygons, symbols, and even pie charts.

Annotation functions are available for both 2-D and 3-D graphics cards. You can apply 3-D annotation even if a 2-D graphics card is installed or a 2-D driver (Win32 or X11) is loaded. For best results, however, ANSYS, Inc. recommends installing a quality 3-D graphics card is installed and the appropriate 3-D or Open GL device driver.

The ! and \$ characters are not available for text annotation.

The following annotation topics are available:

- [16.1.2-D Annotation](#)
- [16.2. Creating Annotations for Models](#)
- [16.3. 3-D Annotation](#)
- [16.4. 3-D Query Annotation](#)

16.1. 2-D Annotation

2-D text and graphics annotations are formed as a 2-D overlay on the graphics screen. Because this overlay exists as an imaginary plane, when you transform your model (by changing the scaling, focus, viewing angle, magnification, etc.), your carefully-constructed annotation will not move with the model. Because of this, 2-D annotation should be used primarily for finalized output (reports and printouts) and for representations of the model's state at various stages in the analysis. 3-D annotations will remain anchored to a specific location on the model, and are discussed later in this chapter.

Access 2-D annotation functions via the GUI at **Utility Menu> PlotCtrls> Annotation> Create 2D Annotation**. Every annotation function performed from the GUI places one or more underlying commands in the log file so that you can accurately reproduce the display if the log file is later submitted for batch input. Annotation commands that might appear in such a session log include **/ANNOT**, **/ANUM**, **/TLABEL**, **/LINE**, **/LARC**, **/LSYMBOL**, **/POLYGON**, **/PMORE**, **/PCIRCLE**, **/PWEDGE**, **/TSPEC**, **/PSPEC**, and **/LSPEC**.

The following annotation primitives are available from the 2-D annotation dialog box:

- Text
- Lines
- Rectangles
- Circles
- Arcs
- Polygons

- Wedges
- Arrows
- Dimensions
- Pies
- Symbols

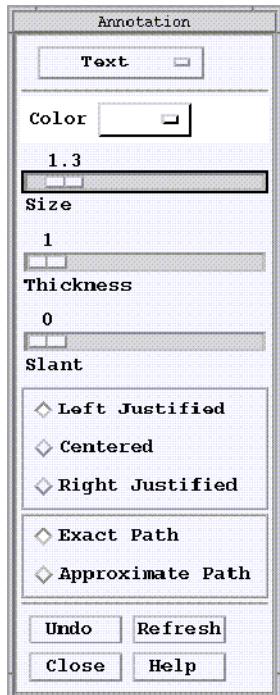
Use the Options setting to copy, move, resize or delete existing annotations.

16.2. Creating Annotations for Models

When you select **Utility Menu> PlotCtrls> Annotation> Create 2D Annotation**, the text annotation dialog box shown below appears. Text annotation can be applied either as stroke text (line-draw characters) or as bitmap fonts.

Bitmap fonts are available on most systems, with the number and type varying from system to system. Bitmap fonts must be enabled (**Utility Menu> PlotCtrls> Annotation> Enable Bitmap Font**) before the annotation is created.

Figure 16.1: Stroke Text Annotation Dialog Box



The fields and buttons presented in the annotation dialog box change when you reset the annotation entity type. For example, if you reset the annotation entity to arcs, the dialog box shown in [Figure 16.1: Stroke Text Annotation Dialog Box \(p. 262\)](#) changes to display the options available for annotation arcs (arc color, solid or dashed lines, and arc width). Regardless of the annotation entity used, the annotation dialog box always displays four action buttons:

Undo -- Erases the last annotation entity created.

Refresh -- Redisplays the annotation, which is useful after move and delete operations.

Close -- Closes the annotation dialog box.

Help -- Displays online help for the dialog of the currently selected annotation entity.

After you create annotations, you can control their display by selecting **Utility Menu> PlotCtrls> Annotation> Display Annotation**. Accessing this menu pick toggles annotation display on and off.

16.3. 3-D Annotation

3-D text and graphics annotations are assigned XYZ coordinates and exist in 3-D space. When using 3-D annotation, the following anchor locations are available:

- Nodes
- Elements
- Key Points
- Lines
- Areas
- Volumes
- All
- At XYZ
- On View

Because 3-D annotation is applied in relation to the XYZ coordinates of the anchor, you can transform your model with the annotation maintaining the spatial relationship with the model. In some cases, changing the perspective or the size of the model changes the apparent relationship between the annotation and the model. The overall 3-D dimensions of the model are defined by a bounding box. If portions of your model's bounding box lie outside of the visible area of your graphics window (if you are zoomed in on a specific area of your model), it can affect the placement of your 3-D annotations. Zooming out usually overcomes this problem. Unlike 2-D annotation, 3-D annotation is valid for the global Cartesian (**CSYS,0**) coordinate system only.

Access 3-D annotation functions via the GUI at **Utility Menu> PlotCtrls> Annotation> Create 3D Annotation**.

Every annotation function performed via the GUI places one or more underlying command(s) in the log file, enabling you to accurately reproduce the display if the log file is later submitted for batch input.

The following annotation primitives are available from the 3-D annotation dialog box:

- Text
- Lines
- Areas
- Symbols
- Arrows

Use the Options setting to copy, move, resize or delete existing annotations.

16.4. 3-D Query Annotation

Query Annotation enables you to retrieve model information directly from the database and apply it to the model. The Model and Results Query Pickers provide a **Generate 3-D Anno** check box that enables the annotation function.

You can obtain basic model information, results data and even simple geometric/loading information (force per unit area, angle between lines, etc.) by graphically picking the desired items.

As with standard 3-D Annotation, use the Options setting to copy, move, resize or delete 3-D Query Annotations, and 3-D Query Annotation is valid for the Cartesian (**CSYS**,0) coordinate system only.

For more information, see [Graphical Picking](#) in the *Operations Guide*.

Chapter 17: Animation

Animation is useful for graphically interpreting many analysis results, especially nonlinear or time-dependent behavior. The program provides tools that enable you to animate any type of display.

The following animation topics are available:

- 17.1. Creating Animated Displays
- 17.2. Using the Basic Animation Commands
- 17.3. Using One-Step Animation Macros
- 17.4. Animation in the Windows Environment

17.1. Creating Animated Displays

The easiest way to generate animation is to use the GUI functions available via **Utility Menu> PlotCtrls> Animate**. The functions execute animation commands internally, which you can type in directly if you prefer.

17.2. Using the Basic Animation Commands

You can display several frames in rapid succession to achieve an animation effect via the **/SEG** and **ANIM** commands:

The **/SEG** command stores graphics data in the terminal's local segment (graphics operation) or pixmap (screen dot) memory (depending on the graphics device). The storage occurs at the same time that a graphics action command produces a display. To display the stored frames in a sequence, issue the **ANIM** command.

Example 17.1: Typical Command Stream for Animation

```
/SEG,DELE      ! Deletes all currently stored segments
/SEG,MULTI     ! Stores subsequent displays in segment memory
...
...           ! Plot-creation commands to generate a sequence of images
...           ! (See below for options)
/SEG,OFF       ! Turns off the frame-capture function
ANIM,15        ! Cycles through the stored sequence 15 times
```

To create the series of frames for your animation sequence, you can either issue a frame-by-frame series of graphics action commands, or invoke a predefined macro to generate the sequence automatically. The predefined macros are **ANCNTR**, **ANCUT**, **ANDATA**, **ANDSCL**, **ANFLOW**, **ANHARM**, **ANISOS**, **AN-MODE**, **ANTIME**, and **ANDYNA**.

The available amount of local segment or pixmap memory, and the memory requirements of each frame limit the number of frames you can include in an animated sequence. On most workstations and PCs, the amount of memory required depends on the number of pixels in each frame.

Although you can create animations of multiple window schemes, animations created with OpenGL display lists (**/DV3D**, ANIM, 0) do not retain the windowing scheme information. You *can* save multiple windows via the X11/WIN32 drivers, or via the OpenGL driver with **/DV3D,ANIM,Key** in effect (where *Key* is not zero).

17.3. Using One-Step Animation Macros

The following table offers convenient one-step macro alternatives to the basic animation commands:

Table 17.1: Animation Macros

Action	Macro	Comments
Generate an animated sequence of a contoured deformed shape in POST1	ANCNTR	Before using, execute a display command that contains deformation, contouring, or both (such as PLNSOL,S,EQV)
Apply a traveling wave animation to graphics data in a modal <i>cyclic symmetry</i> analysis in POST1	ANCYC	For more information, see Applying a Traveling Wave Animation to the Cyclic Model in the Cyclic Symmetry Analysis Guide
Generate an animated sequence of a cutting plane through a contoured deformed shape in POST1	ANCUT	Before using, execute a display command that contains contouring.
Generate a sequential contour animation over a range of results data	ANDATA	This macro can create an animation sequence based on the last plot action command (such as PLDISP).
Generate an animated sequence of a deformed shape in the POST1 postprocessor	ANDSCL	Before using, execute a display command that contains deformation (such as PLDISP).
Generate an animated sequence of charged particle motion	ANFLOW	Before using, execute a command that produces particle trace on an element display (such as PLTRAC).
Generate a time-transient animation of time-harmonic results of the last plot action command (such as PLNSOL,B,SUM).	ANHARM	The animation converts the complex solution variables (real and imaginary sets) into time varying results over one period.
Generate an animated sequence of an isosurface of contoured deformed shape in POST1	ANISOS	Before using, execute a display command that contains contouring.
Generates an animated sequence of a deformed mode shape in POST1	ANMODE	Before using, execute a command that contains deformation.
Generate an animated sequence of a contoured deformed shape varying over time in POST1	ANTIME	Before using, execute a display command that contains deformation, contouring, or both. Your solution must include time variance.

17.4. Animation in the Windows Environment

The Mechanical APDL program on Windows platforms uses the Microsoft standard AVI file format to store animation frames (video only).

The following related topics are available:

[17.4.1. Understanding AVI File Support](#)

[17.4.2. Other Uses for AVI Files](#)

17.4.1. Understanding AVI File Support

Animation capabilities are available among the options under **Utility Menu> PlotCtrls** and via the [animation macros \(p. 266\)](#). If you are animating a deformed shape or different mode shapes of your analysis, the program stores the animation frames in a file called `Jobname.AVI` (where `Jobname` is the jobname for the current session). After completing this step, the program opens the Windows Media Player.

