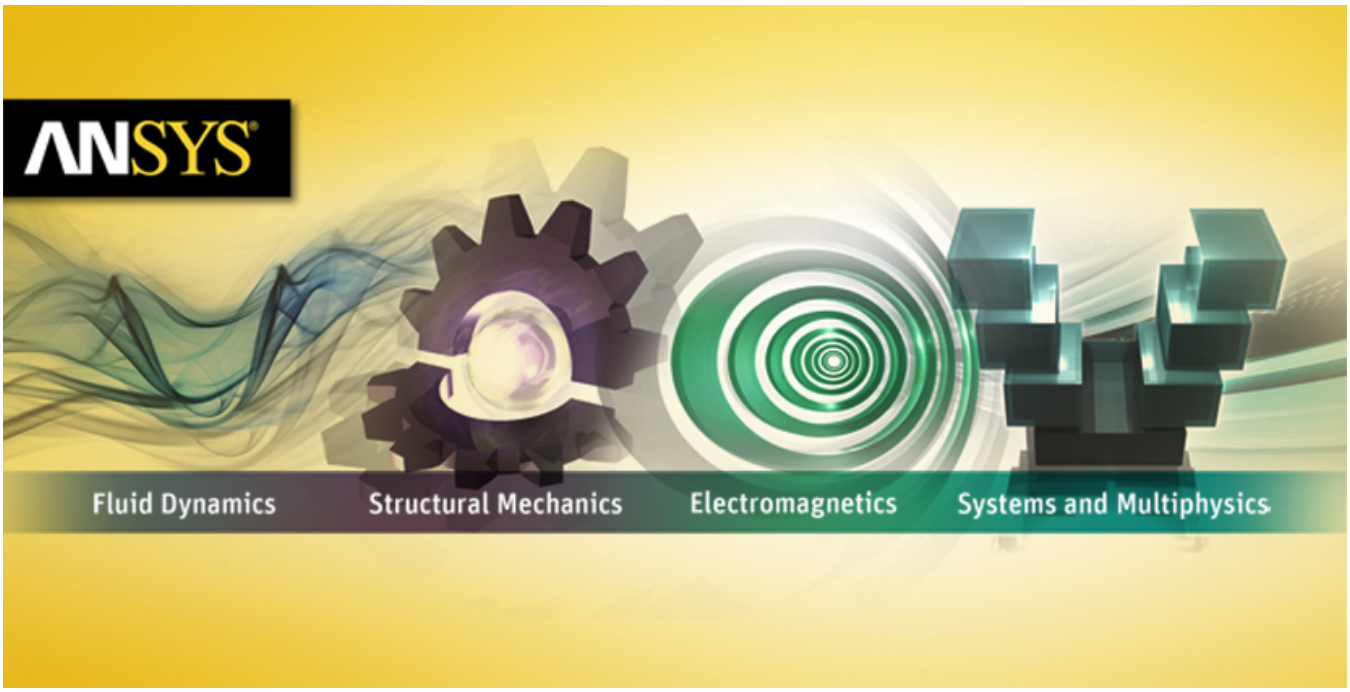


ANSYS Mechanical APDL Modeling and Meshing Guide

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

Release 15.0
November 2013

ANSYS, Inc. is certified to ISO 9001:2008.

Copyright and Trademark Information

Disclaimer Notice

U.S. Government Rights

Third-Party Software

Published in the U.S.A.

Table of Contents

List of Figures

List of Tables

Chapter 1: Understanding Model Generation

The ultimate purpose of a finite element analysis is to recreate mathematically the behavior of an actual engineering system. In other words, the analysis must be an accurate mathematical *model* of a physical prototype. In the broadest sense, this model comprises all the nodes, elements, material properties, real constants, boundary conditions, and other features that are used to represent the physical system.

The following model-generation topics are available:

1.1. What Is Model Generation?

In ANSYS terminology, *model generation* usually takes on the narrower meaning of generating the nodes and elements that represent the spatial volume and connectivity of the actual system. Thus, *model generation* in this discussion means the process of *defining the geometric configuration of the model's nodes and elements.* The ANSYS program offers the following approaches to model generation:

- Creating a solid model within ANSYS.
- Using direct generation.
- Importing a model created in a computer-aided design (CAD) system.

1.2. Typical Steps Involved in Model Generation Within ANSYS

A common modeling session might follow this general outline (detailed information on italicized subjects can be found elsewhere in this guide):

- Begin by *planning your approach*. Determine your objectives, decide what basic form your model will take, choose appropriate element types, and consider how you will establish an appropriate mesh density. You will typically do this general planning before you initiate your ANSYS session.
- Enter the preprocessor (PREP7) to initiate your model-building session. Most often, you will build your model using *solid modeling* procedures.
- Establish a *working plane*.
- Generate basic geometric features using *geometric primitives* and *Boolean operators*.
- Activate the appropriate *coordinate system*.
- Generate other solid model features from the *bottom up*. That is, create *keypoints*, and then define *lines*, *areas*, and *volumes* as needed.
- Use more *Boolean operators* or *number controls* to join separate solid model regions together as appropriate.

- Create tables of element attributes (*element types*, *real constants*, *material properties*, and *element coordinate systems*).
- Set *element attribute pointers*.
- Set *meshing controls* to establish your desired mesh density if desired. This step is not always required because default element sizes exist when you enter the program (see Generating the Mesh (p. 101)). (If you want the program to refine the mesh automatically, exit the preprocessor at this point, and activate *adaptive meshing*.)
- Create nodes and elements by *meshing* your solid model.
- After you have generated nodes and elements, add features such as *surface-to-surface contact elements*, *coupled degrees of freedom*, and *constraint equations*.
- Save your model data to `Jobname.DB`.
- Exit the preprocessor.

Note

The solid modeling features of ANSYS are known to have robustness issues. By careful planning and use of alternative strategies, you can successfully create the model required for analysis. However, you may be better served using your CAD modeler to create your model or using DesignModeler under the ANSYS Workbench environment to create your model.

1.2.1. Comparing Solid Modeling and Direct Generation

You can use two different methods to generate your model: *solid modeling* and *direct generation*. With *solid modeling*, you describe the geometric boundaries of your model, establish controls over the size and desired shape of your elements, and then instruct the ANSYS program to generate all the nodes and elements automatically. By contrast, with the *direct generation* method, you determine the location of every node and the size, shape, and connectivity of every element prior to defining these entities in your ANSYS model.

Although some automatic data generation is possible, the direct generation method is essentially a hands-on, "manual" method that requires you to keep track of all your node numbers as you develop your finite element mesh. This detailed bookkeeping can become tedious for large models, contributing to the potential for modeling errors. Solid modeling is usually more powerful and versatile than direct generation, and is commonly the preferred method for generating your model.

In spite of the many advantages of solid modeling, you might occasionally encounter circumstances where direct generation will be more useful. You can easily switch back and forth between direct generation and solid modeling, using the different techniques as appropriate to define different parts of your model.

Detailed discussions of solid modeling and direct generation can be found in Solid Modeling (p. 33) and Direct Generation (p. 201), respectively. To help you judge which method might be more suitable for a given situation, the relative advantages and disadvantages of the two approaches are summarized here.

1.2.1.1. Solid Modeling

On the plus side, solid modeling

- Is generally more appropriate for large or complex models, especially 3-D models of solid volumes.
- Allows you to work with a relatively small number of data items.
- Allows geometric operations (such as dragging and rotations) that cannot be done with nodes and elements.
- Supports the use of "primitive" areas and volumes (such as polygonal areas and cylindrical volumes) and Boolean operations (intersections, subtractions, etc.) for "top down" construction of your model.
- Is *required* for adaptive meshing.
- Is required in order to do area mesh refinement after loads have been applied (solid model loads are also required).
- Readily allows modifications to geometry.
- Facilitates changes to element distribution; you are not bound to one analysis model.

However, solid modeling

- Can sometimes require large amounts of CPU time.
- Can (for small, simple models) sometimes be more cumbersome, requiring more data entries than direct generation.
- Can sometimes "fail" (the program will not be able to generate the finite element mesh) under certain circumstances.

1.2.1.2. Direct Generation

On the plus side, direct generation

- Is convenient for small or simple models.
- Provides you with complete control over the geometry and numbering of every node and every element.

However, direct generation

- Is usually too time consuming for all but the simplest models; the volume of data you must work with can become overwhelming.
- Cannot be used with adaptive meshing.
- Makes it difficult to modify the mesh (tools such as area mesh refinement, SmartSizing, etc. cannot be used).
- Can become tedious, requiring you to pay more attention to every detail of your mesh; tedium can sometimes cause you to become more prone to committing errors.

1.3. Importing Solid Models Created in CAD systems

As an alternative to creating your solid models within ANSYS, you can create them in your favorite CAD system and then import them into ANSYS for analysis, by saving them in the IGES file format or in a file format supported by an ANSYS Connection product. Creating a model using a CAD package has the following advantages:

- You avoid a duplication of effort by using existing CAD models to generate solid models for analysis.
- You use more familiar tools to create models.

However, models imported from CAD systems may require extensive repair if they are not of suitable quality for meshing.

For more information on importing solid models from IGES files, see Importing Solid Models from IGES Files (p. 97). For more information on importing solid models from other types of files, see the *Connection User's Guide*.

Chapter 2: Planning Your Approach

As you begin to create your model, you will make a number of decisions that determine how you will mathematically simulate the physical system. For example: What are the objectives of your analysis? Will you model all, or just a portion, of the physical system? How much detail will you include in your model? What kinds of elements will you use? How dense should your finite element mesh be?

In general, you will attempt to balance computational expense (such as CPU time) against precision of results as you answer these questions. The decisions you make in the planning stage of your analysis will largely govern the success or failure of your analysis efforts.

This first step of your analysis relies not on the capabilities in the ANSYS program, but on your own education, experience, and professional judgment. Only you can determine what the objectives of your analysis must be. The objectives you establish at the start will influence the remainder of your choices as you generate the model.

The following model-planning topics are available:

2.1. Choosing a Model Type (2-D, 3-D, etc.)

Your finite element model may be categorized as being 2-D or 3-D, and as being composed of point elements, line elements, area elements, or solid elements. Of course, you can intermix different kinds of elements as required (taking care to maintain the appropriate compatibility among degrees of freedom). For example, you might model a stiffened shell structure using 3-D shell elements to represent the skin and 3-D beam elements to represent the ribs. Your choice of model dimensionality and element type will often determine which method of model generation will be most practical for your problem.

LINE models can represent 2-D or 3-D beam or pipe structures, as well as 2-D models of 3-D axisymmetric shell structures. Solid modeling usually does not offer much benefit for generating line models; they are more often created by direct generation methods.

2-D SOLID analysis models are used for thin planar structures (plane stress), "infinitely long" structures having a constant cross section (plane strain), or axisymmetric solid structures. Although many 2-D analysis models are relatively easy to create by direct generation methods, they are usually easier to create with solid modeling.

3-D SHELL models are used for thin structures in 3-D space. Although some 3-D shell analysis models are relatively easy to create by direct generation methods, they are usually easier to create with solid modeling.

3-D SOLID analysis models are used for thick structures in 3-D space that have neither a constant cross section nor an axis of symmetry. Creating a 3-D solid analysis model by direct generation methods usually requires considerable effort. Solid modeling will nearly always make the job easier.

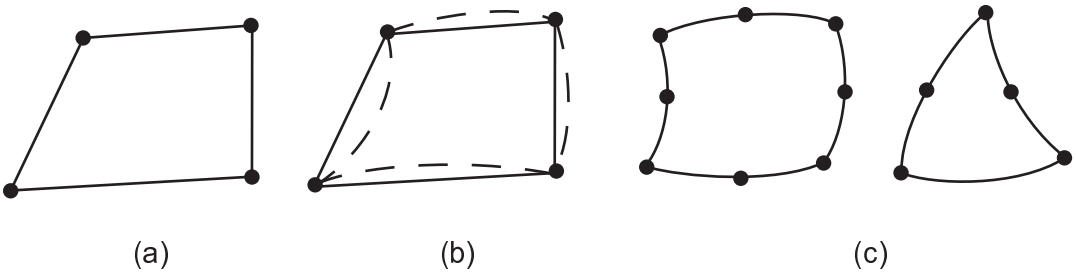
2.2. Choosing Between Linear and Higher Order Elements

The ANSYS program's element library includes two basic types of area and volume elements: *linear* (with or without extra shapes), and *quadratic*. These basic element types are represented schematically in Figure 2.1: Area and Volume Types (p. 6). Let's examine some of the considerations involved in choosing between these two basic element types:

Basic area and volume types available in the ANSYS program

(a) Linear isoparametric
(b) Linear isoparametric with extra shapes
(c) Quadratic

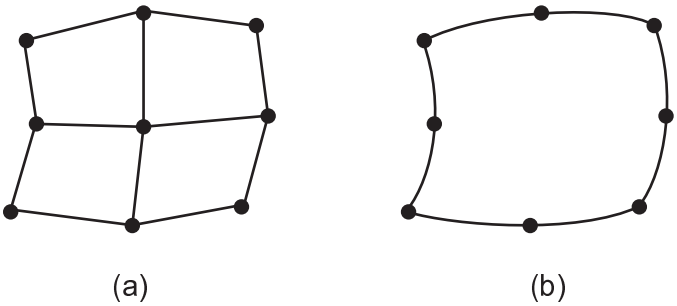
Figure 2.1: Area and Volume Types



2.2.1. Linear Elements (No Midside Nodes)

For structural analyses, these corner node elements with extra shape functions will often yield an accurate solution in a reasonable amount of computer time. When using these elements, it is important to avoid their degenerate forms in critical regions. That is, avoid using the triangular form of 2-D linear elements and the wedge or tetrahedral forms of 3-D linear elements in high results-gradient regions, or other regions of special interest. You should also take care to avoid using excessively distorted linear elements. In *nonlinear* structural analyses, you will usually obtain better accuracy at less expense if you use a fine mesh of these linear elements rather than a comparable coarse mesh of quadratic elements.

Figure 2.2: Comparable Grids



Examples of (a) linear and (b) quadratic elements are shown in Figure 2.2: Comparable Grids (p. 6).

When modeling a curved shell, you must choose between using curved (that is, quadratic) or flat (linear) shell elements. Each choice has its advantages and disadvantages. For most practical cases, the majority

of problems can be solved to a high degree of accuracy in a minimum amount of computer time with flat elements. You must take care, however, to ensure that you use enough flat elements to model the curved surface adequately. Obviously, the smaller the element, the better the accuracy. It is recommended that the 3-D flat shell elements not extend over more than a 15° arc. Conical shell (axisymmetric line) elements should be limited to a 10° arc (or 5° if near the Y axis).

For most non-structural analyses (thermal, magnetic, etc.), the linear elements are nearly as good as the higher order elements, and are less expensive to use. Degenerate elements (triangles and tetrahedra) usually produce accurate results in non-structural analyses.

2.2.2. Quadratic Elements (Midside Nodes)

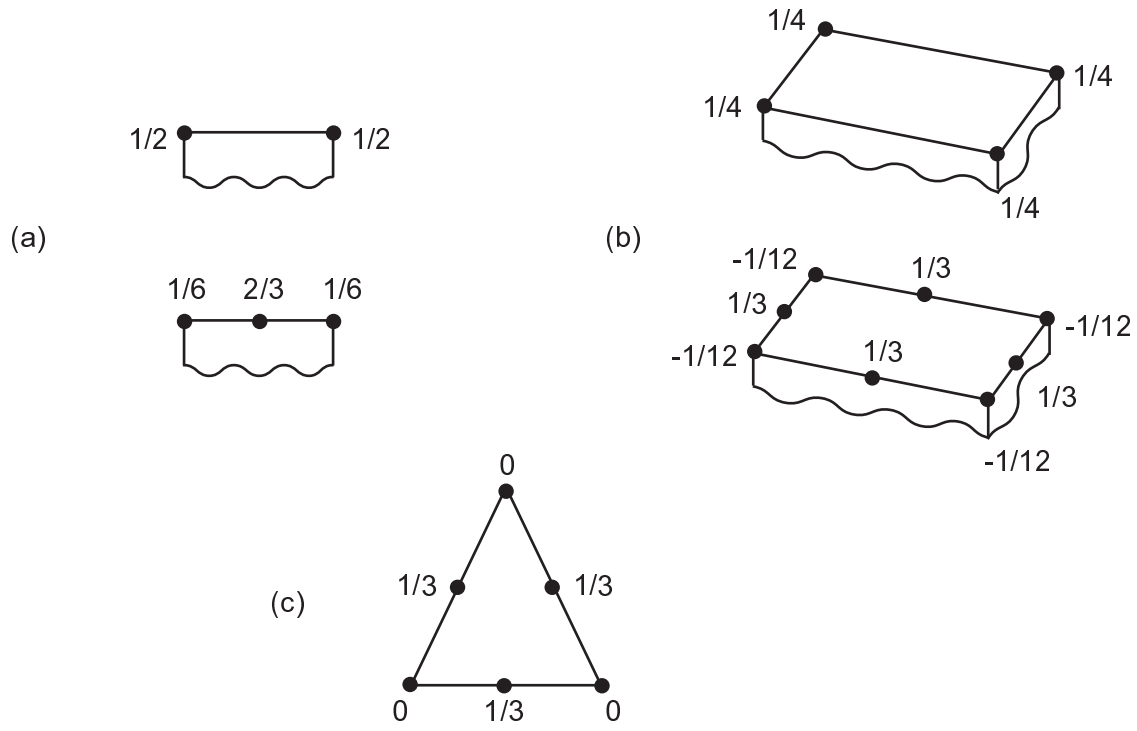
For linear structural analyses with degenerate element shapes (that is, triangular 2-D elements and wedge or tetrahedral 3-D elements), the quadratic elements will usually yield better results at less expense than will the linear elements. However, in order to use these elements correctly, you need to be aware of a few peculiar traits that they exhibit:

- Distributed loads and edge pressures are not allocated to the element nodes according to "common sense," as they are in the linear elements. (See Figure 2.3: Equivalent Nodal Allocations (p. 8).) Reaction forces from midside-node elements exhibit the same nonintuitive interpretation.
- 3-D thermal elements with midside nodes subject to convection loading inherently distribute the heat flow such that it flows in one direction at the midside node and in the other direction at the corner nodes.
- Mass at the midside nodes is greater than at the corner nodes. When selecting master degrees of freedom in a substructure (or CMS) generation, you must include midside nodes as master nodes in order to achieve better mass or surface load representation.

  Equivalent nodal allocations of a unit uniform surface load are shown in Figure 2.3: Equivalent Nodal Allocations (p. 8). The following scenarios are depicted:

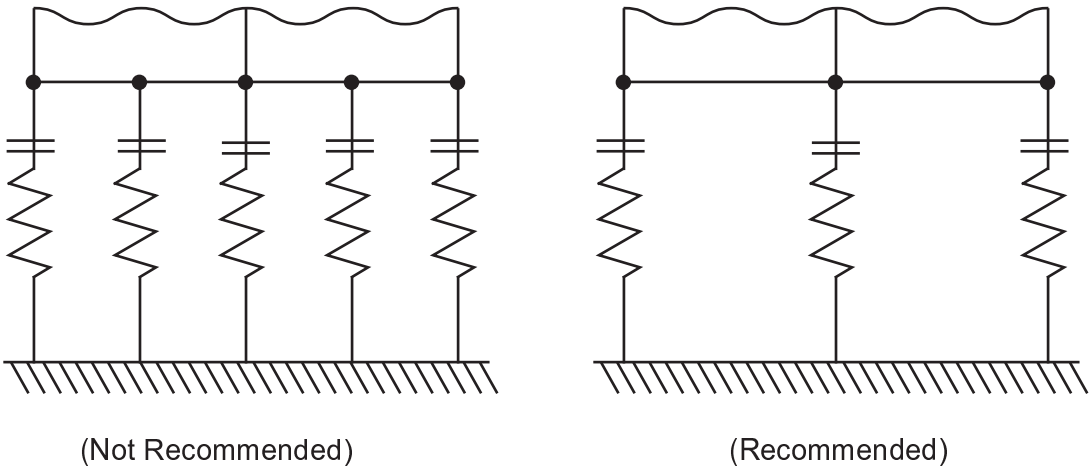
  (a) 2-D elements
  (b) 3-D elements
  (c) triangular 3-D elements

Figure 2.3: Equivalent Nodal Allocations



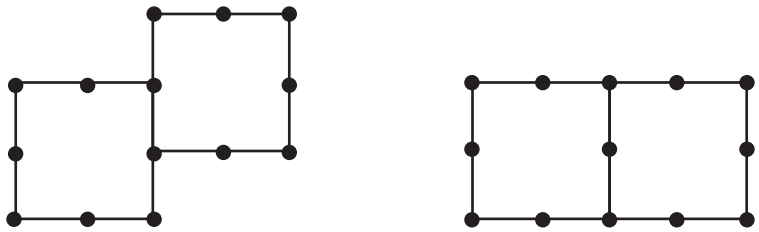
- In dynamic analyses where wave propagation is of interest, midside-node elements are not recommended because of the nonuniform mass distribution.
- Do not define nodal-based contact elements (such as COMBIN40, CONTA175, and CONTA178) at, or connect gap elements to, faces with midside nodes. Similarly for thermal problems, do not apply radiation links or nonlinear convection surfaces to edges with midside nodes. Where nodal-based contact is necessary on surfaces with midside nodes, the midside nodes should be removed, if possible. This caution does not apply to the surface-to-surface, line-to-line, and line-to-surface contact elements (TARGE169, TARGE170, CONTA171, CONTA172, CONTA173, CONTA174, CONTA176, and CONTA177). Meshing of solid models provides ways to omit certain midside nodes.

Figure 2.4: Avoid Midside Nodes at Gaps and Nodal-based Contact Surfaces



- When constraining degrees of freedom at an element edge (or face), all nodes on the face, including the midside nodes, must be constrained.
- The corner node of an element should only be connected to the corner node, and not the midside node of an adjacent element. Adjacent elements should have connected (or common) midside nodes.

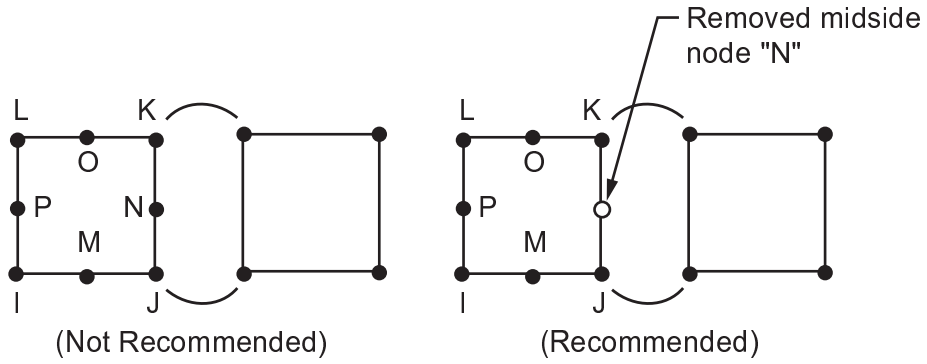
Figure 2.5: Avoid Midside-to-Corner Node Connections Between Elements



(Not Recommended) (Recommended)

- For elements having midside nodes, it is generally preferred that each such node be located at the straight-line position halfway between the corresponding corner nodes. There are, however, situations where other locations may be more desirable:
  - Nodes following curved geometric boundaries will usually produce more accurate analysis results - and all ANSYS meshers place them there by default.
  - Even internal edges in some meshes may have to curve to prevent elements from becoming inverted or otherwise overly distorted. ANSYS meshers sometimes produce this type of curvature.
  - It is possible to mimic a *crack-tip singularity* with "quarter point" elements, with midside nodes deliberately placed off-center. You can produce this type of specialized area mesh in ANSYS by using the **KSCON** command (**Main Menu> Preprocessor> Meshing> Size Cntrls> Concentrat KPs> Create**).
- Midside node positions are checked by the element shape test described below. (For information about controlling element shape checking, see Generating the Mesh (p. 101) of this manual.)
  - All solid and shell elements except 3-node triangles and 4-node tetrahedra are tested for uniformity of the mapping between "real" 3-D space and the element's own "natural" coordinate space. A large *Jacobian ratio* indicates excessive element distortion, which may or may not be caused by poorly located midside nodes. For details about Jacobian ratio tests, refer to the section on element shape testing in the *Mechanical APDL Theory Reference*.
- If you do not assign a location for a midside node, the program will automatically place that node midway between the two corner nodes, based on a linear Cartesian interpolation. Nodes located in this manner will also have their nodal coordinate system rotation angles linearly interpolated.
- Connecting elements should have the same number of nodes along the common side. When mixing element types it may be necessary to remove the midside node from an element. For example, node N of the 8-node element shown below should be removed (or given a zero-node number when the element is created [**E**]) when the element is connected to a 4-node element.

Figure 2.6: Avoid Mismatched Midside Nodes at Element Interconnections



Note

The program will automatically remove midside nodes along the common sides of linear and quadratic elements in the following situation: one area (or volume) is meshed [**AMESH**, **VMESH**, **FVMESH**] with linear elements, then an adjacent area (or volume) is meshed with quadratic elements. Midside nodes will not be removed if the order of meshing is reversed (quadratic elements followed by linear elements).

- A removed midside node implies that the edge is and remains straight, resulting in a corresponding increase in the stiffness. It is recommended that elements with removed nodes be used only in transition regions and not where simpler linear elements with added shape functions will do. If needed, nodes may be added or removed after an element has been generated, using one of the following methods:
  Command(s): EMID, **EMODIF**
  GUI: Main Menu> Preprocessor> Modeling> Move/Modify> Elements> Add Mid Nodes
  Main Menu> Preprocessor> Modeling> Move/Modify> Elements> Remove Mid Nd
  Main Menu> Preprocessor> Modeling> Move/Modify> Elements> Modify Nodes

- A quadratic element has no more integration points than a linear element. For this reason, linear elements will usually be preferred for nonlinear analyses.

- One-element meshes of higher-order quadrilateral elements such as PLANE183 may produce a singularity due to zero energy deformation.

- In postprocessing, the program uses only corner nodes for section and hidden line displays. Similarly, nodal stress data for printout and postprocessing are available only for the corner nodes.

- In graphics displays, midside-node elements, which actually use a curved edge in the element formulation, are displayed with straight-line segments (unless PowerGraphics is used). Models will therefore look "cruder" than they actually are.

2.3. Limitations on Joining Different Elements

You must be careful when you directly join elements that have differing degrees of freedom (DOFs), because there will be inconsistencies at the interface. When elements are not consistent with each other, the solution may not transfer appropriate forces or moments between different elements.

To be consistent, two elements must have the same DOFs; for example, they must both have the same number and type of displacement DOFs and the same number and type of rotational DOFs. Furthermore, the DOFs must overlay (be tied to) each other; that is, they must be continuous across the element boundaries at the interface.

Consider three examples of the use of inconsistent elements:

- **Elements having a different number of DOFs are inconsistent.**

  SHELL181 has three displacement and three rotational DOFs per node. SOLID185 elements have three displacement DOFs per node, but lack rotational DOFs. If a SOLID185 element is joined to a SHELL181 element, the nodal forces corresponding to displacement DOFs are transmitted to the solid element; however, the nodal moments corresponding to the rotational DOFs of the SHELL181 element are not transmitted to the SOLID185 element.

- **Elements having the same number of DOFs may nevertheless be inconsistent.**

  For example, a beam element and a shell element may both have three DOFs per node; however, the shell element may have three displacement DOFs (UX, UY and UZ) while the beam element has only two (UX and UY), so the UZ result reflects the stiffness of the shell element only. Furthermore, the shell element may not have the rotational DOF (ROTZ) that the beam element has, so the nodal moment corresponding to the beam element's rotational DOF is not transmitted to the shell element; the interface behaves as if the beam were "pinned."

- **Both 3-D beam elements and 3-D shell elements have six DOFs per node, but may be joined in a manner that is inconsistent.**

  The ROTZ degree of freedom of the shell element (the drilling mode) is associated with the in-plane rotational stiffness. This is normally a fictitious stiffness; that is, it is not the result of a mathematical calculation of the true stiffness; thus, the ROTZ degree of freedom of the shell element is not a true DOF. It is therefore inconsistent to connect only one node of a 3-D beam element to a 3-D shell element such that a rotational DOF of the beam element corresponds to the ROTZ of the shell element; beams should not be joined to shells in such a manner.

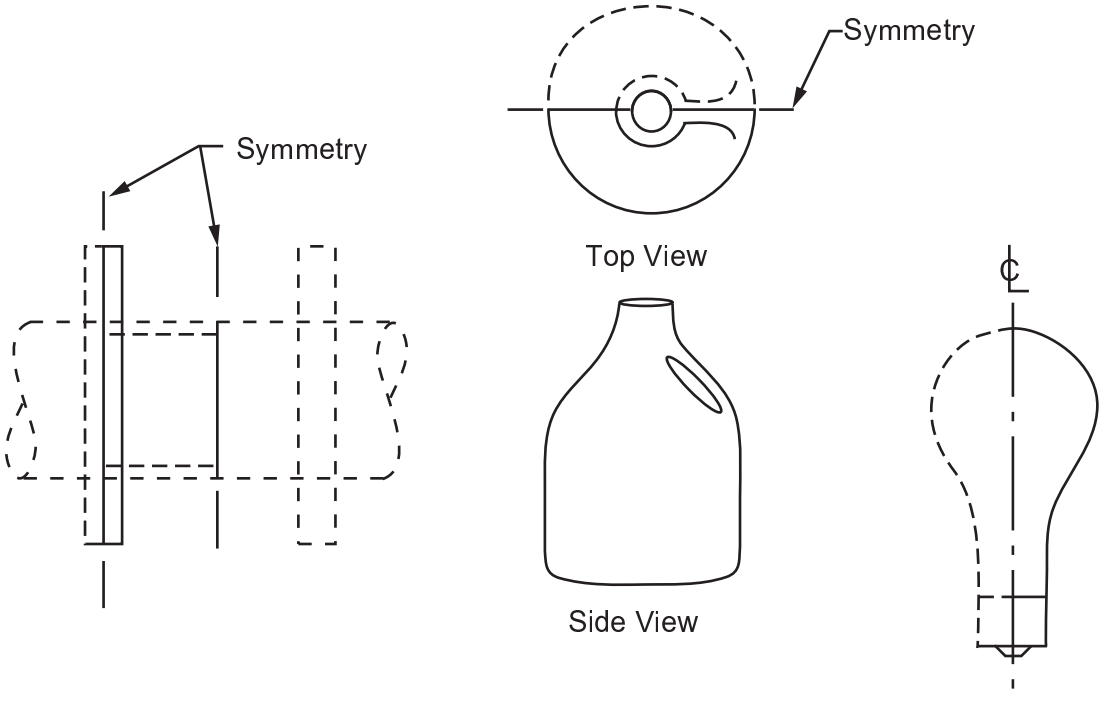
Similar inconsistencies may exist between other elements with differing number and/or types of DOFs.

Such problems may not invalidate your analysis, but you should at least be aware of the conditions at the interface between two different element types.

2.4. Finding Ways to Take Advantage of Symmetry

Many objects have some kind of symmetry, be it repetitive symmetry (such as evenly spaced cooling fins on a long pipe), reflective symmetry (such as a molded plastic container), or axisymmetry (such as a light bulb). When an object is symmetric in all respects (geometry, loads, constraints, and material properties), you can often take advantage of that fact to reduce the size and scope of your model.

Figure 2.7: Examples of Symmetry



2.4.1. Some Comments on Axisymmetric Structures

Any structure that displays geometric symmetry about a central axis (such as a shell or solid of revolution) is an *axisymmetric structure*. Examples would include straight pipes, cones, circular plates, domes, and so forth.

Models of axisymmetric 3-D structures may be represented in equivalent 2-D form. You may expect that results from a 2-D axisymmetric analysis will be more accurate than those from an equivalent 3-D analysis.

By definition, a *fully* axisymmetric model can only be subjected to axisymmetric loads. In many situations, however, axisymmetric structures experience *non*-axisymmetric loading. You must use a special type of element, known as a *general axisymmetric element*, to create a 2-D model of an axisymmetric structure with nonaxisymmetric loads. See General Axisymmetric Elements in the *Element Reference* for details.

2.4.1.1. Some Special Requirements for Axisymmetric Models

Special requirements for axisymmetric models include:

- The axis of symmetry *must* coincide with the global Cartesian Y-axis.
- Negative nodal X-coordinates are not permitted.
- The global Cartesian Y-direction represents the axial direction, the global Cartesian X-direction represents the radial direction, and the global Cartesian Z-direction corresponds to the circumferential direction.
- Your model should be assembled using appropriate element types:

  - For axisymmetric models, use applicable 2-D solids with KEYOPT(3) = 1, and/or axisymmetric shells. In addition, various link, contact, combination, and surface elements can be included in a model that also contains axisymmetric solids or shells. (The program will not realize that these "other" elements are axisymmetric unless axisymmetric solids or shells are present.) If the *Element Reference* does not discuss axisymmetric applications for a particular element type, do not use that element type in an axisymmetric analysis.

  - For axisymmetric harmonic models, use *only* axisymmetric harmonic elements.

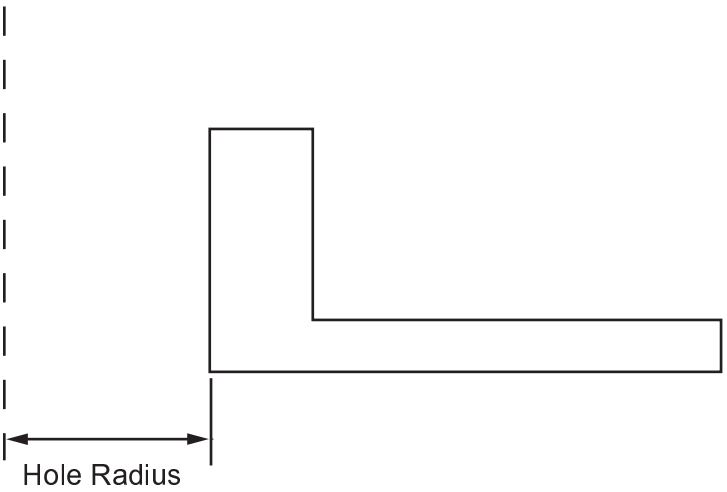
- The SHELL61 element cannot lie on the global Y-axis.

- For models containing 2-D solid elements in which shear effects are important, at least two elements through the thickness should be used.

2.4.1.2. Some Further Hints and Restrictions

If your structure contains a hole along the axis of symmetry, don't forget to provide the proper spacing between the Y-axis and the 2-D axisymmetric model. (See Figure 2.8: An X-direction Offset Represents an Axisymmetric Hole (p. 13).) See Loading in the *Basic Analysis Guide* for a discussion of axisymmetric loads.

Figure 2.8: An X-direction Offset Represents an Axisymmetric Hole



2.5. Determining How Much Detail to Include

Small details that are unimportant to the analysis should not be included in the solid model, since they will only make your model more complicated than necessary. However, for some structures, "small" details such as fillets or holes can be locations of maximum stress, and might be quite important, depending on your analysis objectives. You *must* have an adequate understanding of your structure's expected behavior in order to make competent decisions concerning how much detail to include in your model.

In some cases, only a few minor details will disrupt a structure's symmetry. You can sometimes ignore these details (or, conversely, treat them as being symmetric) in order to gain the benefits of using a smaller symmetric model. You must weigh the gain in model simplification against the cost in reduced accuracy when deciding whether or not to deliberately ignore unsymmetric features of an otherwise symmetric structure.

2.6. Determining the Appropriate Mesh Density

A question that frequently arises in a finite element analysis is, "How fine should the element mesh be in order to obtain reasonably good results?" Unfortunately, no one can give you a definitive answer to this question; *you* must resolve this issue for yourself. Some of the techniques you might employ to resolve this question include:

- Use *adaptive meshing* to generate a mesh that meets acceptable energy error estimate criteria. (This technique is available only for linear static structural or steady state thermal problems. Your judgement as to what constitutes an "acceptable" error level will depend on your analysis requirements.) Adaptive meshing *requires* solid modeling.

- Compare the results of a preliminary analysis with independently derived experimental or known accurate analytical results. Refine the mesh in regions where the discrepancy between known and calculated results is too great. (For all area meshes and for volume meshes composed of tetrahedra, you can refine the mesh locally with the **NREFINE**, **EREFINE**, **KREFINE**, **LREFINE**, and **AREFINE** commands (**Main Menu> Preprocessor> Meshing> Modify Mesh> Refine At>** ***entity type***).)

- Perform an initial analysis using what seems to you to be a "reasonable" mesh. Reanalyze the problem using twice as many elements in critical regions, and compare the two solutions. If the two meshes give nearly the same results, then the mesh is probably adequate. If the two meshes yield substantially different results, then further mesh refinement might be required. You should keep refining your mesh until you obtain nearly identical results for succeeding meshes.

- If mesh-refinement testing reveals that only a portion of your model requires a finer mesh, you can use *submodeling* to "zoom in" on critical regions.

Mesh density is extremely important. If your mesh is too coarse, your results can contain serious errors. If your mesh is too fine, you will waste computer resources, experience excessively long run times, and your model may be too large to run on your computer system. To avoid such problems, always address the issue of mesh density *before* you begin your model generation.

Chapter 3: Coordinate Systems

The ANSYS program uses several types of coordinate systems, each used for a different purpose:

- *Global* and *local* coordinate systems are used to locate geometry items (nodes, keypoints, etc.) in space.
- The *display* coordinate system determines the system in which geometry items are listed or displayed.
- The *nodal* coordinate system defines the degree of freedom directions at each node and the orientation of nodal results data.
- The *element* coordinate system determines the orientation of material properties and element results data.
- The *results* coordinate system is used to transform nodal or element results data to a particular coordinate system for listings, displays, or general postprocessing operations (POST1).

The working plane, which is separate from the coordinate systems discussed here, is for locating geometric primitives during the modeling process. See Using Working Planes (p. 25) for more information about the working plane.

The following coordinate system topics are available:

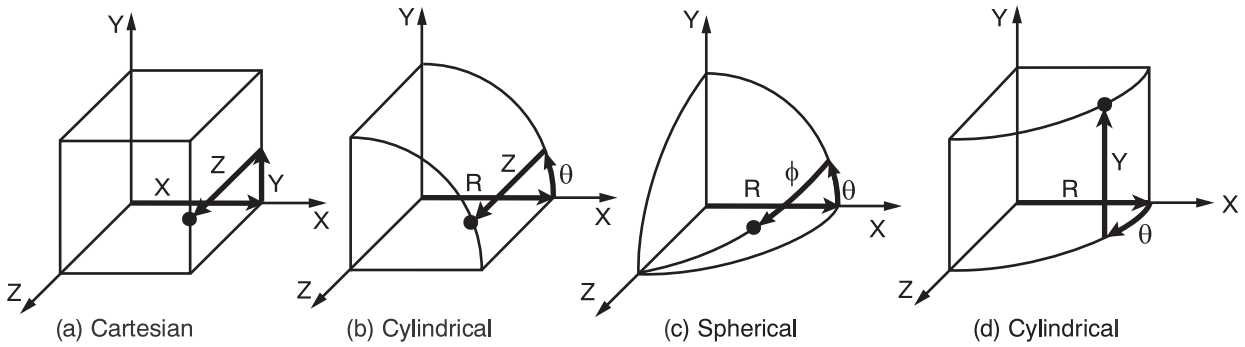
3.1. Global and Local Coordinate Systems

Global and local coordinate systems are used to locate geometry items. When you define a node or a keypoint, its coordinates are interpreted in the global Cartesian system by default. For some models, however, it may be more convenient to define the coordinates in a system other than global Cartesian. You can input the geometry in any of three predefined (global) coordinate systems, or in any number of user defined (local) coordinate systems.

3.1.1. Global Coordinate Systems

A global coordinate system can be thought of as an absolute reference frame. Three predefined global systems are available: *Cartesian, cylindrical,* and *spherical*. All three of these systems are right-handed and, by definition, share the same origin. They are identified by their coordinate system (C.S.) numbers: 0 for Cartesian, 1 and 5 for cylindrical, and 2 for spherical. (See Figure 3.1: Global Coordinate Systems (p. 16).)

Figure 3.1: Global Coordinate Systems



- *(a) Cartesian* (X, Y, Z components) coordinate system 0 (C.S.0)
- *(b) Cylindrical* (R, θ, Z components) coordinate system 1 (C.S.1)
- *(c) Spherical* (R, θ, ϕ components) coordinate system 2 (C.S.2)
- *(d) Cylindrical* (R, θ, Y components) coordinate system 5 (C.S.5)

3.1.2. Local Coordinate Systems

In many cases, it may be necessary to establish your own coordinate system, whose origin is offset from the global origin, or whose orientation differs from that of the predefined global systems. (See Figure 3.2: Euler Rotation Angles (p. 17) for an example of a coordinate system defined by used for local, nodal, or working plane coordinate system rotations.) Such user defined coordinate systems, known as *local* coordinate systems, can be created in the following ways:

- Define the local system in terms of global Cartesian coordinates.
  Command(s): LOCAL
  GUI: Utility Menu> WorkPlane> Local Coordinate Systems> Create Local CS> At Specified Loc
- Define the local system in terms of existing nodes.
  Command(s): CS
  GUI: Utility Menu> WorkPlane> Local Coordinate Systems> Create Local CS> By 3 Nodes
- Define the local system in terms of existing keypoints.
  Command(s): CSKP
  GUI: Utility Menu> WorkPlane> Local Coordinate Systems> Create Local CS> By 3 Keypoints
- Define the local system to be centered at the origin of the presently defined working plane.
  Command(s): CSWPLA
  GUI: Utility Menu> WorkPlane> Local Coordinate Systems> Create Local CS> At WP Origin
- Define the local system in terms of the active coordinate system with the **CLOCAL** command (see The Active Coordinate System (p. 18)). (There is no GUI equivalent for the **CLOCAL** command.)