To animate contours, the program displays a dialog box from which you can select animation options. The program generates the frames and Windows Media Player displays them.

The Replay animation option starts Windows Media Player. If you have stored an animation sequence during the current session, the file name associated with it is passed to Windows Media Player automatically.

You can animate other quantities or generate an animation in other parts of the program via the [**/SEG**](#) command.

AVI files cannot be created directly in batch mode.

The following topics related to AVI file support are available:

17.4.2. Other Uses for AVI Files

While running Windows Media Player, you can use OLE (Object Linking and Embedding) to export your animation to other applications. Select the **Copy** option under the **Edit** menu. Then, you can embed the animation object in another OLE-compliant application. For example, you can embed animation objects in Microsoft Word or Microsoft PowerPoint.

After an animation object is embedded in an application, you can double-click on the object to start playing back your animation. To share your compound document with others, give them the `Jobname.AVI` file that you created and a copy of the file containing the embedded animation sequence.

Chapter 18: External Graphics

Besides creating and controlling graphics that you can view directly from within the program, you can export the contents of the graphics window to either to a printer or to a graphics file.

The following external graphics topic is available:

[18.1. External Graphics Options](#)

18.1. External Graphics Options

You can export the contents of the graphics window (full screen options are also available for some platforms), either to a printer or to a graphics file. The following topics are available:

[18.1.1. Printing Graphics in Windows](#)

[18.1.2. Exporting Graphics in Windows](#)

[18.1.3. Printing Graphics in Linux](#)

[18.1.4. Exporting Graphics in Linux](#)

18.1.1. Printing Graphics in Windows

You can obtain hard copy output by selecting **Utility Menu> PlotCtrls> Hard Copy**. You can print the contents of the graphics window, or create an exportable graphics file. When you select the **To Printer** option, the Windows printer dialog for the designated printer appears. You can modify printing options including page layout, output resolution and document handling.

Printer spooling options are commonly used to free up the processor more quickly (especially in Z-buffered mode). In Type 4 or Polygon mode, spooling may cause some elements to not plot, or to be improperly placed. Select the **Print Directly to Printer** option when these types of printing problems are encountered.

18.1.2. Exporting Graphics in Windows

Selecting **Utility Menu> PlotCtrls> Hard Copy** and then the **To File** option displays the Graphics Hard Copy dialog. Several popular file-export formats (BMP, JPEG, TIFF and PNG) along with limited page layout and configuration options are available. You can export your output window (or full screen) contents into a large number of commercially available publishing or presentation software applications.

Note:

If you are running in batch mode and want to export graphic files, you must issue the **/SHOW** command before commands that produce a graphical output.

ANSYS, Inc. JPEG software is based in part on the work of the Independent JPEG Group, Copyright 1998, Thomas G. Lane.

18.1.2.1. PNG Format

The PNG format is an extremely capable, portable file format with widespread acceptance in many computer applications, including the web. It is a lossless, true-color image format that minimizes the distortion, mottling and pallet limitations found on other formats, while still retaining excellent compression performance. The program creates PNG files with the assistance of the following LIBPNG and ZLIB packages:

- LIBPNG version 1.0.8 - July 2000

Copyright 1995, 1996, Guy Eric Schalnat, Group 42, Inc.

Copyright 1996, 1997 Andreas Dilger

Copyright 1998-2000 Glenn Randers-Pehrson

- ZLIB Version 1.1.3

Copyright 1995 - 1998 Jean Loup Gailly and Mark Adler.

18.1.2.2. ClearType and Reverse Video

Windows operating systems with ClearType may generate plots with blurry text using the Reverse Video option in the Graphics Hard Copy dialog box. Instead, use the Reverse Video option (**Utility Menu> PlotCtrls> Style> Colors> Reverse Video**) for the graphics window.

18.1.2.3. Create Exportable Graphics

You can create exportable graphics via **Utility Menu> PlotCtrls> Redirect Plots**. In addition to the page layout and configuration options found in the Graphics Hard Copy dialog box, this method provides additional position, resolution and scaling options. When you use this option, the output functions for most of these formats are controlled from within the program, instead of by the operating system. This selection also provides VRML 3-D rendering output, along with screen dump and animation options. The **Redirect Plots** export defaults to raster mode, even if vector is the prescribed **/DEVICE** mode. Ensure that the check box in the dialog box is checked for the desired output.

18.1.2.4. Export Windows Metafiles

You can export Windows Metafiles directly from the program (Windows systems only) via **Utility Menu> PlotCtrls> Write Metafile**. The subsequent dialog boxes provide limited page layout and configuration options.

Because most of the export methods listed above use the system's video output to generate the file format, they will not work unless you are in interactive mode. If you are not in interactive mode, and you want to export any of the above graphics types, use the **/SHOW** command. This method enables you to generate output files when you run your analysis from a batch file. The Courier and Helvetica font files used for text output within the JPEG, PNG and TIFF batch outputs are not accessed

from your operating system. Permission to use these files is granted by Adobe Systems, Inc. and Digital Equipment Corp. in association with these functions only.

18.1.3. Printing Graphics in Linux

You can print from within the program. Selecting **Utility Menu> PlotCtrls> Hard Copy** displays the Hard Copy dialog box. Print options, including page layout, reverse video and grey scale are available from this dialog box.

The black background provided in the graphics window is often unsuitable for printing. Selecting Reverse Video displays the graphic on a white background. Contour colors and other colors selected from the palette are unaffected by this option.

18.1.4. Exporting Graphics in Linux

The PS Hard Copy dialog box (**Utility Menu> PlotCtrls> Hard Copy**) also provides a number of file export formats (JPEG, TIFF, and PNG) along with limited page layout options. These files can be used in various word processing and desktop publishing applications.

As in Windows, you must use the **/SHOW** command in order to generate file exports during batch runs.

To obtain additional export formats, select **Utility Menu> PlotCtrls> Redirect Plots**. You can select from JPEG, TIFF, PNG, and VRML. These formats are suitable for a wide range of applications outside of the program.

The **Redirect Plots** export defaults to raster mode, even if vector is the prescribed **/DEVICE** mode. Ensure that the check box in the dialog box is checked for the desired output.

Chapter 19: The Report Generator

The report generator allows you to capture graphical and numerical data at any time throughout the analysis process and then assemble an HTML-based report using the captured data.

To capture data, you can use the report generator interactively or in batch mode. To assemble a report using the captured data, you can use any of these tools:

- The report generator itself (either interactively or in batch mode)
- A third-party (external) HTML editor
- Third-party (external) presentation software.

Using the report generator is a straightforward process, as follows:

1. Start the report generator and specify a directory to store your data and report(s).
2. Capture data (images, animations, tables and listings) that you want to include in your report.
3. Assemble your report using the captured data.

The following topics concerning the report generator are available:

[19.1. Starting the Report Generator](#)

[19.2. Capturing an Image](#)

[19.3. Capturing Animation](#)

[19.4. Capturing a Data Table](#)

[19.5. Capturing a Listing](#)

[19.6. Assembling a Report](#)

[19.7. Setting Report Generator Defaults](#)

19.1. Starting the Report Generator

To start the report generator, select **Utility Menu> File> Report Generator**. *Result:* The **Report Generation** window appears, as shown:

Figure 19.1: Report Generator GUI



The buttons activate the following functions (from left to right): **Image Capture**, **Animation Capture**, **Table Capture**, **Listing Capture**, **Report Assembler**, and **Settings**. When using the report generator, move the mouse pointer over any button to see a description of that button's function.

19.1.1. Specifying a Location for Captured Data and Reports

Your captured data, and any reports that you assemble, reside in a directory that you specify when you start the report generator. The default directory is `jobname_report`.

If the directory that you specify does not exist, the report generator creates it (after prompting you to approve the new directory). If the directory already exists, you have the option to *append* (add) captured data to any existing data in the directory or to *overwrite* (erase) the contents of the directory and start over.

When you specify a directory for your data and reports, the report generator writes this command to the log file:

```
~eui,'ansys::report::setdirectory directory'
```

19.1.2. Understanding the Behavior of the Graphics Window

The report generator constrains image size to accommodate most printers and paper sizes. When you start the report generator, it resizes the **Graphics** window to obtain the optimum image size.

After starting the report generator, do not adjust the size of the **Graphics** window; otherwise, unpredictable results may occur.

To further facilitate printing, the image foreground changes to black and the background changes to white. (You can modify the report generator's settings to prevent the default color change. For more information, see [Setting Report Generator Defaults \(p. 286\)](#).)

When you close the report generator, it restores the **Graphics** window's original size and color scheme.

19.1.3. A Note About the Graphics File Format

The report generator uses the Portable Network Graphics (PNG) format to store images. PNG files are small and exhibit little or no color loss.

19.2. Capturing an Image

This section describes how to capture and store a still image, either interactively or within a batch run.

The report generator saves images to the `images` subdirectory of your specified report directory. The name of each file is `imagen.png`, where `n` is a sequential numeric identifier beginning at 1 and incrementing as you capture additional images.

19.2.1. Interactive

Follow these steps to capture a still image using the report generator GUI:

1. Click the **Image Capture** button.

Result: The **Image Capture** dialog appears.

2. Specify a caption for the captured image (for example, "Pentagonal Prism").

The caption can contain APDL parameters in the format `%parm%`. (Specify "%%" if you want to display the "%" character in your caption.)

3. Click the **OK** button.

Result: The report generator issues this report command to the program and saves the image to your report directory:

```
~eui,'ansys::report::imagecapture "caption"'
```

19.2.2. Batch

The following line must appear near the beginning of your batch code *before* any report commands:

```
~eui,'package require ansys'
```

To capture a still image via a batch run, insert this report command at the point in the run where you want to capture an image:

```
~eui,'ansys::report::imagecapture "caption"'
```

19.3. Capturing Animation

This section describes how to capture and store an animation sequence, either interactively or within a batch run. (Animation capture is possible only in postprocessing after issuing a **SET** command.)