When a local coordinate system is defined, it becomes the active coordinate system. As you create a local system, you assign it a C.S. identification number (which must be 11 or greater). You can create (or delete) local coordinate systems in any phase of your ANSYS session. To delete a local system, use one of the following methods:
Command(s): CSDELE

GUI: Utility Menu> WorkPlane> Local Coordinate Systems> Delete Local CS

To view the status of all global and local coordinate systems, use one of the following methods:
Command(s): CSLIST
GUI: Utility Menu> List> Other> Local Coord Sys

Your local coordinate systems can be Cartesian, cylindrical, or spherical, similar in form to the three predefined global systems. Note that you may define local cylindrical and spherical coordinate systems in either circular or elliptical configuration. Additionally, you can define a *toroidal* local coordinate system, as illustrated in Figure 3.3: Coordinate System Types (p. 18).

Note

Solid modeling operations in a toroidal coordinate system are not recommended. Areas or volumes generated may not be what you expect.

Figure 3.2: Euler Rotation Angles

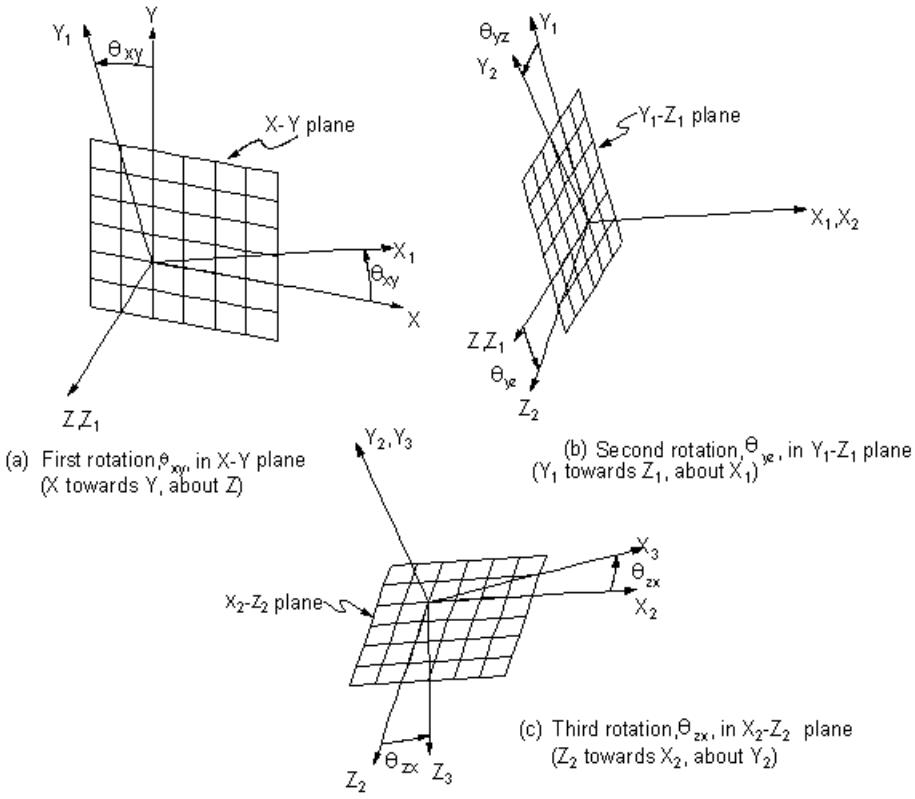
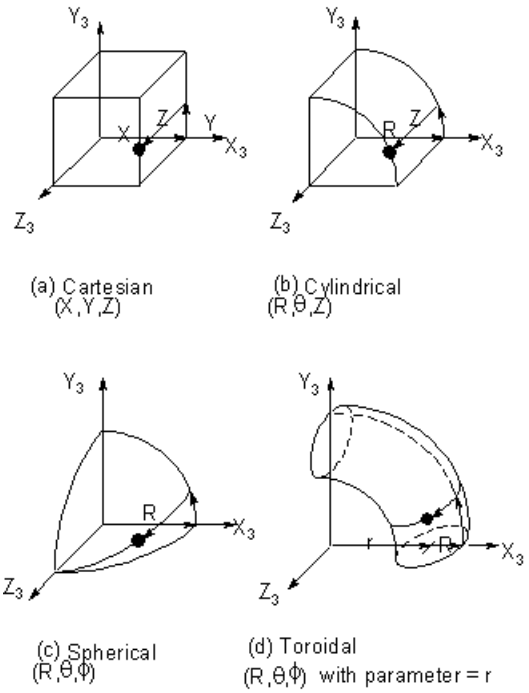


Figure 3.3: Coordinate System Types



3.1.3. The Active Coordinate System

You may define as many coordinate systems as you like, but only one of these systems may be *active* at a time. The choice of active coordinate system is determined as follows: Initially, the global Cartesian system is active by default. Each time you define a local coordinate system, that newly-defined system then automatically becomes the active one. If you want to activate one of the global coordinate systems or some other previously defined coordinate system, use one of the following methods:
Command(s): CSYS
GUI: Utility Menu> WorkPlane> Change Active CS to> Global Cartesian
Utility Menu> WorkPlane> Change Active CS to> Global Cylindrical
Utility Menu> WorkPlane> Change Active CS to> Global Spherical
Utility Menu> WorkPlane> Change Active CS to> Specified Coord Sys
Utility Menu> WorkPlane> Change Active CS to> Working Plane

You can activate a coordinate system in any phase of your ANSYS session. That same coordinate system will remain active in all subsequent phases until you change it explicitly.

Note

When you define a keypoint or a node, the program response labels the coordinates as X, Y, and Z, regardless of which coordinate system is active. You should make the appropriate mental substitutions if the active coordinate system is not Cartesian (R, θ, Z for cylindrical and R, θ, Φ for spherical or toroidal).

3.1.4. Surfaces

Specifying a constant value for a single coordinate implies a *surface*. For example, X = 3 represents the Y-Z plane (or surface) at X = 3 in a Cartesian system. Implied surfaces are used with various operations, such as selecting (**xSEL** commands) and moving (**MOVE**, **KMOVE**, etc.) entities. Some surfaces of constant

value (C) are illustrated in the following figures. These surfaces may be located in either global or local coordinate systems to allow for any desired orientation. Note that for surfaces in elliptical coordinate systems, a constant R value (R = C) represents the value of R along the X-axis.

Figure 3.4: Surfaces of Constant Value

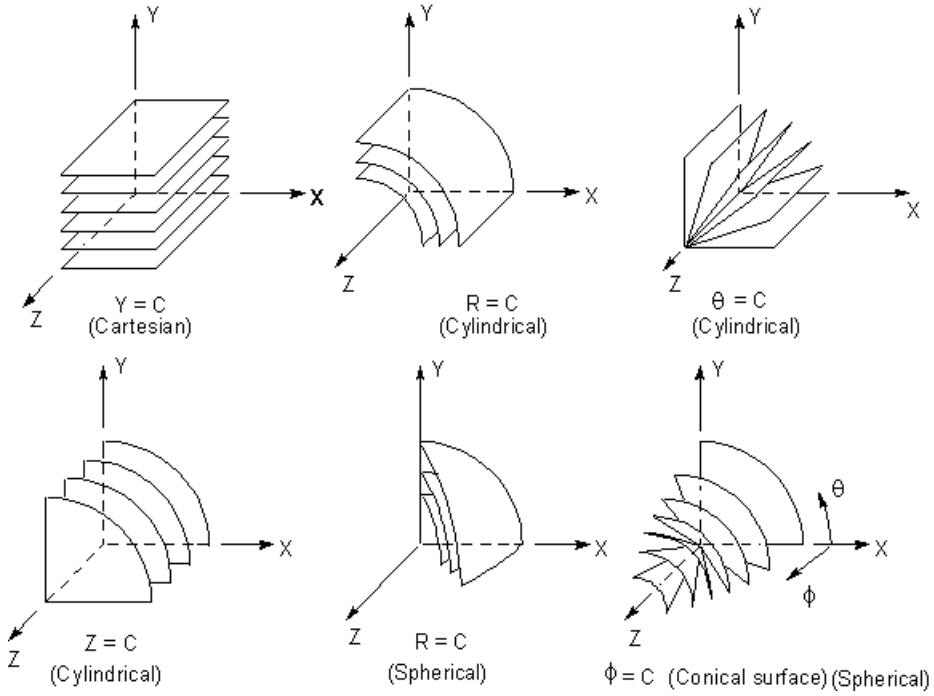
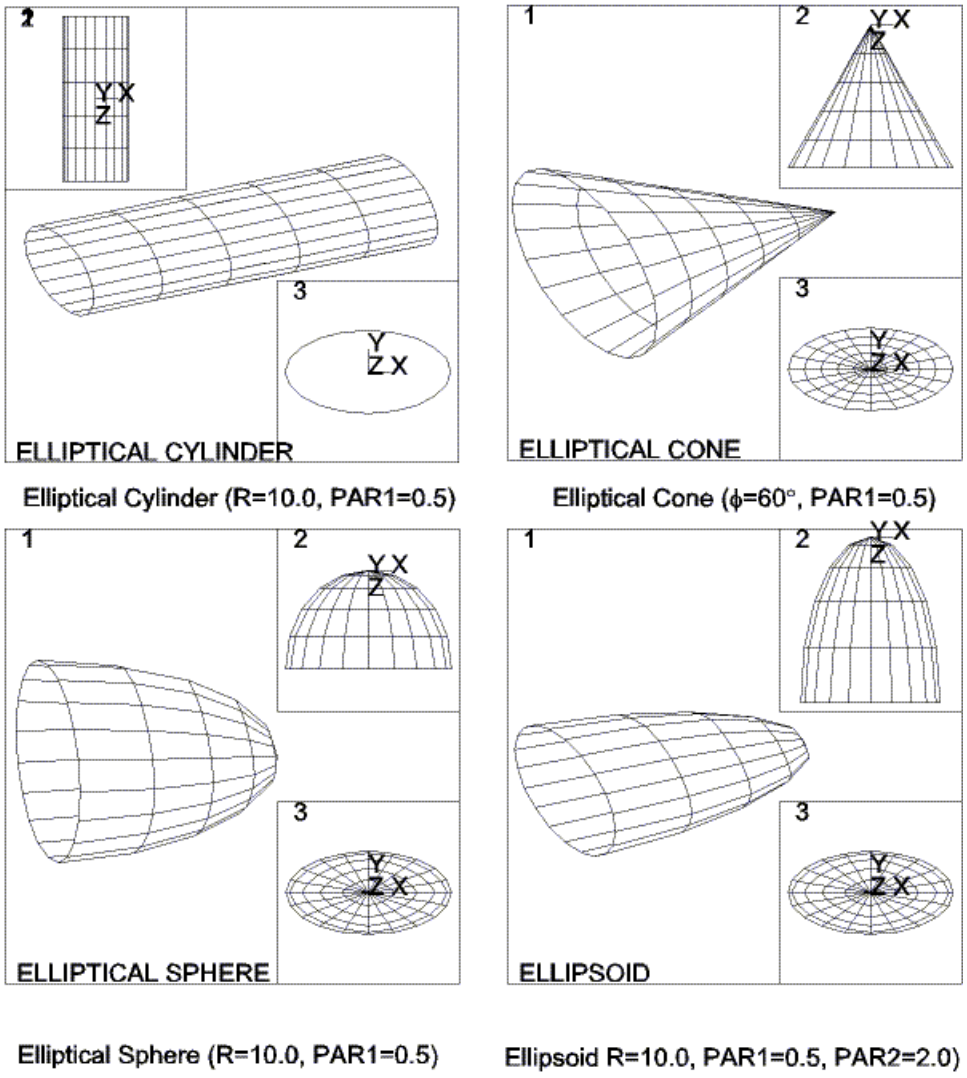


Figure 3.5: Meshed Surfaces of Constant Value



3.1.5. Closed Surfaces and Surface Singularities

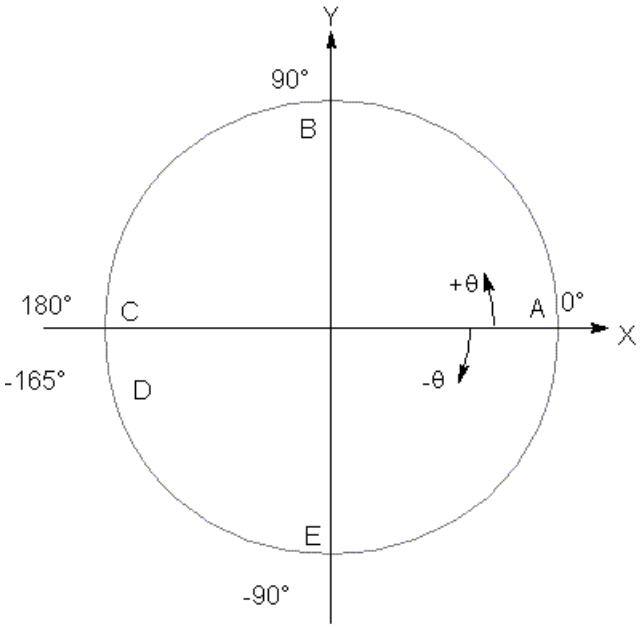
Open surfaces are assumed to be infinite. Cylindrical circular surfaces have a singularity at $\theta = \pm 180°$, as shown in Figure 3.6: Singularity Points (p. 21), so that a fill generation of a string of nodes [**FILL**] or keypoints [**KFILL**] does not cross the 180° line. A fill operation defined from A to C will pass through B. A fill operation from A to D will pass through E. A fill operation from C to D will pass through B, A, and E.

For a specified cylindrical coordinate system, you can move the singularity point to $\theta = 0°$ (or 360°) so that a fill operation from C to D will not pass through B, A, or E. To move the singularity point, use one of the following methods:

Command(s): CSCIR

GUI: Utility Menu> WorkPlane> Local Coordinate Systems> Move Singularity

Figure 3.6: Singularity Points



A similar singularity occurs in the toroidal coordinate system at $\Phi = \pm 180°$ and can also be moved by the above methods. Singularities also occur in the spherical coordinate system at $\Phi = \pm 90°$, such that these locations should not be used.

Note that solid model *lines* will not be affected by these singularity locations. A curved line between two keypoints will take the shortest path in the angular direction, without regard to the location of the singularity point. (As a result, curved lines cannot span an arc of more than 180°.) Thus, in the figure above, circular lines from B to D or from D to B will pass through C.

3.2. Display Coordinate System

By default, a listing of nodes or keypoints always shows their global Cartesian coordinates, even if they were defined in a different coordinate system. You can change the *display coordinate system* used in such listings by one of the following methods:
Command(s): DSYS
GUI: Utility Menu> WorkPlane> Change Display CS to> Global Cartesian
Utility Menu> WorkPlane> Change Display CS to> Global Cylindrical
Utility Menu> WorkPlane> Change Display CS to> Global Spherical
Utility Menu> WorkPlane> Change Display CS to> Specified Coord Sys

Changing the display coordinate system *will also affect your graphical displays*. Unless you desire a specific effect in your displays, you should usually reset the display coordinate system to C.S. 0 (the global Cartesian system) before issuing any graphics display action commands (such as **NPLOT**, **EPLOT**, etc.). (Keypoint plots [**KPLOT**], line plots [**LPLOT**], area plots [**APLOT**], and volume plots [**VPLOT**] are not affected by **DSYS**.)

3.3. Nodal Coordinate Systems

While global and local coordinate systems *locate* geometry items, the nodal coordinate system *orients* the degree of freedom directions at each node. Each node has its own nodal coordinate system, which, by default, is parallel to global Cartesian (regardless of the active coordinate system in which the node

was defined). You can rotate the nodal coordinate system at any node to a desired orientation using one of the following methods:

- Rotate the nodal coordinate system into the active coordinate system. That is, the nodal X-axis is rotated to be parallel to the X or R axis of the active system, the nodal Y-axis is rotated to be parallel to the Y or θ axis of the active system, and the nodal Z-axis is rotated to be parallel to the Z or Φ axis of the active system.
  Command(s): NROTAT
  GUI: Main Menu> Preprocessor> Modeling> Create> Nodes> Rotate Node CS> To Active CS
  Main Menu> Preprocessor> Modeling> Move/Modify> Rotate Node CS> To Active CS

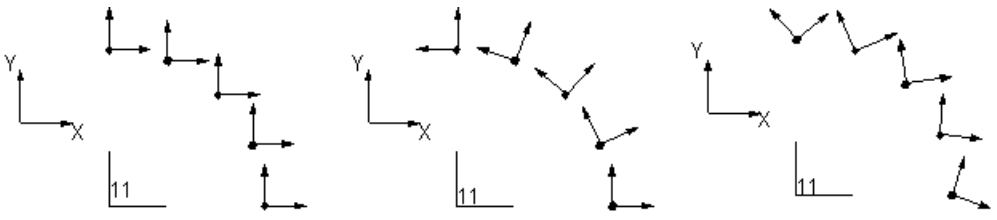
- Rotate the nodal coordinate system by known rotation angles. (Since you will usually not know these rotation angles explicitly, you will probably find the **NROTAT** method to be more useful.) You can define the rotation angles at the time the node is created [**N**], or you can specify rotation angles for existing nodes [**NMODIF**].
  Command(s): N
  GUI: Main Menu> Preprocessor> Modeling> Create> Nodes> In Active CS
  Command(s): NMODIF
  GUI: Main Menu> Preprocessor> Modeling> Create> Nodes> Rotate Node CS> By Angles
  Main Menu> Preprocessor> Modeling> Move/Modify> Rotate Node CS> By Angles

- Rotate the nodal coordinate system by direction cosine components.
  Command(s): NANG
  GUI: Main Menu> Preprocessor> Modeling> Create> Nodes> Rotate Node CS> By Vectors
  Main Menu> Preprocessor> Modeling> Move/Modify> Rotate Node CS> By Vectors

- Rotate the nodal coordinate system to surface normal.
  Command(s): NORA
  GUI: Main Menu> Preprocessor> Modeling> Move/Modify> RotateNode> To Surf Norm> On Areas

You can list the nodal coordinate rotation angles with respect to the global Cartesian system using one of the following methods:
Command(s): NLIST
GUI: Utility Menu> List> Nodes
Utility Menu> List> Picked Entities> Nodes

Figure 3.7: Nodal Coordinate Systems



(a) Default orientation - parallel to global Cartesian system (C.S. 0)

(b) Rotated [NROTAT] parallel to local cylindrical system (C.S. 11)

(c) Rotated [NROTAT] parallel to global cylindrical system (C.S. 1)

3.3.1. Data Interpreted in the Nodal Coordinate System

Input data that are interpreted in the nodal coordinate system include component values of the following:

- Degree of freedom constraints
- Forces
- Master DOF
- Coupled nodes
- Constraint equations

The following results data are reported in the nodal coordinate system on the output file and in POST26:

- Degree of freedom solution
- Nodal loads
- Reaction loads

In POST1, results data are reported in terms of the *results* coordinate system [**RSYS**], not the nodal coordinate system.

3.4. Element Coordinate Systems

Every element has its own coordinate system, the *element coordinate system,* that determines the direction of orthotropic material properties, applied pressures, and results (such as stresses and strains) for that element. All element coordinate systems are right-handed orthogonal systems.

The default orientations for most elements' coordinate systems fit the following patterns:

- *Line elements* usually have the element X-axis directed from their node I toward their node J.
- *Shell elements* usually have the element X-axis similarly directed (from I toward J), the Z-axis normal to the shell surface (with the positive direction determined by the right-hand rule around the element from node I to J to K), and the Y-axis perpendicular to the X and Z axes.
- For *2-D and 3-D solid elements*, the element coordinate system is usually parallel to the global Cartesian system.

However, not all elements correspond to these patterns; see specific element descriptions in the *Element Reference* for the default element coordinate system orientation for such elements.

Many element types have key options (KEYOPTs; input at the time the element is defined [**ET**] or on the **KEYOPT** command) that allow you to change the default element coordinate system orientation. For area and volume elements, you can also change the orientation to align the element coordinate system with a previously defined local system by using one of the following methods:
Command(s): ESYS
GUI: Main Menu> Preprocessor> Meshing> Mesh Attributes> Default Attribs
Main Menu> Preprocessor> Modeling> Create> Elements> Elem Attributes

If you specify both KEYOPTs and **ESYS**, the **ESYS** definition overrides. For some elements, you can define a further rotation, relative to the previous orientation, by entering an angle as a real constant.

3.5. The Results Coordinate System

Results data are calculated during solution and 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 [**ETABLE**].

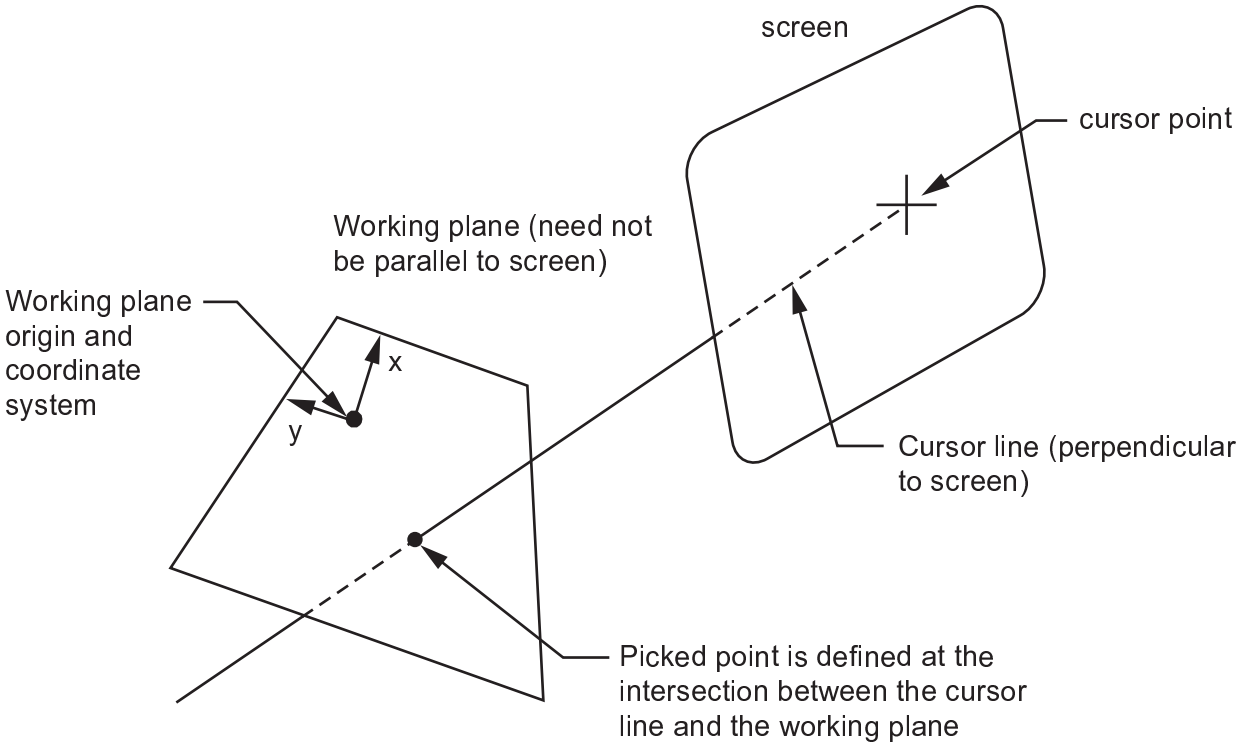
You can change the active results coordinate system to another system (such as the global cylindrical system or a local coordinate system), or to the coordinate systems used during solution (i.e., the nodal and element coordinate systems). If you then list, display, or operate on the results data, they are rotated to this results coordinate system first. Use one of the following methods to change the results coordinate system:
Command(s): RSYS
GUI: Main Menu> General Postproc> Options for Output
Utility Menu> List> Results> Options

See The General Postprocessor (POST1) of the *Basic Analysis Guide* for details on rotating results to a different coordinate system for postprocessing.

Chapter 4: Using Working Planes

Although your cursor appears as a *point* on your screen, it actually represents a *line* through space, normal to the screen. In order to be able to pick a *point* with your cursor, you first need to define an imaginary plane that, when intersected by the normal line of your cursor, will yield a unique point in space. This imaginary plane is called a *working plane*. Another way to think of the interaction between your cursor and your working plane is to picture your cursor as a point that moves around *on your working plane*. The working plane, then, acts as a "tablet" on which you write with your cursor. (The working plane need *not* be parallel to your display screen.)

Figure 4.1: Relationships Among Display Screen, Cursor, Working Plane, and Picked Point



A working plane is an infinite plane with an origin, a 2-D coordinate system, a snap increment, and a display grid. You can define only one working plane at a time. (Creating a new working plane eliminates your existing working plane.) The working plane is separate from the coordinate systems; for example, the working plane can have a different point of origin and rotation than the active coordinate system. To learn how to force the active coordinate system to track the working plane, see Working Plane Tracking (p. 30).

The following working plane topics are available:

4.1. Creating a Working Plane

By default, when you initiate your ANSYS session, there is a working plane located on the global Cartesian X-Y plane, with its x and y axes colinear with the global Cartesian X and Y axes.

The following topics are available to help you create and use a working plane:

4.1.1. Defining a New Working Plane

You can define a new working plane by using any of these methods:

- To define a working plane by three points, or by locating it on the plane normal to a viewing vector at a specified point, use one of these methods:
  Command(s): WPLANE
  GUI: Utility Menu> WorkPlane> Align WP with> XYZ Locations

- To define a working plane by three nodes, or by locating it on the plane normal to a viewing vector at a specified node, use one of these methods:
  Command(s): NWPLAN
  GUI: Utility Menu> WorkPlane> Align WP with> Nodes

- To define a working plane by three keypoints, or by locating it on the plane normal to a viewing vector at a specified keypoint, use one of these methods:
  Command(s): KWPLAN
  GUI: Utility Menu> WorkPlane> Align WP with> Keypoints

- To define a working plane by locating it on the plane normal to a viewing vector at a specified point on a line, use one of these methods:
  Command(s): LWPLAN
  GUI: Utility Menu> WorkPlane> Align WP with> Plane Normal to Line

- To define a working plane by locating it on the X-Y (or R-θ) plane of an existing coordinate system, use one of these methods:
  Command(s): WPCSYS
  GUI: Utility Menu> WorkPlane> Align WP with> Active Coord Sys
  Utility Menu> WorkPlane> Align WP with> Global Cartesian
  Utility Menu> WorkPlane> Align WP with> Specified Coord Sys

4.1.2. Controlling the Display and Style of the Working Plane

To obtain the current status (that is, the location, orientation, and enhancements) of the working plane, use one of these methods:
Command(s): WPSTYL,STAT
GUI: Utility Menu> List> Status> Working Plane

To reset the working plane to its default location and style, use the command **WPSTYL**,DEFA.

4.1.3. Moving the Working Plane

You can move a working plane to a new location (that is, a new origin) using any of the following methods (all of which translate the working plane to a new location parallel to its original location):

- To move the working plane origin to the average location of keypoints, use one of these methods:
  Command(s): KWPAVE
  GUI: Utility Menu> WorkPlane> Offset WP to> Keypoints

- To move the working plane origin to the average location of nodes, use one of these methods:
  Command(s): NWPAVE
  GUI: Utility Menu> WorkPlane> Offset WP to> Nodes

- To move the working plane origin to the average of specified points, use one of these methods:
  Command(s): WPAVE
  GUI: Utility Menu> WorkPlane> Offset WP to> Global Origin
  Utility Menu> WorkPlane> Offset WP to> Origin of Active CS
  Utility Menu> WorkPlane> Offset WP to> XYZ Locations

- To offset the working plane, use one of these methods:
  Command(s): WPOFFS
  GUI: Utility Menu> WorkPlane> Offset WP by Increments

4.1.4. Rotating the Working Plane

You can rotate your working plane to a new orientation in two ways: by rotating the working plane's x-y coordinate system within the plane, or by rotating the entire plane to a new position. (If you do not know the rotation angles explicitly, you might find it easier to simply define a new working plane at the correct orientation using one of the methods described above.) To rotate the working plane, use one of these methods:
Command(s): WPROTA
GUI: Utility Menu> WorkPlane> Offset WP by Increments

4.1.5. Recreating a Previously-defined Working Plane

Although you cannot actually "save" a working plane, you *can* create a local coordinate system at the working plane origin, and then use this local coordinate system to recreate a previously-defined working plane.

- To create a local coordinate system at the working plane origin, use one of these methods:
  Command(s): CSWPLA
  GUI: Utility Menu> WorkPlane> Local Coordinate Systems> Create Local CS> At WP Origin

- To use the local coordinate system to recreate a previously-defined working plane, use one of these methods:
  Command(s): WPCSYS
  GUI: Utility Menu> WorkPlane> Align WP with> Active Coord Sys
  Utility Menu> WorkPlane> Align WP with> Global Cartesian
  Utility Menu> WorkPlane> Align WP with> Specified Coord Sys

4.2. Working Plane Enhancements

Using the **WPSTYL** command or GUI path described earlier, you can enhance your working plane with a *snap increment*, a *display grid*, *retrieval tolerance*, and *coordinate type*. Then, you can force your coordinate system to follow your working plane as the working plane is moved using one of these methods:
Command(s): CSYS
GUI: Utility Menu> WorkPlane> Change Active CS to> Global Cartesian
Utility Menu> WorkPlane> Change Active CS to> Global Cylindrical
Utility Menu> WorkPlane> Change Active CS to> Global Spherical
Utility Menu> WorkPlane> Change Active CS to> Specified Coordinate Sys
Utility Menu> WorkPlane> Change Active CS to> Working Plane
Utility Menu> WorkPlane> Offset WP to> Global Origin

The following additional topics are available to help you refine your working plane:

4.2.1. Snap Increment

It is difficult, if not impossible, to position your cursor at a precisely defined spot on the working plane. In order to pick with precision, you can use the **WPSTYL** command or GUI path to establish a *snap increment*. Once a snap increment is defined, any point that you create by picking will be located at the nearest snap point in your working plane. Stated mathematically, when your cursor falls within the range

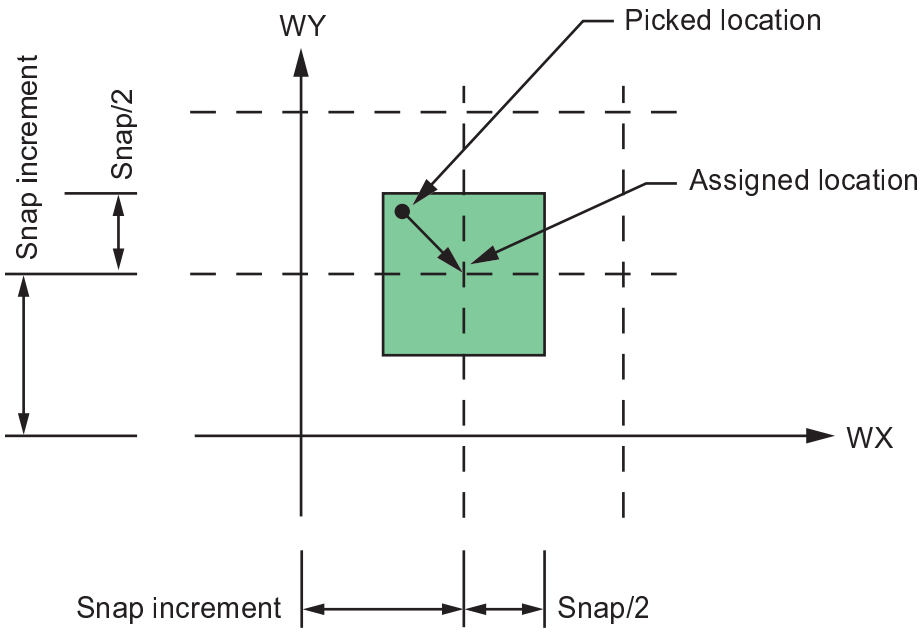
$$N*\text{SNAP} - \text{SNAP}/2 \leq X < N*\text{SNAP} + \text{SNAP}/2$$

for any integer *N*, the x coordinate picked is assigned the value

$$x_p = N*\text{SNAP}.$$

(The same snap increment is used for both x and y coordinates, where x and y are in terms of the working plane coordinate system.) You can visualize the snap increment as creating a pattern of square boxes, as shown below. Any locational pick you make will "snap" to the center of its box.

Figure 4.2: Snap Increment



4.2.2. Display Grid

You can create a square *display grid* to help you visualize the location and orientation of your working plane. The grid spacing, style, and boundaries are established with the **WPSTYL** command. (This grid is *not* related in any way to your snap points.) By issuing **WPSTYL** without arguments, you can toggle the display of the grid on and off.

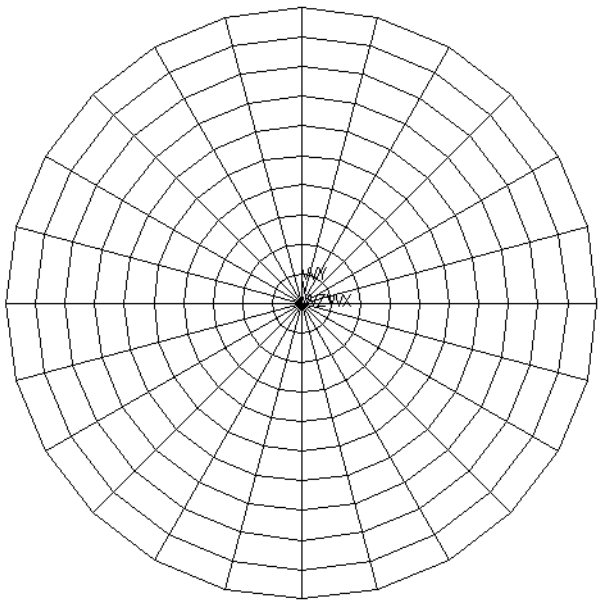
4.2.3. Retrieval Tolerance

An existing entity that you want to pick might lie close to, but not exactly on, your working plane. By specifying a *retrieval tolerance* with the **WPSTYL** command or GUI path, you can instruct the program to consider entities that are within that tolerance to be on the working plane. This tolerance, in effect, gives "thickness" to your working plane, for retrieval picking purposes.

4.2.4. Coordinate Type

There are two types of working planes that you can choose from: Cartesian and polar. Discussion up to this point has concentrated on Cartesian working planes but polar working planes may be used if your geometry is easily described in polar (radius, theta) coordinates. Figure 4.3: Polar Working Plane Grid (p. 30) shows a polar working plane grid that was activated with the **WPSTYL** command. Picking with a polar working plane works the same way as picking on a Cartesian working plane. Grid point locations for the snap feature are located by specifying the radial distance between snap points (*SNAP* on **WPSTYL**) and the angle between snap points (*SNAPANG*).

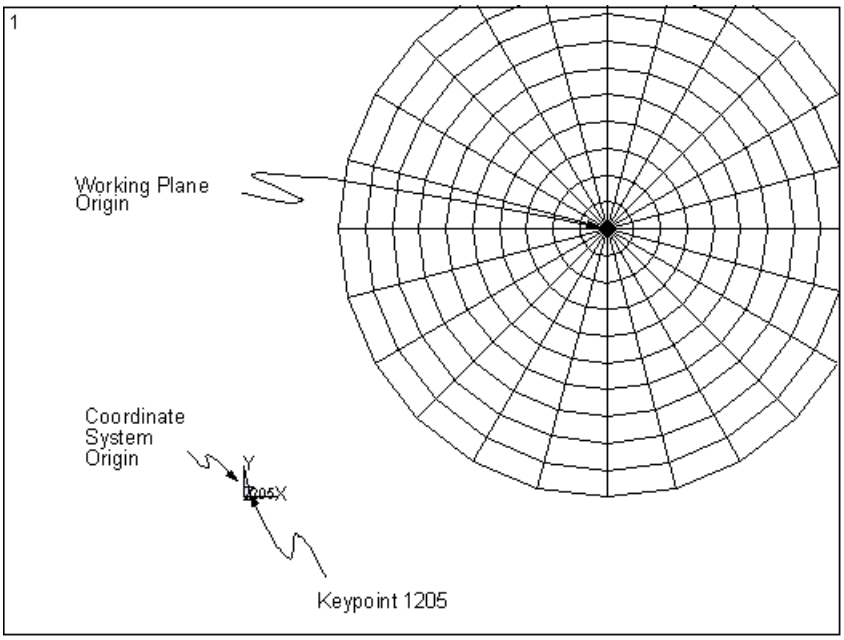
Figure 4.3: Polar Working Plane Grid



4.2.5. Working Plane Tracking

If you've used working planes in conjunction with coordinate systems to define your geometry, you've probably discovered that working planes are completely separate from coordinate systems. When you change or move the working plane, for instance, the coordinate system does not change to reflect the new working plane type or location. This can be frustrating if you are using a combination of picking (based on the working plane), and keyboard input of entities such as keypoints (based on active coordinate system). For instance, if you move the working plane from its default position, then wish to define a keypoint at the new origin of the working plane with keyboard input (that is **K**,1205,0,0,0), you'll find that the keypoint is located at the coordinate system origin rather than the working plane origin (see Figure 4.4: Working Plane/Coordinate System Mismatch (p. 30)).

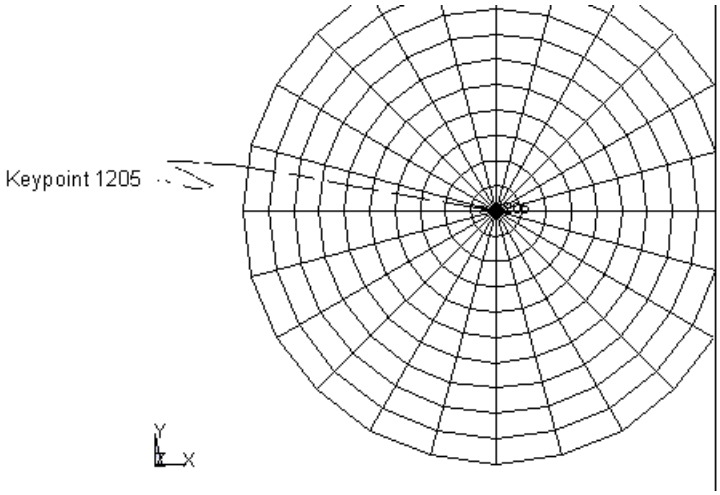
Figure 4.4: Working Plane/Coordinate System Mismatch



If you find yourself forcing the active coordinate system to follow the working plane around as you model, consider using an option on the **CSYS** command or GUI path to do this automatically. The command **CSYS**,WP or **CSYS**,4 will force the active coordinate system to be of the same system type (for example, Cartesian) and in the same location as the working plane. Then, as long as you leave the active coordinate system be WP or 4, as you move the working plane, the coordinate system will move with it. The coordinate system is also updated if you change the type of working plane that you are using. For instance, if you change the working plane from Cartesian to polar, the active coordinate system will change from Cartesian to cylindrical.

To revisit the example discussed above, suppose that you wish to place a keypoint at the origin of your working plane after you've moved that plane. You moved your plane, as before, but this time you activated working plane tracking (**CSYS**,WP) before you moved the plane. Now, when you use the keyboard to locate your keypoint (that is **K**,1205,0,0,0), the keypoint is placed at the origin of the working plane because the coordinate system is in the same location as the working plane (see Figure 4.5: Matched Working Plane Coordinate System (CSYS,WP) (p. 31)).

Figure 4.5: Matched Working Plane Coordinate System (CSYS,WP)



Chapter 5: Solid Modeling

The purpose of using a solid model is to relieve you of the time-consuming task of building a complicated finite element model by direct generation. Some solid modeling and meshing operations can help you to speed up the creation of your final analysis model

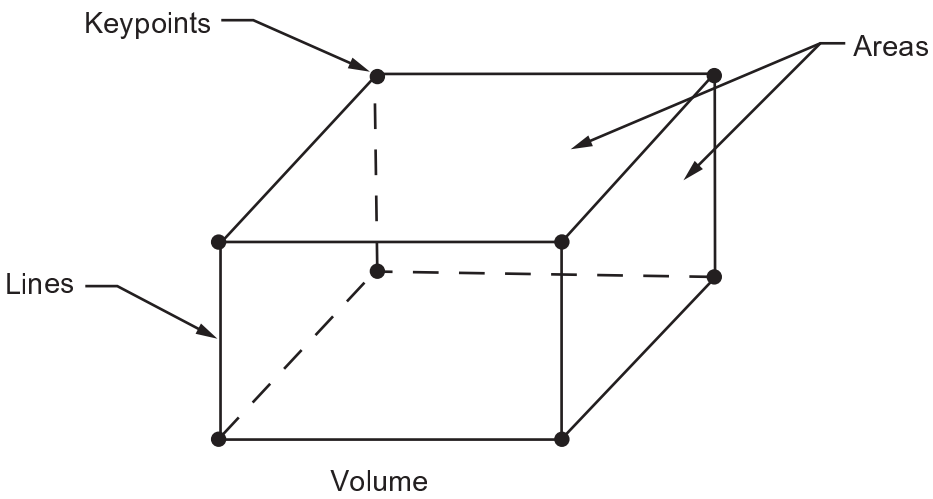
The solid modeling features of ANSYS are known to have robustness issues. With careful planning and alternative strategies, you can successfully create the model required for analysis. However, you may be better served using your CAD modeler or ANSYS DesignModeler under the ANSYS Workbench environment to create your model.

The followng solid modeling topics are available:

5.1. An Overview of Solid Modeling Operations

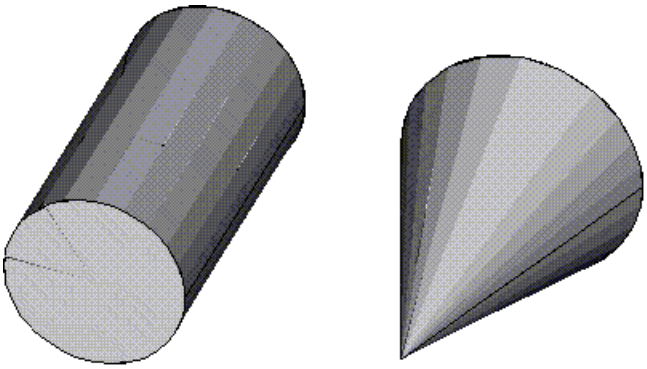
The points that define the vertices of your model are called *keypoints* and are the "lowest-order" solid model entities. If, in building your solid model, you first create your keypoints, and then use those keypoints to define the "higher-order" solid model entities (that is, lines, areas, and volumes), you are said to be building your model "from the bottom up." Models built from the bottom up are defined within the currently active coordinate system.