The report generator saves all individual image files comprising an animation sequence to a subdirectory (of your specified report directory) named `animseq_n`, where *n* is a sequential numeric identifier beginning at 1 and incrementing as you capture additional animations. The functions for accessing the animation reside in `ansysAnimations.js`, a JavaScript file in the report directory.

19.3.1. Interactive

Follow these steps to capture an animation sequence using the report generator GUI:

1. Click the **Animation Capture** button.

Result: The **Animation Capture** dialog appears.

2. Specify a caption for the captured animation (for example, "Prism Deformed Shape Animation Result").

The caption can contain APDL parameters in the format `%parm%`. (Specify "%%" if you want to display the "%" character in your caption.)

3. Specify the type of animation sequence to capture (such as mode shape, deformed shape, etc.) as applicable.
4. Click the **OK** button.

Result: The report generator issues this report command to the program:

```
~eui,'ansys::report::animcapture "caption"'
```

Also, the animation settings window associated with the animation type you selected (for example, **Animate Mode Shape** or **Animate Deformed Shape**) appears.

5. Modify the animation settings or accept the default settings, then click the **OK** button.

Result: The report generator saves the animation sequence.

19.3.2. Batch

The following line must appear near the beginning of your batch code *before* any report commands:

```
~eui,'package require ansys'
```

To capture an animation sequence via a batch run via a batch run, insert this report command at the point in the run where you want to capture an animation:

```
~eui,'ansys::report::animcapture "caption"'
```

Follow the report command with an animation command (for example, **ANTIME** or **ANDATA**).

19.4. Capturing a Data Table

This section describes how to capture and store a data table, either interactively or within a batch run.

The report generator appends captured table data to `ansysTables.js`, a file in your specified report directory containing the JavaScript functions for accessing your table data. (The file contains code to generate HTML as well as comments that hold the table information in a tab-delimited form, allowing you to paste the table data into software other than an HTML document.) The report generator assigns the name `table_n` to each captured table, where `n` is a sequential numeric identifier beginning at 1 and incrementing as you capture additional tables.

19.4.1. Interactive

Follow these steps to capture a data table using the report generator GUI:

1. Click the **Table Capture** button.

Result: The **Table Capture** dialog appears.

2. Specify a caption for the captured table (for example, "Prism Material Properties Table").

The caption can contain APDL parameters in the format `%parm%`. (Specify "%%" if you want to display the "%" character in your caption.)

- Select a predefined table type from the list. (The report generator filters the list of available table types based on the current analysis.)

If you select the **"Material properties"** table type, specify the currently defined materials via the **Material Selection** field.

Note:

The program does not display a material property which has no value associated with it.

If you select the **Custom Table** option, specify the table size (that is, the number of columns and rows) via the **Custom Table Size** field.

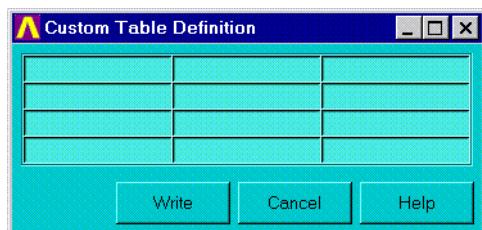
- Click the **OK** button.

Result: The report generator saves your *captured* table data. However, if you have selected the **Custom Table** option, see [Creating a Custom Table \(p. 277\)](#).

19.4.1.1. Creating a Custom Table

If you are creating a custom table, the **Custom Table Definition** dialog appears after you click the **Table Capture** dialog's **OK** button. The dialog contains an empty table of the dimensions that you specified. For example, assuming that you specified a table of three columns and four rows, the dialog looks like this:

Figure 19.2: Custom Table Definition



Type your custom information, which can include the following valid entries, into each cell:

Valid entry type	Example
Text	Maximum Deflection
Text with HTML tags	Maximum Stress [Kg/mm ²]
A *GET command	*get,,node,10,u,y
A *GET command with HTML tags	{<I>} {*get,,node,10,u,y} {</I>}
	<i>Important:</i> The "{" and "}" characters are necessary for parsing purposes.

After typing entries into each cell, click the **Write** button to save your custom table.

19.4.2. Batch

The following line must appear near the beginning of your batch code *before* any report commands:

```
~eui,'package require ansys'
```

To capture a data table via a batch run, insert this report command at the point in the run where you want to capture the table:

```
~eui,'ansys::report::tablecapture tableID "caption" materialID'
```

The *tableID* value is a table identifier representing one of the following predefined table types:

Table ID	Description
1	Creates a table of the finite element entities used in the analysis
2	Creates a table of properties for the requested material ID used in the analysis
3	Creates a table of the loads applied in the analysis
4	Reaction Forces
5	Reaction Moments
6	Max Displacements
7	Directional Stress
8	Shear Stress
9	Principal Stress
10	Equivalent Stress and Stress Intensity
11	Thermal Gradients
12	Thermal Flux
13	Thermal Temperatures
14	Natural Frequencies
15	Rotation
16	Temperature
17	Pressure
18	Electric Potential
19	Fluid Velocity
20	Current
21	Electromotive Force Drop
22	Turbulent Kinetic Energy
23	Turbulent Energy Dissipation
24	Component Total Strain
25	Shear Total Strain
26	Principal Total Strain
27	Total Strain Intensity and Total Equivalent Strain
28	Component Elastic Strain

Table ID	Description
29	Shear Elastic Strain
30	Principal Elastic Strain
31	Elastic Strain Intensity and Elastic Equivalent Strain
32	Component Plastic Strain
33	Shear Plastic Strain
34	Principal Elastic Strain
35	Plastic Strain Intensity and Plastic Equivalent Strain
36	Component Creep Strain
37	Shear Creep Strain
38	Principal Creep Strain
39	Creep Strain Intensity and Creep Equivalent Strain
40	Component Thermal Strain
41	Shear Thermal Strain
42	Principal Thermal Strain
43	Thermal Strain Intensity and Thermal Equivalent Strain
44	Component Pressure Gradient and Sum
45	Component Electric Field and Sum
46	Component Electric Flux Density and Sum
47	Component Magnetic Field Intensity and Sum
48	Component Magnetic Flux Density and Sum

If *tableID* is 2 (Material Properties), one additional argument is required:

- *materialID* -- Corresponds to a material ID

19.5. Capturing a Listing

This section describes how to capture the results of a command, either interactively or within a batch run.

The report generator appends listing data to `ansysListings.js`, a file in your specified report directory containing the JavaScript functions for accessing the listing. (The file contains code to generate HTML as well as comments that hold the list information, allowing you to paste the listing into software other than an HTML document.) The report generator assigns the name `listing_n` to each captured listing, where *n* is a sequential numeric identifier beginning at 1 and incrementing as you capture additional listings.

If you intend to use a captured listing in an HTML report (assembled using either the report generator or a third-party HTML tool), be aware that HTML sizes the text smaller if its width is greater than 132 columns; however, all text associated with the listing may still not fit on a printed page.

19.5.1. Interactive

Follow these steps to capture a listing using the report generator GUI:

1. Click the **Listing Capture** button.

Result: The **Listing Data Capture** dialog appears.

2. Specify a caption for the listing (for example, "Prism Model Area Listing").
3. Specify the command to issue to generate the output text.
4. Click the **OK** button.

Result: The report generator issues this report command to the program and saves the listing:

```
~eui,'ansys::report::datacapture "caption" ansysCommand'
```

19.5.2. Batch

The following line must appear near the beginning of your batch code *before* any report commands:

```
~eui,'package require ansys'
```

To capture a listing via a batch run, insert this report command at the point in the run where you want to capture a listing:

```
~eui,'ansys::report::datacapture "caption" ansysCommand'
```

19.6. Assembling a Report

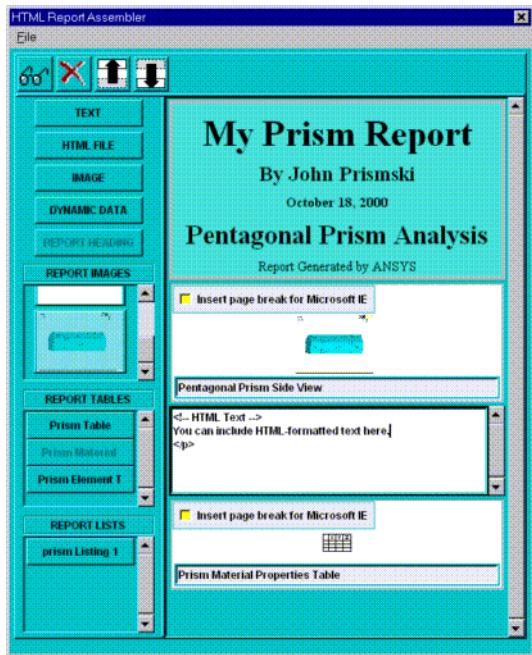
This section describes how to assemble your captured image and text data into a report interactively, within a batch run, or manually using the JavaScript interface.

19.6.1. Interactive Report Assembly

Follow these steps to assemble your report using the report generator GUI:

1. Click the **HTML Report Assembler** button.

Result: The **HTML Report Assembler** window appears, as shown:

Figure 19.3: HTML Report Assembler Window

Click the buttons and thumbnail images in the left panel to add components to your report. The larger panel on the right is your work area, displaying the components that you have chosen to include in the report.

2. Assemble the components of your report.

Click the buttons and thumbnail images in the left panel to add your captured images and text, and other components, as follows:

Button or Field	Purpose
TEXT	Inserts a text area allowing you to type HTML-formatted text into your report.
HTML FILE	Inserts a specified <i>existing</i> HTML file.
IMAGE	Inserts a specified image file (in PNG, JPG, JPEG or GIF format). The report assembler copies the image to your specified report directory and inserts a thumbnail image in the work-area panel.
DYNAMIC DATA	Inserts a text area allowing you to type commands, the results of which appear in your report. The dynamic data becomes part of the HTML code written to the log file. The report generator also writes the <code>ansys::report::interpdynamicdata</code> command to the log file; the command must process the HTML code in order for the results of the command(s) to appear in your report.
REPORT HEADING	Inserts a specified title, author name and subtitle, and includes the current date. The heading component always appears at the top of your report.

Button or Field	Purpose
REPORT IMAGES	Inserts any or all of your captured images and animations. Move the mouse pointer over a thumbnail to see the caption that you assigned to the corresponding image or animation when you captured it. Click any thumbnail image to insert the image or animation that it represents into your report.
REPORT TABLES	Inserts a captured data table. A button appearing in this field is labeled with the first 15 characters of the caption that you assigned to the corresponding table when you captured it. Move the mouse pointer over a button to see the caption that you assigned to the corresponding table when you captured it. Click on a button to insert the table that it represents into your report.
REPORT LISTS	Inserts a captured raw-data output listing. A button appearing in this field is labeled with the first 15 characters of the caption that you assigned to the corresponding listing when you captured it. Move the mouse pointer over a thumbnail to see the caption that you assigned to the corresponding listing when you captured it. Click on a button to insert the listing that it represents into your report.

3. Preview and clean up your report, as follows:

To perform this editing task ...	Do this:
Preview your report	Click this button in the toolbar: 
Delete a report component	Select (click on) the component that you want to delete, then click on this button in the toolbar: 
Change the order of the report components	Select (click) the component that you want to move. Click either of these buttons to move the component up or down, respectively:  If your report contains a heading, it remains fixed at the top of the report.
Change the caption of a report component	Click on the caption field that you want to change to place the cursor within the text. Type your changes directly into the field.
Prevent Microsoft Internet Explorer (IE) from splitting a component between two pages during printing	Check the box labeled Insert page break for Microsoft IE for the appropriate component. This feature is especially useful <i>after</i> you have printed an initial draft of your report from within IE and determined how your printed report looks.

4. Save your work.

Select **File> Save** periodically (or **File> Save and Close** to save your report and close the report assembler window).

19.6.2. Batch Report Assembly

Issue the ***CREATE** command in batch mode to open and append HTML tags to your report file. By default, the report assembler uses the ***CREATE** command to write the report to the log file.

19.6.3. Report Assembly Using the JavaScript Interface

For maximum flexibility and usefulness, the report generator employs JavaScript, a coding language for Web browsers supported by Microsoft Internet Explorer and Netscape Navigator. JavaScript allows easy encapsulation of data and access to that data.

The report generator creates JavaScript functions (in form) as you capture image and text data. Therefore, rather than using the report generator's built-in report assembler, you can use the JavaScript functions to manually assemble your HTML-based report.

19.6.3.1. Inserting an Image

The following example JavaScript code creates an image in the HTML report that you are assembling. If the specified image is not available, your report will contain a message indicating the problem.

```
<script LANGUAGE="JavaScript1.2" SRC="ansysImages.js"> </script>
<script>
imgName('imgCaption');
</script>
```

Following is an explanation of the JavaScript code:

<script LANGUAGE="JavaScript1.2" SRC="ansysImages.js"> </script>

Loads the `ansysImages.js` file to provide the images. You must include this line of code at least once in your HTML document and *before* calling the `imgName` function. Typically, this line appears in the `<HEAD>` section of an HTML document.

<script>

The HTML tag to begin JavaScript code.

`imgName`

A unique image name as it appears in the `ansysImages.js` file.

`imgCaption`

The caption to display for the image. This value is a string and must be enclosed in single quotation marks. It can include HTML tags around the text as well. If not specified, the function uses the default image caption.

</script>

The HTML tag to end JavaScript code.

19.6.3.2. Inserting an Animation

The following example JavaScript code creates an animation sequence in the HTML report that you are assembling. If the specified animation is not available, your report will contain a message indicating the problem.

```
<script LANGUAGE="JavaScript1.2" SRC="ansysAnimations.js"> </script>
<script>
animName('animCaption',animTime, 'animDirect');
</script>
```

Following is an explanation of the JavaScript code:

<script LANGUAGE="JavaScript1.2" SRC="ansysAnimations.js"> </script>

Loads the `ansysAnimations.js` file to provide the animation sequences. You must include this line of code at least once in your HTML document and *before* calling the `animName` function. Typically, this line appears in the `<HEAD>` section of an HTML document.

<script>

The HTML tag to begin JavaScript code.

animName

A unique animation name as it appears in the `ansysAnimations.js` file.

animCaption

The caption to display for the animation sequence. This value is a string and must be enclosed in single quotation marks. It can include HTML tags around the text as well. If not specified, the function uses the default animation caption.

animTime

The time delay (in milliseconds) between each display of an individual image in the animation sequence. This value is limited by the capabilities of your computer hardware; therefore, a value lower than 500 typically has little effect on the animation. If not specified, the function assumes the default value of 500.

animDirect

The direction of play, as follows:

forward

After displaying the last individual image of the animation sequence, the function repeats the animation from the beginning (that is, starting with the first image and incrementing).

back

After displaying the last individual image of the animation sequence, the function repeats the animation in the opposite direction (that is, starting with the prior image and decrementing). After displaying the first image again, the animation repeats in the forward direction.

If not specified, the function assumes the default value of `back`.

</script>

The HTML tag to end JavaScript code.

19.6.3.3. Inserting a Data Table

The following example JavaScript code creates a data table in the HTML report that you are assembling. If the specified table is not available, your report will contain a message indicating the problem.

```
<script LANGUAGE="JavaScript1.2" SRC="ansysTables.js"> </script>
<script>
tableName('tableCaption');
</script>
```

Following is an explanation of the JavaScript code:

<script LANGUAGE="JavaScript1.2" SRC="ansysTables.js"> </script>

Loads the `ansysTables.js` file to provide the data table. You must include this line of code at least once in your HTML document and *before* calling the `tableName` function. Typically, this line appears in the `<HEAD>` section of an HTML document.

<script>

The HTML tag to begin JavaScript code.

tableName

A unique data table name as it appears in the `ansysTables.js` file.

tableCaption

The caption to display for the table. This value is a string and must be enclosed in single quotation marks. It can include HTML tags around the text as well. If not specified, the function uses the default table caption.

</script>

The HTML tag to end JavaScript code.

19.6.3.4. Inserting a Listing

The following example JavaScript code creates an output listing in the HTML report that you are assembling. If the specified listing is not available, your report will contain a message indicating the problem.

```
<script LANGUAGE="JavaScript1.2" SRC="ansysListings.js"> </script>
<script>
listingName('listingCaption');
</script>
```

Following is an explanation of the JavaScript code:

<script LANGUAGE="JavaScript1.2" SRC="ansysListings.js"> </script>

Loads the `ansysListings.js` file to provide the listing. You must include this line of code at least once in your HTML document and *before* calling the `listingName` function. Typically, this line appears in the `<HEAD>` section of an HTML document.

<script>

The HTML tag to begin JavaScript code.

listingName

A unique listing name as it appears in the `ansysListings.js` file.

listingCaption

The caption to display for the listing. This value is a string and must be enclosed in single quotation marks. It can include HTML tags around the text as well. If not specified, the function uses the default listing caption.

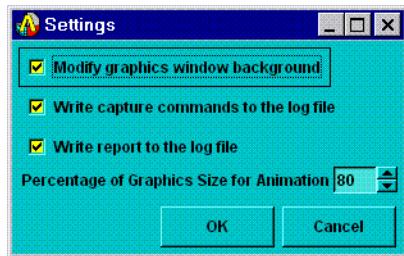
</script>

The HTML tag to end JavaScript code.

19.7. Setting Report Generator Defaults

This section describes how to change settings affecting report generator operation. Click on the **Settings** button to open the **Settings** dialog, as shown:

Figure 19.4: Report Generator Settings Dialog



You can control whether or not the report generator:

- Reverses the foreground and background colors in the **Graphics** window (and hides the background image) when you start the report generator
- Writes capture commands to the log file

- Writes your assembled report to the log file.

You can also use the **Settings** dialog to set the percentage value that the report generator uses to reduce image sizes for animations.

By default, all options are activated (that is, all check boxes contain check marks) and the report generator uses an image size of 100 percent of the **Graphics** window size for animations. Any changes that you make in the settings window become the new defaults for subsequent report generator sessions until you change them again.

Chapter 20: File Management and Files

The Mechanical APDL program uses various permanent and temporary files during an analysis. The following file-management topics are available to help you customize and manage files:

- 20.1. File Management Overview
- 20.2. Changing the Default File Name
- 20.3. Sending Output to Screens, Files, or Both
- 20.4. Text and Binary Files
- 20.5. Reading Your Own Files into the Program
- 20.6. Writing Your Own Files from the Program
- 20.7. Assigning File Names
- 20.8. Reviewing Contents of Binary Files (AUX2)
- 20.9. Operating on Results Files (AUX3)
- 20.10. Other File Management Commands

20.1. File Management Overview

The program uses files extensively for data storage and retrieval, especially when solving an analysis. The files are named `filename.ext`, where `filename` defaults to the jobname, and `ext` is a unique two- to four-character extension identifying the contents of the file.

The jobname is a name that you can specify via the **/FILNAME** command. If you specify no jobname, it defaults to `file`.

Example 20.1: File Names

If the jobname is `bolt`, you may have files at the end of an analysis which could include:

<code>bolt.db</code>	Database file
<code>bolt.emat</code>	Element matrices
<code>bolt.err</code>	Error and warning messages
<code>bolt.log</code>	Command input history
<code>bolt.rst</code>	Results file

Files generated and then deleted before the end of an analysis session are called *temporary* files. Files that remain after the session are called *permanent* files. See [Table 20.1: Program-Generated Temporary Files \(p. 291\)](#) and [Table 20.2: Program-Generated Permanent Files \(p. 292\)](#).

20.2. Changing the Default File Name

When you start the program, you can change the default jobname from `file` to a name that is more meaningful. To do so, activate the program as follows:

```
ansys211 -j newjobname
```

The value `-j` is an option indicating that a new jobname, `newjobname`, follows. After this command executes, all program files generated during the current run will have a filename of `newjobname.ext`.

If a job is running in the background, *do not execute the program interactively* in the same directory unless you use a different jobname.

The program can process blanks in file and directory names. If a blank appears in the file name, enclose the file name within single quotes.