Figure 5.1: Bottom Up Construction



Building your model from the top down: The ANSYS program also gives you the ability to assemble your model using *geometric primitives*, which are fully-defined lines, areas, and volumes. As you create a primitive, the program automatically creates all the "lower" entities associated with it. If your modeling effort *begins* with the "higher" primitive entities, you are said to be building your model "from the top down." You can freely combine bottom up and top down modeling techniques, as appropriate, in any model. Remember that geometric primitives are built within the working plane while bottom up techniques are defined against the active coordinate system. If you are mixing techniques, you may wish to consider using the **CSYS**,WP or **CSYS**, 4 command to force the coordinate system to follow the working plane.

Figure 5.2: Top Down Constructions (Primitives)

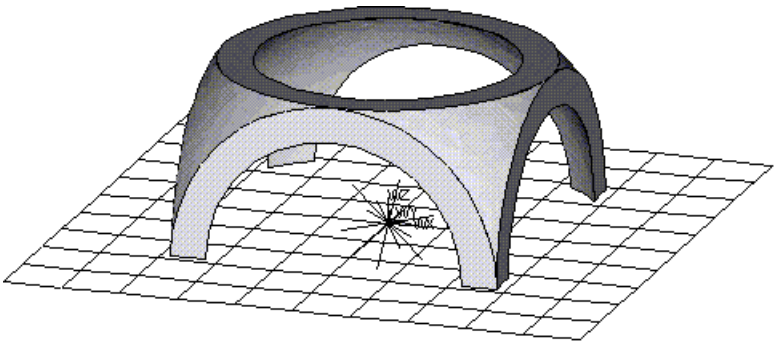


Note

Solid modeling operations in a toroidal coordinate system are not recommended. Areas or volumes generated may not be what you expect.

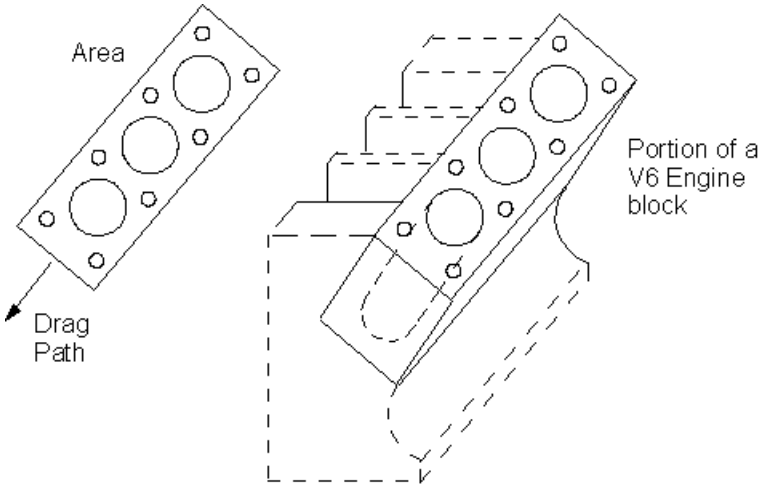
Using Boolean operators: You can "sculpt" your solid model using intersections, subtractions, and other Boolean operations. Booleans allow you to work directly with higher solid model entities to create complex shapes. (Both bottom up and top down creations can be used in Boolean operations.)

Figure 5.3: Create Complex Shapes With Boolean Operations



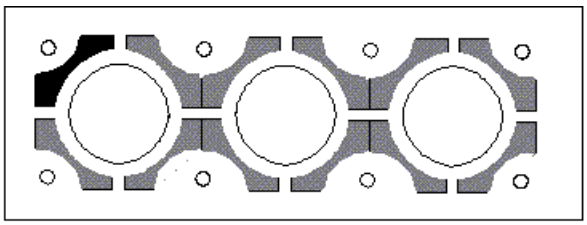
Dragging and rotating: Boolean operators, although convenient, can be computationally expensive. Sometimes a model can be constructed more efficiently by dragging, or rotating.

Figure 5.4: Dragging an Area to Create a Volume (VDRAG)



Moving and copying solid model entities: A complicated area or volume that appears repetitively in your model need only be constructed once; it can then be moved, rotated, and copied to a new location on your model. You might also find it more convenient to place geometric primitives in their proper location by moving them, rather than by changing the working plane.

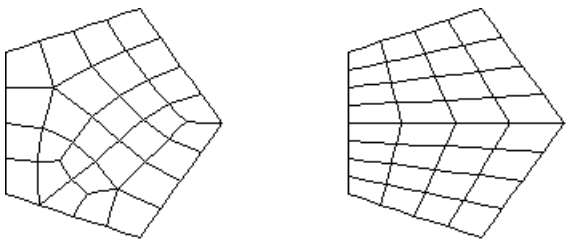
Figure 5.5: Copying an Area



A complicated area
(black) can be copied as
many times as needed
(gray)

Meshing: Your ultimate objective in building a solid model is to *mesh* that model with nodes and elements. Once you have completed the solid model, set element attributes, and established meshing controls, you can then turn the ANSYS program loose to generate the finite element mesh. By taking care to meet certain requirements, you can request a "mapped" mesh containing all quadrilateral, all triangular, or all brick elements.

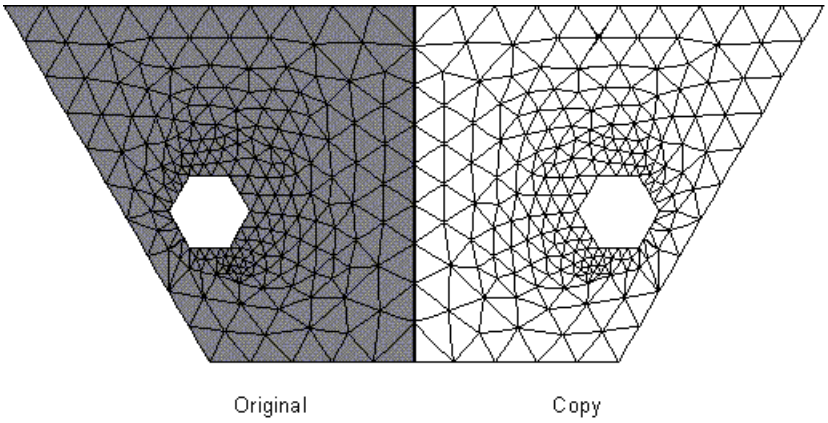
Figure 5.6: Free and Mapped Meshes



Moving and copying nodes and elements: Automatic meshing is a huge improvement over direct generation of nodes and elements, but it can sometimes be computationally time consuming. If your model contains repetitive features, you might find that the most efficient approach to model generation would be to model and mesh a pattern region of your model, then generate copies of that meshed

region. (Copying a mesh in this manner will generally be faster than separately meshing repeated features.)

Figure 5.7: Copying a Meshed Area



Solid model loads: In the ANSYS program, loads are normally associated with nodes and elements. However, using solid modeling, you will often find it inconvenient to define loads at nodes and elements. Fortunately, you may assign loads directly to the solid model; when you initiate the solution calculations (with a **SOLVE** command), the program will automatically transfer these solid model loads to the finite element model.

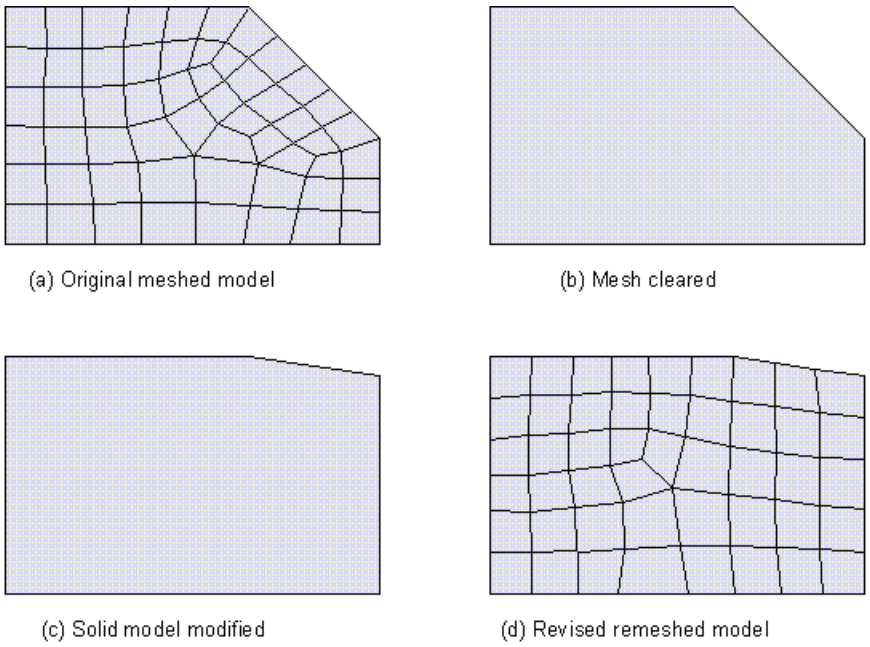
Revising your model (clearing and deleting): Before you can revise your model, you need to be aware of the hierarchy of solid model and finite element model entities. A lower order entity cannot be deleted if it is attached to a higher-order entity. Thus, a volume cannot be deleted if it has been meshed, a line cannot be deleted if it is attached to an area, and so forth. If an entity is attached to any loads, deleting or redefining that entity will delete the attached loads from the database. The hierarchy of modeling entities is as listed below:

Highest	Elements (and Element Loads)
	Nodes (and Nodal Loads)
	Volumes (and Solid-Model Body Loads)
	Areas (and Solid-Model Surface Loads)
	Lines (and Solid-Model Line Loads)
Lowest	Keypoints (and Solid-Model Point Loads)

If you need to revise a solid model after it has been meshed, you must usually first delete all the nodes and elements in the portion of the model to be revised, using the **xCLEAR** commands (**Main Menu> Preprocessor> Meshing> Clear**). Once the solid model is cleared, you can delete (from the top down) and redefine solid model entities to revise your model. As an alternative to clearing, deleting, and redefining, you can sometimes consider modifying the keypoints directly, using one of these methods:
Command(s): KMODIF
GUI: Main Menu> Preprocessor> Modeling> Move/Modify> Keypoints> Set of KPs
Main Menu> Preprocessor> Modeling> Move/Modify> Keypoints> Single KP

By using **KMODIF**, you automatically clear and redefine the attached lines, areas, and volumes. After you have revised your solid model, you will need to remesh the portions that were cleared.

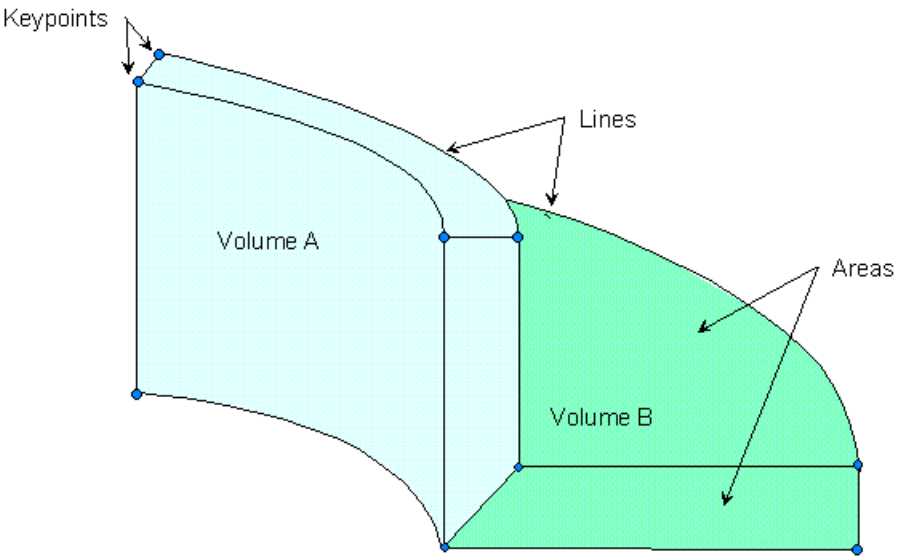
Figure 5.8: Revising a Meshed Solid Model



5.2. Creating Your Solid Model from the Bottom Up

Any solid model, whether assembled from the bottom up or from the top down, is defined in terms of keypoints, lines, areas, and volumes. Figure 5.9: Basic Solid Model Entities (p. 37) illustrates these entities.

Figure 5.9: Basic Solid Model Entities



Keypoints are the vertices, lines are the edges, areas are the faces, and volumes are the interior of the object. Notice that there is a hierarchy in these entities: volumes, the highest-order entities, are bounded by areas, which are bounded by lines, which in turn are bounded by keypoints.

Caution

Solid modeling operations in a toroidal coordinate system are not recommended. Areas or volumes generated may not be what you expect.

In bottom up construction, you first create keypoints and use those keypoints to define higher-order solid model entities (lines, areas, and volumes).

5.2.1. Keypoints

When building your model from the bottom up, you begin by defining the lowest-order solid model entities, *keypoints*. Keypoints are defined within the currently active coordinate system. You can then define lines, areas, and volumes connecting these keypoints. You do not always have to explicitly define all entities in ascending order to create higher-order entities: you can define areas and volumes directly in terms of the keypoints at their vertices. The intermediate entities will then be generated automatically as needed. For example, if you define a brick-like volume in terms of the eight keypoints at its corners, the program will automatically generate the bounding areas and lines.

To define individual keypoints, use one of the methods listed in the following table. See the *Command Reference* for more information on the individual commands.

Define a Keypoint	**Command**	**GUI**
in the active coordinate system	**K**	**Main Menu> Preprocessor> Modeling> Create> Keypoints> In Active CS** **Main Menu> Preprocessor> Modeling> Create> Keypoints> On Working Plane**
at a given location on an existing line	**KL**	**Main Menu> Preprocessor> Modeling> Create> Keypoints> On Line** **Main Menu> Preprocessor> Modeling> Create> Keypoints> On Line w/Ratio**

Once you create an initial pattern of keypoints, you can generate additional keypoints and work with existing keypoints using the methods described in the following table. Many Boolean operations (see Sculpting Your Model with Boolean Operations (p. 56)) will also create keypoints.

	Command	**GUI**
create a keypoint between two existing keypoints	**KBETW**	**Main Menu> Preprocessor> Modeling> Create> Keypoints> KP between KPs**
generate keypoints between two keypoints	**KFILL**	**Main Menu> Preprocessor> Modeling> Create> Keypoints> Fill between KPs**

	Command	GUI
create a keypoint at the center of a circular arc defined by three locations	**KCENTER**	**Main Menu> Preprocessor> Modeling> Create> Keypoints> KP at Center**
generate additional keypoints from a pattern of keypoints	**KGEN**	**Main Menu> Preprocessor> Modeling> Copy> Keypoints**
generate a scaled pattern of keypoints from a given keypoint pattern	**KSCALE**	This command cannot be accessed from a menu.
generate a reflected set of keypoints	**KSYMM**	**Main Menu> Preprocessor> Modeling> Reflect> Keypoints**
transfer a pattern of keypoints to another coordinate system	**KTRAN**	**Main Menu> Preprocessor> Modeling> Move/Modify> Transfer Coord> Keypoints**
define a default location for undefined nodes or keypoints	**SOURCE**	This command cannot be accessed from a menu.
calculate and move a keypoint to an intersection	**KMOVE**	**Main Menu> Preprocessor> Modeling> Move/Modify> Keypoint> To Intersect**
define a keypoint at an existing node location	**KNODE**	**Main Menu> Preprocessor> Modeling> Create> Keypoints> On Node**
calculate the distance between two keypoints	**KDIST**	**Main Menu> Preprocessor> Modeling> Check Geom> KP distances**
modify the coordinates defining a keypoint [1]	**KMODIF**	**Main Menu> Preprocessor> Modeling> Move/Modify> Keypoints> Set of KPs** **Main Menu> Preprocessor> Modeling> Move/Modify> Keypoints> Single KP**

1. Modifying the coordinates will automatically clear any meshed region attached to the specified keypoint, and will also redefine the higher-order entities attached to that keypoint. (A keypoint can also be redefined using the **K** command, but only if it is still "free"; that is, not yet attached to a line, nor meshed.)

You can maintain keypoints using the methods listed in the following table.

Maintenance	Command	GUI
list defined keypoints	**KLIST**	**Utility Menu> List> Keypoint> Coordinates +Attributes** **Utility Menu> List> Keypoint> Coordinates only** **Utility Menu> List> Keypoint> Hard Points**
select keypoints	**KSEL**	**Utility Menu> Select> Entities**
display selected keypoints	**KPLOT**	**Utility Menu> Plot> Keypoints> Keypoints** **Utility Menu> Plot> Specified Entities> Keypoints**

Maintenance	Command	GUI
delete unmeshed keypoints	**KDELE**	**Main Menu> Preprocessor> Modeling> Delete> Keypoints**

If you have issued the command **/PNUM**,KP,1 (**Utility Menu> PlotCtrls> Numbering**), keypoint numbers will also appear in line, area, and volume displays [**LPLOT**, **APLOT**, and **VPLOT**] for keypoints attached to these higher-order entities.

5.2.2. Hard Points

Hard points are actually a special type of keypoints. You can use hard points to apply loads or obtain data from arbitrary points on lines and areas within your model. Hard points do not modify either the geometry or topology of your model. Most of the keypoint commands, such as **FK**, **KLIST**, and **KSEL**, can be applied to hard points. In addition, hard points have their own set of commands and extension in the GUI.

If you issue any commands that update the geometry of an entity, such as Boolean or simplification commands, any hard points associated with that entity are deleted. Therefore, you should add any hard points after completing the solid model. If you delete an entity that has associated hard points, those hard points are either

- Deleted along with the entity (if the hard point is not associated with any other entities).
- Detached from the deleted entity (if the hard point is associated with additional entities).

You cannot manipulate hard points with commands to copy, move, modify, or modify keypoints. Hard points have their own suite of commands and GUI controls. Mapped meshing is not supported when hard points are used. Hard point information cannot be written to the IGES file. The `Jobname.cdb` file can be written with the **CDWRITE**,DB option.

You can define hard points on existing lines or areas. In both cases, you can define the location of hard points on such entities by

- Picking (unavailable for models imported from IGES files).
- Specifying ratios (available for lines only).
- Specifying global X, Y, and Z coordinates.

To create hard points, use one of the methods listed in the following table.

Define individual hard points	Command	GUI
on an existing line	**HPTCREATE**, LINE	**Main Menu> Preprocessor> Modeling> Create> Keypoints> Hard PT on line> Hard PT by ratio** **Main Menu> Preprocessor> Modeling> Create> Keypoints> Hard PT on line> Hard PT by coordinates** **Main Menu> Preprocessor> Modeling> Create> Keypoints> Hard PT on line> Hard PT by picking**
on an existing area	**HPTCREATE**, AREA	**Main Menu> Preprocessor> Modeling> Create> Keypoints> Hard PT on line> Hard PT by coordinates**

Define individual hard points	Command	GUI
		Main Menu> Preprocessor> Modeling> Create> Keypoints> Hard PT on line> Hard PT by picking

You can select hard points or geometric entities with associated hard points using any of the methods described in the following table.

Select	Command	GUI
hard points	**KSEL**	**Utility Menu> Select> Entities** [1]
lines with associated hard points	**LSEL**	**Utility Menu> Select> Entities** [2]
areas with associated hard points	**ASEL**	**Utility Menu> Select> Entities** [3]

1. Choose both `Keypoints` and `By Hard Points` in the Select Entities dialog box.
2. Choose both `Lines` and `By Hard Points` in the Select Entities dialog box.
3. Choose both `Areas` and `By Hard Points` in the Select Entities dialog box.

You can maintain hard points using the methods described in the table below.

Maintenance	Command	GUI
list hard points	**KLIST**	**Utility Menu> List> Keypoint> Hard Points**
list lines with associated hard points	**LLIST**	This command cannot be accessed from a menu.
list areas with associated hard points	**ALIST**	This command cannot be accessed from a menu.
display hard points	**KPLOT**	**Utility Menu> Plot> Keypoints> Hard Points**
delete hard points	**HPTDELETE**	**Main Menu> Preprocessor> Modeling> Delete> Hard Points**

5.2.3. Lines

Lines are mainly used to represent the edges of an object. As with keypoints, lines are defined within the currently active coordinate system. You do not always need to define all lines explicitly, because the program will generate the necessary lines in many instances when an area or volume is defined. Lines are required if you want to generate line elements (such as beams) or to create areas from lines.

If you need to explicitly define a line, you can use one of the methods described in the following table.

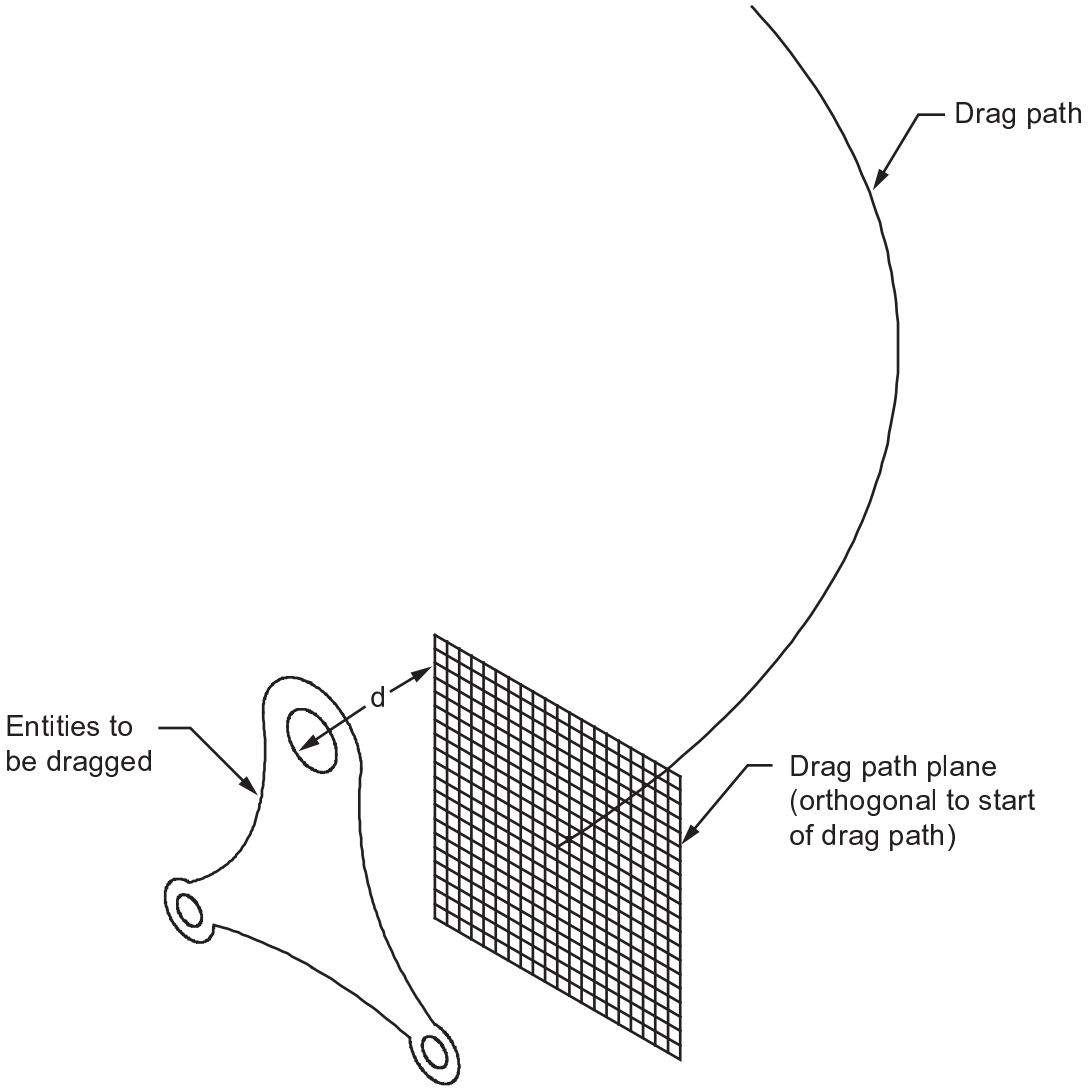
Generate a	Command	GUI
"straight" or cubic line between two specified keypoints	**L**	**Main Menu> Preprocessor> Modeling> Create> Lines> Lines> In Active Coord**

Generate a	Com-mand	GUI
circular line through three keypoints (or between two keypoints if you specify a radius)	**LARC**	**Main Menu> Preprocessor> Modeling> Create> Lines> Arcs> By End KPs & Rad** **Main Menu> Preprocessor> Modeling> Create> Lines> Arcs> Through 3 KPs**
cubic line from a spline fit to a series of keypoints	**BSPLIN**	**Main Menu> Preprocessor> Modeling> Create> Lines> Splines> Spline thru KPs** **Main Menu> Preprocessor> Modeling> Create> Lines> Splines> Spline thru Locs** **Main Menu> Preprocessor> Modeling> Create> Lines> Splines> With Options> Spline thru KPs** **Main Menu> Preprocessor> Modeling> Create> Lines> Splines> With Options> Spline thru Locs**
circular arc line	**CIRCLE**	**Main Menu> Preprocessor> Modeling> Create> Lines> Arcs> By Cent & Radius** **Main Menu> Preprocessor> Modeling> Create> Lines> Arcs> Full Circle**
segmented spline through a series of keypoints	**SPLINE**	**Main Menu> Preprocessor> Modeling> Create> Lines> Splines> Segmented Spline** **Main Menu> Preprocessor> Modeling> Create> Lines> Splines> With Options> Segmented Spline**
straight line at an angle with a line	**LANG**	**Main Menu> Preprocessor> Modeling> Create> Lines> Lines> At Angle to Line** **Main Menu> Preprocessor> Modeling> Create> Lines> Lines> Normal to Line**
line at an angle with two existing lines	**L2ANG**	**Main Menu> Preprocessor> Modeling> Create> Lines> Lines> Angle to 2 Lines** **Main Menu> Preprocessor> Modeling> Create> Lines> Lines> Norm to 2 Lines**
line at the end of, and tangent to, an existing line	**LTAN**	**Main Menu> Preprocessor> Modeling> Create> Lines> Lines> Tan to 2 Lines**
line tangent to two lines	**L2TAN**	**Main Menu> Preprocessor> Modeling> Create> Lines> Lines> Tan to 2 Lines**
shortest line between two keypoints on an area	**LAREA**	**Main Menu> Preprocessor> Modeling> Create> Lines> Lines> Overlaid on Area**
line by sweeping a keypoint pattern along a path [1]	**LDRAG**	**Main Menu> Preprocessor> Modeling> Operate> Extrude> Lines> Along Lines**

Generate a	Com-mand	GUI
circular line by rotating a keypoint pattern about an axis	**LROTAT**	**Main Menu> Preprocessor> Modeling> Operate> Extrude> Lines> About Axis**
fillet line between two intersecting lines	**LFILLT**	**Main Menu> Preprocessor> Modeling> Create> Lines> Line Fillet**
truly straight line no matter what coordinate system is active	**LSTR**	**Main Menu> Preprocessor> Create> Lines> Lines> Straight Line**

1. For best results when using drag operations (**LDRAG**), minimize the distance (shown as distance "d" in Figure 5.10: Drag Operation Suggestions (p. 43)) from the drag path plane to the entities to be dragged. Also the entity plane should be as close to parallel to the drag path plane as possible. Both of these guidelines can be met if the entities to be dragged are in the drag path plane. The drag path plane is automatically defined to be orthogonal to and located at the start of the drag path.

Figure 5.10: Drag Operation Suggestions



For those commands that create a "straight" line, the actual shape of the line will be determined by the active coordinate system. Thus, for a "straight" line in a Cartesian coordinate system, dX/dL, dY/dL, and dZ/dL will each be constant along the length (L) of that line, producing a line that is truly straight. For a "straight" line in a cylindrical coordinate system, dR/dL, dθ/dL, and dZ/dL will each be constant, producing a line that could be a helical spiral (if all three components of slope were nonzero).

Copy a pattern of lines to generate additional lines using any of the methods described in the following table.

	Com-mand	GUI
generate additional lines from a pattern of lines	**LGEN**	**Main Menu> Preprocessor> Modeling> Copy> Lines** **Main Menu> Preprocessor> Modeling> Move/Modify> Lines**
generate lines from a line pattern by symmetry re-flection	**LSYMM**	**Main Menu> Preprocessor> Modeling> Reflect> Lines**
transfer a pattern of lines to another coordinate system	**LTRAN**	**Main Menu> Preprocessor> Modeling> Move/Modify> Transfer Coord> Lines**

You can modify an existing line by redefining it (by reissuing the **L** command) or by using one of the methods described in the following table.

Modification	Com-mand	GUI
divide up a line into smaller lines	**LDIV**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Divide> Line into 2 Ln's** **Main Menu> Preprocessor> Modeling> Operate> Booleans> Divide> Line into N Ln's** **Main Menu> Preprocessor> Modeling> Operate> Booleans> Divide> Lines w/ Options**
combine a line with a second line	**LCOMB**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Add> Lines**
extend a line at one end	**LEXTND**	**Main Menu> Preprocessor> Modeling> Operate> Extend Line**

You can also use several geometric primitives and Boolean commands to generate or modify lines. See Creating Your Solid Model from the Top Down: Primitives (p. 52) for more information on geometric primitives and Sculpting Your Model with Boolean Operations (p. 56) for more information on Boolean operations.

You can maintain lines using the methods listed in the following table.

Maintenance	Com-mand	GUI
list lines	**LLIST**	**Utility Menu> List> Lines** **Utility Menu> List> Picked Entities> Lines**

Maintenance	Com-mand	GUI
display lines	**LPLOT**	**Utility Menu> Plot> Lines** **Utility Menu> Plot> Specified Entities> Lines**
select lines	**LSEL**	**Utility Menu> Select> Entities**
delete lines	**LDELE**	**Main Menu> Preprocessor> Modeling> Delete> Line and Below** **Main Menu> Preprocessor> Modeling> Delete> Lines Only**

Only unmeshed lines that are not attached to an area can be redefined, modified, or deleted. **LDIV**, **LCOMB**, and **LFILLT** provide exceptions to this rule. You can use these three commands to modify unmeshed lines, even if they are attached to areas. The attached areas will be updated, even if they are attached to volumes.

If you have issued the command **/PNUM**,Line,1 (**Utility Menu> PlotCtrls> Numbering**), line numbers will also appear in area and volume displays (**APLOT** and **VPLOT**) for lines that are attached to these higher-order entities.

5.2.4. Areas

Flat areas are used to represent 2-D solid objects (such as flat plates or axisymmetric solids). Curved as well as flat areas are used to represent 3-D surfaces, such as shells, and the faces of 3-D solid objects. Areas are required if you wish to use area elements or if you wish to create volumes from areas. Most commands that create areas will also automatically generate the necessary lines and keypoints; similarly, many areas can be conveniently generated by defining volumes.

You can use any of the methods described in the following table to explicitly define areas. Several geometric primitives and Boolean commands can also be used to generate or modify areas. See Creating Your Solid Model from the Top Down: Primitives (p. 52) for more information on geometric primitives and Sculpting Your Model with Boolean Operations (p. 56) for more information on Boolean operations.

Define an area	Com-mand	GUI
in terms of its vertices (through keypoints)	**A**	**Main Menu> Preprocessor> Modeling> Create> Areas> Arbitrary> Through KPs**
in terms of its boundaries (a set of lines defining the perimeter)	**AL**	**Main Menu> Preprocessor> Modeling> Create> Areas> Arbitrary> By Lines**
by “sweeping” a line along a characteristic path [1]	**ADRAG**	**Main Menu> Preprocessor> Modeling> Operate> Extrude> Along Lines**
by rotating a line pattern about an axis	**AROTAT**	**Main Menu> Preprocessor> Modeling> Operate>Extrude> About Axis**
of constant fillet radius at the intersection of two areas [2]	**AFILLT**	**Main Menu> Preprocessor> Modeling> Create> Areas> Area Fillet**

Define an area	Command	GUI
by “skinning” a surface through guiding lines	**ASKIN**	**Main Menu> Preprocessor> Modeling> Create> Areas> Arbitrary> By Skinning**
by offsetting an existing area (similar to what happens when a balloon is inflated or deflated) [3]	**AOFFST**	**Main Menu> Preprocessor> Modeling> Create> Areas> Arbitrary> By Offset**

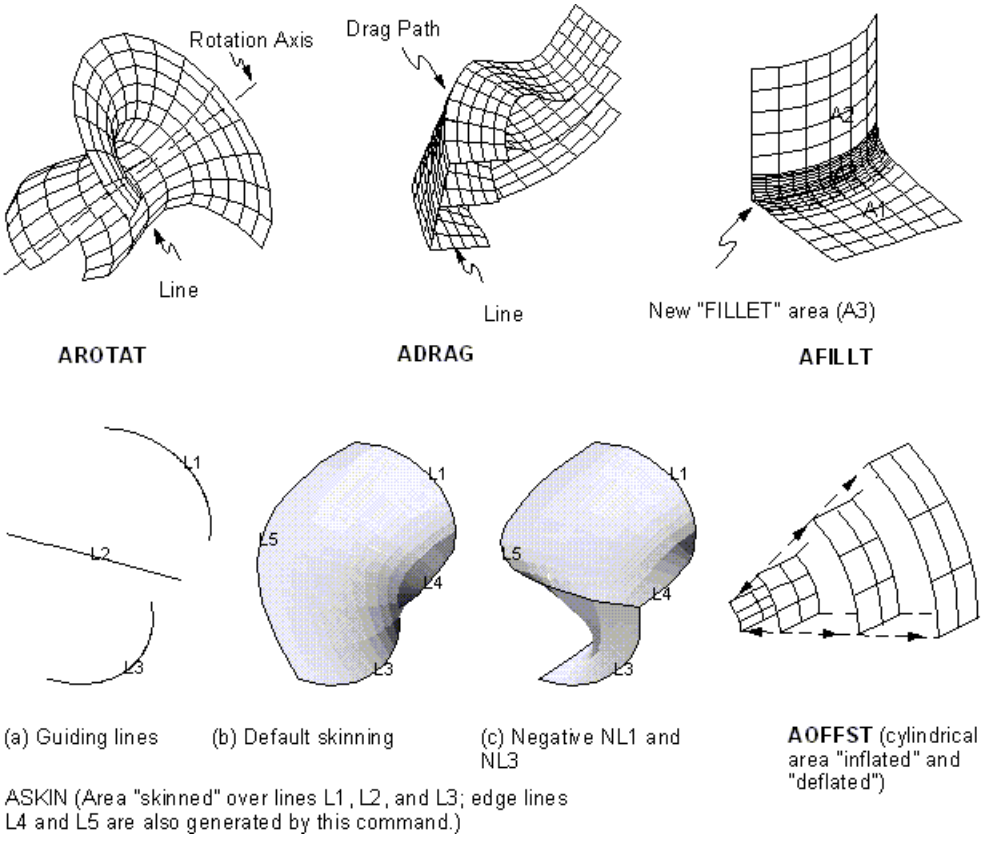
1. For best results when using drag operations (**ADRAG**), minimize the distance (shown as distance "d" in Figure 5.10: Drag Operation Suggestions (p. 43)) from the drag path plane to the entities to be dragged. Also the entity plane should be as close to parallel to the drag path plane as possible. Both of these guidelines can be met if the entities to be dragged are in the drag path plane. The drag path plane is automatically defined to be orthogonal to and located at the start of the drag path.

2. You might experience difficulties due to the underlying Boolean operations used by this command. You may wish to use rotate or extrude operations (**VROTAT** or **VEXT**) to perform the same operations. See When a Boolean Operation Fails (p. 86) for additional information.

3. You might experience difficulties if you attempt to "deflate" an area by a distance that equals or exceeds its least radius of curvature. Using this process will *not* generate a new surface when such problems arise (you will receive a warning if the process fails).

Figure 5.11: Area Command Operations (p. 46) for examples of the operations you can perform with the **AROTAT**, **ADRAG**, **AFILLT**, **ASKIN**, and **AOFFST** commands.

Figure 5.11: Area Command Operations



You can copy existing areas to generate additional areas using the methods described in the following table. Several geometric primitives and Boolean commands can also be used to generate or modify areas. See Creating Your Solid Model from the Top Down: Primitives (p. 52) for more information on geometric primitives and Sculpting Your Model with Boolean Operations (p. 56) for more information on Boolean operations.

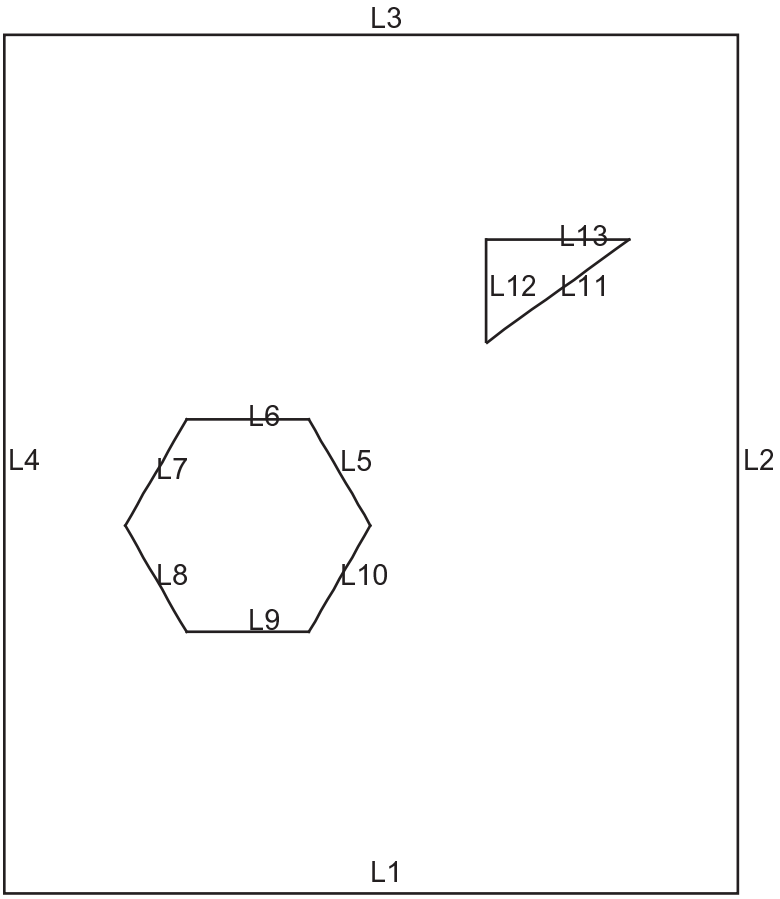
	Command	GUI
Generate additional areas from a pattern of areas	**AGEN**	**Main Menu> Preprocessor> Modeling> Copy> Areas** **Main Menu> Preprocessor> Modeling> Move/Modify> Areas> Areas**
Generate areas form an area pattern by symmetry reflection	**ARSYM**	**Main Menu> Preprocessor> Modeling> Reflect> Areas**
Transfer a pattern of areas to another coordinate system	**ATRAN**	**Main Menu> Preprocessor> Modeling> Move/Modify> Transfer Coord> Areas**
Copy a portion of an area	**ASUB**	**Main Menu> Preprocessor> Modeling> Create> Areas> Arbitrary> Overlaid on Area**

Use the methods listed in the following table to maintain areas. Only unmeshed areas that are not attached to a volume can be redefined or deleted.

Maintenance	Command	GUI
List defined areas [1]	**ALIST**	**Utility Menu> List> Areas** **Utility Menu> List> Picked Entities> Areas**
Display areas	**APLOT**	**Utility Menu> Plot> Areas** **Utility Menu> Plot> Specified Entities> Areas**
Select areas	**ASEL**	**Utility Menu> Select> Entities**
Delete areas	**ADELE**	**Main Menu> Preprocessor> Modeling> Delete> Area and Below** **Main Menu> Preprocessor> Modeling> Delete> Areas Only**

1. The area (topographical measure) of an area (solid model entity) will be included in an **ALIST** only if you previously executed **ASUM** (**Main Menu> Preprocessor> Operate> Calc Geom Items**). The "loop" numbers shown refer to the closed strings of lines that define the boundaries of an area, as illustrated in Figure 5.12: Loops Bound an Area (p. 48).

Figure 5.12: Loops Bound an Area




```
NO.LOOP      LINES                 AREA      #NODES #ELEM   MAT  REAL  TYP ESYS
  1  1     1     2     3     4    N/A            0     0     0     0    0    0
     2    12    13    11
     3     5    10     9     8
           7     6
```

5.2.5. Volumes

Volumes are used to represent 3-D objects, and are required only if you wish to use volume elements. Most commands that create volumes will also automatically generate the necessary lower-order entities.

To define volumes, use any of the methods described in the following table.Several geometric primitives and Boolean commands can also be used to generate or modify areas. See Creating Your Solid Model from the Top Down: Primitives (p. 52) for more information on geometric primitives and Sculpting Your Model with Boolean Operations (p. 56) for more information on Boolean operations.