20.3. Sending Output to Screens, Files, or Both

One of the files commonly referred to throughout the documentation is the output file (`Jobname.OUT`). If you are running on a Linux system and you want to send analysis output only to the screen, open the launcher via the `launcher211` command, select the **Preferences** tab, and select **Screen Only** for the **Send Output To** option. The output is then displayed directly to your output window. If you select **Screen and File**, an actual text file named `Jobname.OUT` is also written in your current working directory.

When you launch the program and direct output to both screen and file, the program does not immediately display output in the output window. The I/O buffer must be filled or flushed first. Errors and warnings will flush the I/O buffer. You can also issue certain commands (such as **/OUTPUT**, **NLIST**, or **KLIST**) to force-flush the I/O buffer.

Windows systems do not support the **Screen and File** option. The default behavior is to print output to the output window. You can also redirect your text output to a file (**/OUTPUT**).

20.4. Text and Binary Files

Depending on how files are used, the program writes them in text (ASCII) form or binary form. For example, `ERR` and `LOG` files are text, while `DB`, `EMAT`, and `RST` files are binary. In general, files that you may need to read (and edit) are written in text form, and all other files are written in binary form.

Below are some tips for using binary files:

- When transferring files via FTP (File Transfer Protocol), you must set the **BINARY** option before doing the transfer.
- Most binary files *must* have write permission to be used, even if the data is only being read from the file. However, the database files (`file.DB`) and results files (`file.RST`, `file.RTH`, etc.) can be read-only. When you save a read-only `file.DB`, the existing read-only file is saved to a `file.DBB`. You cannot save the read-only `file.DB` a second time, however, because it will attempt to overwrite the `file.DBB` file.

Binary files are not backward-compatible with previous releases. You cannot use binary files generated by the current release with an earlier release. Some files are forward-compatible. For a list of those files, see [Table 20.2: Program-Generated Permanent Files \(p. 292\)](#).

20.4.1. Program-Generated Files

The following tables list the files that the program writes:

Table 20.1: Program-Generated Temporary Files

Identifier	Type	Contents
ANO	Text	Graphics annotation commands (/ANNOT)
BAT	Text	Input data copied from batch input file (/BATCH)
DSPxxxx	Binary	Scratch files for the sparse solver
EROT	Binary	Rotated element matrices
EVC	Binary	Scratch file for PCG Lanczos eigensolver
EVL	Binary	Scratch file for PCG Lanczos eigensolver
LNxx	Binary	Scratch files for the sparse solver (x = 1-42)
LOCK	Binary	Prevents more than one job with the same name from running in the same directory
LV	Binary	Scratch file from substructure generation pass with more than one load vector.
PAGE	Binary	Page file for virtual memory (database space)
Pnn	Binary	Scratch files for matrix assembly procedure
PCn	Binary	Scratch files for PCG solver and matrix assembly procedure
PDA	Binary	Scratch file for PCG solver and matrix assembly procedure
PMA	Binary	Scratch file for PCG solver and matrix assembly procedure
SNODExxx	Binary	Scratch files for Supernode eigensolver
SSCR	Binary	Scratch file from substructure generation pass
vtA0Q	Binary	Scratch files for harmonic analysis using the VT method and/or certain eigensolvers
vtAQ0	Binary	Scratch files for harmonic analysis using the VT method and/or certain eigensolvers
vtAQ1	Binary	Scratch files for harmonic analysis using the VT method and/or certain eigensolvers
vtFQ	Binary	Scratch files for harmonic analysis using the VT method and/or certain eigensolvers
vtQ	Binary	Scratch files for harmonic analysis using the VT method and/or certain eigensolvers
vtWrk	Binary	Scratch files for harmonic analysis using the VT method and/or certain eigensolvers
vtdX	Binary	Scratch files for harmonic analysis using the VT method and/or certain eigensolvers

Many permanent files are upwardly compatible. Files that can generally be used by future program releases have a **Y** in the **Upward** column.

Table 20.2: Program-Generated Permanent Files

Identifier	Type	Upward	Contents
ANF	Text	Y	ANSYS Neutral Format geometry file
ASI	Binary	Y	Results file from a structural analysis on an FSI interface
BCLV	Binary	Y	Superelement static correction vectors from the generation pass
BCS	Text	-	Stores performance information when running the sparse solver
BFIN	Text	-	Interpolated body forces written as BF commands (BFINT)
CBDO	Text	-	Interpolated DOF data written as D Commands (CBDOF)
CDB	Text	Y	Text database file (CDWRITE)
CMAP	Text	-	Color map file
CMD	Text	Y	Commands written by *CFWRITE
CMS	Binary	Y	Component Mode Synthesis file
CND	Text	Y	Nonlinear diagnostics file that tracks contact quantities throughout the solution (NLDIAG)
CNM	Text	Y	Contact pair output data (CNTR)
DB	Binary	Y	Database file (SAVE , /EXIT)
DBB	Binary	Y	Copy of database file created when a nonlinear analysis terminates abnormally (used for traditional restart)
DBE	Binary	-	Database file from VMESH failure in batch mode
DSP	Text	-	Stores performance information when running the sparse solver
DSUB	Binary	-	Superelement DOF solution from use pass
DSPsymb	Binary	-	Factorized stiffness matrix (also known as the triangularized stiffness matrix) from the sparse solver
ELEM	Text	Y	Element definitions (EWRITE)
EMAT	Binary	-	Element matrices
ERR	Text	-	Error and warning messages
ESAV	Binary	-	Element saved data ESAV files created by nonlinear analyses may not be upwardly compatible
FULL	Binary	-	Assembled global stiffness and mass matrices
GST	Text	-	Graphical solution tracking file
IGES	Text	Y	IGES file from solid model data (IGESOUT)
LDHI	Text	Y	Loading and boundary conditions of load steps (used for multiframe restart)
LGW	Text	Y	Database command log file (LGWRITE)
Lnn	Binary	Y	Load case file (where nn = load case number) (LCWRITE)

Identifier	Type	Upward	Contents
LMODE	Binary	-	Modal analysis frequencies and left mode shapes (MODOPT)
LN22	Binary	-	Factorized stiffness matrix (also known as the triangularized stiffness matrix) from the sparse solver
LOG	Text	Y	Command input history
MAPPING	Text	Y	Mapping data (HBMAT)
MATRIX	Text/Binary	Y	Mapping data in Harwell-Boeing format (HBMAT)
MCF	Text	Y	Modal coordinates from harmonic or transient analysis
MCOM	Text	Y	Mode combination commands from spectrum analysis
MLV	Binary	-	Modal analysis element load vector data
M _{nnn}	Binary	Y	Modal displacements, velocities, and accelerations records and solution commands for a single substep of a load step (used for multiframe restart of a mode-superposition transient analysis)
MNTR	Text	-	Nonlinear analysis convergence monitoring
MODE	Binary	-	Modal analysis frequencies and mode shapes; buckling analysis load multipliers and mode shapes
MODESYM	Binary	-	Modal analysis frequencies and mode shapes
MP	Text	Y	Material property definitions (MPWRITE)
NDXXX	Text	-	Element IDs violating criteria
NLH	Text	Y	Nonlinear diagnostics file that tracks results or contact quantities throughout the solution (NLHIST)
NODE	Text	Y	Node definitions (NWRITE)
NRXXX	Text	Y	Stores Newton-Raphson iteration information when the nonlinear diagnostic tool is active (NLDIAG , NRRE, ON)
OSAV	Binary	-	Copy of <code>ESAV</code> file from last converged substep
OUT	Text	-	Output file
PARM	Text	Y	Parameter definitions (PARSAV)
PCS	Text	-	Stores performance information when running the PCG solver
PRS	Binary	-	Stores response spectrum (SPRS, MPRS, and DDAM) information (mode coefficients, etc.) when remote modal files are used.
PSAV	Binary	-	Copy of <code>OSAV</code> file from last substep (used only for semi-implicit analysis)
PSD	Binary	-	Stores random vibration (PSD) information (modal covariance matrices, etc.)
PVTS	Text	-	Stores pivot information when running the sparse solver
RCN	Binary	Y	Results file for the initial contact state
RDB	Binary	Y	Database at the start of the first substep of the first load step (used for multiframe restart)
RD _{nn}	Binary	Y	Database from structural analyses after nn times of rezoning
RDSP	Binary	-	Reduced displacements

Identifier	Type	Upward	Contents
RFRQ	Binary	-	Reduced complex displacements
RMF	Binary	Y	Results file from a static mean flow analysis
RMG	Binary	Y	Results file from a magnetic field analysis
RMSH	Text	-	Remeshing activity monitor file for mesh nonlinear adaptivity analysis
R _n nnn	Binary	-	Element saved records, solution commands, and status for a single substep of a load step (used for multiframe restart of static and full transient analyses)
RS _n n	Binary	Y	Results file from structural analyses after nn times of rezoning
RSM	Binary	-	Radiosity mapping data file for radiosity surface elements (SURF251 , SURF252)
RST	Binary	Y	Results file from structural and coupled-field analyses
RSTP	Binary	Y	Results file from a linear perturbation analysis
RTH	Binary	Y	Results file from a thermal analysis
SECF	Text	-	Stores output quantities associated with result sections (OUTPR , RSFO)
SELD	Binary	-	Superelement load vector data from generation pass
S _n n	Text	Y	Load step files (where nn = load step number) (LSWRITE)
SORD	Text	-	Superelement name and number from use pass
STAT	Text	-	Status of a batch run
SUB	Binary	Y	Superelement matrix file from generation pass
TB	Text	Y	Hyperelastic material constants
USUB	Binary	Y	Renamed DSUB File for input to substructure expansion pass

20.4.2. File Compression

Many file compression utilities exist for Linux (such as **gzip**) and Windows (such as **WinZip**). The program cannot read compressed files. You can compress your models to save space when archiving, so long as you uncompress them before trying to read them into the program.

The **/FCOMP** command enables you to control compression of the results files, database files, certain restart files (.Rnnn), and certain files created during a nonlinear analysis (.osav) while they are being written. This command offers two different ways to do compression:

- Compression based on a sparsification scheme (default).

The average file compression is 10-50 percent. The solution time and time reading the results file or database file are not significantly increased and may decrease due to reduced I/O size requirements.