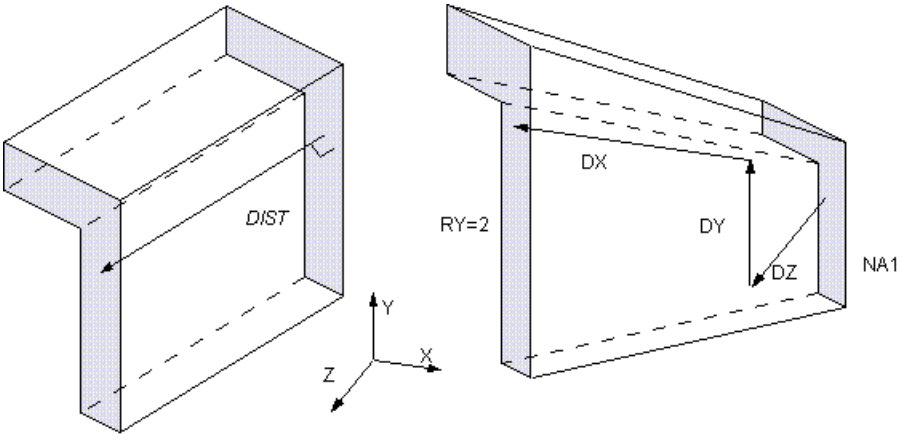
Define a volume	Command	GUI
in terms of its vertices (through keypoints)	**V**	**Main Menu> Preprocessor> Modeling> Create> Volumes> Arbitrary> Through KPs**
in terms of its boundaries (in terms of a set of areas defining the surfaces)	**VA**	**Main Menu> Preprocessor> Modeling> Create> Volumes> Arbitrary> By Areas**

Define a volume	Command	GUI
by "sweeping" an area pattern along a characteristic path [1]	**VDRAG**	**Main Menu> Preprocessor> Operate> Extrude> Along Lines**
by rotating an area pattern about an axis	**VROTAT**	**Main Menu> Preprocessor> Modeling> Operate> Extrude> About Axis**
by offsetting an area	**VOFFST**	**Main Menu> Preprocessor> Modeling> Operate> Extrude> Areas> Along Normal**
by extruding and scaling a pattern of areas in the active coordinate system	**VEXT**	**Main Menu> Preprocessor> Modeling> Operate> Extrude> Areas> By XYZ Offset**

1. For best results when using drag operations (**VDRAG**), minimize the distance (shown as distance "d" in Figure 5.10: Drag Operation Suggestions (p. 43)) from the drag path plane to the entities to be dragged. Also the entity plane should be as close to parallel to the drag path plane as possible. Both of these guidelines can be met if the entities to be dragged are in the drag path plane. The drag path plane is automatically defined to be orthogonal to and located at the start of the drag path.

Figure 5.13: Volume Command Operations (p. 49) illustrates the **VOFFST** and **VEXT** command operations.

Figure 5.13: Volume Command Operations



To generate additional volumes from existing volumes, use the methods described in the following table.

Generate volumes	Command	GUI
from a pattern of volumes	**VGEN**	**Main Menu> Preprocessor> Modeling> Copy> Volumes** **Main Menu> Preprocessor> Modeling> Move/Modify> Volumes**
from a volume pattern by symmetry reflection	**VSYMM**	**Main Menu> Preprocessor> Modeling> Reflect> Volumes**
transfer a pattern of volumes to another coordinate system	**VTRAN**	**Main Menu> Preprocessor> Modeling> Move/Modify> Transfer Coord> Volumes**

You can maintain volumes using the methods described in the following table. Note that only unmeshed volumes can be redefined or deleted.

Maintenance	Command	GUI
list volumes [1]	**VLIST**	**Utility Menu> List> Picked Entities> Volumes** **Utility Menu> List> Volumes**
display volumes	**VPLOT**	**Utility Menu> Plot> Specified Entities> Volumes** **Utility Menu> Plot> Volumes**
select volumes	**VSEL**	**Utility Menu> Select> Entities**
delete volumes	**VDELE**	**Main Menu> Preprocessor> Modeling> Delete> Volume and Below** **Main Menu> Preprocessor> Modeling> Delete> Volumes Only**

1. A volume listing indicates that the volumes are composed of a number of *shells*. A shell is the volumetric equivalent of a loop - a set of entities that define a continuous closed boundary.

5.2.5.1. Extruding Volumes

If the area that is being extruded (used as the pattern for the resulting volume) with the **VROTAT**, **VEXT**, **VOFFST**, or **VDRAG** command is meshed (or belongs to a meshed volume), that mesh will be used as a pattern for the mesh of the volume that is created . (Compare these commands to the **VSWEEP** command, which is described in Sweeping Volumes (p. 52).)

Follow these steps to extrude your mesh:

1. Mesh the area that is to be extruded, dragged, offset, or rotated (using MESH200 elements).

2. Select an appropriate 3-D element type [**ET**] (match the shape and number of nodes to the MESH200 element). Activate the selection [**TYPE**].

3. Specify the desired number of element divisions in the extruded, rotated, or offset direction (*NDIV* argument on **ESIZE** command). If using **VDRAG**, specify the number of element divisions on the drag path line(s) (**LESIZE** or **ESIZE**,,*NDIV*).

4. Issue the **VROTAT**, **VEXT**, **VOFFST**, or **VDRAG** command.

Concatenated areas [**ACCAT**] or areas that have concatenated lines [**LCCAT**] cannot be extruded. You can get around the concatenated line limitation by first meshing the area(s), then deleting the concatenated lines, and finally extruding the area(s) into meshed volume(s).

If element attributes have been associated with the pattern area via the **ACCAT** command, the opposite area generated by the **VDRAG**, **VEXT**, **VOFFST**, or **VROTAT** operation will also have those attributes (i.e., the element attributes from the pattern area are copied to the opposite area). Note that only the area opposite the pattern area will have the same attributes as the pattern area; the areas adjacent to the pattern area will not.

Use the **EXTOPT** command (**Main Menu> Preprocessor> Modeling> Operate> Extrude> Elem Ext Opts** or **Main Menu> Preprocessor> Meshing> Mesh> Volume Sweep> Sweep Opts**) to make the generation of meshed volumes from 2–D models easier. **EXTOPT** controls options relating to the gen-

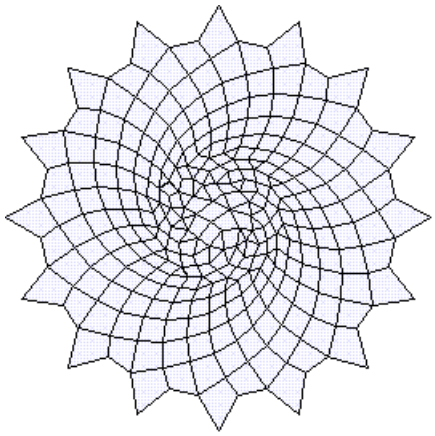
eration of volume elements from area elements using the **VEXT**, **VROTAT**, **VOFFST**, **VDRAG**, and **VSWEEP** commands. It enables carryover of material attributes, real constant attributes, element coordinate system attributes, and section attributes of the pattern area elements to the created volume elements (except for **VSWEEP** as noted below).

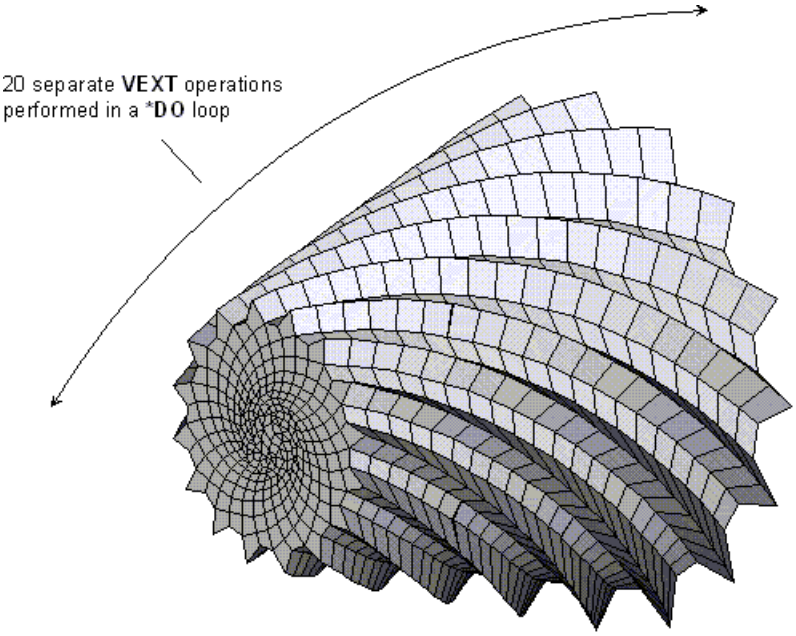
The **EXTOPT** control options include the following:

- When using **VEXT**, **VROTAT**, **VOFFST**, or **VDRAG**, you can set controls to carryover material attributes, real constant attributes, element coordinate attributes, and section attributes of the pattern area elements to the meshed volume elements. (When using **VSWEEP**, since the volume already exists, use the **VATT** command to assign attributes before sweeping.)
- When using **VEXT**, **VROTAT**, **VOFFST**, or **VDRAG**, you can set controls to carryover particular attributes (materials, real constants, element coordinate systems, or section properties) of the pattern area elements to the generated volume elements or you can set controls to use particular attributes (material, real constants, element coordinate systems, or section properties) of the current **MAT**, **REAL**, **ESYS** and **SECNUM** command settings for the generated volume elements.
- You can set controls to set the number of element divisions and spacing ratio in the direction of volume generation.
- When using **VEXT**, **VROTAT**, **VOFFST**, or **VDRAG**, you can set controls to clear the pattern area mesh when volume generations are done. (When you are using **VSWEEP**, if selected, the area meshes on the pattern (source), target, and/or side areas clear when volume sweeping is done.)

The carryover of the attributes of the pattern area elements saves you time that would otherwise be required to prepare the 3-D model extrusion of multiple areas with differing attributes.

Figure 5.14: Extruding (and Scaling) Meshed Areas Into Meshed Volumes





5.2.5.2. Sweeping Volumes

Use the **VSWEEP** command (**Main Menu> Preprocessor> Meshing> Mesh> Volume Sweep> Sweep**) to fill an existing unmeshed volume with elements by sweeping the mesh from an adjacent area through the volume.

When you use the **VROTAT**, **VEXT**, **VOFFST**, and **VDRAG** commands to extrude a meshed area into a meshed volume, ANSYS creates the volume and the volume mesh simultaneously. In contrast, you use the **VSWEEP** command in an existing unmeshed volume. Thus **VSWEEP** is particularly useful when you are importing a solid model that was created in a CAD program.

For detailed information about volume sweeping, see Generating a Volume Mesh By Sweeping (p. 154) of this manual, as well as the description of the **VSWEEP** command in the *Command Reference*.

5.3. Creating Your Solid Model from the Top Down: Primitives

In top down construction, you use geometric primitives (fully-defined lines, areas, and volumes) to assemble your model. As you create a primitive, the program automatically creates all the “lower” entities associated with it. A *geometric primitive* is a commonly used solid modeling shape (such as a sphere or regular prism) that can be created with a single ANSYS command.

Because primitives are higher-order entities that can be constructed *without* first defining any keypoints, model generation that uses primitives is sometimes referred to as "top down" modeling. (When you create a primitive, the program automatically creates all the necessary lower-order entities, including keypoints.) Geometric primitives are created within the working plane.

You can freely combine bottom up and top down modeling techniques, as appropriate, in any model. Remember that geometric primitives are built within the working plane while bottom up techniques

are defined against the active coordinate system. If you are mixing techniques, you may wish to consider using the **CSYS**,WP or **CSYS**, 4 command to force the coordinate system to follow the working plane.

Caution

Solid modeling operations in a toroidal coordinate system are not recommended. Areas or volumes generated may not be what you expect.

5.3.1. Creating Area Primitives

Any area primitives you create will lie flat on the working plane and will be oriented according to the working plane coordinate system. Area primitives must have surface areas greater than zero (that is, you cannot create a degenerate area as a means of defining a line).

The interface between two touching primitives will create a seam of discontinuity in the finite element model, unless you take steps to "weld" that seam shut, using commands such as **NUMMRG**, **AADD**, or **AGLUE**.

You can define area primitives using the methods described in the following table.

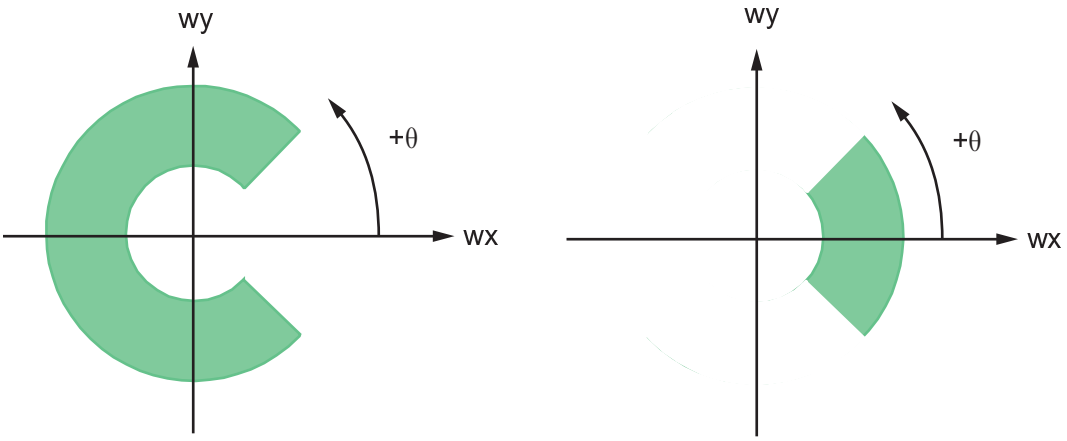
Create a	Command	GUI
rectangular area anywhere on the working plane	**RECTNG**	**Main Menu> Preprocessor> Modeling> Create> Areas> Rectangle> By Dimensions**
rectangular area by corner points	**BLC4**	**Main Menu> Preprocessor> Modeling> Create> Areas> Rectangle> By 2 Corners**
rectangular area by center and corner points	**BLC5**	**Main Menu> Preprocessor> Modeling> Create> Areas> Rectangle> By Centr & Cornr**
circular area centered about the working plane origin	**PCIRC**	**Main Menu> Preprocessor> Modeling> Create> Circle> By Dimensions**
circular area anywhere on the working plane	**CYL4**	**Main Menu> Preprocessor> Modeling> Create> Circle> Annulus** or **> Partial Annulus** or **> Solid Circle**
circular area by end points	**CYL5**	**Main Menu> Preprocessor> Modeling> Create> Circle> By End Points**
regular polygonal area centered about the working plane origin	**RPOLY**	**Main Menu> Preprocessor> Modeling> Create> Polygon> By Circumscr Rad** or **> By Inscribed Rad** or **> By Side Length**
regular polygonal area anywhere on the working plane	**RPR4**	**Main Menu> Preprocessor> Modeling> Create> Polygon> Hexagon** or **> Octagon** or **> Pentagon** or **> Septagon** or **> Square** or **> Triangle**
arbitrary polygonal area based on working plane coordinate pairs	**POLY** [1]	This command cannot be accessed from a menu.

1. You must use the **PTXY** command to define coordinate pairs before issuing the **POLY** command.

When you define an arc segment of a circular geometric primitive (**PCIRC** and **CYL4** discussed above, or **CONE**, **CYLIND**, **SPHERE**, and **TORUS** discussed in the next section on volume primitives) the arc sector *begins* at the algebraically smaller angle, extends in a positive angular direction, and *ends* at the

larger angle. (The input order of *THETA1*, *THETA2* on these commands does *not* define the starting and ending angles of the arc sector.) The following figure illustrates how these commands work:

Figure 5.15: Arc Sectors of Circular Geometric Primitives



PCIRC,1,2,45,315
PCIRC,1,2,315,45

PCIRC,1,2,-45,45
PCIRC,1,2,45,-45

5.3.2. Creating Volume Primitives

Volume primitives are positioned relative to the working plane as outlined in their command descriptions.

The interface between two touching primitives will create a seam of discontinuity in the finite element model, unless you take steps to "weld" that seam shut, using commands such as **NUMMRG**, **VGLUE**, or **VADD**.

You can define volume primitives using the methods described in the following table.

Create a	Com-mand	GUI
block volume based on working plane coordinates	**BLOCK**	**Main Menu> Preprocessor> Modeling> Create> Volumes> Block> By Dimensions**
block volume by corner points	**BLC4**	**Main Menu> Preprocessor> Modeling> Create> Volumes> Block> By 2 Corners & Z**
block volume by center and corner points	**BLC5**	**Main Menu> Preprocessor> Modeling> Create> Volumes> Block> By Centr,Cornr,Z**
cylindrical volume centered about the working plane origin	**CYLIND**	**Main Menu> Preprocessor> Modeling> Create> Volumes> Cylinder> By Dimensions**
cylindrical volume anywhere on the working plane	**CYL4**	**Main Menu> Preprocessor> Modeling> Create> Volumes> Cylinder> Hollow Cylinder** or **> Partial Cylinder** or **> Solid Cylinder**
cylindrical volume by end points	**CYL5**	**Main Menu> Preprocessor> Modeling> Create> Volumes> Cylinder> By End Pts & Z**

Create a	Command	GUI
regular prism volume centered about the working plane origin	**RPRISM**	**Main Menu> Preprocessor> Modeling> Create> Volumes> Prism> By Circumscr Rad** or **> By Inscribed Rad** or **> By Side Length**
prism volume anywhere on the working plane	**RPR4**	**Main Menu> Preprocessor> Modeling> Create> Volumes> Prism> Hexagonal** or **> Octagonal** or **> Pentagonal** or **> Septagonal** or **> Square** or **> Triangular**
arbitrary prism based on working plane coordinate pairs	**PRISM** [1]	This command cannot be accessed from a menu.
spherical volume centered about the working plane origin	**SPHERE**	**Main Menu> Preprocessor> Modeling> Create> Volumes> Sphere> By Dimensions**
spherical volume anywhere on the working plane	**SPH4**	**Main Menu> Preprocessor> Modeling> Create> Volumes> Sphere> Hollow Sphere** or **> Solid Sphere**
spherical volume by diameter end points	**SPH5**	**Main Menu> Preprocessor> Modeling> Create> Volumes> Sphere> By End Points**
conical volume centered about the working plane origin	**CONE**	**Main Menu> Preprocessor> Modeling> Create> Volumes> Cone> By Dimensions**
conical volume anywhere on the working plane	**CON4**	**Main Menu> Preprocessor> Modeling> Create> Volumes> Cone> By Picking**
toroidal volume [2]	**TORUS**	**Main Menu> Preprocessor> Modeling> Create> Volumes> Torus**

1. You must use the **PTXY** command to define coordinate pairs before issuing the **PRISM** command.

2. See Creating a Torus or Toroidal Sector (p. 55) for more information on toroidal volumes.

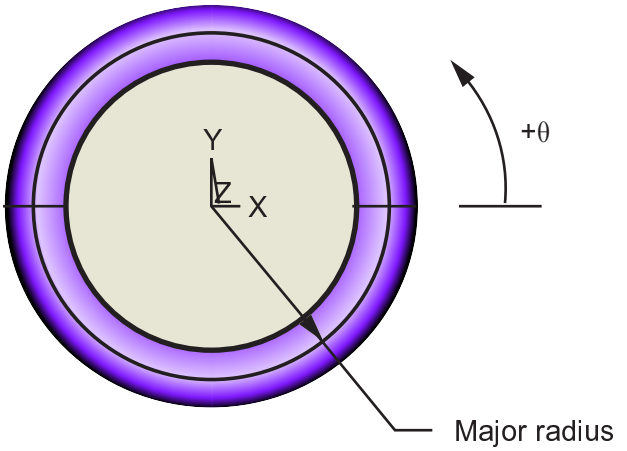
5.3.2.1. Creating a Torus or Toroidal Sector

You can use the **TORUS**,*RAD1*,*RAD2*,*RAD3*,*THETA1*,*THETA2* command to create either a *torus* or a *toroidal sector*. To create a *torus*, you do not need to specify values for *THETA1* or *THETA2*. You must specify three values to define the radii of the torus (*RAD1*, *RAD2*, and *RAD3*). You can specify the radii in any order. The smallest of the values is the inner minor radius, the intermediate value is the outer minor radius, and the largest value is the major radius. (There is one exception regarding the order of the radii values - if you want to create a solid torus, specify zero or blank for the inner minor radius, in which case the zero or blank *must* occupy either the *RAD1* or *RAD2* position.) At least two of the values that you specify must be positive values; they will be used to define the outer minor radius and the major radius.

To create the torus shown in Figure 5.16: Torus Primitive (p. 56), the command **TORUS**,5,1,2 was issued. Due to the sizes of the specified radii values relative to one another, 5, 1, and 2 were used to define the major radius, inner minor radius, and outer minor radius of the torus, respectively. Since no values for *THETA1* and *THETA2* were specified, the default values of 0 and 360 were used as the starting and ending angles of the torus. (See Figure 5.17: Toroidal Sector (p. 56) for a view of a toroidal sector showing all radii.)

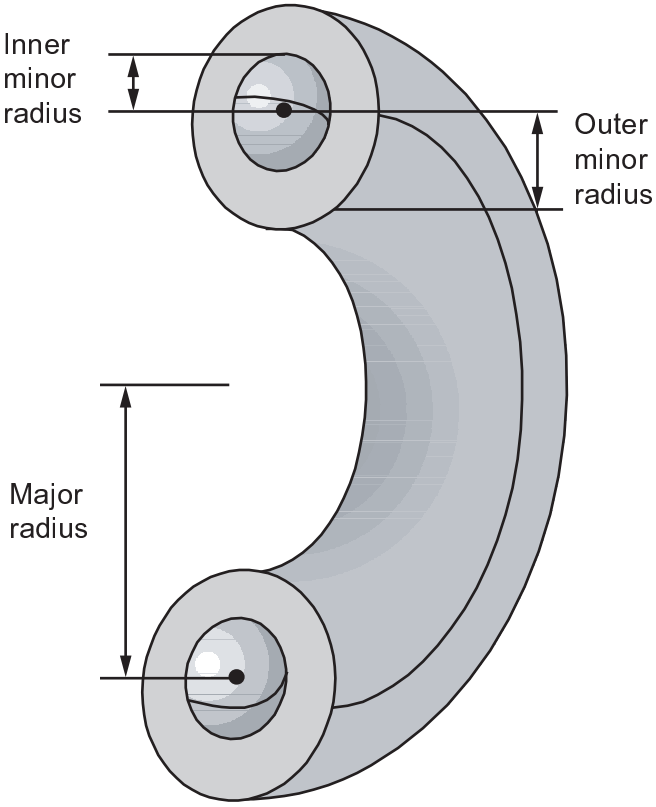
See the description of the **TORUS** command in the *Command Reference* for additional details.

Figure 5.16: Torus Primitive



To create the *toroidal sector* shown in Figure 5.17: Toroidal Sector (p. 56), the command **TORUS**,5,1,2,0,180 was issued; where 5, 1, and 2 are the major radius, inner minor radius, and outer minor radius of the torus; and 0 and 180 are the starting and ending angles of the torus.

Figure 5.17: Toroidal Sector



5.4. Sculpting Your Model with Boolean Operations

Boolean algebra provides a means for combining *sets* of data, using such logical operators as intersect, union, subtract, etc. The ANSYS program allows you to apply these same Boolean operators to your solid model, so that you can modify your solid model constructions more easily.

You can apply Boolean operations to almost *any* solid model construction, whether it was created from the top down or from the bottom up. The only exceptions are that Boolean operations are not valid

for entities created by concatenation (see Free or Mapped Mesh (p. 101)) and that some Boolean operations cannot always be performed on entities that contain degeneracies. (See Solid Model Loads (p. 83) later in this chapter.)

Also, all solid-model loads and element attributes should be defined *after* you complete your Boolean operations. If you are using Booleans to modify an existing model, you should take care to redefine your element attributes and solid-model loads.

Note

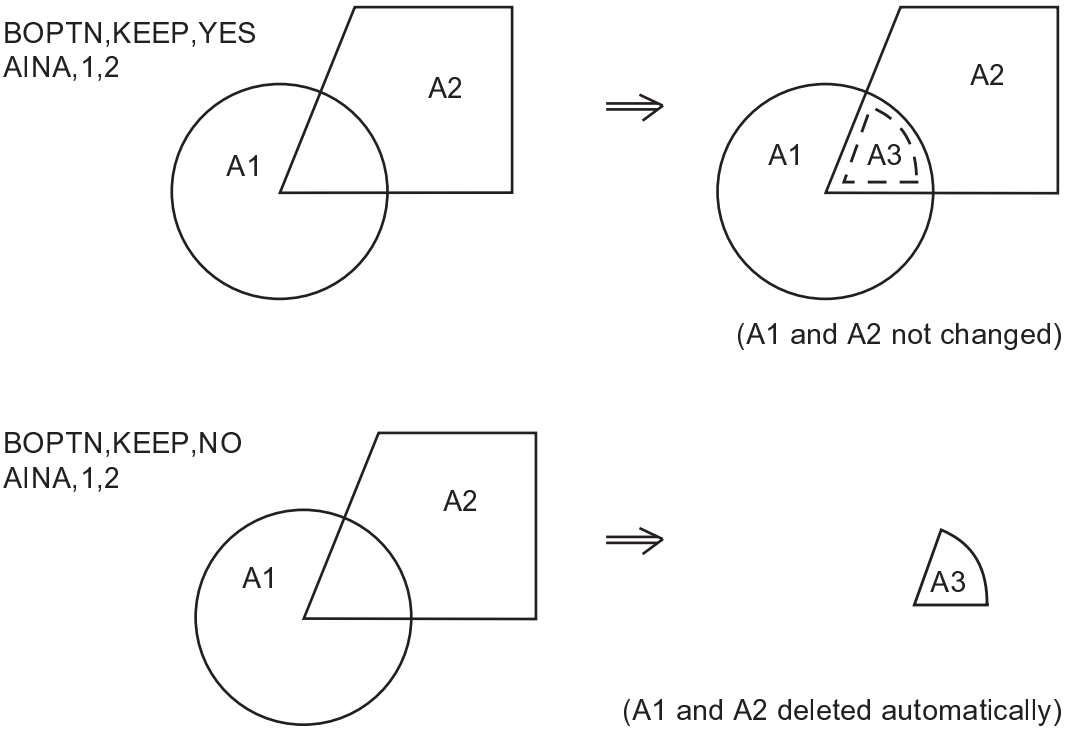
Boolean operations and other solid modeling operations can be unreliable, either failing to work or corrupting/contaminating the database. You should SAVE a copy of your database before a Boolean operation in order to cleanly recover from a failure. See Considerations and Cautions for Solid Modeling (p. 86) for additional information.

5.4.1. Boolean Operation Settings

You can specify Boolean operation options using the **BOPTN** command (**Main Menu> Preprocessor> Modeling> Operate> Booleans> Settings**).

When you perform a Boolean operation on two (or more) entities, you are faced with the decision of whether or not you want to keep the original entities. You can control this action with the KEEP label of the **BOPTN**,*Lab*,*Value* command, as illustrated schematically in Figure 5.18: Boolean Keep Options (p. 57):

Figure 5.18: Boolean Keep Options



Boolean operations on lower-order entities that are attached to higher-order entities are generally permitted.

Boolean operations cannot be performed on meshed entities. You must clear the mesh from the entity before performing the Boolean operation.

The label NWARN on the **BOPTN** command allows you to control warning messages. A value of "0" will result in a warning message if a Boolean operation has no effect. A value of "1" will suppress all warning or error messages if a Boolean operation has no effect. A value of "-1" will allow error messages if a Boolean operation has no effect. The default value of this label is "0".

The VERSION label can be used to control which numbering scheme ANSYS will use to number the entities produced by Boolean operations. By default, ANSYS will number entities using the Revision 5.2 numbering scheme, but you can force ANSYS to use the Revision 5.1 numbering scheme instead. Normally, you will want to use the default numbering scheme when running Revision 5.2 or newer versions. However, if you are reading input that was created at Revision 5.1, you should activate Revision 5.1 numbering [**BOPTN**,VERSION,RV51] before reading in the file [**/INPUT**] so that the input will run properly.

Note

A command input stream created at Revision 5.1 may produce different entity numbering at Revision 5.2 or Revision 5.3 unless the Revision 5.1 numbering scheme is specified [**BOPTN**,VERSION,RV51].

The label DEFA will reset all **BOPTN** settings to default values. The label STAT lists the status of present settings.

5.4.2. Entity Numbering After Boolean Operations

The numbering scheme assigns numbers to Boolean output entities based on information relating to their *topology* and *geometry*. The *topology* information used for an area, for example, includes the number of loops it is defined by, the number of lines (that is, 3-sided area, 4-sided area, etc.) making up the area, the line numbers of any original lines (lines existing before the Boolean operation) in the area, keypoint numbers of any original keypoints, etc. The *geometry* information used for an area consists of the coordinates of its centroid, endpoints, and other "control points" relative to some arbitrary reference coordinate system. "Control points" are defined by the NURBS used to parametrically describe your model. The numbering scheme first assigns numbers (beginning with the next available number) to those output entities that can be uniquely identified by their topology. Any remaining entities are then assigned numbers based on their geometry. Unfortunately, entity numbering based on geometry may not be consistent across runs where the geometry changes. Therefore, when geometry-based entity numbering occurs, the program issues the following warning message:

```
*** WARNING ***

Entity numbers from the Boolean operation were assigned based on geometry.
If you are planning to do optimization, (or input looping), do not rely
on the entity numbers for loads, etc.  To suppress this warning, issue "BOPT,NWARN,0".
```

5.4.3. Intersect

An *intersection* defines a new set of entities which is common to every original entity included in the operation. In other words, an intersection represents the region of overlap of two or more entities. The new set can be of the same or lower dimension as the original entities. For instance, the intersection of two lines can be a keypoint (or a set of keypoints), or it can be a line (or set of lines). The Boolean intersect commands are as follows:

Create the intersection of	Com-mand	GUI
lines	**LINL**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Intersect> Common> Lines**
areas	**AINA**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Intersect> Common> Areas**
volumes	**VINV**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Intersect> Common> Volumes**
a line with an area	**LINA**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Intersect> Line with Area**
an area with a volume	**AINV**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Intersect> Area with Volume**
a line with a volume	**LINV**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Intersect> Line with Volume**

5.4.3.1. Illustrations of Intersection Operations

The following figures illustrate the intersection operations listed above:

Figure 5.19: LINL (Line Intersect Line)

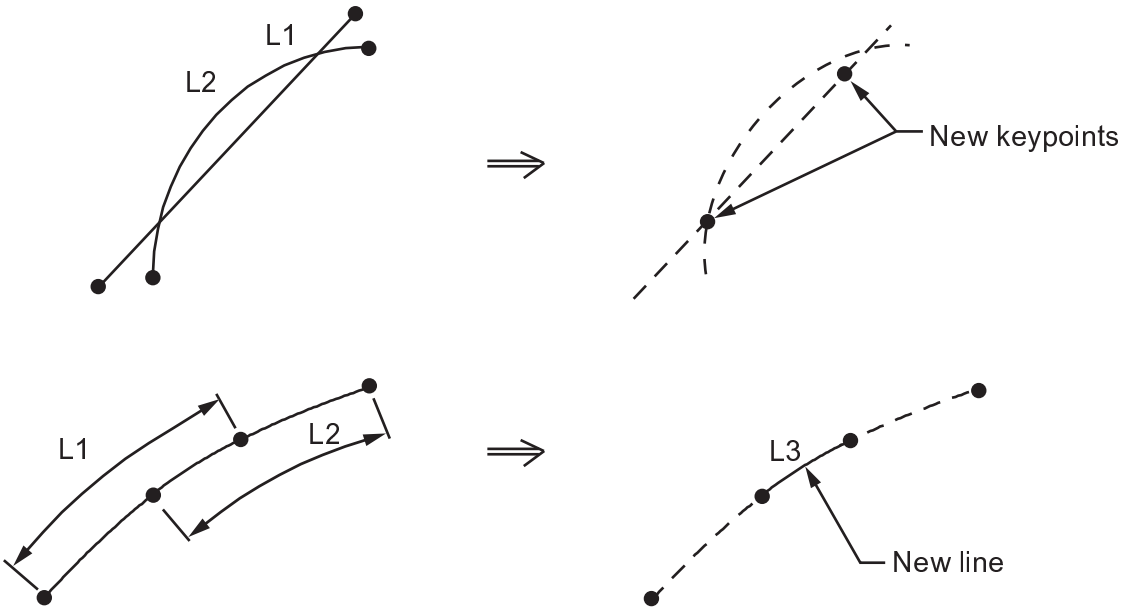


Figure 5.20: AINA (Area Intersect Area)

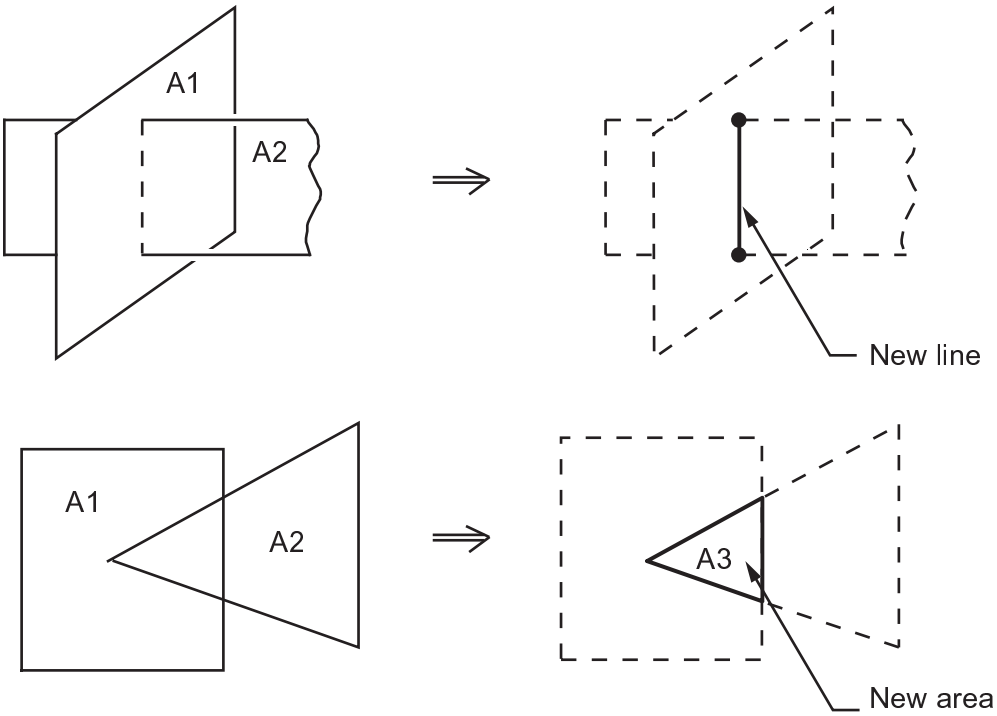


Figure 5.21: VINV (Volume Intersect Volume)

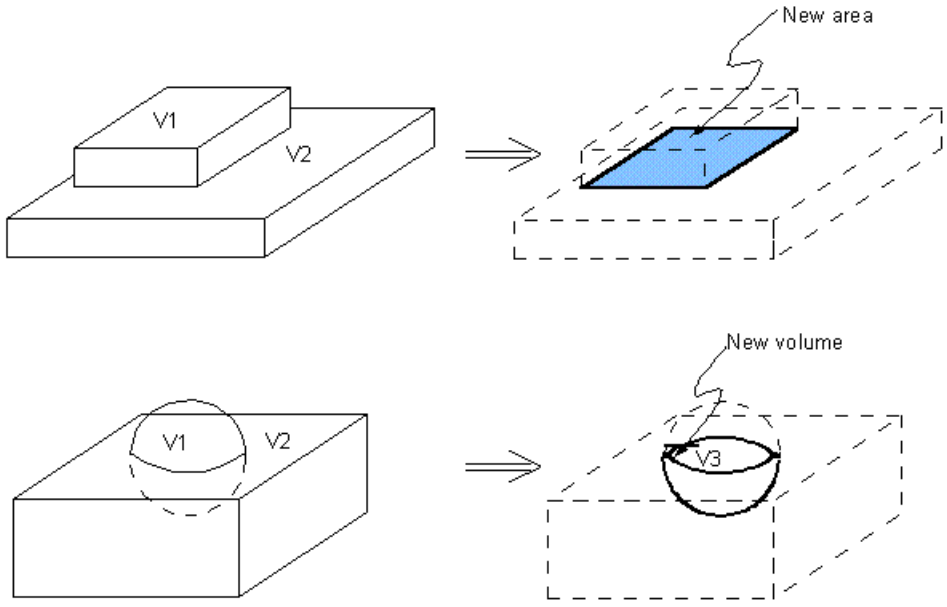


Figure 5.22: LINA (Line Intersect Area)

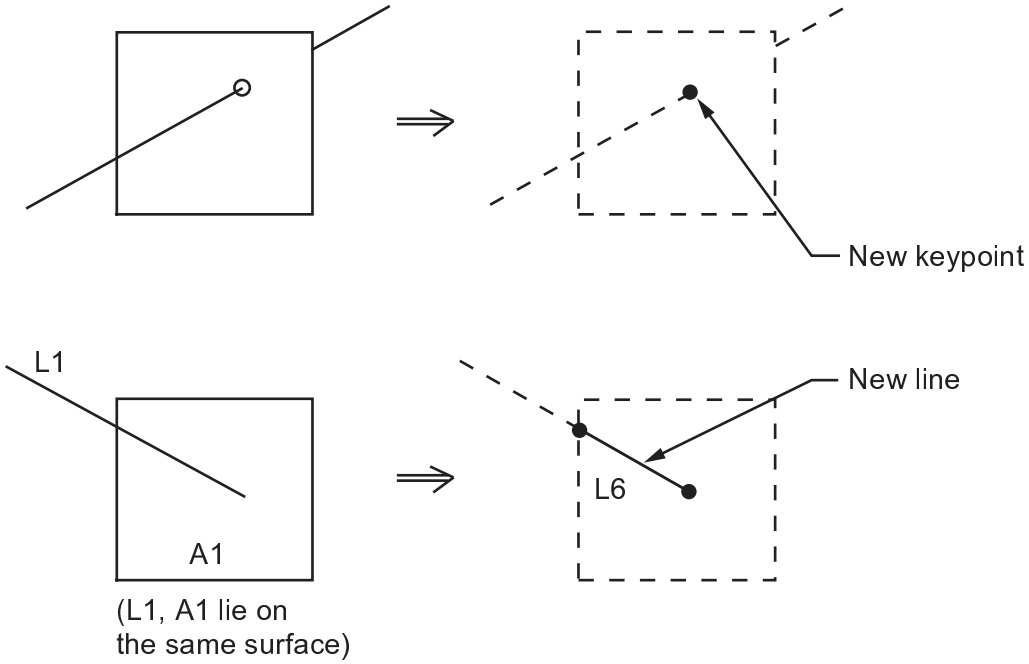


Figure 5.23: AINV (Area Intersect Volume)

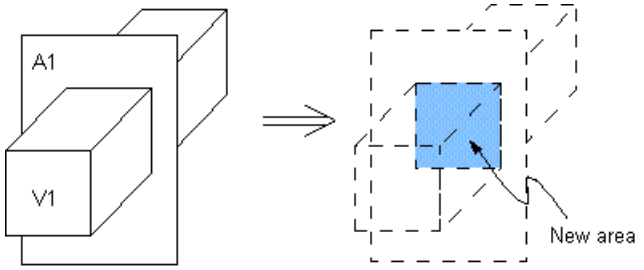
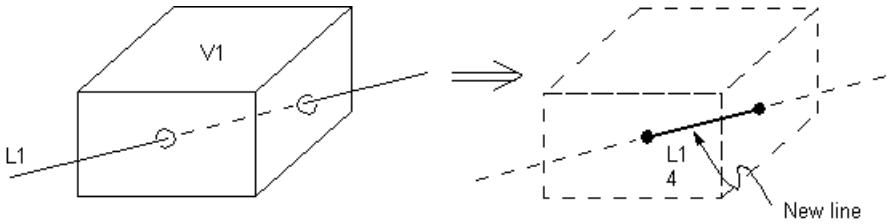


Figure 5.24: LINV (Line Intersect Volume)



5.4.4. Pairwise Intersect

A pairwise intersection defines a new set of entities which is any overlapping set of entities included in the operation. In other words, a pairwise intersection represents the region of overlap of at least any two of the original entities. The new set can be of the same or lower dimension as the original entities. For instance, the pairwise intersection of a set of lines can be a keypoint (or a set of keypoints), or it can be a line (or set of lines). The Boolean pairwise intersect commands are as follows:

Find the pairwise intersection of	Command	GUI
lines	**LINP**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Intersect> Pairwise> Lines**
areas	**AINP**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Intersect> Pairwise> Areas**
volumes	**VINP**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Intersect> Pairwise> Volumes**

5.4.4.1. Illustrations of Pairwise Intersection Operations

Figure 5.25:**LINP** (Line Intersect Pairwise) (p. 62), Figure 5.26:**AINP** (Area Intersect Pairwise) (p. 63), and Figure 5.27:**VINP** (Volume Intersect Pairwise) (p. 64) illustrate the pairwise intersection operations listed above:

Figure 5.25: LINP (Line Intersect Pairwise)

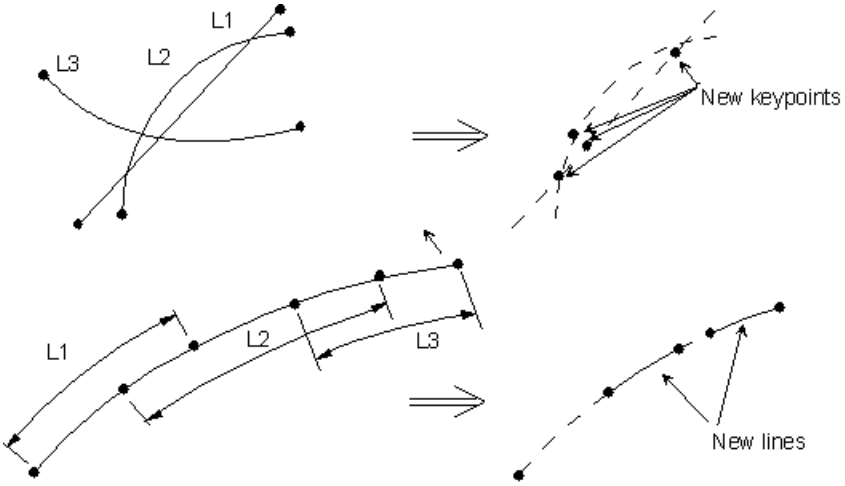


Figure 5.26: AINP (Area Intersect Pairwise)

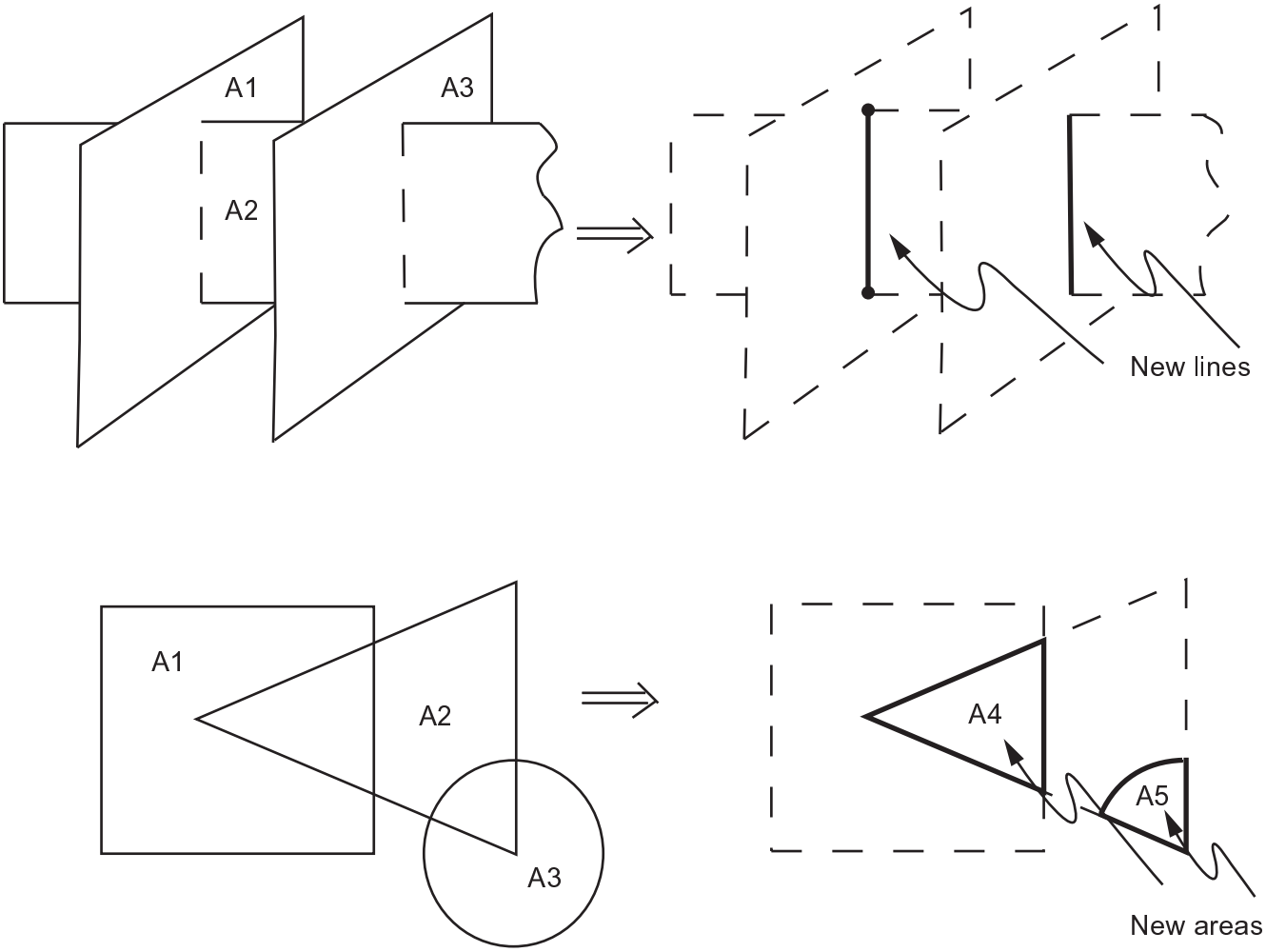
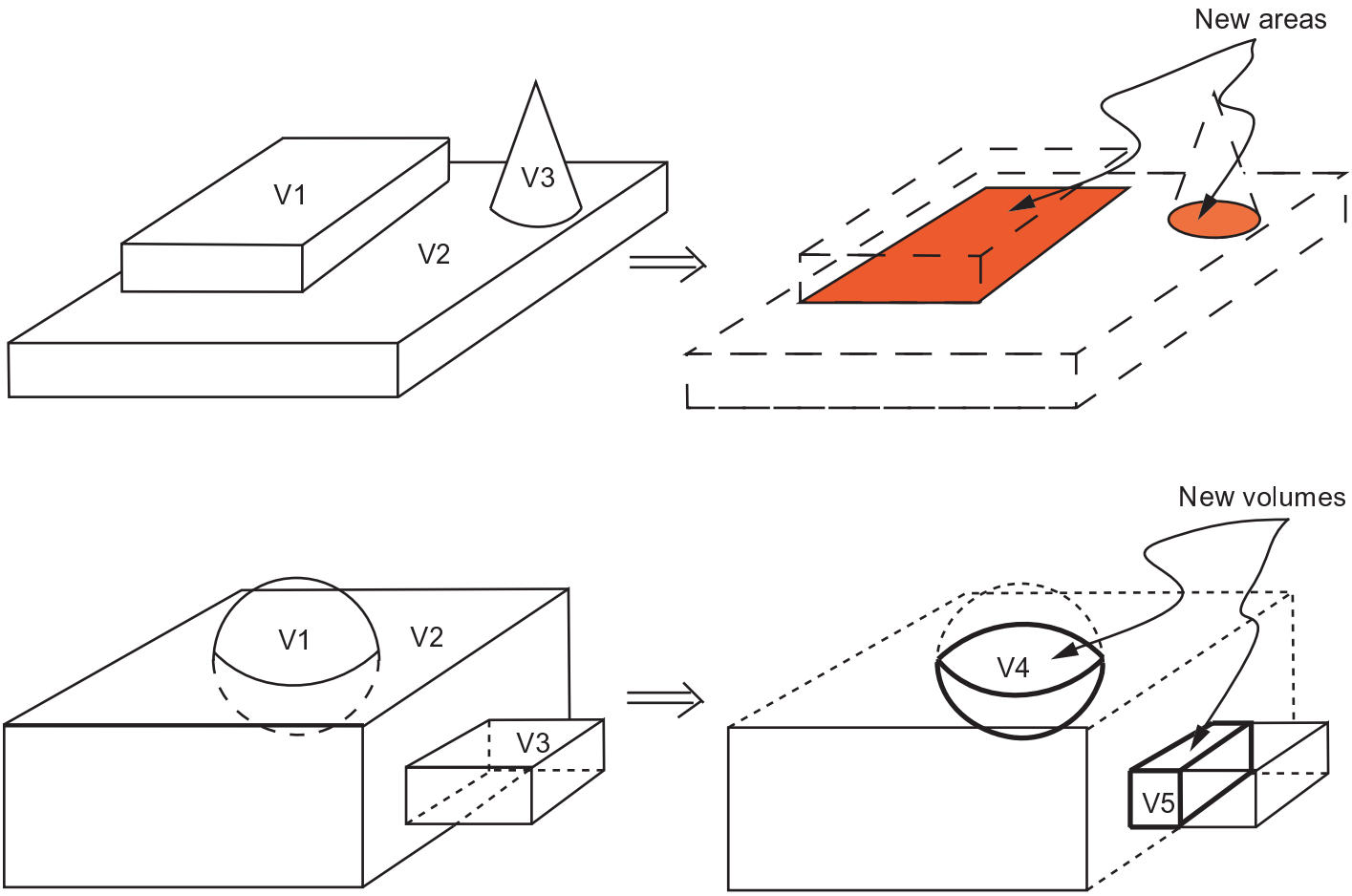


Figure 5.27: VINP (Volume Intersect Pairwise)



5.4.5. Add

An *addition* of entities defines a new entity that includes all parts of the originals. (This operation is also known mathematically as a *union*, *joining,* or *summation*.) The resulting entity is a single seamless whole, containing no internal divisions. (As a practical matter, "added" entities will often not mesh as well as will "overlapped" entities.) Only volumes or coplanar 2-D areas can be added in the ANSYS program. Areas added may contain holes within the area; that is, internal loops. The Boolean add commands are as follows:

Add	Com-mand	GUI
separate areas to create a single area	AADD	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Add> Areas**
separate volumes to create a single volume	VADD	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Add> Volumes**

5.4.5.1. Illustrations of Addition Operations

The following figures illustrate the add operations listed above.

Figure 5.28: AADD (Add Areas)

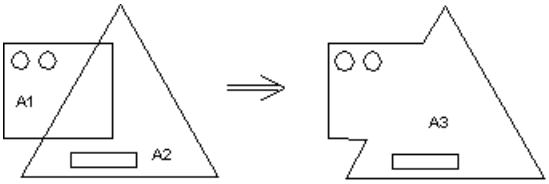
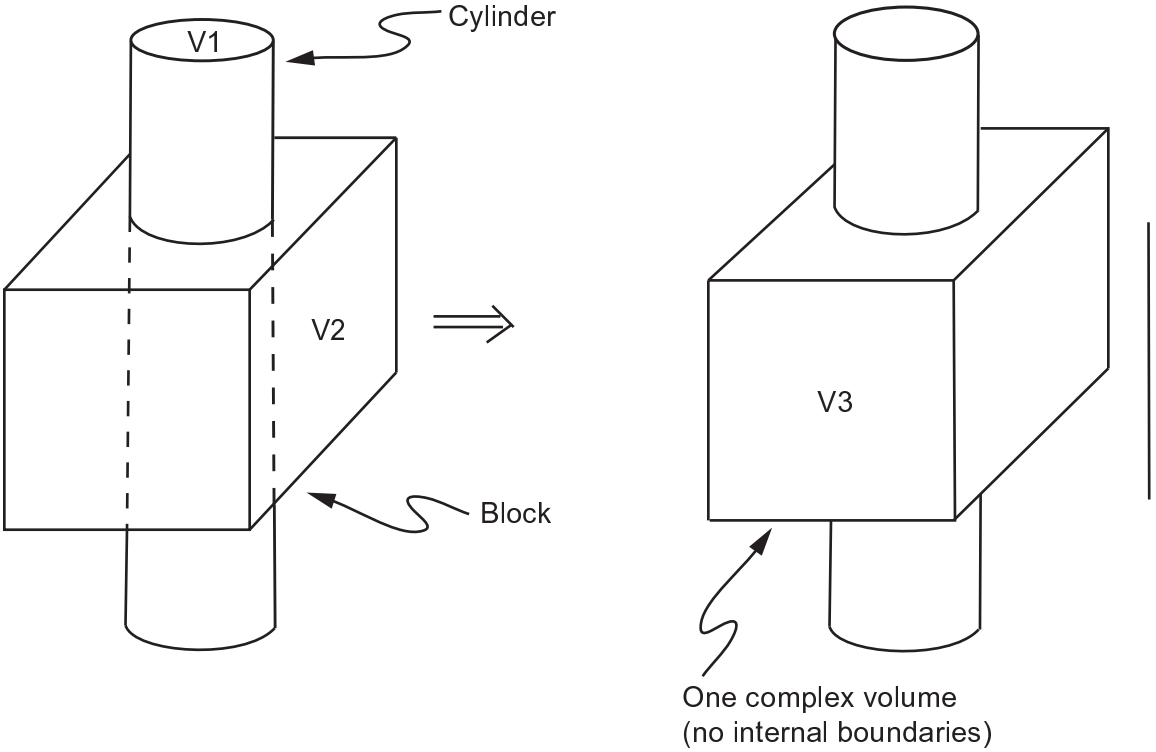


Figure 5.29: VADD (Add Volumes)



5.4.6. Subtract

If you subtract one entity (E2) from another (E1), you will obtain one of two results: Either you will create a new entity or entities (E1 - E2 $\geq$ E3) that is of the same dimensionality as E1 and that contains no overlap with E2, or, if the overlap is of a lower dimensionality, you will simply divide E1 into two or more new entities (E1 - E2 $\geq$ E3 and E4).

If the command field *SEPO* on the subtract command is set to blank (default), the subtraction of entities can result in lines with a common end point, or areas with a common line boundary, or volumes sharing a common boundary area. If the command field is set to "SEPO", the resulting entities will no longer share common boundaries but have distinct but coincident boundaries. This latter operation is not valid if the overlap of entities does not divide one of the input entities into at least two distinct lines, areas, or volumes. The Boolean subtract commands (and their corresponding GUI paths) are as follows:

Subtract	Com-mand	GUI
lines from lines	LSBL	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Subtract> Lines**

Subtract	Com-mand	GUI
		Main Menu> Preprocessor> Modeling> Operate> Booleans> Subtract> With Options> Lines **Main Menu> Preprocessor> Modeling> Operate> Booleans> Divide> Line by Line** **Main Menu> Preprocessor> Modeling> Operate> Booleans> Divide> With Options> Line by Line**
areas from areas	**ASBA**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Subtract> Areas** **Main Menu> Preprocessor> Modeling> Operate> Booleans> Subtract> With Options> Areas** **Main Menu> Preprocessor> Modeling> Operate> Booleans> Divide> Area by Area** **Main Menu> Preprocessor> Modeling> Operate> Booleans> Divide> With Options> Area by Area**
volumes from volumes	**VSBV**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Subtract> Volumes** **Main Menu> Preprocessor> Modeling> Operate> Booleans> Subtract> With Options> Volumes**
areas from lines	**LSBA**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Divide> Line by Area** **Main Menu> Preprocessor> Modeling> Operate> Booleans> Divide> With Options> Line by Area**
volumes from lines	**LSBV**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Divide> Line by Volume** **Main Menu> Preprocessor> Modeling> Operate> Booleans> Divide> With Options> Line by Volume**
volumes from areas	**ASBV**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Divide> Area by Volume** **Main Menu> Preprocessor> Modeling> Operate> Booleans> Divide> With Options> Area by Volume**
lines from areas	**ASBL** [1]	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Divide> Area by Line** **Main Menu> Preprocessor> Modeling> Operate> Booleans> Divide> With Options> Area by Line**
areas from volumes	**VSBA**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Divide> Volume by Area** **Main Menu> Preprocessor> Modeling> Operate> Booleans> Divide> With Options> Volume by Area**

1. The *SEP0* field is not available on the **ASBL** command.

You can subtract multiple entities from a single entity. All entity subtraction commands are of the form *e*SB*e,* where "*e*" represents multiple or single entities.

You can set either entity field value of the subtract operation to ALL . If ALL is used in the minuend field, the entity or entities listed or picked in the subtrahend field will be removed from all selected entities. If ALL is used in the subtrahend field, all selected entities will be subtracted from those listed in the minuend field. If ALL is used in both minuend and subtrahend fields for subtraction of like entities, nothing will happen; that is, the result will be your input entities.

The *KEEP(X)* argument fields of the entity subtraction commands allow you to selectively keep or delete entities. For instance, the *KEEPA* and *KEEPL* arguments on the **ASBL** command allow you to keep or delete the areas and/or lines used in an ASBL operation. This is in contrast to the **BOPTN**,KEEP,*Value* command which demands you either keep or delete all input entities. The *KEEPL* and *KEEPA* arguments override previous settings made with the **BOPTN** command (**Main Menu> Preprocessor> Modeling> Operate> Booleans> Settings**). If these two fields are left blank, the default settings are controlled by the **BOPTN** command. The default setting for **BOPTN** is to delete all entities that are used as inputs to entity subtraction commands.

5.4.6.1. Illustrations of Subtraction Operations

Figure 5.30:**LSBL** (Line Subtract Line) (p. 67) through Figure 5.37:**VSBA** (Volume Subtract Area) (p. 70) illustrate simple entity subtraction operations. See the descriptions of the **LSBL**, **ASBA**, **VSBV**, **LSBA**, **LSBV**, **ASBV**, **ASBL**, and **VSBA** commands in the *Command Reference* for more information.

Figure 5.30: LSBL (Line Subtract Line)

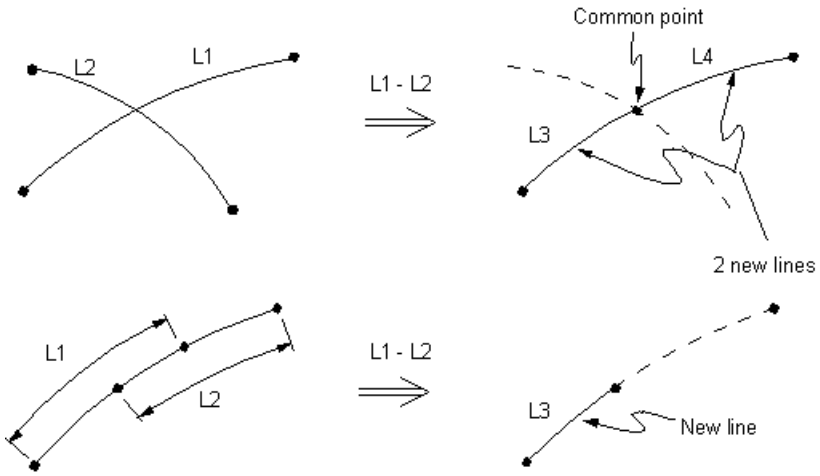


Figure 5.31: ASBA (Area Subtract Area)

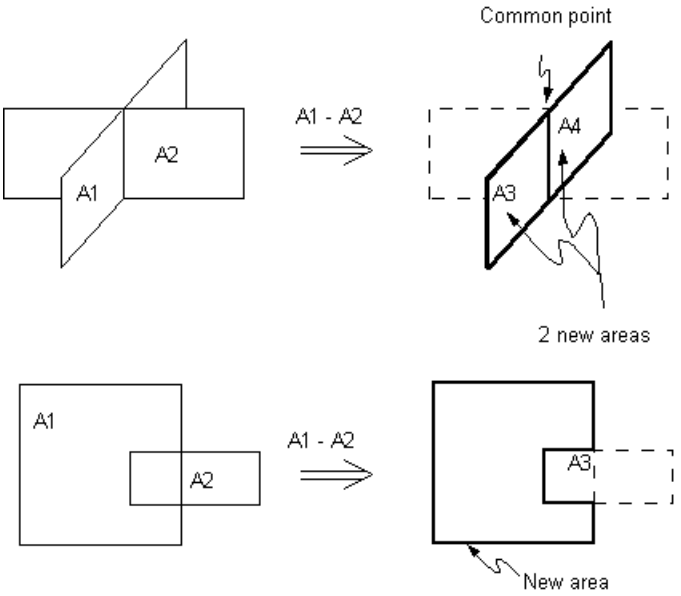


Figure 5.32: VSBV (Volume Subtract Volume)

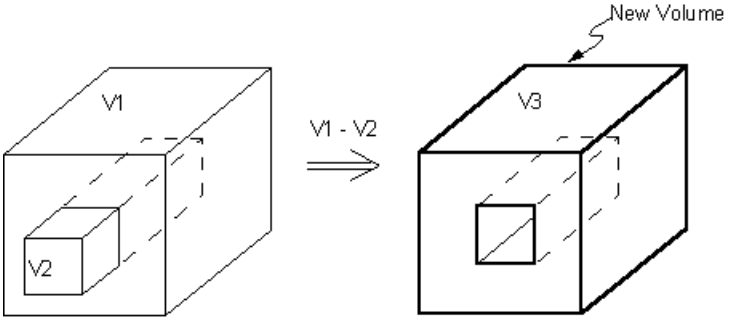


Figure 5.33: LSBA (Line Subtract Area)

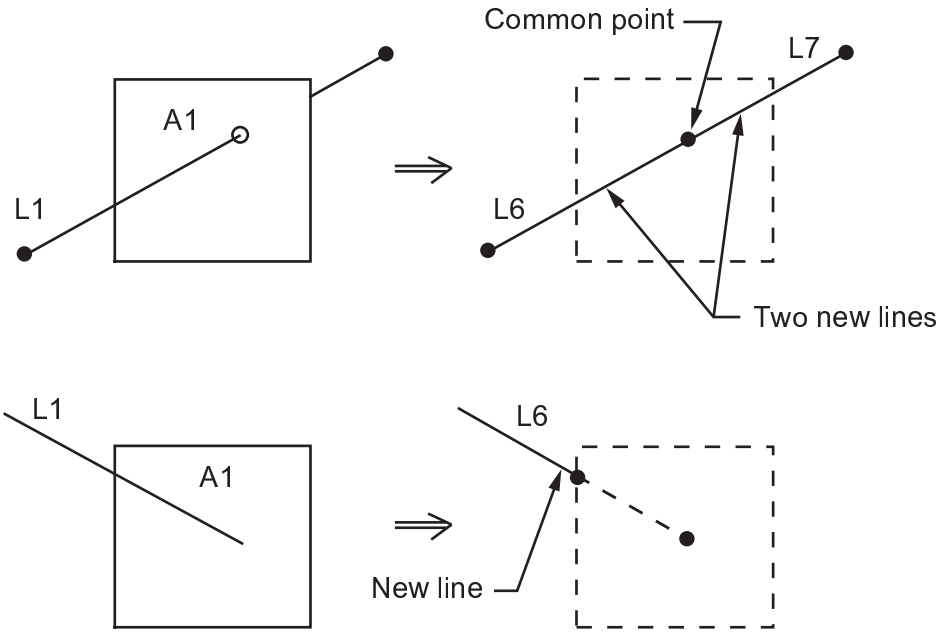


Figure 5.34: LSBV (Line Subtract Volume)

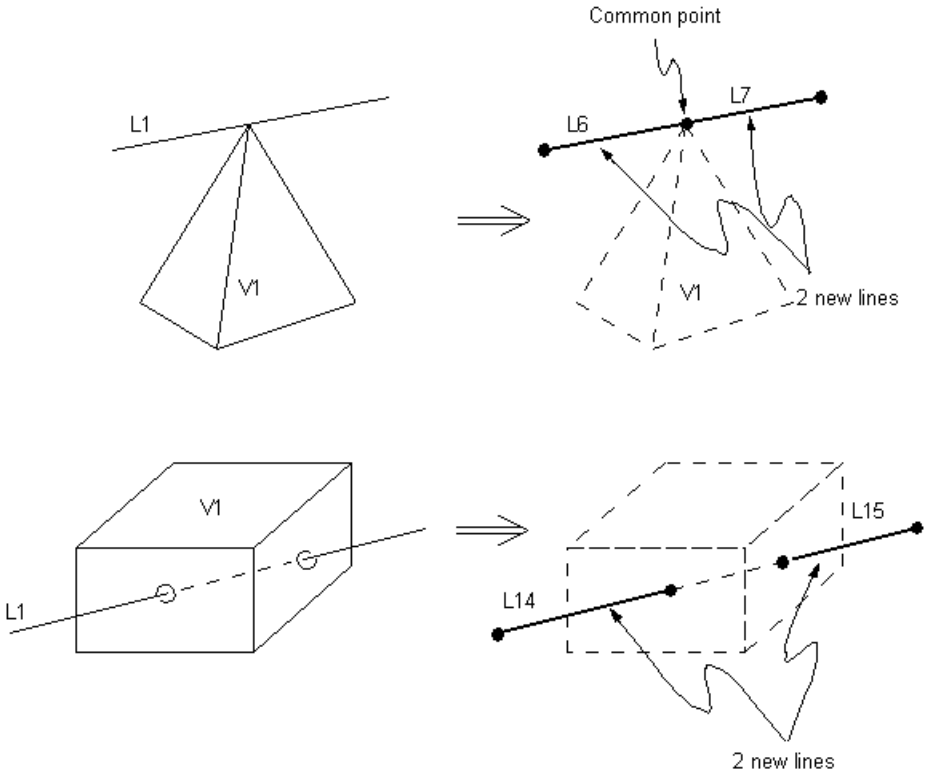


Figure 5.35: ASBV (Area Subtract Volume)

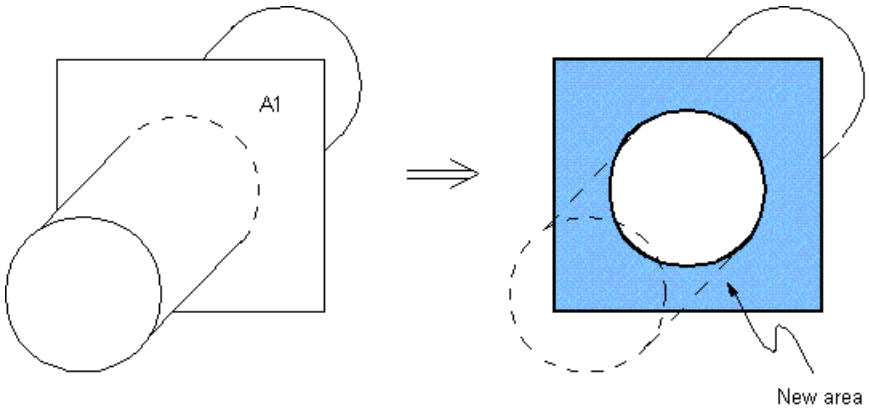


Figure 5.36: ASBL (Area Subtract Line)

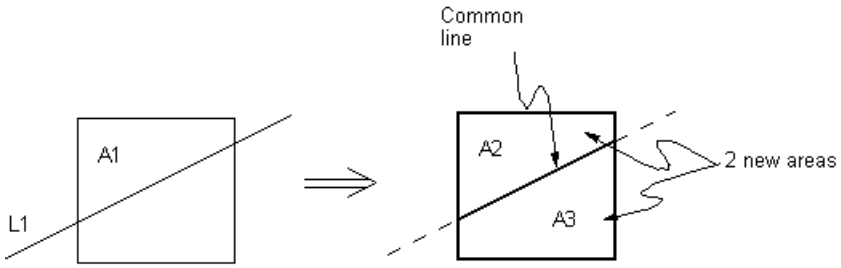
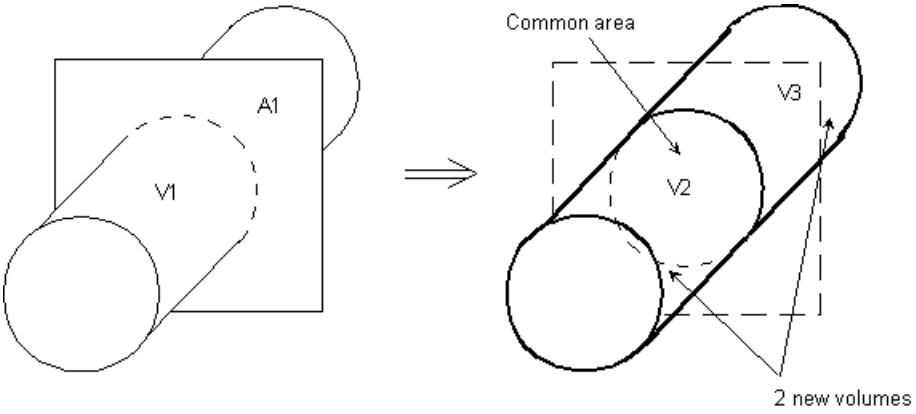


Figure 5.37: VSBA (Volume Subtract Area)



The entity subtraction commands can have multiple inputs. You can subtract one entity from multiple entities, or you can subtract multiple entities from one entity. You may also subtract multiple entities from multiple entities. Figure 5.38:**LSBL** (Multiple Line Subtract a Line) (p. 70) to Figure 5.45:**VSBA** (Single Volume Subtract Multiple Areas) (p. 72) illustrate multiple entity subtractions.

Figure 5.38: LSBL (Multiple Line Subtract a Line)

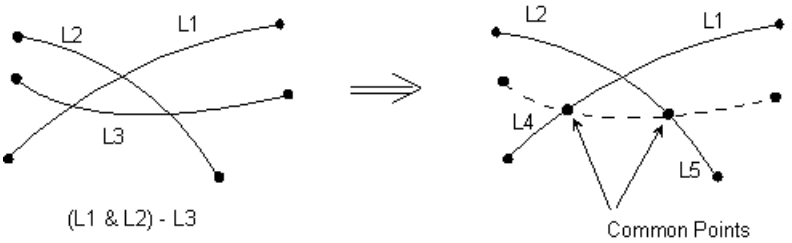


Figure 5.39: ASBA (Multiple Area Subtract an Area)

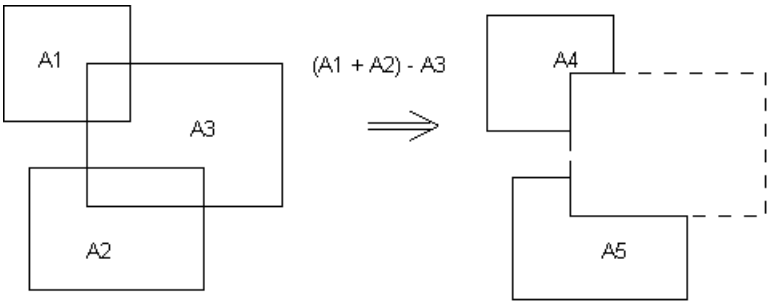


Figure 5.40: VSBV (Multiple Volume Subtract a Volume)

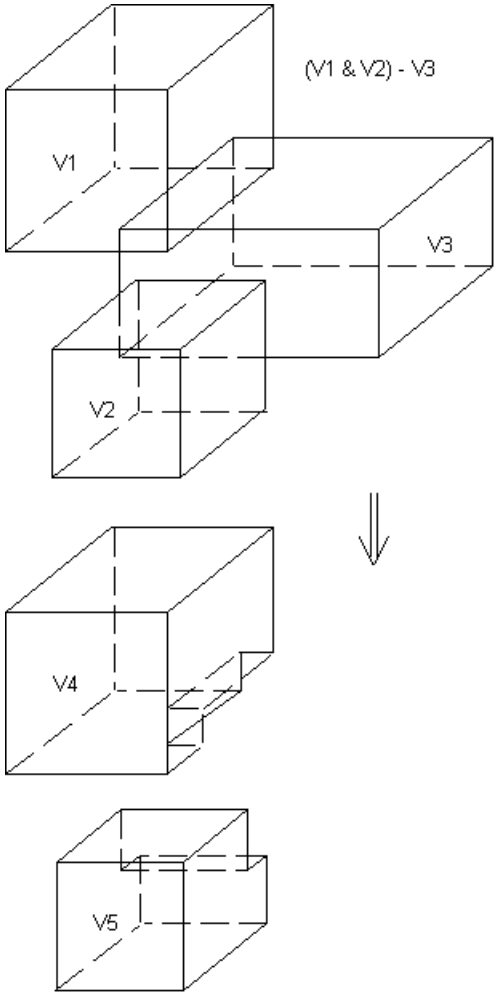


Figure 5.41: LSBA (Multiple Line Subtract an Area)

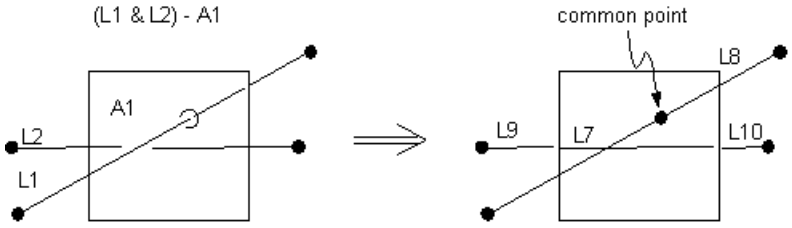


Figure 5.42: LSBV (Multiple Line Subtract a Volume)

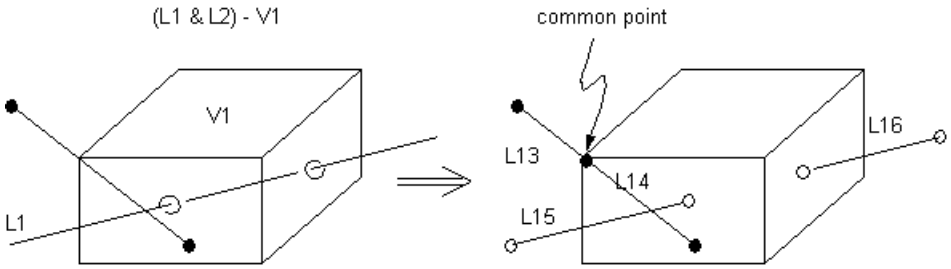


Figure 5.43: ASBV (Multiple Area Subtract a Volume)

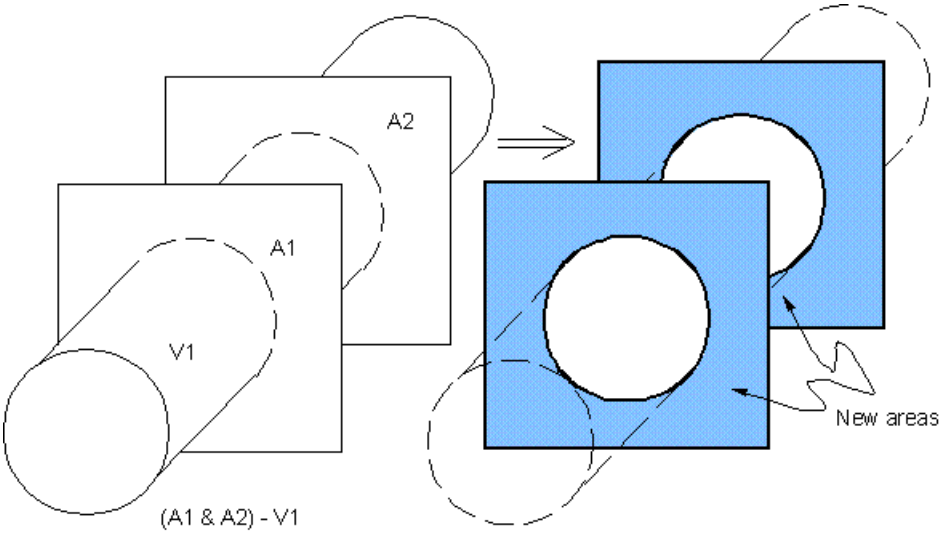


Figure 5.44: ASBL (Multiple Area Subtract a Line)

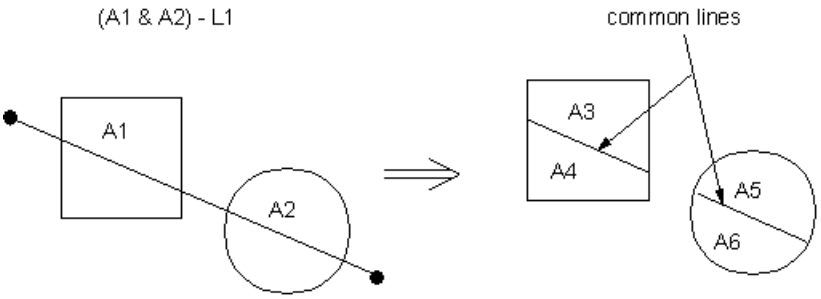
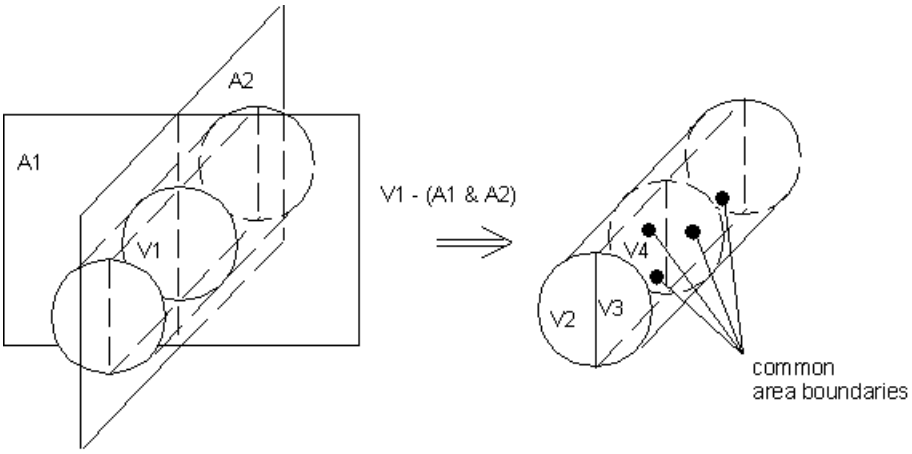


Figure 5.45: VSBA (Single Volume Subtract Multiple Areas)



5.4.7. Working Plane Subtract

The working plane can be subtracted from an entity to divide it into two or more entities. The working plane can be subtracted from lines, areas, and volumes by using the commands and GUI paths described below. For each of these subtract commands, the *SEP0* field can be used to determine whether the entities produced will have shared or coincident but separate boundaries. The KEEP field can be used to retain or delete input entities regardless of the setting of the **BOPTN** command (**Main Menu> Preprocessor> Modeling> Operate> Booleans> Settings**).

The working plane is often used to cut up an existing model prior to map meshing. You can use any of the methods described in the following table to subtract the working plane from an entity.

Subtract the intersection of the working plane from	Command	GUI
lines	**LSBW**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Divide> Line by WrkPlane** **Main Menu> Preprocessor> Modeling> Operate> Booleans> Divide> With Options> Line by WrkPlane**
areas	**ASBW**	**Main Menu> Preprocessor> Operate> Divide> Area by WrkPlane** **Main Menu> Preprocessor> Modeling> Operate> Booleans> Divide> With Options> Area by WrkPlane**
volumes	**VSBW**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Divide> Volu by WrkPlane** **Main Menu> Preprocessor> Modeling> Operate> Booleans> Divide> With Options> Volu by WrkPlane**

5.4.7.1. Illustrations of Working Plane Subtraction Operations

The following figures illustrate the working plane subtraction operations listed above:

Figure 5.46: LSBW (Line Subtract Working Plane)

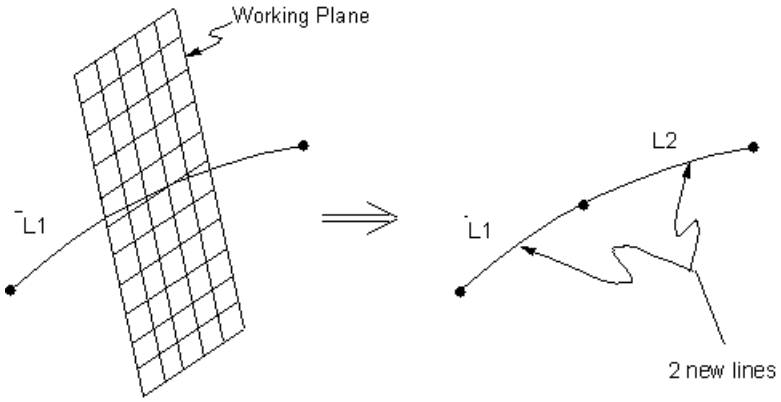


Figure 5.47: ASBW (Area Subtract Working Plane)

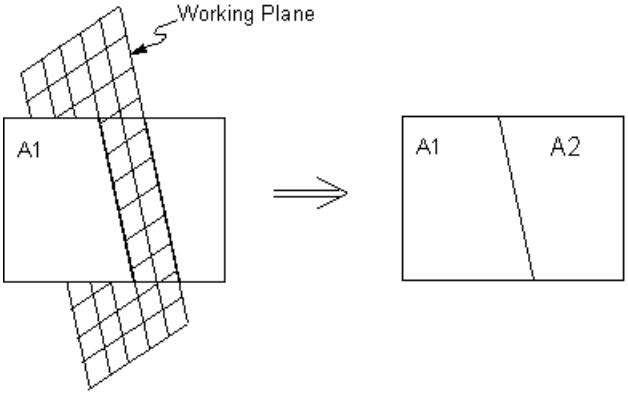
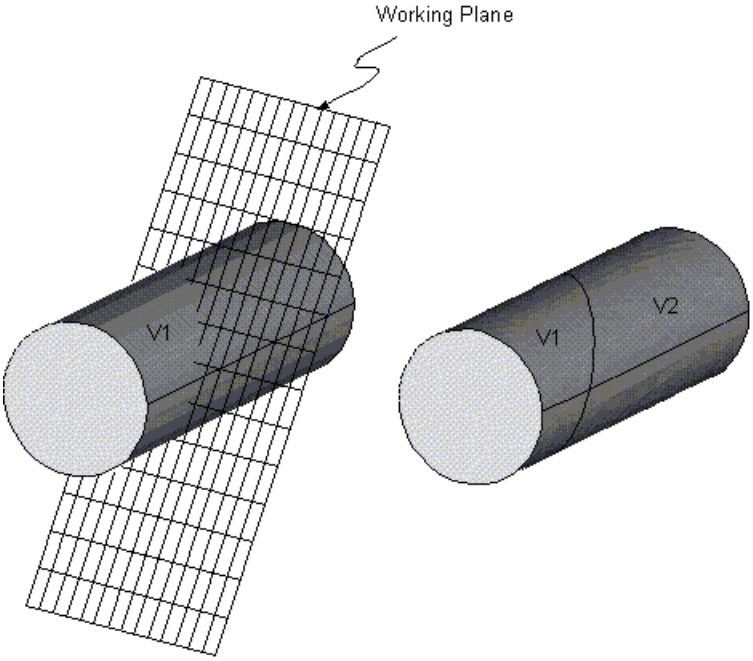


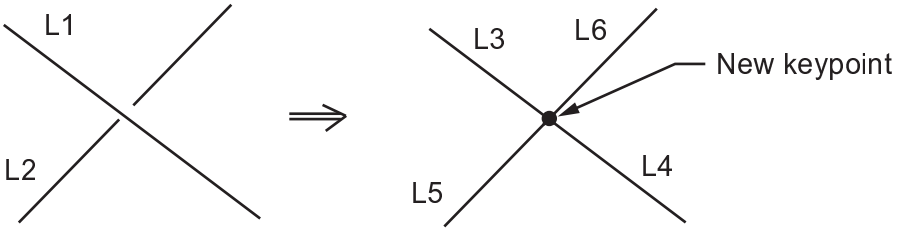
Figure 5.48: VSBW (Volume Subtract Working Plane)



5.4.8. Classify

Classification is similar to subtraction, except that both the original entities are replaced by new entities. At present, only line-line classification is possible in the ANSYS program. To perform line-line classification, use the **LCSL** command. You cannot access the **LCSL** command directly in the GUI. The following figure illustrates the classification operation.

Figure 5.49: LCSL (Line Classify Line)



5.4.9. Overlap

The *overlap* commands will join two or more entities to create three or more new entities that encompass all parts of the originals. The end result is similar to an "add" operation, except that boundaries will be created around the overlap zone. Thus, the overlap operation produces a number of relatively uncomplicated regions, as compared to the single relatively complicated region created by the add operation. For this reason, overlapped entities will often mesh better than added entities.