- Compression based on the zlib algorithm; you control the level of compression by inputting a number from 0 (no compression) to 5.

The average file will compress about 25 percent at the cost of added solution time (more so if running in distributed-memory parallel mode) or time to save the database. The time to read the results (for example, the **SET** command) or resume the database is also increased.

Compressing the files is useful for speeding up file transfers (for example when solving on a network cluster) or reducing the disk space required to archive the analysis data.

Note that results files from modal analyses will compress very little if the stresses are written to the MODE file (*MSUPkey* = YES on **MXPAND**).

20.5. Reading Your Own Files into the Program

In many situations, you will need to read in your own files. The file may be a text file of commands or a binary data file.

Issue the **/INPUT** command to read in a text file containing commands (such as *Jobname.LOG* or *Camshaft_Assembly.INP*).

Example 20.2: Reading a Text File

This command reads the file *MATERIAL.INP* from the current directory:

```
/INPUT,MATERIAL,INP
```

Table 20.3: Commands for Reading in Special-Purpose Text Files

This command...	Reads in this type of text file:
*USE	Macros
PARRES	Parameter (<i>Jobname.PARM</i>)
ERead	Element (<i>Jobname.ELEM</i>)
NREAD	Node (<i>Jobname.NODE</i>)
MPREAD	Material property (<i>Jobname.MP</i>)
INISTATE	Initial state (<i>Jobname.IST</i>)

Table 20.4: Commands for Reading in Binary Files

This command...	Reads in this type of binary file:
RESUME	Database (<i>Jobname.DB</i>)
SET [1]	Results (<i>Jobname.RST</i> , <i>Jobname.RTH</i> , <i>Jobname.RMG</i>)

1. in the POST1 postprocessor

20.6. Writing Your Own Files from the Program

Besides the files generated automatically during an analysis, you can force files to be written as necessary. A commonly used file-write command is **/OUTPUT**, used to redirect text output from the screen to a file.

Example 20.3: Forcing an Output File to Be Written

These commands redirect POST1 stress printout to a file:

```
/OUTPUT,STRESS,OUT! Output to file STRESS.OUT  
PRNSOL,COMP! Component stresses  
/OUTPUT! Output back to screen
```

Table 20.5: Other Commands for Writing Files

Command	Purpose
SAVE	Writes the database to Jobname.DB
PARSAV	Writes parameters to Jobname.PARM
EWRITE	Writes element definitions to Jobname.ELEM
NWRITE	Writes node definitions to Jobname.NODE
MPWRITE	Writes material properties to Jobname.MP
/SHOW	Writes graphics output to a standard graphic file format

You can also redirect graphics output (plots) from the screen to a neutral graphics file.

20.7. Assigning File Names

You can use the **/FILNAME** command at the Begin level to assign a jobname for all subsequently written files.

Issue the **/ASSIGN** command to assign a different name, extension, and directory to a file.

Example 20.4: Assigning File Names

This command reassigns the element matrix file (identifier EMAT) to MYFILE.DAT in the SAVE_DIR directory:

```
/ASSIGN,EMAT,SAVE_DIR/MYFILE,DAT,SAVE_DIR/
```

The "/" is a delimiter separating the directory name (SAVE_DIR) from the file name (MYFILE). The delimiter is system-dependent; therefore, use the delimiter(s) appropriate for your system.

You can assign only a specific set of files. See the **/ASSIGN** command description for the complete list.

20.8. Reviewing Contents of Binary Files (AUX2)

The AUX2 auxiliary processor (**/AUX2**) enables you to print binary files in a readable format. Use the processor primarily to verify file formats (for debugging purposes).

The output from a dumped binary file is unlabeled and must be correlated with known formats (documented in the [Guide to Interfacing with ANSYS](#).)

A complete binary file dump can produce many pages of printout. The *Format* argument on the **FORM** command enables you to control the amount of output.

Issue the **HBMAT** command to dump any matrix written on the assembled global matrix file (.FULL file) or the superelement matrix file (.SUB file). The matrix is written to a new file (.MATRIX) in the standard Harwell-Boeing format. (The Harwell-Boeing format is column-oriented; that is, non-zero matrix values are stored with their corresponding row indices in a sequence of columns. Because matrix files are stored by row and not column, the transpose of the matrix is actually written when the **HBMAT** command is used with a non-symmetric matrix.)

Issue the **PSMAT** command to write a postscript file containing a graphic representation of any matrix on the .FULL file. The matrix is symbolized by a grid in which colored cells represent the nonzero coefficients of the matrix.

20.9. Operating on Results Files (AUX3)

The AUX3 auxiliary processor (**/AUX3**) enables you to operate on results files by deleting sets or by changing values such as the load step, load substep, cumulative iteration, or time.

20.10. Other File Management Commands

Table 20.6: Special-Purpose File Management Commands

Command	Purpose
/COPY	Copy existing binary files from within the program
/CLOG	Copy the log file during an interactive session
/RENAME	Rename files
/DELETE	Delete files
/FDELE	Delete specific files during a solution run (to save disk space)

Chapter 21:Memory Management and Configuration

In order to maximize performance on your particular system, it is helpful to understand the memory-management scheme and some frequently used terms concerning computer memory. However, there is usually no need to concern yourself with memory-management issues. The program's memory manager allocates extra memory from the system when it needs to *in almost all cases*.

The following memory-management topics are available:

- 21.1. Work and Swap Space Requirements
- 21.2. How the Program Uses Work Space
- 21.3. How and When to Perform Memory Management
- 21.4. Using the Configuration File
- 21.5. Understanding Memory Error Messages

To learn how to improve analysis performance, see the *Parallel Processing Guide* and the *Performance Guide*.

21.1. Work and Swap Space Requirements

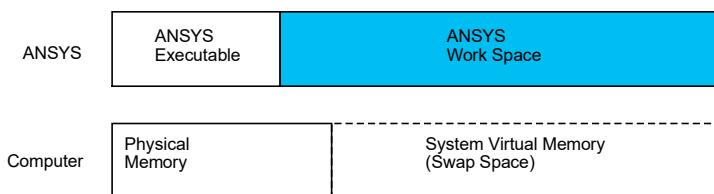
The Mechanical APDL program requires memory in which to reside and memory to use as work space. The work space default is 2 GB (2048 MB) on Linux and Windows systems.

As shown in [Figure 21.1: Comparing Available Memory \(p. 299\)](#), the total memory required for the program can exceed the amount of physical memory available. The additional memory comes from *system virtual memory*, which is simply a portion of a computer's hard disk drive used by the system to supplement physical memory.

The disk space used for system virtual memory is called *swap space*, and its associated hidden system file is called the *swap file*. (sometimes called a *page file*). Other systems maintain multiple files or even dedicated disk sectors for virtual memory. The amount of swap space required for the program depends mainly on the amount of physical memory available and the amount of work space allocated.

Because hard disk drives are much slower than system memory, and writing data from memory to disk (or reading data from disk to memory) incurs significant overhead, it is best to avoid virtual memory usage insofar as possible; otherwise, performance is severely degraded.

Figure 21.1: Comparing Available Memory



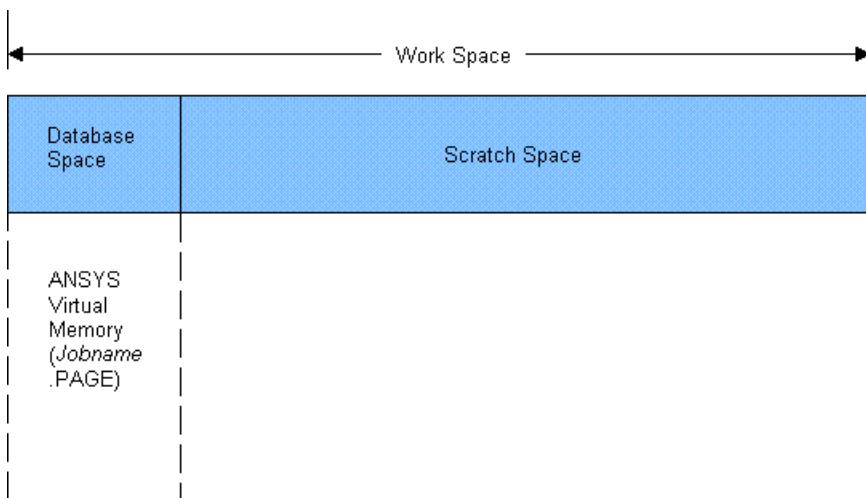
System virtual memory is used to satisfy additional memory requirements.

21.2. How the Program Uses Work Space

The program uses its work space (the shaded portion in [Figure 21.1: Comparing Available Memory \(p. 299\)](#)) via two components: *database space* and *scratch space*, as shown in [Figure 21.2: Work Space \(p. 300\)](#). Database space works with the program database (model geometry, material properties, loads, etc.). Scratch space is where all internal calculations are done (such as element matrix formulation, equation solution, and Boolean calculations).

The default work space is 2 GB (2048 MB), of which half is assigned to database space and half to scratch space. These sizes represent the amount of memory that is allocated upon program startup for the database and scratch spaces.

Figure 21.2: Work Space



If your model database is too big to fit in the initial database space, the program attempts to allocate additional memory to hold it. If it cannot, the program uses *Mechanical APDL virtual memory*, which is a file written by the program that is used for data overflow. The primary difference between *system virtual memory* and *Mechanical APDL virtual memory* is that the former uses system functions to swap data between memory and disk, whereas the latter uses Mechanical APDL programming instructions.

The file used for Mechanical APDL virtual memory is called the *page file* and has the name `Jobname.PAGE`. Its size depends entirely on the size of the database. When the page file is first written, the program issues a message to that effect. Use of the page file is not desirable because it is a less efficient way of processing data. You may be able to prevent it by allocating more database space (discussed in [How and When to Perform Memory Management \(p. 301\)](#)).

If internal calculations are unable to fit within the initial scratch space, the program attempts to allocate additional memory to meet the requirements. If the program is successful, an alert message appears indicating that additional memory has been allocated.

In general, you should have enough physical memory to comfortably run an analysis job. If you are using virtual memory only temporarily or using a relatively small amount of virtual memory, the performance degradation is typically small. Using a significant amount of virtual memory, particularly during solution, can degrade performance by as much as a factor of 10.