Overlapping is valid only if the overlap region has the same dimensionality as the original entities. The Boolean overlap commands (and their corresponding GUI paths) are as follows:

Overlap	Command	GUI
lines	**LOVLAP**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Overlap> Lines**
areas	**AOVLAP**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Overlap> Areas**
volumes	**VOVLAP**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Overlap> Volumes**

5.4.9.1. Illustrations of Overlap Operations

The following figures illustrate the overlap operations listed above:

Figure 5.50: LOVLAP (Line Overlap Line)

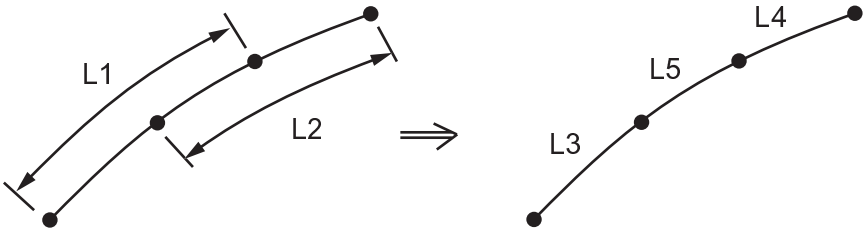


Figure 5.51: AOVLAP (Area Overlap Area)

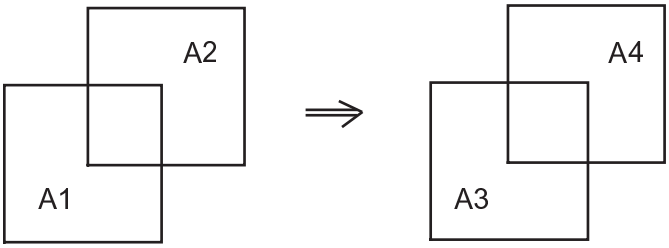
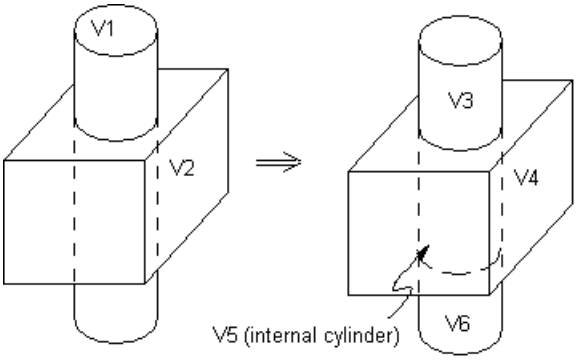


Figure 5.52: VOVLAP (Volume Overlap Volume)



5.4.10. Partition

The *partition* commands will join two or more entities to create three or more new entities that encompass all parts of the originals. The end result is similar to an "overlap" operation if the overlap is of the same dimensionality as the original entities. However, unlike the *overlap* operations, non-overlapping input entities will not be deleted. The Boolean partition commands are as follows:

Partition	Command	GUI
lines	**LPTN**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Partition> Lines**
areas	**APTN**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Partition> Areas**
volumes	**VPTN**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Partition> Volumes**

5.4.10.1. Illustrations of Partition Operations

Figure 5.53:**LPTN** (Line Partition) (p. 76), Figure 5.54:**APTN** (Area Partition) (p. 77), and Figure 5.55:**VPTN** (Volume Partition) (p. 77) illustrate partition operations. See the descriptions of the **LPTN**, **APTN**, and **VPTN** commands in the *Command Reference* for more information.

Figure 5.53: LPTN (Line Partition)

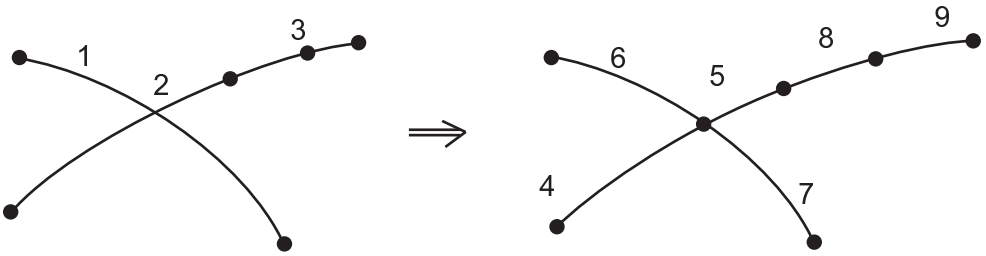


Figure 5.54: APTN (Area Partition)

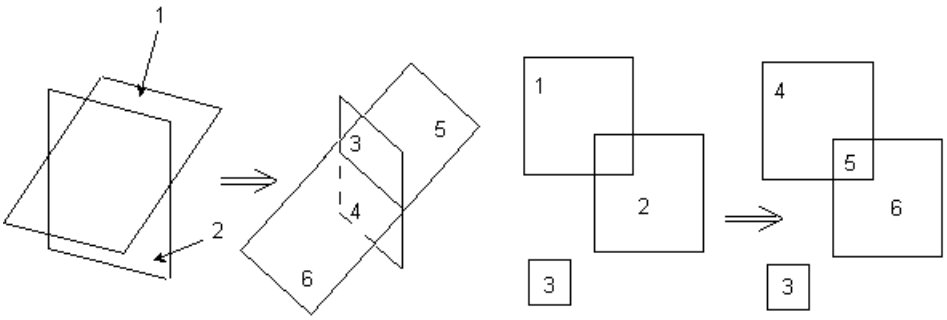
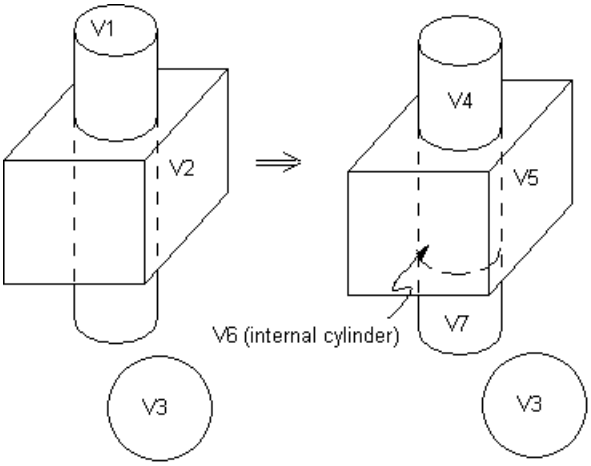


Figure 5.55: VPTN (Volume Partition)



5.4.11. Glue (or Merge)

Glue is similar to overlap, except that it applies only to cases in which the intersection between entities occurs at a boundary, and is one dimension lower than the original entities. The entities maintain their individuality (they are not "added"), but they become connected at their intersection (they "talk" to each other), as shown in the illustrations below. The Boolean glue commands (and their corresponding GUI paths) are as follows:

Generate new	Com-mand	GUI
lines by “gluing” lines	**LGLUE**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Glue> Lines**
areas by “gluing” areas	**AGLUE**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Glue> Areas**
volumes by “gluing” volumes	**VGLUE**	**Main Menu> Preprocessor> Modeling> Operate> Booleans> Glue> Volumes**

5.4.11.1. Illustrations of Glue Operations

The following figures illustrate the glue operations listed above:

Figure 5.56: LGLUE (Line Glue Line)

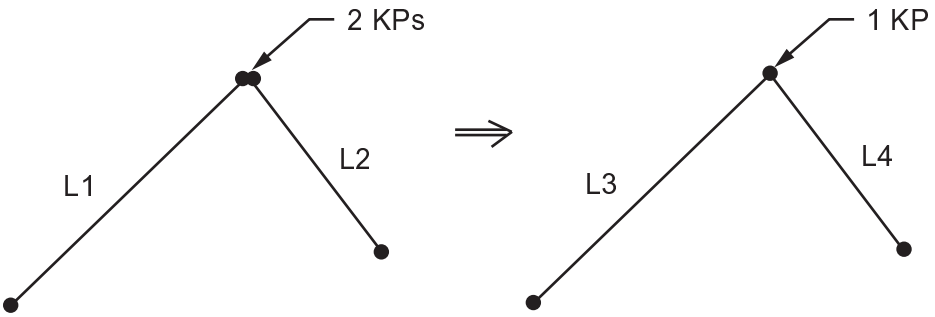


Figure 5.57: AGLUE (Area Glue Area)

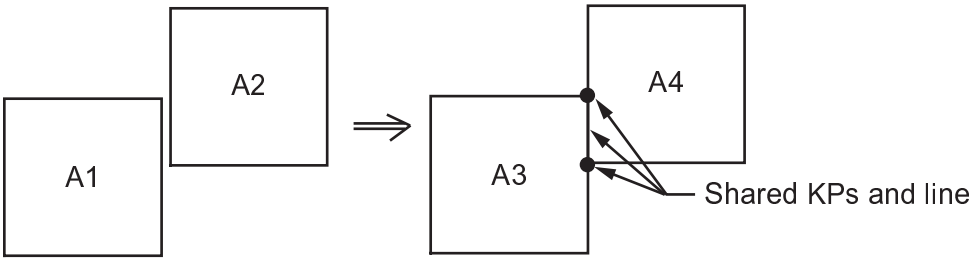
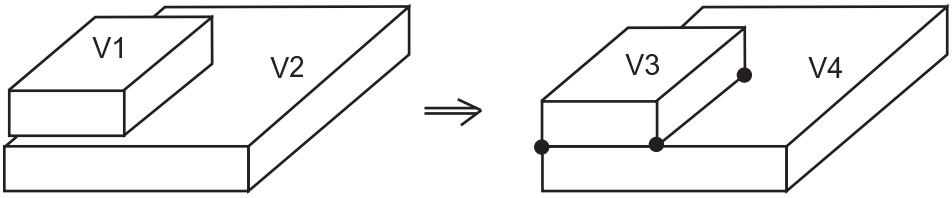


Figure 5.58: VGLUE (Volume Glue Volume)



5.4.12. Alternatives to Boolean Operations

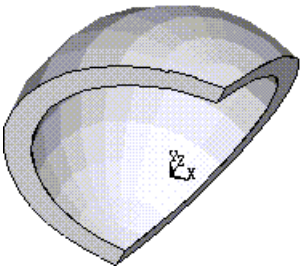
Boolean operations can sometimes be relatively slow and expensive. As a general guideline, if you can replace a Boolean command with another, similarly convenient command, do so. Some of the alternative procedures that can sometimes be used in place of Booleans are described below.

Dragging and Rotating: A complicated prismatic or cylindrical volume might be defined just as conveniently, but more efficiently, using **VDRAG** and **VROTAT** (as described below) in place of Boolean operations. A good example would be a model of a block with a number of holes drilled through it.

Extruding and Offsetting: A 2-D cross section can also be offset or extruded into a 3-D volume using **VEXT** and **VOFFST** (as described below). See Creating Your Solid Model from the Bottom Up (p. 37) earlier in this chapter, which also discusses how to extrude, offset, rotate, or drag meshed areas into meshed volumes.

Utilizing the Options on Primitive Commands: A number of primitive commands allow you to define a relatively complicated shape in a single command. For example, you can create a hollow spherical segment of specified wall thickness, using a single **SPHERE** command (**Main Menu> Preprocessor> Modeling> Create> Volumes> Sphere> By Dimensions**):

Figure 5.59: Hollow Spherical Segment Created With One Command



It is clear from this example how exercising the full ability of one primitive command can sometimes save the expense of performing several Boolean operations.

5.5. Updating after Boolean Operations

Some Boolean commands will automatically update entities after the Boolean operation is performed on attached lower-order entities. For instance, if you use an **AADD** Boolean operation (**Main Menu> Preprocessor> Modeling> Operate> Booleans> Add> Areas**) to add several areas together that are attached to a single volume, that volume will be updated by replacing the original areas with the newly produced area. This releases you from the work of deleting the higher-order entity (volume in this case) and rebuilding it with bottom up techniques. The following commands perform automatic updating of higher-order entities:

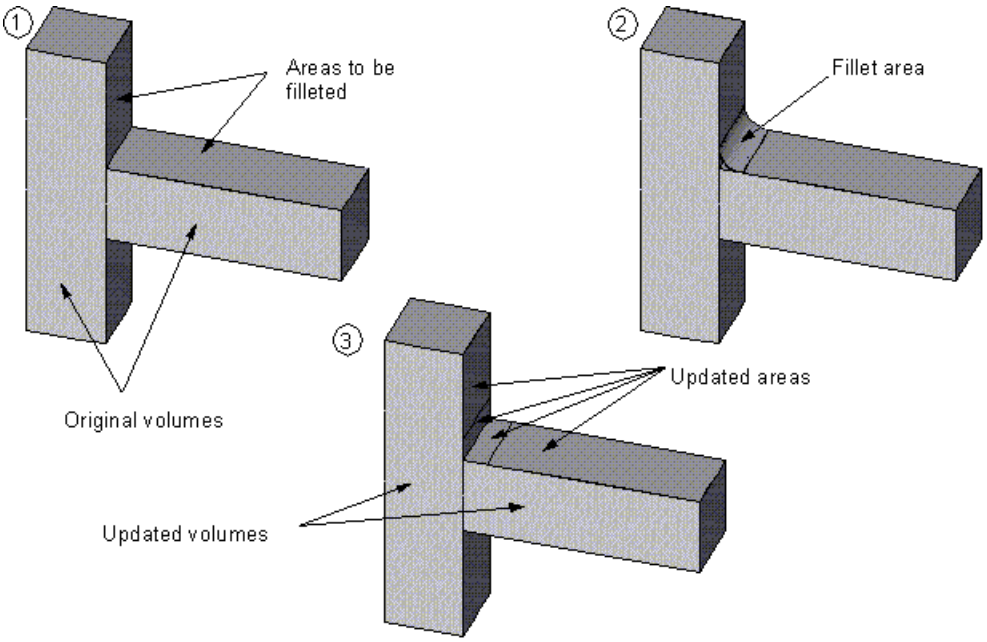
Table 5.1: Commands That Automatically Update Entities

Command	Entities directly modified by command	Entities that can be updated
AADD	Areas	Volumes
ASBA	Areas and Lines	Areas and Volumes
ASBV	Areas and Lines	Areas and Volumes
ASBL	Areas and Lines	Areas and Volumes
AFILLT	N/A	Areas and Volumes
LSBL	Lines	Areas
LSBA	Lines	Areas
LSBV	Lines	Areas
LCSL	Lines	Areas

Updating can occur only if the Boolean operation produces entities that are equal to the original entity. For instance, if an **ASBA** operation (**Main Menu> Preprocessor> Modeling> Operate> Booleans> Divide> Area by Area**, and so on) cuts an area into two pieces, the two new pieces will replace the original area, and the underlying volume will be updated to include the new areas. However, if the **ASBA** operation cuts a piece out of the original area (to create a hole for instance), no entities can be updated.

Figure 5.60: Automatic Boolean Updating With **AFILLT** (p. 80) shows an example of automatic Boolean updating. In this example, a fillet area [**AFILLT**] (**Main Menu> Preprocessor> Modeling> Create> Areas> Area Fillet**) is placed between two intersecting volumes. The Boolean fillet operation creates the fillet area, but also automatically updates the areas on the surfaces of the volumes, and the volumes are updated to replace the two original surface areas with the four new surface areas.

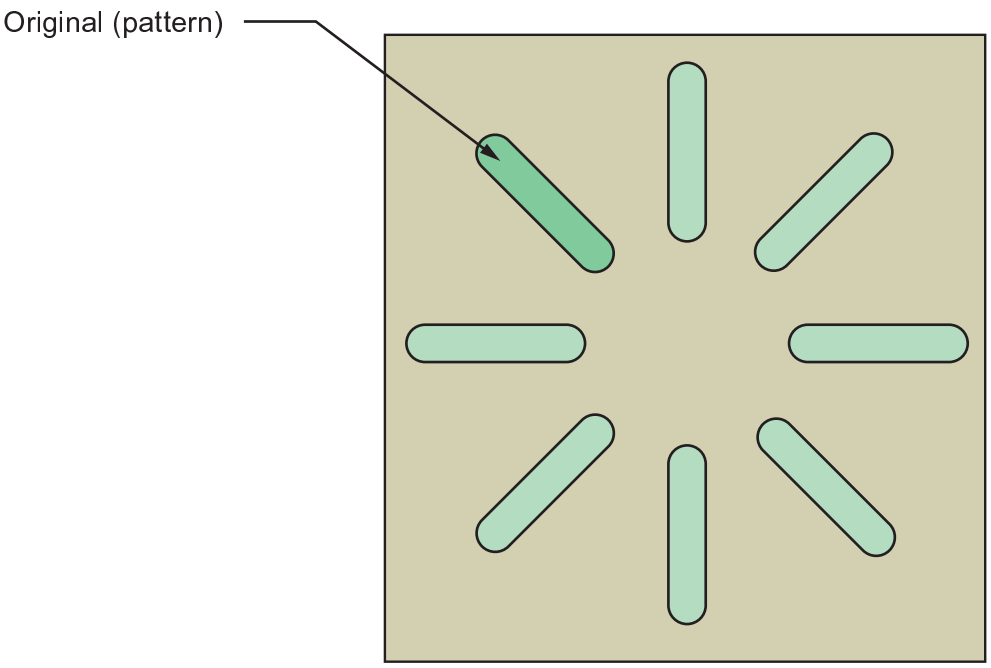
Figure 5.60: Automatic Boolean Updating With AFILLT



5.6. Moving and Copying Solid Model Entities

If your model repetitively uses a relatively complicated area or volume, you need construct that part only once; you can then generate copies of that part in new locations and new orientations as needed. For example, the elongated voids in the plate shown below can be copied from a single such void.

Figure 5.61: Copying an Area



Geometric primitives can also be considered to be "parts." As you create geometric primitives, their location and orientation will be determined by the current working plane. Because it is not always particularly convenient to redefine the working plane for each new primitive that you create, you might find it more practical to allow a primitive to be created at the "wrong" location, and then *move* that

primitive to its correct position. Of course, this operation is not limited to geometric primitives: *any* solid model entity can be copied or moved.

The commands that you can use to move or copy solid model entities include the *xGEN* commands, the *xSYM(M)* commands, and the *xTRAN* commands (and their corresponding GUI paths). Of these, the *xGEN* and *xTRAN* commands will probably be the most useful for moving and rotating a copy of an entity. (Copying a higher-order entity will automatically cause all the lower-order entities associated with it to be copied as well. In addition, if you copy an entity's elements (*NOELEM* = 0), all elements associated with lower-order entities attached to it will also be copied.) You can *move* an entity to a new location by setting *IMOVE* = 1 in the appropriate *xGEN*, *xSYM(M)*, or *xTRAN* command.

5.6.1. Generating Entities from a Pattern

ANSYS provides the following *xGEN* commands and GUI paths:

- To generate additional keypoints from a pattern of keypoints:
  Command(s): KGEN
  GUI: Main Menu> Preprocessor> Modeling> Copy> Keypoints

- To generate additional lines from a pattern of lines:
  Command(s): LGEN
  GUI: Main Menu> Preprocessor> Modeling> Copy> Lines
  Main Menu> Preprocessor> Modeling> Move/Modify> Lines

- To generate additional areas from a pattern of areas:
  Command(s): AGEN
  GUI: Main Menu> Preprocessor> Modeling> Copy> Areas
  Main Menu> Preprocessor> Modeling> Move/Modify> Areas> Areas

- To generate additional volumes from a pattern of volumes:
  Command(s): VGEN
  GUI: Main Menu> Preprocessor> Modeling> Copy> Volumes
  Main Menu> Preprocessor> Modeling> Move/Modify> Volumes

5.6.2. Generating Entities by Symmetry Reflection

ANSYS provides the following *xSYM(M)* commands and GUI paths:

- To generate a reflected set of keypoints:
  Command(s): KSYMM
  GUI: Main Menu> Preprocessor> Modeling> Reflect> Keypoints

- To generate lines from a line pattern by symmetry reflection:
  Command(s): LSYMM
  GUI: Main Menu> Preprocessor> Modeling> Reflect> Lines

- To generate areas from an area pattern by symmetry reflection:
  Command(s): ARSYM
  GUI: Main Menu> Preprocessor> Modeling> Reflect> Areas

- To generate volumes from a volume pattern by symmetry reflection:
  Command(s): VSYMM
  GUI: Main Menu> Preprocessor> Modeling> Reflect> Volumes

5.6.3. Transferring a Pattern of Entities to a Coordinate System

ANSYS provides the following *xTRAN* commands and GUI paths:

- To transfer a pattern of keypoints to another coordinate system:
  Command(s): KTRAN
  GUI: Main Menu> Preprocessor> Modeling> Move/Modify> Transfer Coord> Keypoints

- To transfer a pattern of lines to another coordinate system:
  Command(s): LTRAN
  GUI: Main Menu> Preprocessor> Modeling> Move/Modify> Transfer Coord> Lines

- To transfer a pattern of areas to another coordinate system:
  Command(s): ATRAN
  GUI: Main Menu> Preprocessor> Modeling> Move/Modify> Transfer Coord> Areas

- To transfer a pattern of volumes to another coordinate system:
  Command(s): VTRAN
  GUI: Main Menu> Preprocessor> Modeling> Move/Modify> Transfer Coord> Volumes

5.7. Scaling Solid Model Entities

It is possible to scale defined entities up or down. The **xSCALE** family of commands can be used to scale a single entity or a pattern of entities in the active coordinate system.

Scale factors are applied to the X, Y, and Z keypoint coordinates for each of the four scaling commands. If you are working in a cylindrical coordinate system when doing a scaling operation, X, Y, and Z will be interpreted as R, θ, Z, where θ is an angular offset. In spherical coordinates, X, Y, and Z will be interpreted as R, θ, Φ, where θ and Φ are both angular offsets.

- To generate a scaled set of (meshed) keypoints from a pattern of keypoints:
  Command(s): KPSCALE
  GUI: Main Menu> Preprocessor> Modeling> Operate> Scale> Keypoints

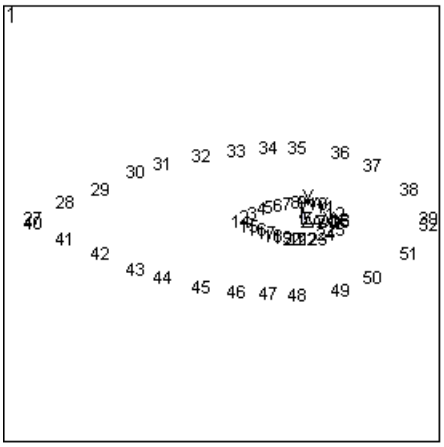
- To generate a scaled set of lines from a pattern of lines:
  Command(s): LSSCALE
  GUI: Main Menu> Preprocessor> Modeling> Operate> Scale> Lines

- To generate a scaled set of areas from a pattern of areas:
  Command(s): ARSCALE
  GUI: Main Menu> Preprocessor> Modeling> Operate> Scale> Areas

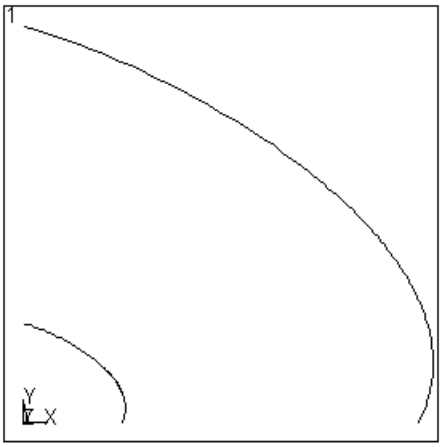
- To generate a scaled set of volumes from a pattern of volumes:
  Command(s): VLSCALE
  GUI: Main Menu> Preprocessor> Modeling> Operate> Scale> Volumes

Samples of scaled entities are shown in Figure 5.62: Scaling Entities (p. 83).

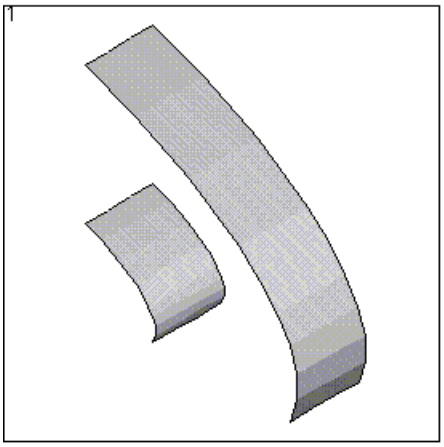
Figure 5.62: Scaling Entities



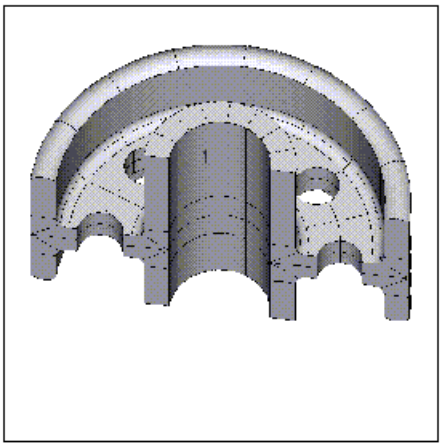
KPSCALE



LSSCALE



ARSCALE



VLSCALE

5.8. Solid Model Loads

You can define loads directly on the solid model at any time before you actually initiate the solution. Thus, solid model loads may be defined before or after finite element meshing. (Details of how to define solid model loads are discussed in Loading in the *Basic Analysis Guide*.) The remainder of this section describes other functions related to solid model loads.

5.8.1. Transferring Solid Model Loads

Solid model loads will be transferred to the finite element model automatically when you begin the solution calculations (when you issue a **SOLVE** command (**Main Menu> Solution> Solve> Current LS**)), or you may transfer them "manually" using the following methods:

- To transfer solid model loads and boundary conditions to the finite element model:
  Command(s): SBCTRAN
  GUI: Main Menu> Preprocessor> Loads> Define Loads> Operate> Transfer to FE> All Solid Lds
  Main Menu> Solution> Define Loads> Operate> Transfer to FE> All Solid Lds

- To transfer solid model body force loads to the finite element model:
  Command(s): BFTRAN

GUI: Main Menu> Preprocessor> Loads> Define Loads> Operate> Transfer to FE> Body Loads
Main Menu> Solution> Define Loads> Operate> Transfer to FE> Body Loads

- To transfer solid model DOF constraints to the finite element model:
  Command(s): DTRAN
  GUI: Main Menu> Preprocessor> Loads> Define Loads> Operate> Transfer to FE> Constraints
  Main Menu> Solution> Define Loads> Operate> Transfer to FE> Constraints

- To transfer solid model forces to the finite element model:
  Command(s): FTRAN
  GUI: Main Menu> Preprocessor> Loads> Define Loads> Operate> Transfer to FE> Forces
  Main Menu> Solution> Define Loads> Operate> Transfer to FE> Forces

- To transfer solid model surface loads to the finite element model:
  Command(s): SFTRAN
  GUI: Main Menu> Preprocessor> Loads> Define Loads> Operate> Transfer to FE> Surface Loads
  Main Menu> Solution> Define Loads> Operate> Transfer to FE> Surface Loads

5.8.2. Displaying Load Symbols

Solid model loads may be displayed at any time after you have turned on the appropriate load symbols:

- To show boundary condition symbols on displays, use the **/PBC** command. This command includes an option to display boundary condition values, as well as symbols. (Turning off **/VSCALE** scaling before issuing the **/PBC** command may help.)

- To show body force loads as contours on displays, use the **/PBF** command.

- To show surface load symbols on model displays, use the **/PSF** command.

All of these commands can be accessed in the GUI by picking **Utility Menu> PlotCtrls> Symbols**.

5.8.3. Turning Off Large Symbols for Node and Keypoint Locations

Large symbols for node and keypoint locations are displayed by default (**/PSYMB**,DOT,1).

- To display smaller symbols for node and keypoint locations, use one of the following methods:
  Command(s): /PSYMB,DOT,0
  GUI: Utility Menu> PlotCtrls> Symbols

5.8.4. Selecting a Format for the Graphical Display of Numbers

You can indicate the field length and the number of digits that are displayed for the numbers appearing on a model.

- To specify a format for the display of numbers, use the following command:

  Command(s):
  /GFORMAT

5.8.5. Listing Solid Model Loads

You can list *all* solid model loads or you can list separate types of solid model loads using the following methods:

- To list *all* solid model loads:
  Command(s): SBCLIST
  GUI: Utility Menu> List> Loads> Solid Model Loads

- To list the body force loads at keypoints:
  Command(s): BFKLIST
  GUI: Utility Menu> List> Loads> Body> On All Keypoints
  Utility Menu> List> Loads> Body> On Picked KPs

- To list the DOF constraints at keypoints:
  Command(s): DKLIST
  GUI: Utility Menu> List> Loads> DOF Constraints> On All Keypoints
  Utility Menu> List> Loads> DOF Constraints> On Picked KPs

- To list the DOF constraints on a line:
  Command(s): DLLIST
  GUI: Utility Menu> List> Loads> DOF Constraints> On All Lines
  Utility Menu> List> Loads> DOF Constraints> On Picked Lines

- To list the DOF constraints on an area:
  Command(s): DALIST
  GUI: Utility Menu> List> Loads> DOF Constraints> On All Areas
  Utility Menu> List> Loads> DOF Constraints> On Picked Areas

- To list the forces at keypoints:
  Command(s): FKLIST
  GUI: Utility Menu> List> Loads> Forces> On All Keypoints
  Utility Menu> List> Loads> Forces> On Picked KPs

- To list the surface loads on lines:
  Command(s): SFLLIST
  GUI: Utility Menu> List> Loads> Surface Loads> On All Lines
  Utility Menu> List> Loads> Surface Loads> On Picked Lines

- To list the surface loads on areas:
  Command(s): SFALIST
  GUI: Utility Menu> List> Loads> Surface Loads> On All Areas
  Utility Menu> List> Loads> Surface Loads> On Picked Areas

5.9. Mass and Inertia Calculations

The **xSUM** commands calculate and print geometry items associated with solid model entities. See Lumped Calculation of Mass Related Information in the *Mechanical APDL Theory Reference* for details.

Note

For very narrow (sliver) areas or very thin volumes, such that the ratio of the minimum to the maximum dimension is less than 0.01, the **ASUM** and **VSUM** commands can provide erroneous area or volume information. To ensure that such calculations are accurate, make certain that you subdivide such areas and volumes so that the ratio of the minimum to the maximum is at least 0.05.

- To calculate and print the center of mass location, moments of inertia, and so on, associated with selected keypoints:
  Command(s): KSUM
  GUI: Main Menu> Preprocessor> Modeling> Operate> Calc Geom Items> Of Keypoints

- To calculate and print the length, center of mass location, moments of inertia, and so on, for selected lines:
  Command(s): LSUM
  GUI: Main Menu> Preprocessor> Modeling> Operate> Calc Geom Items> Of Lines

- To calculate and print the area, center of mass location, moments of inertia, and so on, for selected areas:
  Command(s): ASUM
  GUI: Main Menu> Preprocessor> Modeling> Operate> Calc Geom Items> Of Areas

- To calculate and print the volume, center of mass location, moments of inertia, and so on, for selected volumes:
  Command(s): VSUM
  GUI: Main Menu> Preprocessor> Modeling> Operate> Calc Geom Items> Of Volumes

- To calculate and print all of the previously mentioned keypoint, line, area, and volume geometry items at one time:
  Command(s): GSUM
  GUI: Main Menu> Preprocessor> Modeling> Operate> Calc Geom Items> Of Geometry

5.10. Considerations and Cautions for Solid Modeling

5.10.1. Representation of Solid Model Entities

During the solid modeling process, it is helpful to understand the underlying mathematical operations used by ANSYS. This knowledge is particularly useful when you encounter degeneracies and discontinuities. For example, an error stating that a degeneracy has been detected in a solid model Boolean operation may occur. Knowledge of the mathematical terminology involved can aid in overcoming such an error.

Internally, solid model entities are mathematically represented by *trimmed parametric surfaces*. Trimmed parametric surfaces consist of two components: *parametric geometry* and *topology*. Parametric geometry defines the underlying surface of a model. The term "parametric" refers to the parametric space that mathematically represents the geometric space. Refer to Figure 5.63: Cone Surface Maps to a Parametric Square (p. 87) for an illustration of the relationship between the geometric model and the parametric model. Nonuniform Rational B-splines, or NURBS, are used in the definition of the parametric geometry. The term "topology" refers to the trimming surfaces that bound a model's geometry.

5.10.2. When a Boolean Operation Fails

Boolean operators provide a powerful set of tools that enable you to create complicated geometries with a minimum amount of input. However, you might encounter situations in which a Boolean operation will have difficulties. If this happens, you can still sometimes work around the problem to achieve the desired solid model. Let's examine some of the causes of and cures for Boolean failures.

5.10.2.1. Degeneracies

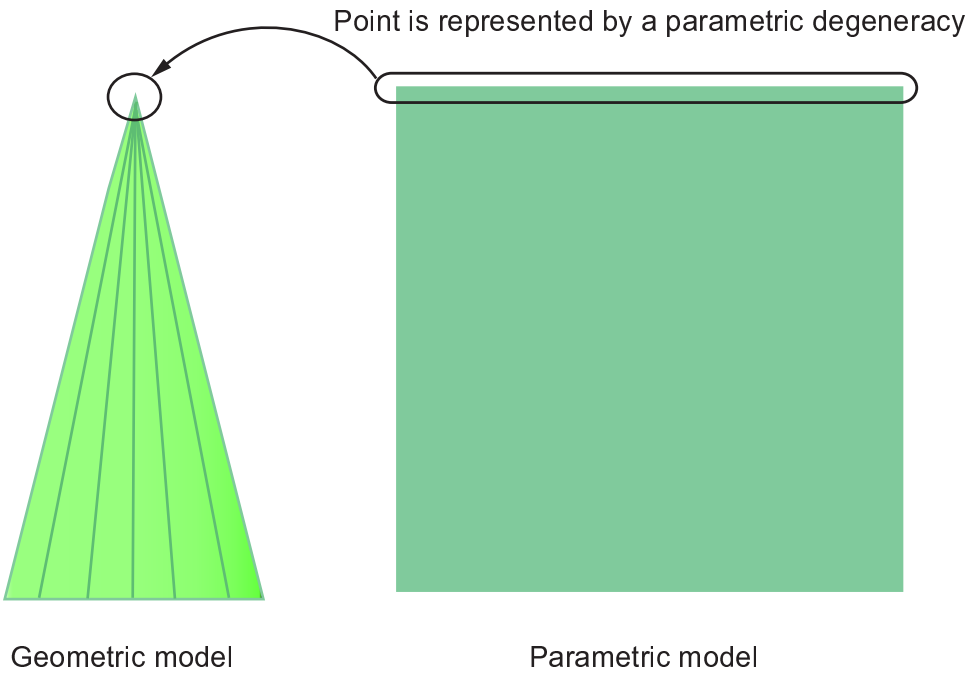
Boolean operations can fail due to degeneracies. Knowing what constitutes a degeneracy, how a degeneracy is formed, and why degeneracies sometimes cause Booleans to fail can help you overcome such

errors. Degeneracies can arise due to issues in geometry or topology. ANSYS will classify degeneracies (which cause a Boolean failure) as either parametric (geometry) or topological.

Parametric degeneracies result from the representation of geometric space with an underlying parametric space. When this "shadow world" of the parametric representation is not in dimensional harmony with the real world of the geometric model, a degeneracy is created. For example, at the apex of a cone, a single *point* on the geometric model is represented by an *edge* in the parametric representation (see Figure 5.63: Cone Surface Maps to a Parametric Square (p. 87)). Such a point is termed a *degenerate edge*, or more simply, a degeneracy. Figure 5.65: Examples of Degeneracies (p. 89) illustrates several common degeneracies formed during modeling.

A degeneracy of this kind is not harmful in and of itself. A model that contains a degeneracy can often be used in Boolean operations, can be meshed successfully, and can yield excellent analysis results. It is only when a degeneracy happens to cause a problem in a Boolean operation that you even need to be aware of its existence.

Figure 5.63: Cone Surface Maps to a Parametric Square



5.10.3. Graphically Identifying Degeneracies

Degeneracies in areas or volumes can be graphically identified using the methods described below. If you are using the command input method, including the *DEGEN* label will place a red star at keypoints associated with degeneracies (see Figure 5.64: Plotting of Geometric Degeneracies (p. 88)).

- To display degeneracies in areas:
  Command(s): APLOT, , , ,DEGE
  GUI: Main Menu> Preprocessor> Modeling> Operate> Booleans> Show Degeneracy> Plot Degen Areas

- To display selected volumes:
  Command(s): VPLOT, , , ,DEGE
  GUI: Main Menu> Preprocessor> Modeling> Operate> Booleans> Show Degeneracy> Plot Degen Volus

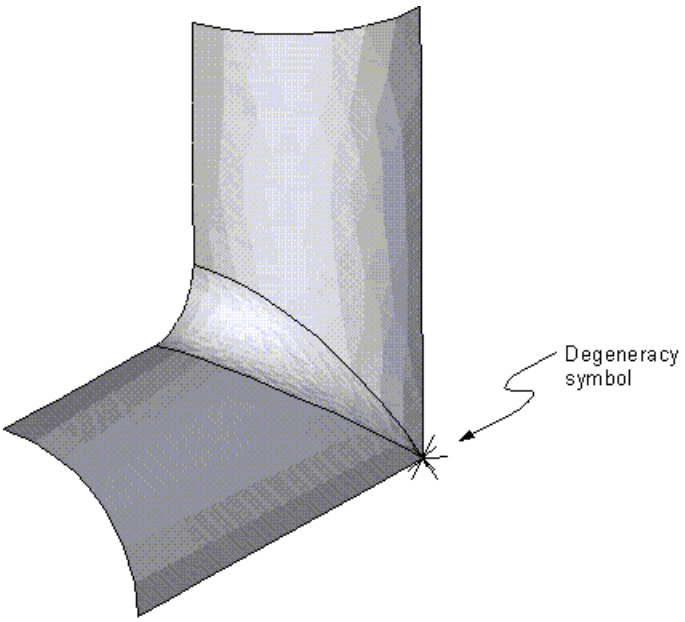
5.10.4. Listing the Keypoints Associated with Degeneracies

You may also choose to list the keypoints associated with degeneracies:

- To list keypoints of an area that lie on a parametric degeneracy:
  Command(s): ADGL
  GUI: Main Menu> Preprocessor> Modeling> Operate> Booleans> Show Degeneracy> List Degen Areas

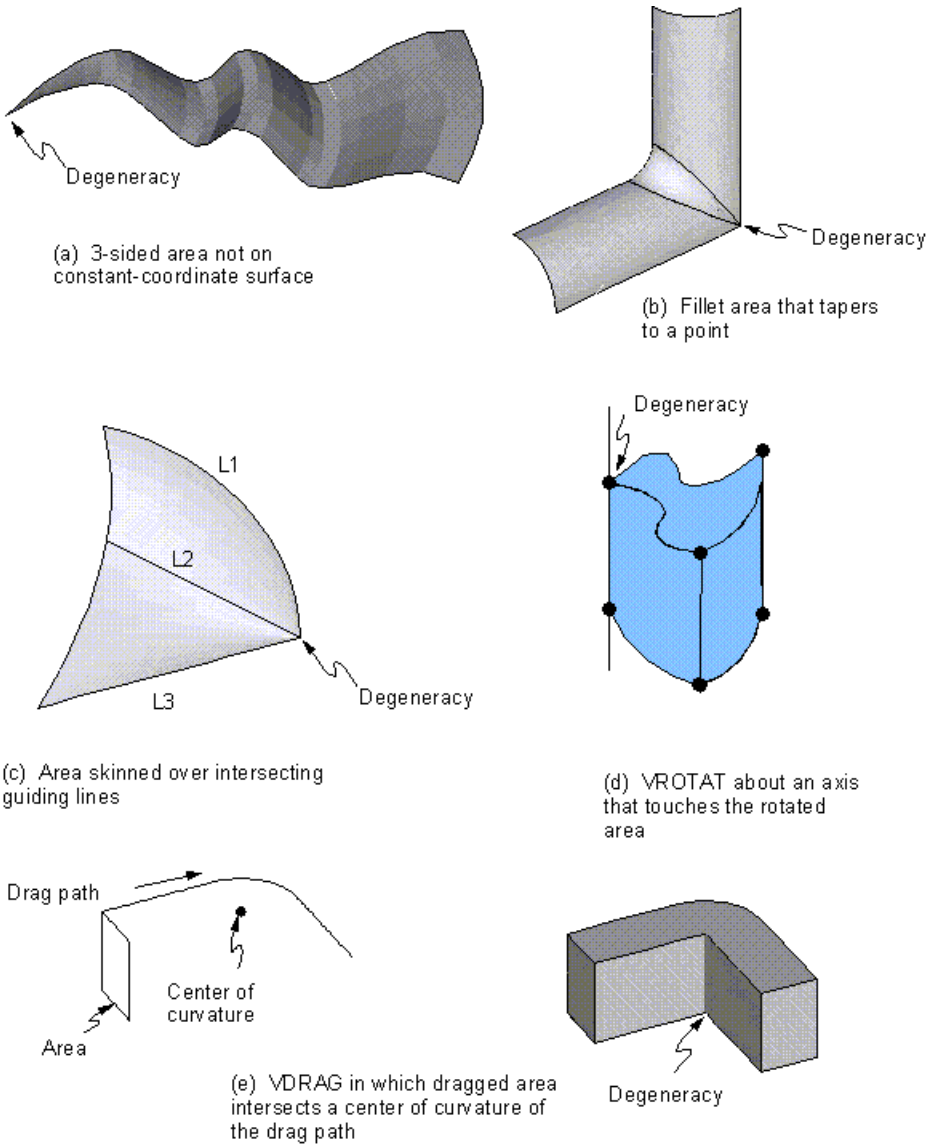
- To list keypoints of a volume that lie on a parametric degeneracy:
  Command(s): VDGL
  GUI: Main Menu> Preprocessor> Modeling> Operate> Booleans> Show Degeneracy> List Degen Volus

Figure 5.64: Plotting of Geometric Degeneracies



Another kind of degeneracy can be detected if a Boolean operation attempts to create a *degenerate boundary* during any phase of its operation. A degenerate boundary is an incomplete or zero-area loop, or an incomplete or zero-volume shell. This type of degeneracy is commonly referred to as a *topological degeneracy*. A Boolean operation will produce an error message if a degeneracy of this nature is detected. Topological degeneracies cannot be plotted since they do not exist prior to a Boolean operation. Examples of topological degeneracies are illustrated in Figure 5.66: Topologically Degenerate Loop (p. 90) and Figure 5.67: Examples of Topologically Degenerate Loops and Shells (p. 90). An example of a Boolean operation resulting in failure is shown in Figure 5.66: Topologically Degenerate Loop (p. 90). In this example, the triangular prism cannot be subtracted [**VSBV**] from the block, because a degenerate loop would be formed on the top surface of the block. Other Boolean operations such as **VADD**, **VOVLAP**, etc. would also fail for these volumes due to this degeneracy.

Figure 5.65: Examples of Degeneracies



A topologically degenerate loop will be created during **VSBV**, **VADD**, **VOVLAP**, or other Boolean operations involving these volumes

Figure 5.66: Topologically Degenerate Loop

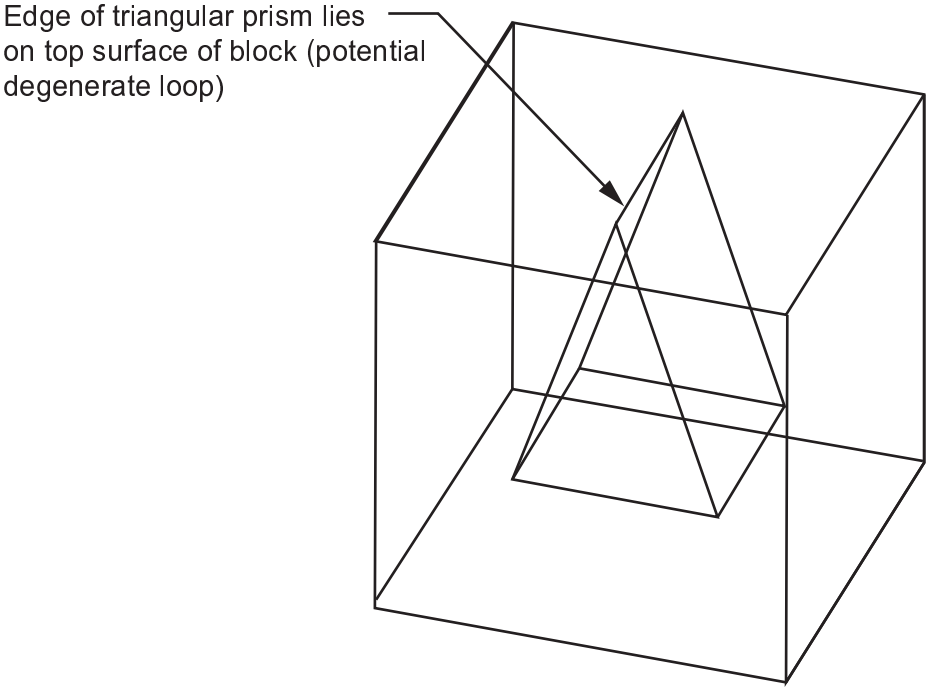
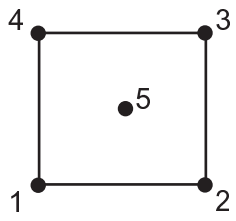
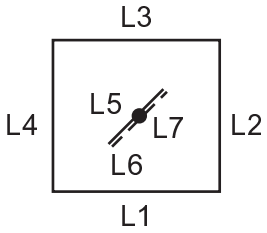


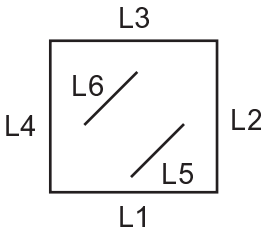
Figure 5.67: Examples of Topologically Degenerate Loops and Shells



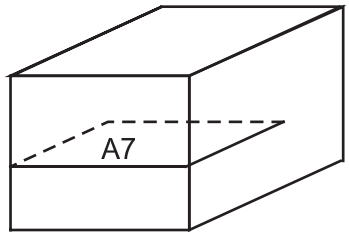
If keypoint 5 is made part of this area, it will be a degenerate loop.



If coincident lines 5, 6, and 7 are made part of this area, they will form a degenerate (zero-area) loop.



If lines 5 or 6 are made part of this area, they will be degenerate loops.

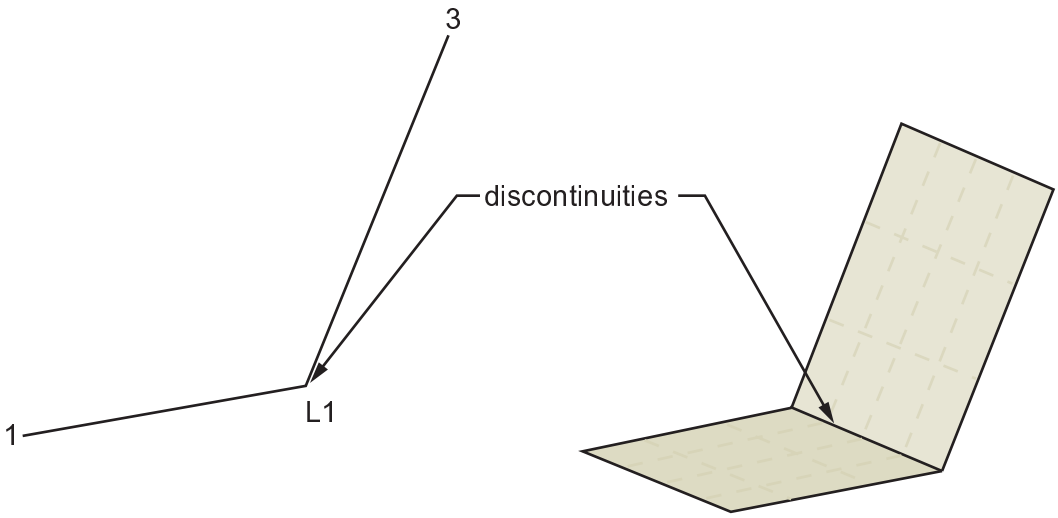


If area 7 is made part of this volume, it will be a degenerate shell.

5.10.4.1. Discontinuities

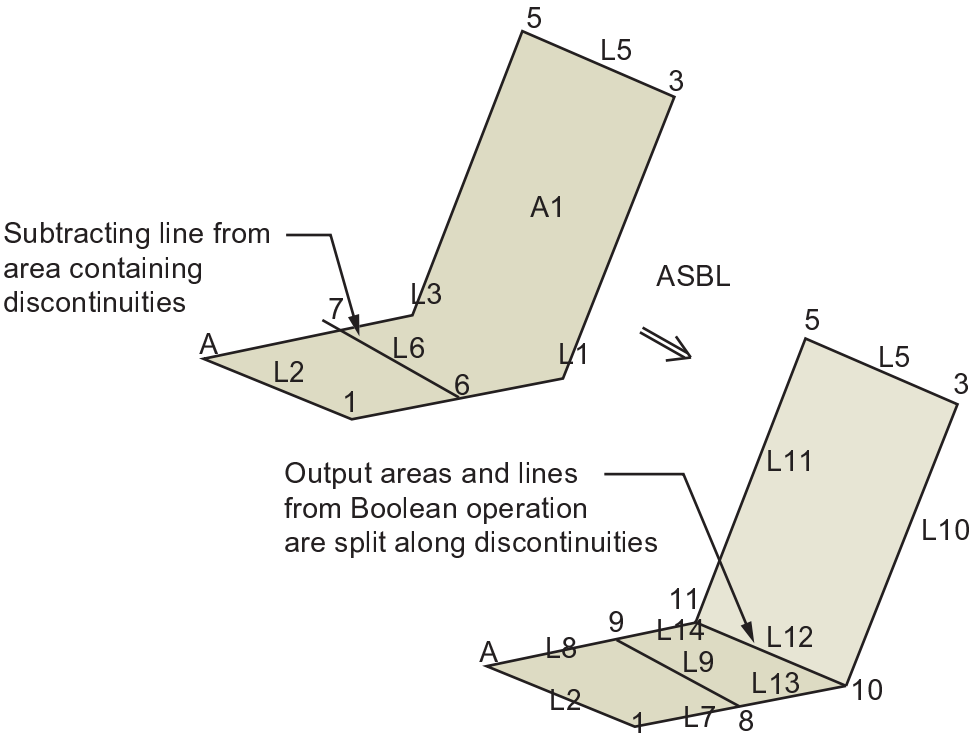
Generally, discontinuities are sharp "kinks" in solid model entities. They can result from the combination of lines [**LCOMB**] with inconsistent end tangents or from an IGES import. See Figure 5.68: Lines and Areas Containing Discontinuities (p. 91) for an illustration of entities containing discontinuities.

Figure 5.68: Lines and Areas Containing Discontinuities



Many solid model operations will support entities containing discontinuities. However, Boolean operations do not directly support discontinuities. Boolean operations will split or divide entities at points or along lines of discontinuities prior to the Boolean operation. For example, Figure 5.69: Boolean Operation Involving Entities With Discontinuities (p. 92) shows an area containing a discontinuity involved in a Boolean subtract operation. Prior to the subtract operation area 1 along with lines 1 and 3 are split at discontinuities.

Figure 5.69: Boolean Operation Involving Entities With Discontinuities



Note

Discontinuities can be associated with both direction and magnitude of tangent vectors. Boolean operations will detect both types of discontinuities.

5.10.4.2. Other Causes of Boolean Failures

Boolean operations can fail due to other circumstances besides degeneracies. For instance, intersections at tangent points can sometimes be difficult for the Boolean operations to handle, particularly with non-primitive constructions. Also, entities that share a boundary (such as two volumes with bounding areas on the same surface) can potentially cause problems in a Boolean operation. Problems might also be encountered if your geometry contains small regions with very high curvatures, or regions with very sharp angles or transitions.

5.10.5. Some Suggested Corrective Actions

If a Boolean operation fails, you can try using one or more of the following procedures to work around the problem.

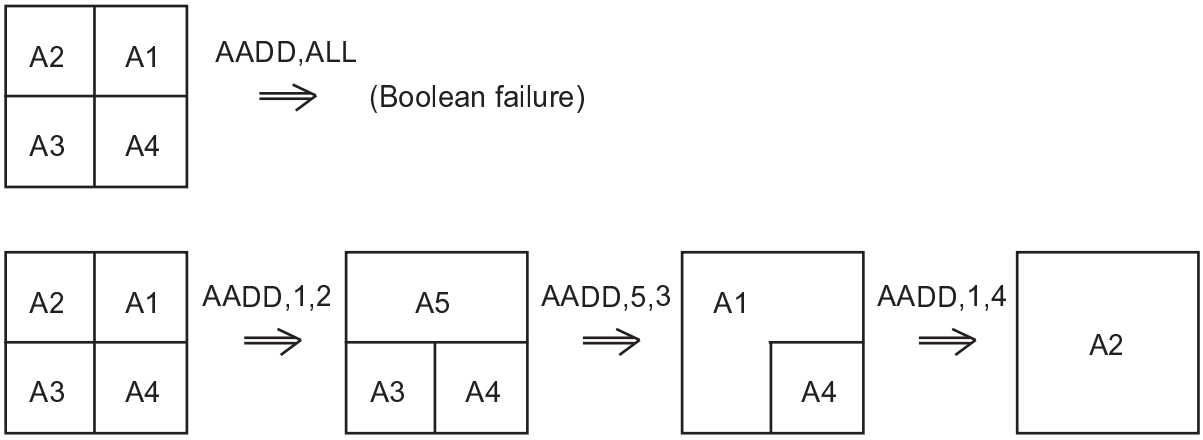
You need not always follow these guidelines as you first build your model. You can often create a model any way you like, and never encounter a Boolean failure. These guidelines are provided as suggested ways to try to recover from a Boolean failure.

Adjust the input geometries, using the following guidelines:

- Whenever possible, try to use geometric primitives as you create your solid model. The results of Boolean operations on non-primitives can sometimes be less accurate and efficient.

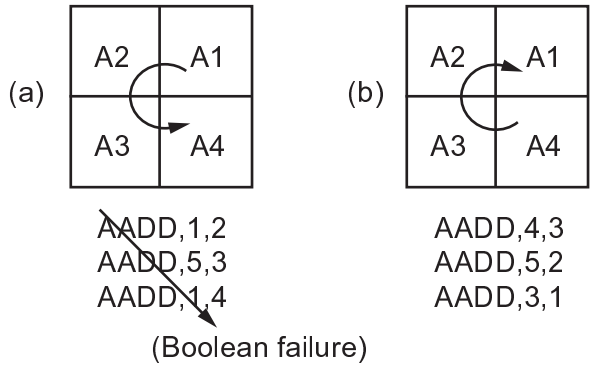
- Try to avoid creating geometries that contain degeneracies, if the degeneracy will lie on a potential intersection curve. A few specific examples of such geometries would include:
  - An untruncated cone primitive. (See Figure 5.63: Cone Surface Maps to a Parametric Square (p. 87).)
  - A 3-sided area that is neither planar nor lies on a constant-coordinate surface in the active coordinate system. (See Figure 5.65: Examples of Degeneracies (p. 89) (a).)
  - A fillet area [**AFILLT**] that tapers to a point. (See Figure 5.65: Examples of Degeneracies (p. 89) (b).)
  - A skinned area [**ASKIN**] for which two or more guiding lines intersect. (See Figure 5.65: Examples of Degeneracies (p. 89) (c).)
  - An area or volume created by rotation [**AROTAT**, **VROTAT**] about an axis that intersects any of the input entities. (See Figure 5.65: Examples of Degeneracies (p. 89) (d).)
  - An area or volume created by dragging [**ADRAG**, **VDRAG**] along a path that has a center of curvature that intersects any of the input entities. (See Figure 5.65: Examples of Degeneracies (p. 89) (e).)
- Try to avoid performing Boolean operations on entities that are tangent to each other. Similarly, try to avoid Boolean operations on entities that have coincident boundaries.
- If your Boolean operation included more than two input entities, break the operation into a *series* of operations with fewer input entities. For example, replace **AADD**,ALL with the series (**AADD**,1,2; **AADD**,5,3; **AADD**,1,4). (See Figure 5.70: Decompose a Single Operation into a Series of Operations (p. 93).) You will have to keep track of the numbers assigned to new Boolean output entities created throughout the series of commands.

Figure 5.70: Decompose a Single Operation into a Series of Operations



- If a failure occurs in a series of Boolean operations, try changing the order of operations. For instance, replace the series (**AADD**,1,2; **AADD**,5,3; **AADD**,1,4) with a reordered series, such as (**AADD**,4,3; **AADD**,5,2; **AADD**,3,1). (See Figure 5.71: Change the Order of a Series of Operations (p. 94).)

Figure 5.71: Change the Order of a Series of Operations



- If the Boolean operation fails, you may receive an error message suggesting you loosen (increase) the tolerance from the default value of 1.0 x 10^{-4} (1.0E-4). This tolerance affects the precision with which Boolean constructions are formed. Sometimes, simply changing this tolerance and reissuing the Boolean command will suffice. At other times, you might find that you need to retrace your steps, recreating the Boolean's input entities using a changed tolerance, before you can successfully proceed with the Boolean operation.

You can loosen the tolerance with the **BTOL**,*PTOL* command, where *PTOL* is the new tolerance:
Command(s): BTOL,*PTOL*
GUI: Main Menu> Preprocessor> Modeling> Operate>Settings

Once you have loosened the tolerance and re-executed the operation successfully, you should return the tolerance to its default value. Doing so will assure precise Boolean constructions later in your modeling effort.

If the geometry is unworkable (no Boolean will work or it cannot be meshed), or if you have encountered a "database corruption" (or contamination) message, then as a work-around use **CDWRITE** to write out the geometry (**CDWRITE**,COMB) using the ANF option (**CDOPT**,ANF) and **CDREAD** to read it back in (after exiting and re-entering ANSYS) to obtain clean geometry.

5.10.6. Other Hints

5.10.6.1. Avoid Regions that Cross over Themselves

You must take care not to define an area or volume that crosses over itself. (You might inadvertently create such an entity by means of an **ADRAG** or **VDRAG** command, for instance.) The ANSYS program cannot always detect such a defect before meshing, but an area or volume that crosses over itself will usually reveal itself in a meshing failure.

5.10.6.2. Use ANSYS Parameters

Use ANSYS parameters for dimensions that might change or that have a lot of digits. Typing a number like 2.8574639 over and over is tedious and is likely to lead to input errors.

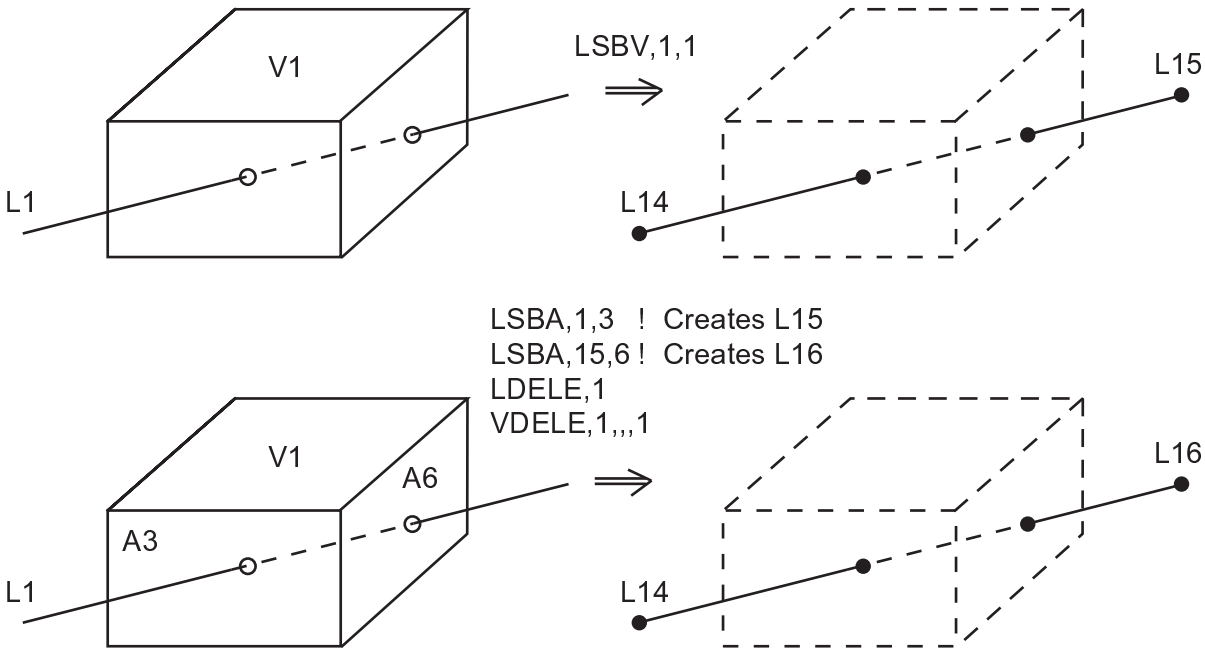
5.10.6.3. Consider Alternatives to Boolean Commands

When appropriate, use reasonable alternatives to Boolean commands to overcome Boolean failures or simply to reduce run times. (For example, dragging or rotating might sometimes be preferred over comparable Boolean modeling operations.)

5.10.6.4. Work with Lower-Dimension Constructions

Modeling sessions can sometimes be made more efficient, both with respect to run times and memory requirements, if Boolean operations can be performed using lower-dimension entities. For instance, replacing a line-volume subtraction [**LSBV**] with a series of line-area subtractions [**LSBA**] and a deletion [**LDELE**] might yield the same results at less cost. Switching to lower-dimension operations can sometimes save CPU time and memory requirements. (See Figure 5.72: Switching to Lower-Dimension (p. 95).) Of course, you must weigh the loss of convenience against any gain that you might realize. Switching to lower-dimension operations will probably not be justified unless you are experiencing out-of-memory errors, extremely long run times, or other such problems.

Figure 5.72: Switching to Lower-Dimension



5.10.6.5. Create Complex Models in Pieces

Consider creating complex solid models in pieces and combining them using the **RESUME** command (**Utility Menu> File> Resume from**) and **CDREAD** command (**Main Menu> Preprocessor> Archive Model> Read**) in the preprocessor (PREP7). The following is an example of this procedure:

```
/PREP7
RESUME,MODEL1,DB
CDWRITE,SOLID          !Write out just solid model (to file.iges)
RESUME,MODEL2,DB       !Note that this 2nd RESUME clears MODEL1 from the
                       !database and reads MODEL2
CDREAD,SOLID           !Reads in the solid model (from file.iges)

!Solid models from MODEL1 and MODEL2 are now
!in the current database

SAVE,MODEL3,DB         !Save combined model to separate file
FINISH
```

The **CDREAD** command automatically renumbers new entities to avoid conflicts with other solid model entities that already exist in the database. In addition, this command automatically executes a **NUMMRG**,KP command to merge duplicate solid-model entities.

5.10.6.6. Don't Forget to SAVE

Make it a habit to save your database [**SAVE**] (**Utility Menu> File> Save as**) before attempting a new or risky operation. This way, if an operation causes your system to hang or crash, or produces some other undesired result, you will have preserved a clean copy of your database to help you recover from such trouble.

5.10.6.7. Tessellation Errors

It is possible, although unlikely, that you might encounter the following error message at times as you are building your solid model:

```
 *** WARNING ***

Error in surface tessellation for area N.  This area can only be
displayed with /FACET,WIRE, and cannot be part of a V, VA, VSUM,
or ASUM operation.
```

Tessellation is the subdividing of an area for purposes of display (that is, surface faceting), area calculation (as in **ASUM**), and volume integration (as in **V**, **VA**, or **VSUM**). A tessellation error produces a *warning* message instead of an *error* message because an area that cannot be tessellated might still mesh successfully.

The types of situations that can cause tessellation errors include: areas having sharp knife edges, areas having too many holes, or areas having holes on extremely warped surfaces.

Note

For very narrow (sliver) areas or very thin volumes, such that the ratio of the minimum to the maximum dimension is less than 0.01, the **ASUM** and **VSUM** commands can provide erroneous area or volume information. To ensure that such calculations are accurate, make certain that you subdivide such areas and volumes so that the ratio of the minimum to the maximum is at least 0.05.

Chapter 6: Importing Solid Models from IGES Files

As an alternative to creating a solid model directly in the program, you can first create a solid model in your favorite CAD system, save it as an IGES file, and then import it. Once successfully imported, you can mesh the model just as you would for any model created in within the program.

The topics here cover the native IGES translation filter, *not* the connection functionality such as the connection for SAT. The connections are separately licensed and separately documented features that are included on the installation media. See the *Connection User's Guide* for more information.

6.1. Working With IGES Files

The Initial Graphics Exchange Specification (IGES) is a vendor-neutral standard format used to exchange geometric models between various CAD and CAE systems. The filter can import partial files, so you can generally import at least part of your file. You can also import multiple files into the same model.

ANSYS provides the SMOOTH method (NURBS-based or RV52) for importing IGES files. This method uses the standard ANSYS geometry database and has no capabilities for automatically creating volumes.

6.1.1. Using the SMOOTH Method

The SMOOTH method provides robust features for importing the model and preparing it for analysis. The method is also ideal for exporting your model to an IGES file or creating new modeling entities on top of the imported model.

6.1.1.1. Importing IGES files using the SMOOTH Method

To set the options for importing an IGES file:
Command(s): IOPTN
GUI: Utility Menu> File> Import> IGES

- Select the SMOOTH method.

To select the IGES file:
Command(s): IGESIN
GUI: File picker dialog box that follows after setting IGES options.

Respond **Yes** when prompted whether the IGES command should be executed.

6.1.1.2. Guidelines for Using the SMOOTH Method

When importing the IGES model, adhere to the following guidelines.

6.1.1.2.1. While Building the Model in the CAD System

- Observe solid modeling procedures with regard to planning, symmetry, and the amount of detail needed for a finite element analysis. For example, for axisymmetric models, the program requires that the global Y axis be the axis of rotation. Refer to Planning Your Approach (p. 5).
- Avoid creating closed curves (that is, a line that starts and ends at the same point and closed surfaces (such as a surface that starts and ends at the same edge). The program cannot store closed curves or closed surfaces. (At least two keypoints are required.) If a closed curve, closed surface, or "trimmed" closed surface--defined by IGES entities 120 *and* 144 or 128 *and* 144--is encountered while reading an IGES file, the program will attempt to split it into two or more entities.
- As much as possible, write to the IGES file using data that the program supports. (See the description of the **IOPTN** command in the *Command Reference*.)

6.1.1.2.2. While Writing the IGES File From the CAD Program

- Transfer only the portion of the geometry required for the analysis. A finite element analysis may not need as much detail as a CAD model requires.
- For trimmed surface transfer, include global XYZ data along with UV data in the IGES file.
- If the model to be analyzed is very large, use the CAD program's selection capabilities to create several IGES files, each containing a portion of the model. The ANSYS program will use the next available entity number as each file is read. You can then use the PREP7 merge feature (**NUMMRG** command or menu path **Main Menu> Preprocessor> Numbering Ctrls> Merge Items**) to merge coincident entities.
- Write the IGES file in ASCII format, with 80 characters per record.
- For the Pro/ENGINEER program, use these additional guidelines:
  - Set the Config.pro option "iges_out_trim_xyz" to "yes."
  - Set the accuracy to 1E-6 and regenerate the model.

6.1.1.2.3. While Reading the IGES File into ANSYS:

- Pay attention to the messages issued by the ANSYS program. Warning messages give details such as IGES entities not transferred and the corresponding ANSYS entity numbers.
- If any IGES entities were not transferred, reconstruct them using ANSYS solid modeling commands. The SMOOTH IGES filter is capable of reading in any rational B-spline curve entity (type 126), or rational B-spline surface entity (type 128) with a degree less than or equal to 20. Attempts to read in B-spline curve or surface entities of degree higher than 20 may result in error messages.
- Duplicate lines and keypoints are possible when transferring a model in from an IGES file. This often happens with CAD models due to the tolerances and practices that they were created with. You sometimes need to "clean up" these solid models with ANSYS commands that merge duplicate entities together (**NUMMRG** command or menu path **Main Menu> Preprocessor> Numbering Ctrls> Merge Items**).

Merging is done automatically when an IGES file is read into ANSYS [**IGESIN**] in AUX15. Default tolerances are used to determine if keypoints should be merged together into a single keypoint. Sometimes the default tolerances are not adequate and must be adjusted.

6.1.1.2.4. While Writing an IGES File from ANSYS:

- Set the system of units (**/UNITS**) before writing the IGES file. This information is captured on the IGES file and is read by many programs that read IGES files. (You cannot access the **/UNITS** command directly in the GUI.)
- Select all lower level solid modeling entities before writing the file (**ALLSEL**,BELOW,ALL or menu path **Utility Menu> Select> Everything Below**).
- If you wish to write out a portion of your model, select only those entities (areas) to be written and all corresponding lower level entities (lines and keypoints). Then unselect any higher level entities (volumes) before writing the file.

Chapter 7: Generating the Mesh

The process for generating a mesh of nodes and elements consists of three general steps:

1. Set the element attributes.
2. Set mesh controls (optional). A large number of mesh controls are available from which to choose.
3. Meshing the model.

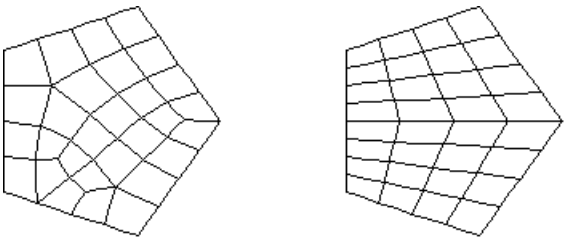
It is not always necessary to set mesh controls because the default mesh controls are appropriate for many models. If no controls are specified, the program will use the default settings (**DESIZE**) to produce a free mesh. Alternatively, you can use the SmartSize feature to produce a better quality free mesh.

The following mesh-generation topics are available:

7.1. Free or Mapped Mesh

Before meshing the model, and even before building the model, it is important to think about whether a free mesh or a mapped mesh is appropriate for the analysis. A *free* mesh has no restrictions in terms of element shapes, and has no specified pattern applied to it. A *mapped* mesh is restricted in terms of the element shape it contains and the pattern of the mesh. A mapped area mesh contains either only quadrilateral or only triangular elements, while a mapped volume mesh contains only hexahedron elements. In addition, a mapped mesh typically has a regular pattern, with obvious rows of elements. If you want this type of mesh, you must build the geometry as a series of fairly regular volumes and/or areas that can accept a mapped mesh.

Figure 7.1: Free and Mapped Meshes



You use the **MSHKEY** command or the equivalent GUI path (both of which are described later) to choose a free or a mapped mesh.

Keep in mind that the mesh controls you use will vary depending on whether a free or mapped mesh is desired. The details of free and mapped meshing will be explained later.

7.2. Setting Element Attributes

Before you generate a mesh of nodes and elements, you must first define the appropriate *element attributes*. That is, you must specify the following:

- The element type
- Real constant set (usually comprising the element's geometric properties, such as thickness or cross-sectional area)
- Material properties set (such as Young's modulus, thermal conductivity, etc.)
- Element coordinate system
- Section ID (current-technology beam elements only--see Generating a Beam Mesh With Orientation Nodes (p. 147))

For beam meshing only, you may also specify orientation keypoints as attributes *of a line*. Generating a Beam Mesh With Orientation Nodes (p. 147) describes beam meshing in detail.

7.2.1. Creating Tables of Element Attributes

To assign attributes to your elements, you must first build tables of element attributes. Typical models include element types (**ET** command or menu path **Main Menu> Preprocessor> Element Type> Add/Edit/Delete**), real constants (**R** command or menu path **Main Menu> Preprocessor> Real Constants**), and material properties (**MP** and **TB** family of commands, menu path **Main Menu> Preprocessor> Material Props>** ***material option***).

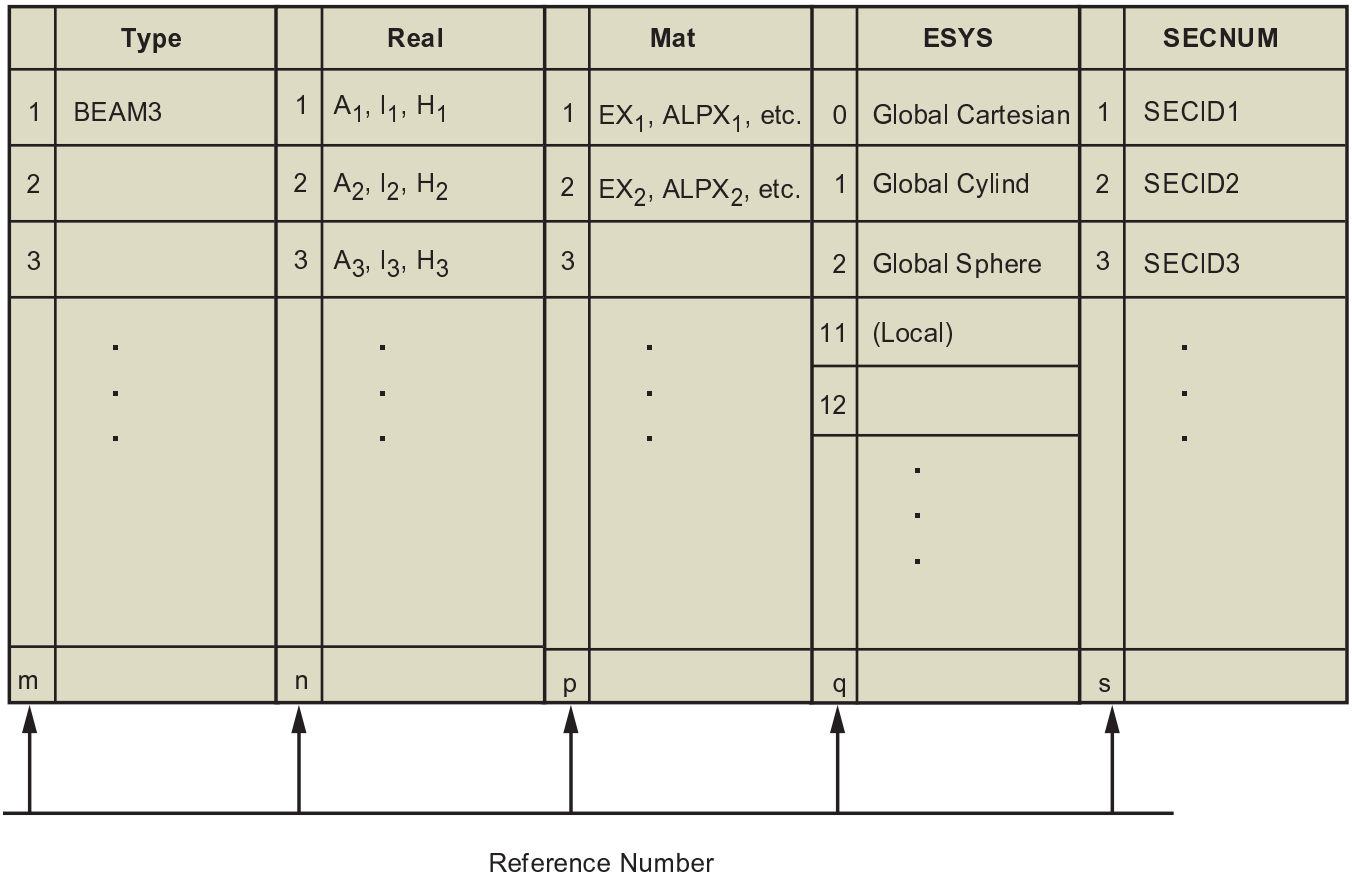
A table of coordinate systems can also be assembled using commands such as **LOCAL**, **CLOCAL**, etc. (**Utility Menu> WorkPlane> Local Coordinate Systems> Create Local CS>** ***option***). This table can be used to assign element coordinate systems to elements. (Not all element types can be assigned a coordinate system in this manner. See Element Coordinate Systems (p. 23) of this manual for information about element coordinate systems. For element descriptions, see the *Element Reference*.)

For beam meshing with BEAM188 or BEAM189 elements, you can build a table of sections using the **SECTYPE** and **SECDATA** commands (**Main Menu> Preprocessor> Sections**).

Orientation keypoints are attributes of a *line*; they are not *element* attributes. You *cannot* create tables of orientation keypoints. See Assigning Element Attributes Before Meshing (p. 103) for more information.

The element attribute tables described above can be visualized as shown in Figure 7.2: Element Attribute Tables (p. 103). (For more information on creating your element attribute tables, see Getting Started in the *Basic Analysis Guide*.)

Figure 7.2: Element Attribute Tables



You can review the contents of the element type, real constant, and material tables by issuing the **ETLIST** (TYPE table), **RLIST** (REAL table), or **MPLIST** (MAT table) commands (or by choosing the equivalent menu path **Utility Menu> List> Properties>** ***property type***). You can review the coordinate system table by issuing **CSLIST** (**Utility Menu> List> Other> Local Coord Sys**). You can review the section table by issuing **SLIST** (**Main Menu> Preprocessor> Sections> List Sections**).

7.2.2. Assigning Element Attributes Before Meshing

Once the attribute tables are assembled, you can assign element attributes to different parts of your model by "pointing" to the appropriate entries in the tables. The pointers are simply a set of reference numbers that include a material number (MAT), a real constant set number (REAL), an element type number (TYPE), a coordinate system number (ESYS) and, for beam meshing with BEAM188, or BEAM189, a section ID number (SECNUM). You can either assign the attributes directly to selected solid model entities, or define a default set of attributes that will be used for elements created in subsequent meshing operations.

Note

As stated earlier, although you can assign orientation keypoints as attributes of a line for beam meshing, you cannot build tables of orientation keypoints. Therefore, to assign orientation keypoints as attributes, you must assign them directly to selected lines; you cannot define a default set of orientation keypoints to be used in subsequent meshing operations. See Generating a Beam Mesh With Orientation Nodes (p. 147) for details about assigning orientation keypoints.

7.2.2.1. Assigning Attributes Directly to the Solid Model Entities

Assigning the element attributes to the solid model entities allows you to preassign attributes for each region of your model. By using this method, you can avoid having to reset attributes in the middle of meshing operations. (Clearing a solid model entity of its nodes and elements will *not* delete attributes assigned directly to the entity.)

Use the commands and GUI paths listed below to assign attributes directly to solid model entities.

- To assign attributes to keypoints:
  Command(s): KATT
  GUI: Main Menu> Preprocessor> Meshing> Mesh Attributes> All Keypoints
  Main Menu> Preprocessor> Meshing> Mesh Attributes> Picked KPs

- To assign attributes to lines:
  Command(s): LATT
  GUI: Main Menu> Preprocessor> Meshing> Mesh Attributes> All Lines
  Main Menu> Preprocessor> Meshing> Mesh Attributes> Picked Lines

- To assign attributes to areas:
  Command(s): AATT
  GUI: Main Menu> Preprocessor> Meshing> Mesh Attributes> All Areas
  Main Menu> Preprocessor> Meshing> Mesh Attributes> Picked Areas

- To assign attributes to volumes:
  Command(s): VATT
  GUI: Main Menu> Preprocessor> Meshing> Mesh Attributes> All Volumes
  Main Menu> Preprocessor> Meshing> Mesh Attributes> Picked Volumes

7.2.2.2. Assigning Default Attributes

You can assign a set of default attributes by simply pointing to various entries in the attribute tables. The pointers that are in effect at the time you create your elements (that is, when you initiate meshing) are used by the program to assign attributes from the tables to the solid model and to the elements. Attributes assigned directly to the solid model entities (as described above) will override the default attributes. Also, if you clear a solid model entity of its nodes and elements, any attributes that were assigned through default attributes will be deleted.

To assign a set of default attributes:
Command(s): TYPE, REAL, MAT, ESYS, SECNUM
GUI: Main Menu> Preprocessor> Meshing> Mesh Attributes> Default Attribs
Main Menu> Preprocessor> Modeling> Create> Elements> Elem Attributes

7.2.2.3. Automatic Selection of the Dimensionally Correct Element Type

In certain cases, the program can choose the correct element type for a meshing or extrusion operation, eliminating the need for you to manually switch between element types when the correct choice is obvious.

Specifically, if you fail to assign an element type directly to a solid model entity [**xATT**] *and* the default element type [**TYPE**] is not dimensionally correct for the operation that you want to perform, *but there is only one dimensionally correct element type in the currently defined element attribute tables,* ANSYS will automatically use that element type to proceed with the operation.

The meshing and extrusion operations affected by this feature are **KMESH**, **LMESH**, **AMESH**, **VMESH**, **FVMESH**, **VOFFST**, **VEXT**, **VDRAG**, **VROTAT**, and **VSWEEP**.

7.2.2.4. Defining a Variable Thickness at Nodes

You can define a thickness at nodes for shell elements.

To define the variable thickness, use either of these methods:
Command(s): RTHICK
GUI: Main Menu> Preprocessor> Real Constants> Thickness Func

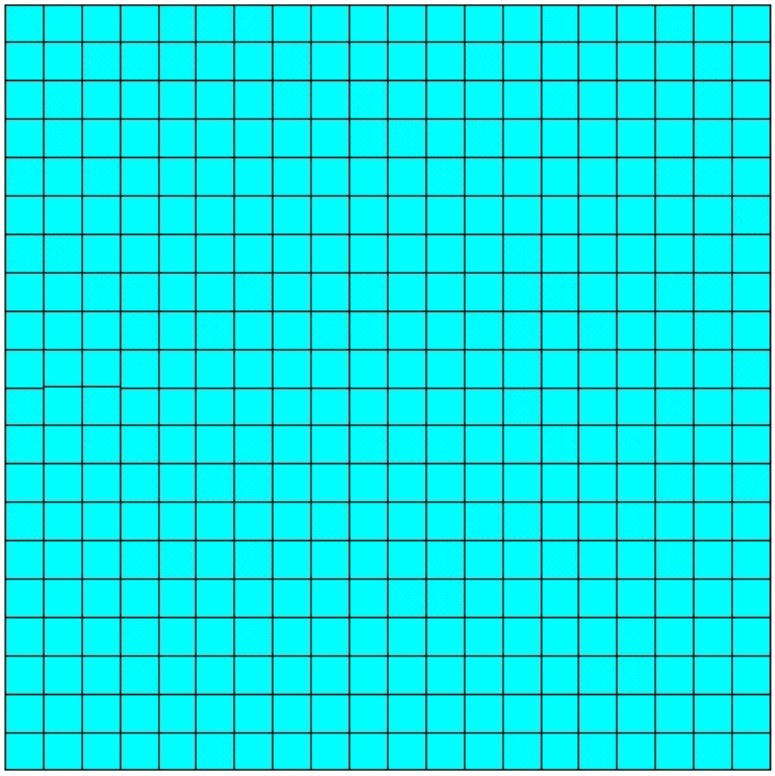
Shell elements are capable of modeling complex distributions of thickness. SHELL181, for example, permits different thicknesses to be assigned at each of its four corner nodes. Each individual element assumes a smooth variation between the given corner values.

Defining a complicated thickness variation on a group of elements can be a challenge. In the worst case, every element needs its own unique set of real constant thicknesses. For some of these situations, using **RTHICK** can simplify the model definition.

This procedure is illustrated by the sample input listing and figure that appears below, which show the creation of a 10 x 10 rectangle filled with 0.5 x 0.5 square SHELL181 elements.

```
/TITLE, RTHICK Example
/PREP7
ET,1,181,,,2
RECT,,10,,10
ESHAPE,2
ESIZE,,20
AMESH,1
EPLO
```

Figure 7.3: Original Elements



The thickness should vary according to the formula: $thickness = 0.5 + 0.2x + 0.02y^2$.