21.3. How and When to Perform Memory Management

The following topics provide guidance as to when the `-m` command line option may be necessary:

- [21.3.1. Determining When to Change the Work Space](#)
- [21.3.2. Changing the Amount of Work Space](#)
- [21.3.3. Changing the Amount of Database Space](#)

21.3.1. Determining When to Change the Work Space

The `-m` command line option allows you to manually set the size of the initial block of memory used by ANSYS. Memory allocated upon startup via the `-m` option exists in one contiguous block. For example, a `-m` setting of 1800 with a `-db` option of 300 instructs the program to allocate an 1800 MB contiguous block of memory with a 300 MB contiguous block reserved for the database and a 1500 MB contiguous block reserved for scratch memory ($1800 - 300 = 1500$).

The current defaults are `-m = 2048` and `-db = 1024`. Ideally, all program memory will be allocated from within the initial block, allowing efficient reuse of memory blocks during various phases of simulation. When the program needs more memory, it will allocate from the system, automatically growing new blocks that are half the size of the initial scratch memory block or the size of the new memory block allocation, whichever is larger.

One reason to change the default memory settings is when a job fails due to insufficient memory that may be caused by fragmented memory. For example, if a large model requires a contiguous block of 1600 MB for the sparse solver, the default memory allocation is insufficient (`-m 2048` MB minus `-db 1024` MB = 1024 MB contiguous memory). In this case, the program tries to allocate an additional 1600 MB block of contiguous memory to satisfy the sparse solver requirement, bringing the total memory requirement to 3648 MB (2048 default plus 1600 additional). This memory request may fail on smaller systems. To accommodate this model within the default memory availability, specify `-db -100`. (Using a negative value prevents the program from allocating additional memory for the database.) The result is an initial memory block of 1948 MB, sufficient to satisfy the sparse solver requirement of 1600 MB.

In some cases, you may want to specifically control how much memory the program can allocate from the system, useful in a multi-user environment where resources such as physical memory are being shared by multiple running analysis jobs. You can set a fixed-memory mode by specifying a negative `-m` value. When the fixed memory mode is used, the initial database and scratch spaces are set per the `-m` (and `-db`) sizes; however, the program is constrained so that it cannot grow any additional memory (for scratch or database). With this option, the program will fail if the memory required by the program exceeds the initial block reserved at startup.

21.3.2. Changing the Amount of Work Space

The easiest way to change the amount of program work space is to use the work space entry option (`-m`) while activating the program, either via the launcher or via the program execution command.

Example 21.1: Setting the Program Work Space at Startup

To request 1400 MB of program work space (instead of the 2 GB default), issue this command:

```
ansys211 -m 1400
```

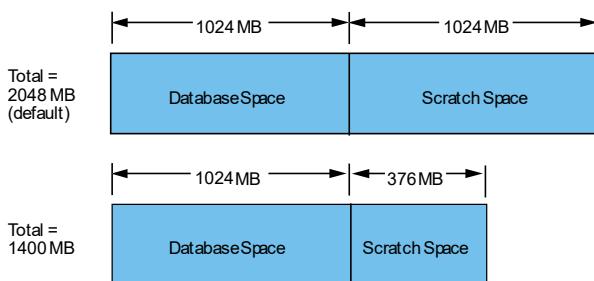
Other ways to change the work space are:

- Specifying the work space size you want on the dialog boxes that appear when you select interactive mode or batch mode from the program launcher.
- Using a different VIRTM_MB value in your config.ans file. A later section in this chapter discusses this file in detail.

Use caution when specifying a value for the `-m` option. Entering an amount larger than needed wastes system resources and can degrade system performance.

Given a fixed amount of database space, changing the amount of work space changes the available scratch space but holds the database space constant:

Figure 21.3: Changing Work Space



21.3.3. Changing the Amount of Database Space

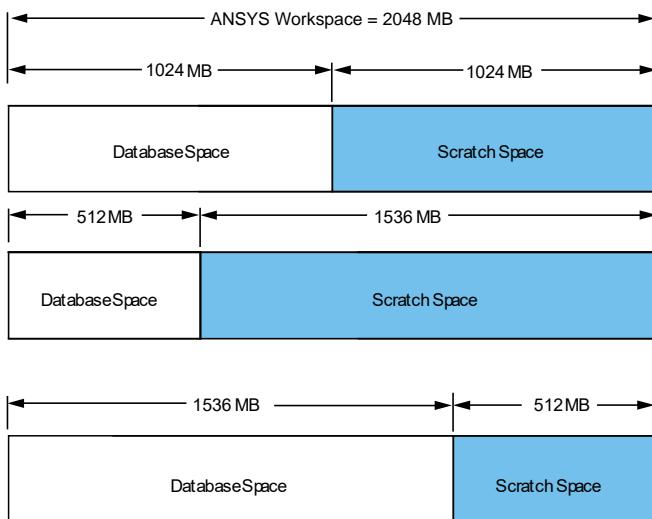
The easiest way to change the amount of database space is to use the database space entry option (`-db`) while activating the program, either via the launcher or via the program execution command.

Example 21.2: Setting the Database Space at Startup

To request 200 MB of database space (instead of the 1024 MB default), issue this command:

```
ansys211 -db 200
```

Given a fixed amount of program work space, allocating more database space leaves less for scratch space and vice versa, as shown:

Figure 21.4: Dividing Work Space

Although all of the above diagrams correspond to a `-m` value of 2048 MB, their `-db` values correspond to 1024, 512, and 1536 MB, respectively.

When you are about to solve a large model and the memory requirements are approaching your system's memory limits, you may need to control the amount of database space differently. Run the solution phase as a batch job with minimal `-db` space (for example, 64 MB). To insure enough scratch space, use a negative value (for example, `-db -64`) to prevent the program from allocating additional memory required to hold the database. By reducing the amount of database space, the amount of memory available for scratch space during solution is increased. Before postprocessing, restart the program with an increased `-db` value and resume the `jobname.db` file.

21.4. Using the Configuration File

When you execute the program, it reads a configuration file, `config.ans`, if one exists. The file controls system-dependent settings such as the size of each file buffer, maximum number of database pages in memory, etc.

The program searches for the `config.ans` file first in the current (working) directory, next in the login (home) directory, and finally in the `/apdl` directory. The search path for `config.ans` is identical to the `start.ans` and `stop.ans` files.

The configuration file is a fixed-format file, consisting of a list of keywords followed by an equal (=) sign and a number. The keyword must begin in column 1, the equal sign must be in column 9, and the number must begin in column 10.

Example 21.3: config.ans File

```
NO_RSTGM=1
NO_ELDBW=0
NUM_BUFR=2
SIZE_BIO=4096
VIRTM_MB=2048
NUM_VPAG=8192
SIZ_VPAG=16384
NUM_DPAG=16777212
MEM_GROW=12
LOCALFIL=0
```

```

CONTACTS=1000
ORDERER_=2
MX_NODES=5000
MX_ELEMS=2000
MX_KEYPT=500
MX_LINES=1000
MX.Areas=300
MX_VOLUS=200
MX_REALS=10
MX_COUPS=10
MX_CEQNS=10
FILESPLT=128
GRW_NBUF=1
MEBA_LIC=0

```

Following are descriptions of each keyword in the example. Because many of the values for config.ans are dependent on the system being used, a range of values for each keyword is provided. On most computer systems, 1 integer word = 4 bytes.

NO_RSTGM

Key that determines whether or not the geometry data will be written to the results file: 0 (write the data) or 1 (do not write the data). This is especially useful for large, complex analyses where the results file would become excessively large during the solution.

NO_ELDBW

Key that determines whether or not to write results into the database after a solution. When *VAL* = 0 (default), write results into the database. When *VAL* = 1, do not write results into the database. This key can be changed at the Begin level via the [**/CONFIG,NOELDB**](#) command.

NUM_BUFR

Number of buffers per file stored in scratch space: 1 to 32. A buffer is a chunk of space used to "hold" data in memory before they are written to the hard disk. The program waits for the buffer to be completely "filled up" and only then "empties" it onto the hard disk. This prevents frequent disk input-output activity, which can be time consuming.

NUM_BUFR

Defaults to 4 and can be changed at the Begin level with the [**/CONFIG,NBUF**](#) command. It is used for the EROT, ESAVE, EMAT, and FULL files. On systems with a large amount of real memory, you can increase NUM_BUFR or SIZE_BIO (or both) to keep the program solution files in memory rather than on disk. This can save a significant amount of disk input-output activity and may be practical for small problems with many substeps.

SIZE_BIO

Size of each file buffer: 1024 to 41943041073741824 integer words (4 KB to 3814 TB). It defaults to 16384 and can be changed (on most systems) at the Begin level with the [**/CONFIG,SZBIO**](#) command. See NUM_BUFR for details.

VIRTM_MB

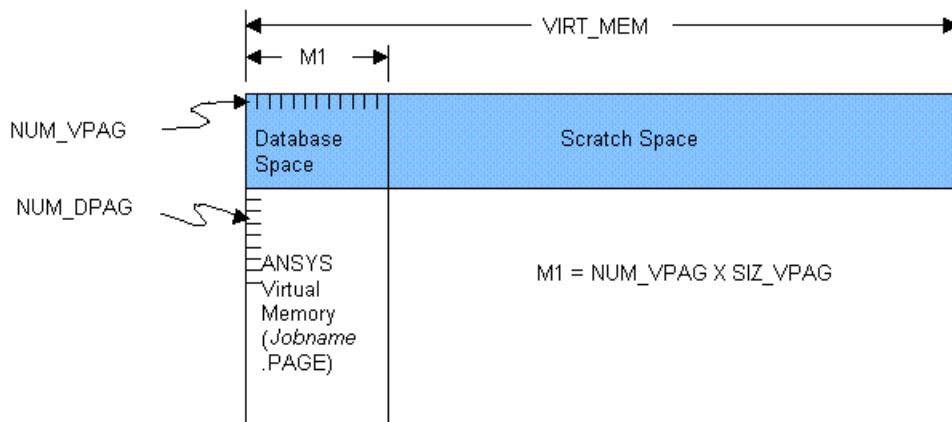
Amount of total program work space requested for the current session. It defaults to 2 GB (2048 MB); it can be changed with the work space entry option, as explained in the basic concepts section

earlier in this chapter. (See [Figure 21.5: Memory Diagram in Terms of Configuration Keywords \(p. 305\)](#).) You can also use VIRT_MEM in place of VIRT_M_B to specify the work space in integer words.

NUM_VPAG

Maximum number of database pages in memory, ranging from 16 to 16384. The default value is 16384. You can change NUM_VPAG or SIZ_VPAG (or both) to change the amount of database space; see [How and When to Perform Memory Management \(p. 301\)](#) in this chapter, and [Figure 21.5: Memory Diagram in Terms of Configuration Keywords \(p. 305\)](#).

Figure 21.5: Memory Diagram in Terms of Configuration Keywords



SIZ_VPAG

Size of each database page: 4096 to 4194304 integer words (16 KB to 16 MB), defaults to 16384 (64 KB). You can change SIZ_VPAG or NUM_VPAG (or both) to change the amount of database space. SIZ_VPAG also affects the size of the page file; see NUM_DPAG.

NUM_DPAG

Number of database pages on disk: NUM_DPAG defaults to 16777212 (the maximum). This number times SIZ_VPAG determines the maximum size of the page file (*Jobname . PAGE*), which is written only if the database becomes too large to fit in the database space in memory. If the page file is written, sufficient disk space must be available to accommodate it, or the program will abort.

MEM_GROW

Starting size of the memory block (in MB) that the program will attempt to allocate should a problem grow larger than will fit in the current scratch space allocation. If the program attempts to allocate additional scratch space, it will start with a memory block size equal to MEM_GROW and then reduce this by halves until it can allocate additional memory. If not specified, MEM_GROW defaults to one-half the initial scratch space. To turn off dynamic memory allocation (use a fixed-memory model), set MEM_GROW=0.

LOCALFIL

Key that determines when files are to be closed: 0 (globally closed) or 1 (locally closed). It defaults to 0 (globally closed) and can be changed at the Begin level with the **/CONFIG,LOCFL** command. This key is applicable only to the EROT, ESAV, EMAT, and FULL files. Locally closed files (LOCALFIL=1) may be deleted earlier during solution if requested with the **/FDELETE** command. This may be helpful

while running large problems. Globally closed files are closed at the end of the run and are not opened and closed each substep. This saves time in analyses with many substeps.

CONTACTS

Number of contact elements that are expected to be in contact at any given time. It defaults to 1000 and can be changed at the Begin level with the **/CONFIG,NCONT** command. This is *not* the same as the total number of contact elements in the model.

Note:

Any of the following nine MX_ keywords that you do not specify is set to 100 the first time the program encounters it. Whenever the current maximum is exceeded, the keyword value automatically doubles. The maximum values are dynamically expanded, even at first encounter.

MX_NODES

Maximum number of nodes. You can change the value (on most systems) at the Begin level with the **/CONFIG,MXND** command.

MX_ELEMS

Maximum number of elements. You can change the value (on most systems) at the Begin level with the **/CONFIG,MXEL** command.

MX_KEYPT

Maximum number of keypoints. You can change the value (on most systems) at the Begin level with the **/CONFIG,MXKP** command.

MX_LINES

Maximum number of lines. You can change the value (on most systems) at the Begin level with the **/CONFIG,MXLS** command.

MX.Areas

Maximum number of areas. You can change the value (on most systems) at the Begin level with the **/CONFIG,MXAR** command.

MX_VOLUS

Maximum number of volumes. You can change the value (on most systems) at the Begin level with the **/CONFIG,MXVL** command.

MX_REALS

Maximum number of sets of real constants (element attributes). You can change the value (on most systems) at the Begin level with the **/CONFIG,MXRL** command.

MX_COUPS

Maximum number of sets of coupled degrees of freedom. You can change the value (on most systems) at the Begin level with the **/CONFIG,MXCP** command.

MX_CEQNS

Maximum number of constraint equations. You can change the value (on most systems) at the Begin level with the **/CONFIG, MXCE** command.

FILESPLT

Integer value indicating the file split point in megawords. A megaword is 1024*1024 4-byte words, or 4 MB. All files that are eligible for splitting will be split into a new file every increment of xxxx megawords (where xxxx is the value specified with this keyword). You can change the value (on most systems) at the Begin level with the **/CONFIG, FSPLIT** command. For more information about file splitting, see [Splitting Files Across File Partitions in the *Operations Guide*](#).

GRW_NBUF

Key that determines whether or not the number of file buffers for each binary file (.ESAV, .EMAT, .FULL, and so on) will automatically grow as needed. By default (GRW_NBUF = 0), the logic is program-controlled and the number of file buffers may or may not grow automatically. When GRW_NBUF = 1, the number of file buffers automatically grows for most binary files to reduce the amount of I/O. This option may require a significantly greater amount of memory than the default setting. Also, when GRW_NBUF = 1, the number of file buffers for the results file will not grow, and files written by the sparse and PCG equation solvers are not affected. When GRW_NBUF = -1, the number of file buffers does not grow automatically for any file.

MEBA_LIC

Key to control automatic checkout of a Mechanical batch license during solve when the capability is not enabled, useful for PrepPost sessions. When MEBA_LIC = 0 (default), automatic license checkout occurs. When MEBA_LIC = 1, automatic license checkout is bypassed. This key can be changed at the Begin level via the **/CONFIG, MEBA_LIC** command.

You can change many of the configuration settings at program start-up using entry options, at the Begin level with the **/CONFIG** command, or with other the program commands. In most cases, therefore, there is no need to create your own config.ans file. Also, the default settings for each computer system have been chosen for efficient running of typical models on typical system configurations. Change them only for atypical models or atypical systems.

Distributed ANSYS

Distributed ANSYS sets the following keywords: NO_ELDBW = 1; NO_RSTGM = 0; FSPLIT = the default value. The NO_ELDBW, NO_RSTGM, and FSPLIT options cannot be changed when using Distributed ANSYS.

21.5. Understanding Memory Error Messages

This model requires more scratch space than available. ANSYS has currently allocated YY MB and was not able to allocate enough additional memory in order to proceed. Please increase the virtual memory on your system, and/or increase the work space memory and rerun ANSYS. Problem terminated.

(Sparse and Block Lanczos solvers only): There is not enough memory for the <Sparse or Block Lanczos> solver to proceed. Please increase the virtual memory on your system and/or

increase the work space memory and re-run ANSYS. Memory currently allocated for the <Sparse or Block Lanczos> solver = YY MB.

(Sparse and Block Lanczos solvers only): There is not enough memory for the <Sparse or Block Lanczos> solver to proceed. Please increase the virtual memory on your system and/or increase the work space memory and re-run ANSYS. Memory currently allocated for the <Sparse or Block Lanczos> solver = YY MB. Memory currently required for the <Sparse or Block Lanczos> solver to continue = YYY MB.

(Distributed Sparse and Block Lanczos solvers only): There is not enough memory for the Distributed <Sparse or Block Lanczos> solver to proceed. Please increase the virtual memory on your system and/or increase the work space memory and re-run ANSYS. Memory currently allocated by ANSYS = YY MB. Memory allocation attempted = YYY MB. Largest block of ANSYS memory available for the Distributed <Sparse or Block Lanczos> solver = YYYY MB.

The messages listed above may occur while you are running Mechanical APDL.

This type of message appears when you are running the program in dynamic memory mode and the program has attempted to allocate additional memory, but it failed because it could not find a large enough contiguous block of memory to proceed.

Specifying a higher -m value on the program execution command and executing the program again may help. However, if a large enough block of memory is still unavailable, changing the -m value will not help. You can also try decreasing the database size to help free up more work space, allowing you to continue with the analysis. Increasing the system virtual memory may also help. Try increasing the system virtual memory so that the physical memory (RAM) plus the virtual memory comfortably exceeds the memory available at the failure point (shown in the error message).

For more information, see [Memory Management and Configuration \(p. 299\)](#).

The memory (-m) size requested is not currently available. Reenter ANSYS command line with less memory requested.

The database (-db) space requested is not currently available. Reenter ANSYS command line with less database space requested.

Either message may occur at program startup.

The memory used by the program resides within the system virtual memory. The required system virtual memory for the amount of memory that you requested for either the work space (-m) or database space (-db) for the program (via either the program command line or the configuration file) is not currently available. Request less work space or database space if possible and execute the program again. If the initial requested memory is required, then wait until sufficient system virtual memory is free and try again.

Another alternative is to increase the system's virtual memory (use the **System** icon in the **Control Panel** in Windows).

The memory (-m) size requested cannot currently be addressed using dynamic memory mode. ANSYS addressing can be changed by turning on fixed memory via the -f command line option.

This message may occur at startup.

Include the `-f` command line option on the program execution command to remedy the problem.

For more information, see [Memory Management and Configuration \(p. 299\)](#).

This model requires more scratch space than ANSYS can address in dynamic memory mode. ANSYS addressing can be changed by turning on fixed memory via the `-f` command line option. Problem terminated.

This message may occur while you are running Mechanical APDL.

Include the `-f` command line option on the program execution command to remedy the problem.

For more information, see [Memory Management and Configuration \(p. 299\)](#).

This model requires more scratch space than available, currently XX words (YY MB). The scratch space may be increased by increasing the work space, currently XX words (YY MB), via the ANSYS command line memory option. Problem terminated.

This message may occur while you are running Mechanical APDL.

If you are running the program in dynamic memory mode and the internal calculations that the program is trying to perform cannot fit in the scratch space, the program will *attempt* to allocate additional memory to meet the requirements. However, this message may still appear if:

- Some portions of the program cannot use the additional memory that was allocated or allocate the memory when it is needed. The processing that the program was trying to perform when this message appeared took place in one of these portions.
- You used the `-f` command line switch so that the program would use a fixed-mode memory addressing scheme. Thus, no dynamic memory allocation is allowed.

If you receive this message, specify a higher `-m` value on the program execution command or change your `-f` command line option and execute the program again.

For more information, see [Memory Management and Configuration \(p. 299\)](#).