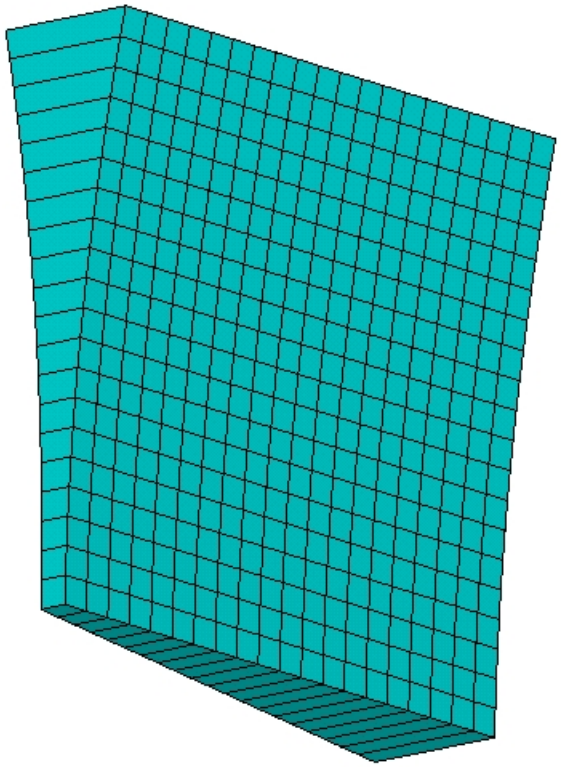
To accomplish this variation, you can create an array parameter that maps thickness to node number. (In other words, the N^{th} member of the array is the desired thickness at node *N*.)

```
*GET,MXNODE,NODE,,NUM,MAXD
*DIM,THICK,,MXNODE
*DO,NODE,1,MXNODE
   *IF,NSEL(NODE),EQ,1,THEN
      THICK(node) = 0.5 + 0.2*NX(NODE) + 0.02*NY(NODE)**2
   *ENDIF
*ENDDO
NODE = $ MXNODE =
```

Finally, assign the thickness in the array parameter to the elements using the **RTHICK** function.

```
RTHICK,THICK(1),1,2,3,4
/ESHAPE,1.0  $ /USER,1  $ /DIST,1,7
/VIEW,1,-0.75,-0.28,0.6  $ /ANG,1,-1
/FOC,1,5.3,5.3,0.27  $ EPLO
```

Figure 7.4: Shell Elements with Thickness Shown



7.3. Mesh Controls

The default mesh controls that the program uses may produce a mesh that is adequate for the model you are analyzing. In this case, you will not need to specify any mesh controls. However, if you do use mesh controls, you must set them before meshing your solid model.

Mesh controls allow you to establish such factors as the element shape, midside node placement, and element size to be used in meshing the solid model. This step is one of the most important of your entire analysis, for the decisions you make at this stage in your model development will profoundly affect the accuracy and economy of your analysis. (See Planning Your Approach (p. 5) of this manual for more detailed discussions of some of the factors you should consider as you set mesh controls.)

7.3.1. The MeshTool

The MeshTool (**Main Menu> Preprocessor> Meshing> MeshTool**) provides a convenient path to many of the most common mesh controls, as well as to the most frequently performed meshing operations. The MeshTool is an interactive "tool box," not only because of the numerous functions (or tools) that it contains, but also because once you open it, it remains open until you either close it or you exit PREP7.

Although all of the functions available via the MeshTool are also available via the traditional commands and menus, using the MeshTool is a valuable shortcut.

The many functions available via the MeshTool include:

- Controlling SmartSizing levels
- Setting element size controls
- Specifying element shape
- Specifying meshing type (free or mapped)
- Meshing solid model entities
- Clearing meshes
- Refining meshes

This guide covers all of these functions in detail. For details about the MeshTool, access it using the path listed above and click on its Help button.

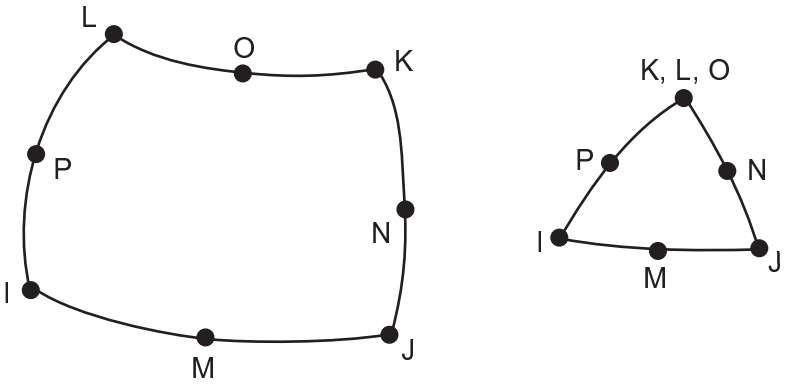
7.3.2. Element Shape

At a minimum, you should set the allowable element shapes if you plan on meshing with an element type that can take on more than one shape. For instance, many area elements can be both triangular and quadrilateral shaped within the same meshed area. Volume elements can often be either hexahedral (brick) or tetrahedral shaped, but a mixture of the two shapes in the same model is not recommended. (An exception to this involves the use of transitional pyramid elements, which is described in Creating Transitional Pyramid Elements (p. 124) of this manual.)

7.3.2.1. A Note About Degenerate Element Shapes

This section assumes that you are somewhat familiar with the concept of degenerate element shapes. For example, consider the PLANE183 element, which is a 2-D structural solid element having eight nodes (I, J, K, L, M, N, O, P). By default, PLANE183 has a quadrilateral shape. However, a triangular-shaped element can be formed by defining the same node number for nodes K, L, and O. Thus PLANE183 can be "degenerated" into a triangle. See Figure 7.5: An Example of a Degenerate Element Shape (p. 108) for an illustration of PLANE183 in both its default and degenerate forms.

Figure 7.5: An Example of a Degenerate Element Shape



Quadrilateral Shape (Default) Triangular Option

Although it can be helpful for you to understand this concept, when specifying element shapes before meshing, you do not have to concern yourself with whether a shape is the default or degenerate shape

of a particular element. Instead, you can think in more simpler terms of the desired element shape itself (quadrilateral, triangle, hexahedra, or tetrahedra).

For details about degenerate element shapes, see the *Element Reference*.

7.3.2.2. Element Shape Specification

To specify element shapes, use either of these methods:
Command(s): MSHAPE,*KEY*,*Dimension*
GUI: Main Menu> Preprocessor> Meshing> MeshTool
Main Menu> Preprocessor> Meshing> Mesher Opts
Main Menu> Preprocessor> Meshing> Mesh> Volumes> Mapped> 4 to 6 sided

There are two factors to consider when specifying element shape: the desired element shape and the dimension of the model to be meshed.

7.3.2.2.1. Command Method

If you are using the **MSHAPE** command, the value of the *Dimension* argument (2-D or 3-D) indicates the dimension of the model to be meshed. The value of the *KEY* argument (0 or 1) indicates the element shape to be used in the mesh:

- When *KEY* = 0, the program meshes with quadrilateral-shaped elements if *Dimension* = 2-D and with hexahedral-shaped elements if *Dimension* = 3-D (as long as the element type supports quadrilateral or hexahedral element shapes, respectively).
- When *KEY* = 1, the program meshes with triangle-shaped elements if *Dimension* = 2-D and with tetrahedral-shaped elements if *Dimension* = 3-D (as long as the element type supports triangle or tetrahedral element shapes, respectively).

7.3.2.2.2. GUI Method (Via the MeshTool)

For increased productivity, the MeshTool (described earlier in this chapter) is the recommended method for specifying element shape. You access the MeshTool via the following GUI path: **Main Menu> Preprocessor> Meshing> MeshTool**. Using the MeshTool, you simply click on the desired element shape that you want to use to mesh the model. From the MeshTool, you can also click on the type of meshing (free or mapped) that you want. (For more information, see Choosing Free or Mapped Meshing (p. 110), "Choosing Free or Mapped Meshing.") Using the MeshTool makes selecting the shape simple, because it presents only those shapes that are compatible with the type of meshing that you are requesting, as well as with the dimension of the model you are meshing. (See Table 7.1: Supported Combinations of Element Shape and Meshing Type (p. 110) for the combinations of element shapes and meshing types supported.)

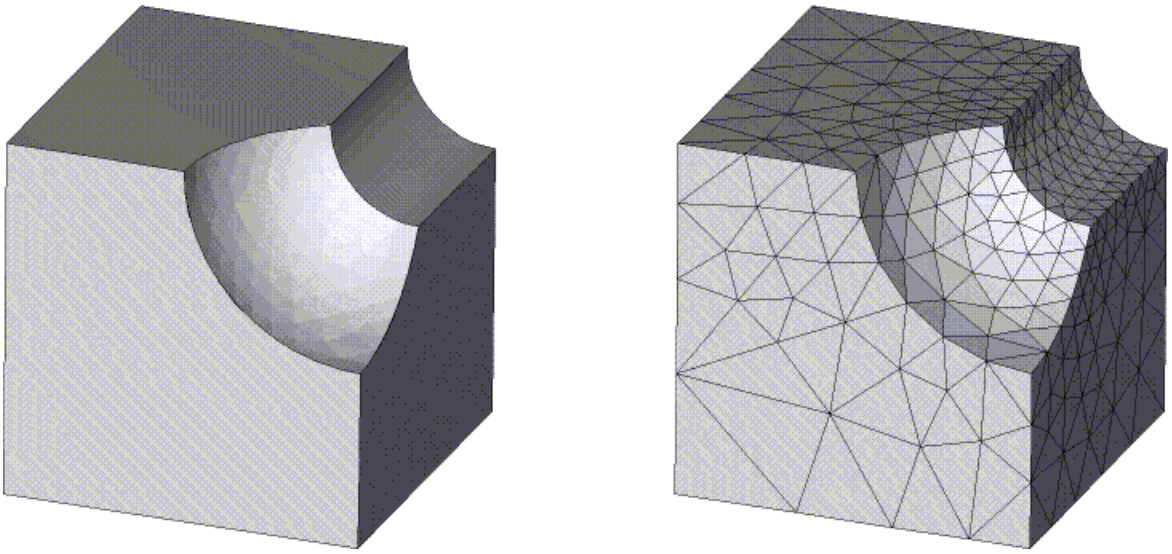
Note

Since element shape specification is closely related to the type of meshing that you request (free or mapped), it may help you to read Choosing Free or Mapped Meshing (p. 110) of this manual ("Choosing Free or Mapped Meshing") before specifying element shape.

In some cases, the **MSHAPE** command and the appropriate meshing command (**AMESH**, **VMESH**, or the equivalent menu path **Main Menu> Preprocessor> Meshing> Mesh> *meshing option***) are all that you will need to mesh your model. The size of each element will be determined by default element

size specifications [**SMRTSIZE** or **DESIZE**]. For instance, the model below in Figure 7.6: Default Element Sizes (p. 110) (left) can be meshed with one **VMESH** command to produce the mesh shown on the right:

Figure 7.6: Default Element Sizes



The element sizes that the program chose for the above model may or may not be adequate for the analysis, depending on the physics of the structure. One way to change the mesh would be to change the default SmartSize level [**SMRTSIZE**] and remesh. For details, see Smart Element Sizing for Free Meshing (p. 111) of this manual.

7.3.3. Choosing Free or Mapped Meshing

In addition to specifying element shape, you may also want to specify the type of meshing (free or mapped) that should be used to mesh your model. You do this by setting the meshing key:
Command(s): MSHKEY
GUI: Main Menu> Preprocessor> Meshing> MeshTool
Main Menu> Preprocessor> Meshing> Mesher Opts

As described in Element Shape Specification (p. 109) you can use the MeshTool (**Main Menu> Preprocessor> Meshing> MeshTool**) to specify meshing type. The MeshTool is the recommended method. Refer to Element Shape Specification (p. 109) for related information.

Together, the settings for element shape [**MSHAPE**] and meshing type [**MSHKEY**] affect the resulting mesh. Table 7.1: Supported Combinations of Element Shape and Meshing Type (p. 110) shows the combinations of element shape and meshing type that the program supports.

Table 7.1: Supported Combinations of Element Shape and Meshing Type

Element Shape	Free Meshing	Mapped Meshing	Mapped If Possible; Otherwise Free Mesh With SmartSizing On
Quadrilateral	Yes	Yes	Yes
Triangle	Yes	Yes	Yes
Hexahedral	No	Yes	No
Tetrahedral	Yes	No	No

Table 7.2: Failure to Specify Element Shape and/or Meshing Type (p. 111) explains what happens when you fail to specify values for these settings.

Table 7.2: Failure to Specify Element Shape and/or Meshing Type

Your Action ...	How it Affects the Mesh ...
You issue the **MSHAPE** command with no arguments.	Quadrilateral-shaped or hexahedral-shaped elements are used to mesh the model, depending on whether you are meshing an area or a volume.
You do not specify an element shape, but you do specify the type of meshing to be used.	The default shape of the element is used to mesh the model with your specified meshing type.
You specify neither an element shape nor the type of meshing to be used.	The default shape of the element is used to mesh the model with whichever type of meshing is the default for that shape.

See the descriptions of the **MSHAPE** and **MSHKEY** commands in the *Command Reference* for more information.

7.3.4. Controlling Placement of Midside Nodes

When meshing with quadratic elements, you can control the placement of midside nodes. Your choices for midside node placement are:

- Midside nodes (if any) of elements on a region boundary follow the curvature of the boundary line or area. This is the default.
- Place midside nodes of all elements so that element edges are straight. This option allows a coarse mesh along curves. However, the curvature of the model is not matched.
- Do not create midside nodes (elements have removed midside nodes).

To control midside node placement:
Command(s): MSHMID
GUI: Main Menu> Preprocessor> Meshing> Mesher Opts

7.3.5. Smart Element Sizing for Free Meshing

Smart element sizing (SmartSizing) is a meshing feature that creates initial element sizes for free meshing operations. SmartSizing gives the mesher a better chance of creating reasonably shaped elements during automatic mesh generation. This feature, which is controlled by the **SMRTSIZE** command, provides a range of settings (from coarse to fine mesh) for meshing.

By default, the **DESIZE** method of element sizing will be used during free meshing (see Default Element Sizes for Mapped Meshing (p. 114)). However, it is recommended that SmartSizing be used instead for free meshing. To turn SmartSizing on, simply specify an element size level on the **SMRTSIZE** command (see the discussion on basic controls below).

Note

If you use SmartSizing on a model that contains only an area, the program uses the area to calculate the guiding element size that it should use to mesh the model. On the other hand,

if you use SmartSizing on a model that contains both an area and a volume, the program uses the volume to calculate the guiding element size for the model. Even if the area in the first model (area only) and the area in the second model (area and volume) are exactly the same, and the SmartSizing setting is the same, the elements that the program uses to mesh the first model will usually not be as coarse as the elements that it uses to mesh the second model. The program does this to prevent volumes from being meshed with too many elements. (However, if you have specified a global element size [**ESIZE**], the size of the elements will be the same for both models, because the program will use the size that you specified as the guiding element size.)

Note

When you use SmartSizing, we recommend that in most cases you specify the desired SmartSizing settings [**SMRTSIZE**] and then mesh the entire model at once [**AMESH**,ALL or **VMESH**,ALL], rather than SmartSizing area by area or volume by volume. This gives SmartSizing the opportunity to reduce element sizes near small features in adjacent regions. However, you should not try to SmartSize in a single operation areas or volumes that just touch (rather than sharing common keypoints, lines, or areas), such as might exist in a model prepared for contact analysis. The near zero proximity can cause SmartSizing to compute very small element sizes and produce an unreasonably fine mesh, with a huge number of nodes and elements. You should mesh each contiguous piece of such a model as a separate group of areas or volumes.

7.3.5.1. The Advantages of SmartSizing

The SmartSizing algorithm first computes estimated element edge lengths for all lines in the areas or volumes being meshed. The edge lengths on these lines are then refined for curvature and proximity of features in the geometry. Since all lines and areas are sized before meshing begins, the quality of the generated mesh is not dependent on the order in which the areas or volumes are meshed. (Remember that for best results, all areas or volumes should be meshed at the same time.)

If quadrilateral elements are being used for area meshing, SmartSizing tries to set an even number of line divisions around each area so that an all-quadrilateral mesh is possible. Triangles will be included in the mesh only if forcing all quadrilaterals would create poorly shaped elements, or if odd divisions exist on boundaries.

7.3.5.2. SmartSizing Controls - Basic versus Advanced

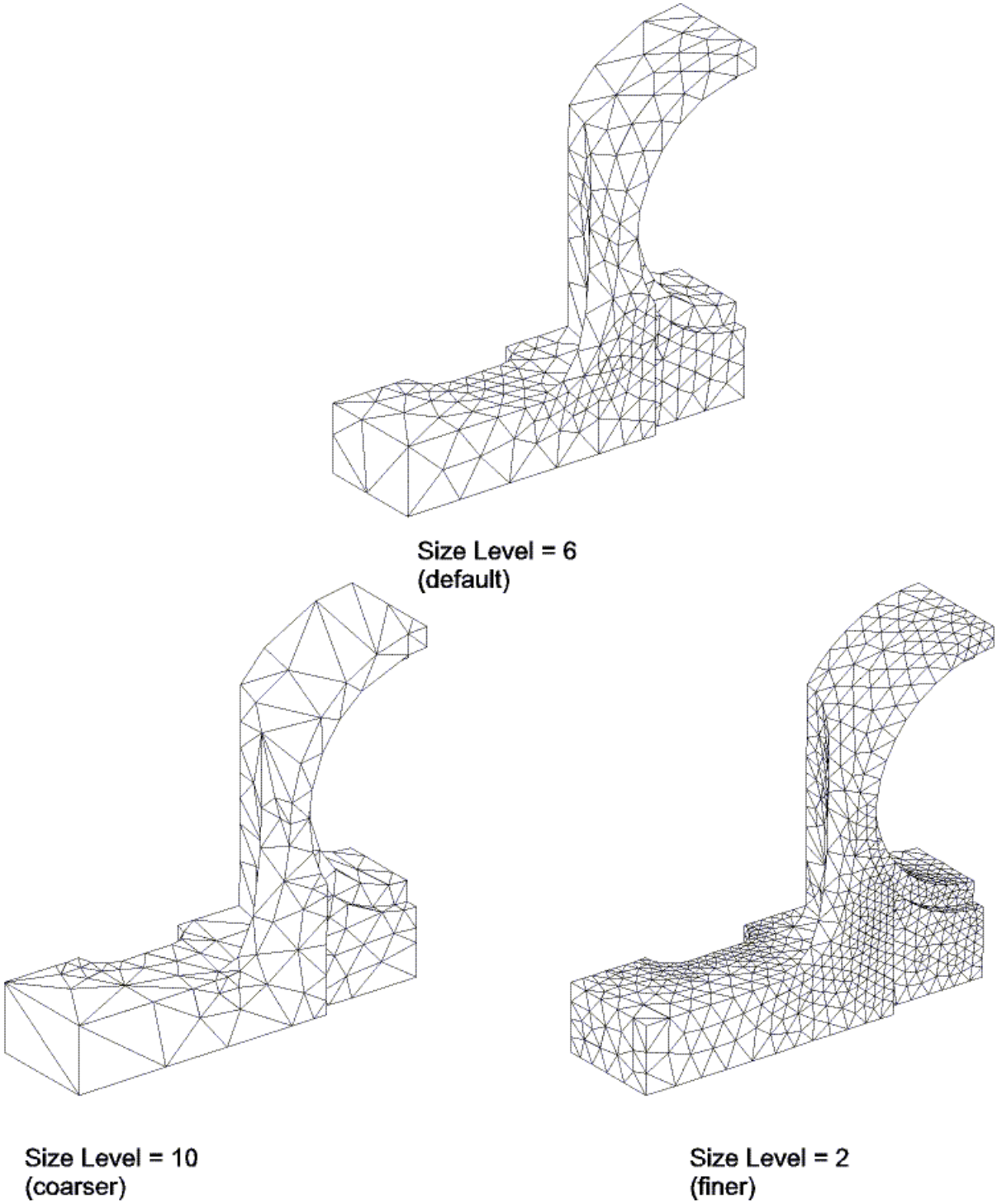
There are two categories of SmartSizing controls: basic and advanced.

7.3.5.2.1. Basic Controls

To use the basic controls, you simply specify a mesh size level from 1 (fine mesh) to 10 (coarse mesh). The program automatically sets a series of individual control values that are used to produce the requested size level. To specify the size level, use one of the following methods:
Command(s): SMRTSIZE,*SIZLVL*
GUI: Main Menu> Preprocessor> Meshing> MeshTool
Main Menu> Preprocessor> Meshing> Size Cntrls> SmartSize> Basic

Figure 7.7: Varying SmartSize Levels for the Same Model (p. 113) shows a model meshed with several different SmartSize settings, including the default size level of 6.

Figure 7.7: Varying SmartSize Levels for the Same Model



7.3.5.2.2. Advanced Controls

You may prefer to use the advanced method, which involves setting the individual control quantities manually. This allows you to "tweak" the mesh to better fit your needs. You can change such things as the small hole and small angle coarsening keys, and the mesh expansion and transition factors (see the description of the **SMRTSIZE** command for a complete list of advanced controls). In addition, you can set a starting element size for SmartSizing with the **ESIZE** command.

Use one of the following methods to set advanced SmartSizing controls:

Command(s): SMRTSIZE and **ESIZE**
GUI: Main Menu> Preprocessor> Meshing> Size Cntrls> SmartSize> Adv Opts

7.3.5.3. Interaction with Other Mesh Controls

Local element sizing controls (discussed later in Local Mesh Controls (p. 116)) can be used in conjunction with SmartSizing. However, if conflicting element sizes are set, the SmartSizing algorithm will handle them as follows:

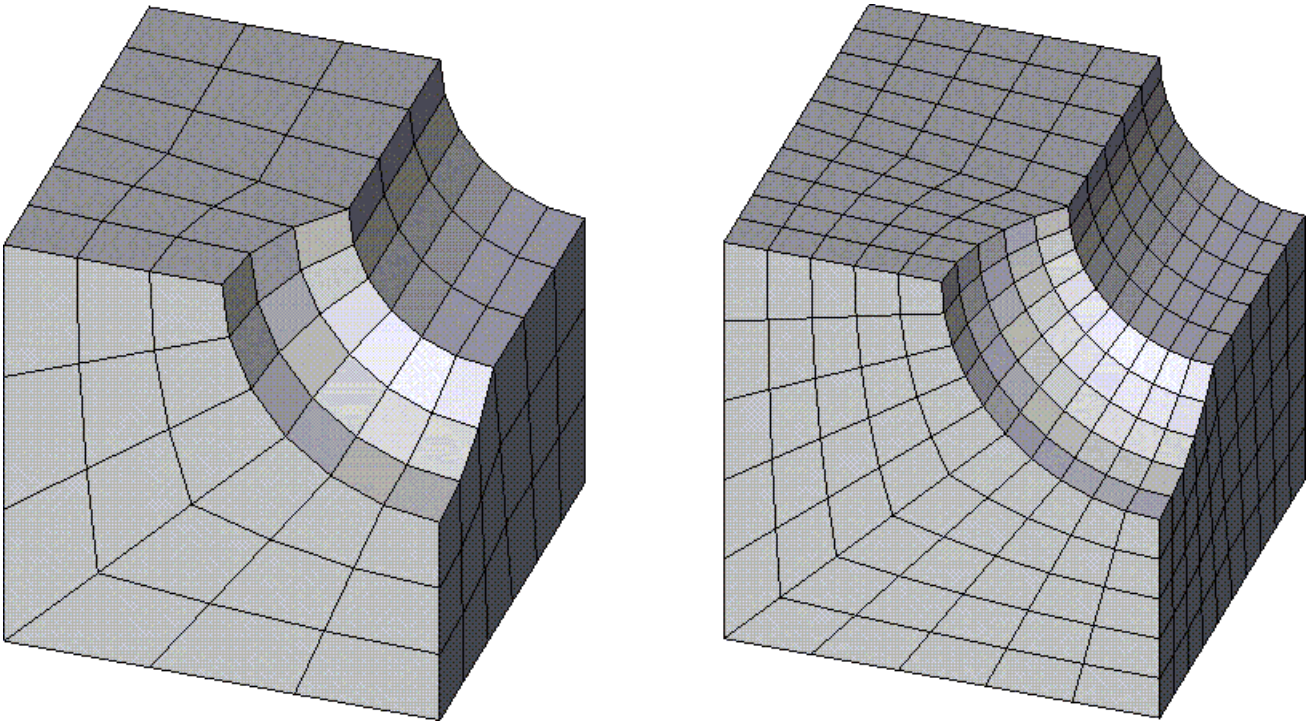
- Any element size specifications at areas (**AESIZE** command or menu path **Main Menu> Preprocessor> Meshing> Size Cntrls> Areas> *option***) will be considered, but may be overridden to accommodate curvature and proximity of features.
- Any element size specifications on lines (**LESIZE** command or menu path **Main Menu> Preprocessor> Meshing> Size Cntrls> Lines> *option***) will be optionally used as defined. (The KYNDIV argument on **LESIZE** allows you to assign specifications that can be overwritten where needed.)
- Any element size specifications at keypoints (**KESIZE** command or menu path **Main Menu> Preprocessor> Meshing> Size Cntrls> Keypoints> *option***) will be considered, but may be overridden to accommodate curvature and proximity of features.
- If a global element size is set (**ESIZE** command or menu path **Main Menu> Preprocessor> Meshing> Size Cntrls> Global> Size**), it will be overridden as necessary to accommodate curvature and proximity of features. If a consistent element size is desired, set the global element size and turn SmartSizing off (**SMRTSIZE**,OFF or menu path **Main Menu> Preprocessor> Meshing> Size Cntrls> SmartSize> Basic**).
- Default element sizes specified with the **DESIZE** command (**Main Menu> Preprocessor> Meshing> Size Cntrls> Global> Other**) are ignored when SmartSizing is on.

7.3.6. Default Element Sizes for Mapped Meshing

The **DESIZE** command allows you to modify such defaults as: the minimum and maximum number of elements that will be attached to an unmeshed line, maximum spanned angle per element, and minimum and maximum edge length. The **DESIZE** command (**Main Menu> Preprocessor> Meshing> Size Cntrls> Global> Other**) is always used to control element sizing for mapped meshing. **DESIZE** settings are also used by default for free meshing. However, it is recommended that you use SmartSizing [**SMRTSIZE**] instead for free meshing operations.

As an example, the mapped mesh on the left in Figure 7.8: Changing Default Element Sizes (p. 115) was produced with the element size defaults that exist when you enter the program. The mesh on the right was produced by modifying the minimum number of elements (*MINL*) and the maximum spanned angle per element (*ANGL*) on the **DESIZE** command.

Figure 7.8: Changing Default Element Sizes



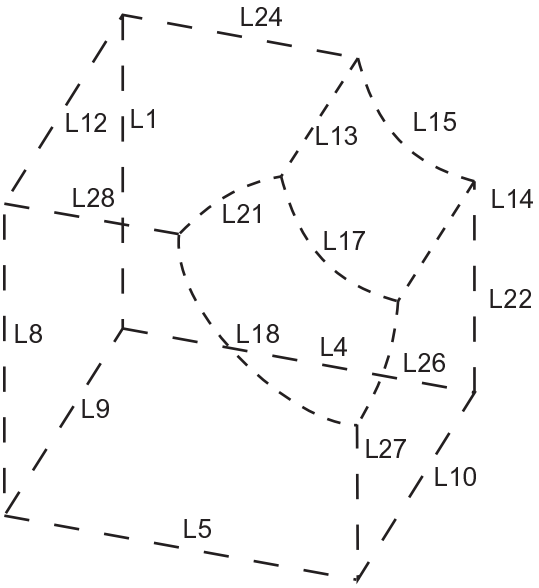
For larger models, it may be wise to preview the default mesh that will result from the **DESIZE** specifications. This can be done by viewing the line divisions in a line display. The steps for previewing a default mesh are as follows:

1. Build solid model.
2. Select element type.
3. Select allowable element shapes [**MSHAPE**].
4. Select mesher (free or mapped) for meshing [**MSHKEY**].
5. Issue **LESIZE**,ALL (this adjusts line divisions based on **DESIZE** specifications).
6. Request a line plot [**LPLOT**].

For instance:

```
.
.
.
ET,1,45   ! 8-node hexahedral-shaped element
MSHAPE,0   ! Use hexahedra
MSHKEY,1  ! Use mapped meshing
LESIZE,ALL  ! Adjust line divisions based on DESIZE
LPLOT
```

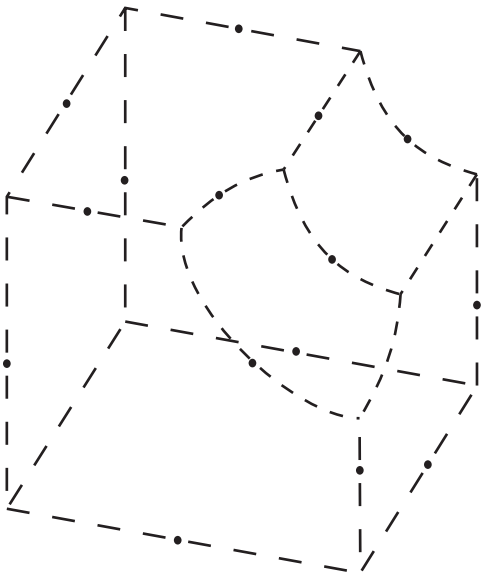
Figure 7.9: Previewing the Default Mesh



If the resulting mesh looks as though it will be too coarse, it can be changed by altering the element size defaults:

```
.
.
.
DESIZE,5,,30,15  ! Change default element sizes
LESIZE,ALL,,,,,1  ! Adjust line divisions based on DESIZE, force adjustments
LPLOT
```

Figure 7.10: Previewing the Modified Mesh



7.3.7. Local Mesh Controls

In many cases, the mesh produced by default element sizes is not appropriate due to the physics of the structure. Examples include models with stress concentrations or singularities. In these cases, you will have to get more involved with the meshing process. You can take more control by using the following element size specifications:

- To control the global element size in terms of the element edge length used on surface boundaries (lines) or the number of element divisions per line:
  Command(s): ESIZE
  GUI: Main Menu> Preprocessor> Meshing> Size Cntrls> Global> Size

- To control the element sizes near specified keypoints:
  Command(s): KESIZE
  GUI: Main Menu> Preprocessor> Meshing> Size Cntrls> Keypoints> All KPs
  Main Menu> Preprocessor> Meshing> Size Cntrls> Keypoints> Picked KPs
  Main Menu> Preprocessor> Meshing> Size Cntrls> Keypoints> Clr Size

- To control the number of elements on specified lines:
  Command(s): LESIZE
  GUI: Main Menu> Preprocessor> Meshing> Size Cntrls> Lines> All Lines
  Main Menu> Preprocessor> Meshing> Size Cntrls> Lines> Picked Lines
  Main Menu> Preprocessor> Meshing> Size Cntrls> Lines> Clr Size

---

Note

When you use the GUI method to set the number of elements on specified lines, and any of those lines is connected to one or more meshed lines, areas, or volumes, the program prompts you to determine whether you want to clear the meshed entities. If you answer yes to the prompt, the program clears the meshed entities. (This occurs only when you perform this operation via the GUI; the program does not prompt you when you use the command method [**LESIZE**].)

All of the size specifications described above can be used together. If conflicting element sizes are set using more than one of the above commands, a specific hierarchy is observed. The hierarchy will vary slightly, depending on whether the **DESIZE** or **SMRTSIZE** method of default element sizing is used.

- *Hierarchy used for* **DESIZE** element sizing. For any given line, element sizes along the line are established as follows:

  – Line divisions specified with **LESIZE** are always honored.

  – If line divisions have not been set for the line, **KESIZE** specifications at its keypoints (if any) are used.

  – If there are no size specifications on the line or on its keypoints, element sizes are established by the **ESIZE** specification.

  – If none of the above size specifications are set, **DESIZE** settings will control element sizes for the line.

- *Hierarchy used for* **SMRTSIZE** element sizing. For any given line, element sizes along the line are established as follows:

  – Line divisions specified with **LESIZE** are always honored.

  – If line divisions have not been set for the line, **KESIZE** specifications at its keypoints (if any) are used, but may be overridden to account for curvature and small geometric features.

– If there are no size specifications on the line or on its keypoints, **ESIZE** specification will be used as a starting element size, but may be overridden to account for curvature and small geometric features.

– If none of the above size specifications are set, **SMRTSIZE** settings will control element sizes for the line.

Note

Line divisions that have been established by **KESIZE** or **ESIZE** and a meshing operation will show up as negative numbers in a line listing [**LLIST**], while line divisions that you set via **LESIZE** show up as positive numbers. The signs of these numbers affect how the program treats the line divisions if you clear the mesh later (**ACLEAR**, **VCLEAR**, etc., or menu path **Main Menu> Preprocessor> Meshing> Clear>** ***entity***). If the number of line divisions is positive, the program does not remove the line divisions during the clearing operation; if the number is negative, the program removes the line divisions (which will then show up as zeros in a subsequent line listing).

If you are performing a linear static structural or linear steady-state thermal analysis, you can let the program establish meshing controls automatically as it adapts element sizes to drive the estimated error in the analysis below a target value. This procedure, known as *adaptive meshing*, is described in Adaptive Meshing in the *Advanced Analysis Guide*.

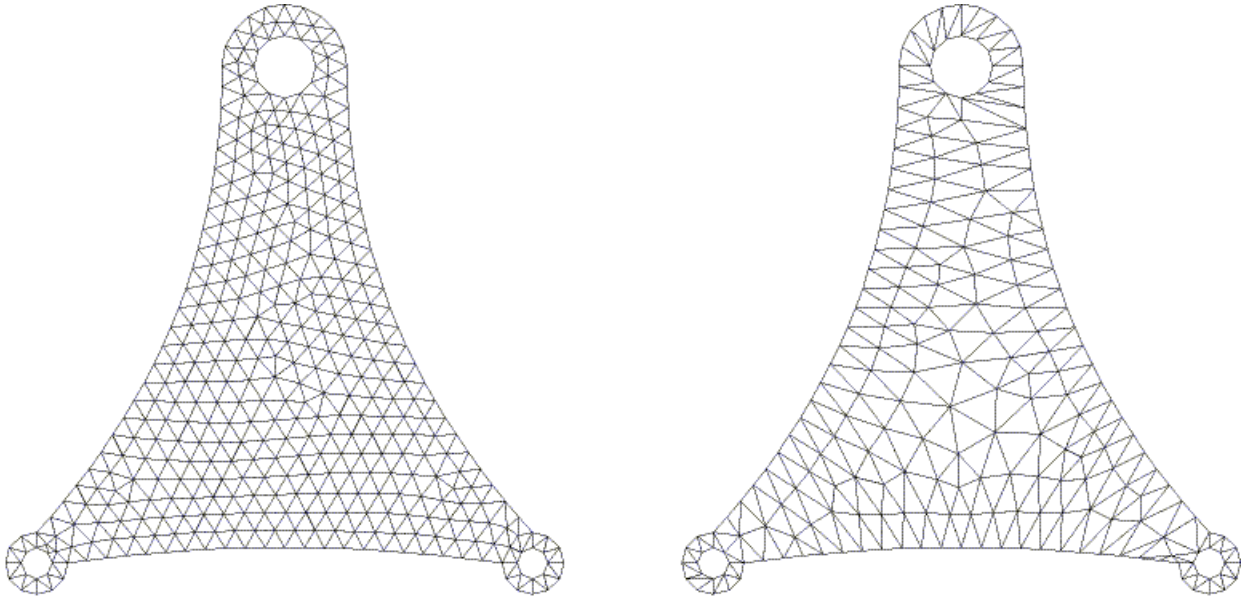
7.3.8. Interior Mesh Controls

The discussion on meshing specifications has focused thus far on the setting of element sizes on the boundaries of the solid model (**LESIZE**, **ESIZE**, etc.). However, you can also control the mesh on the interior of an area where there are no lines to guide the size of the mesh. To do so, use one of the following methods:
Command(s): MOPT
GUI: Main Menu> Preprocessor> Meshing> Size Cntrls> Global> Area Cntrls

7.3.8.1. Controlling Mesh Expansion

The *Lab* = EXPND option on the **MOPT** command can be used to guide the mesh from a fine mesh on the boundary of an area to a coarse mesh on the interior (see Figure 7.11: Area Mesh Without and With Mesh Expansion (p. 119)).

Figure 7.11: Area Mesh Without and With Mesh Expansion



In Figure 7.11: Area Mesh Without and With Mesh Expansion (p. 119), mesh (a) was created based only on the setting of the **ESIZE** command (**Main Menu> Preprocessor> Meshing> Size Cntrls> Global> Size**). Notice that the elements are well shaped, but that 698 elements are required to fill the area since the elements are uniformly sized. (The model is made of a single area.) Using the expand option (*Lab* = EXPND) on the **MOPT** command, mesh (b) was created with far fewer elements because the mesh is allowed to expand from the small element sizes on the boundaries of the area to much larger elements in the interior. Some of the elements of this mesh, however, have poor aspect ratios (for example, those around the small holes). Another weakness of mesh (b) is that the elements change in size (transition) from the small elements to the larger elements, especially near the small holes.

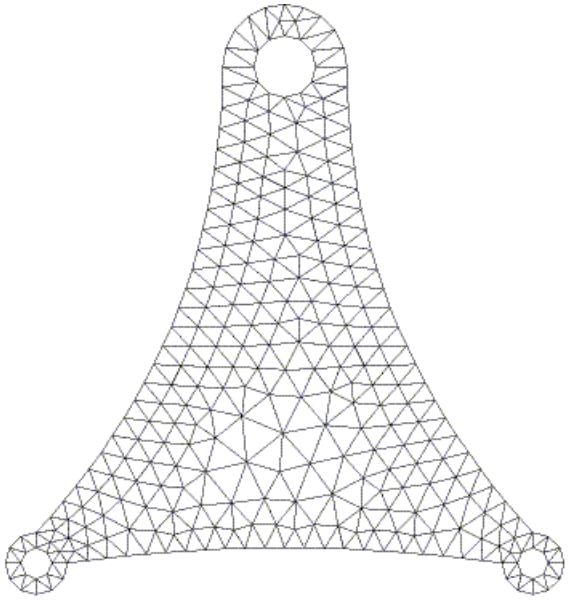
Note

Although this discussion is limited to area mesh expansion [*Lab* = EXPND], you can also use the **MOPT** command to control tetrahedra mesh expansion [*Lab* = TETEXPND]. See the description of the **MOPT** command in the *Command Reference* for more information.

7.3.8.2. Controlling Mesh Transitioning

To improve mesh (b) above, a more gradual transition from small elements on the boundaries to large elements on the interior is needed. The *Lab* = TRANS option on the **MOPT** command can be used to control the rate of transitioning from fine to coarse elements. Figure 7.12: Area Mesh With Expansion and Transition Control (**MOPT** Command) (p. 120) shows the same area meshed with **MOPT**,TRANS,1.3 used in addition to the **MOPT** setting which produced the previous mesh. This mesh has far fewer elements than mesh (a) of Figure 7.11: Area Mesh Without and With Mesh Expansion (p. 119), yet the transition from small elements to larger elements is fairly smooth. Also, the element aspect ratios are significantly better than the elements in mesh (b) of Figure 7.11: Area Mesh Without and With Mesh Expansion (p. 119).

Figure 7.12: Area Mesh With Expansion and Transition Control (MOPT Command)



7.3.8.3. Controlling Which Mesher the Program Uses

You can also use the **MOPT** command to control which surface meshers (triangle and quadrilateral) and which tetrahedra mesher the program uses to perform a meshing operation [**AMESH**, **VMESH**]. The **MOPT** command also allows you to set the order of meshing so smaller areas are meshed first (**MOPT**,AORDER,ON). For large or complex volumes (including those imported from CAD packages) where **SMRTSIZE** proves inefficient, **MOPT**,AORDER,ON can be used instead.

Note

Quadrilateral surface meshes will differ based on which triangle surface mesher is selected. This is true because all free quadrilateral meshing algorithms use a triangle mesh as a starting point.

Command(s): MOPT
GUI: Main Menu> Preprocessor> Meshing> Mesher Opts

Note

The menu path provided above takes you to the Mesher Options dialog box. References to the Mesher Options dialog box appear throughout this section.

7.3.8.3.1. Surface Meshing Options

If you need to mesh multiple volumes, you should consider using the AORDER meshing option in the Mesher Opts dialog box (or issuing the **MOPT**,AORDER,ON command) so your mesh is created in the smallest area first. This helps ensure that your mesh is adequately dense in smaller areas and that the mesh is of a higher quality.

The following options for triangle surface meshing are available:

- Let the program choose which triangle surface mesher to use. This is the recommended setting and the default. In most cases, the program will choose the main triangle mesher, which is the Riemann space mesher. If the chosen mesher fails for any reason, the program invokes the alternate mesher and retries the meshing operation.

  To choose this option, issue the command **MOPT**,AMESH,DEFAULT. In the GUI, access the Mesher Options dialog box and choose Program Chooses in the Triangle Mesher option menu.

- Main triangle surface mesher (Riemann space mesher). The program uses the main mesher, and it does not invoke an alternate mesher if the main mesher fails. The Riemann space mesher is well suited for most surfaces.

  To choose this option, issue the command **MOPT**,AMESH,MAIN. In the GUI, access the Mesher Options dialog box and choose Main from the Triangle Mesher option menu.

- First alternate triangle surface mesher (3-D tri mesher). The program uses the first alternate triangle mesher, and it does not invoke another mesher if this mesher fails. This option is not recommended due to speed considerations. However, for surfaces with degeneracies in parametric space, this mesher often provides the best results. We also recommend that you use this mesher when meshing highly anisotropic regions.

  To choose this option, issue the command **MOPT**,AMESH,ALTERNATE. In the GUI, access the Mesher Options dialog box and choose Alternate from the Triangle Mesher option menu.

- Second alternate triangle surface mesher (2-D parametric space mesher). The program uses the second alternate triangle mesher, and it does not invoke another mesher if this mesher fails. This option is not recommended for use on surfaces with degeneracies (spheres, cones, and so on) or on poorly parameterized surfaces because poor meshes may be created.

  To choose this option, issue the command **MOPT**,AMESH,ALT2. In the GUI, access the Mesher Options dialog box and choose Alternate 2 from the Triangle Mesher option menu.

The options listed below are available for quadrilateral surface meshing. Keep in mind that quadrilateral surface meshes will differ based on which triangle surface mesher is selected. This is true because all free quadrilateral meshing algorithms use a triangle mesh as a starting point.

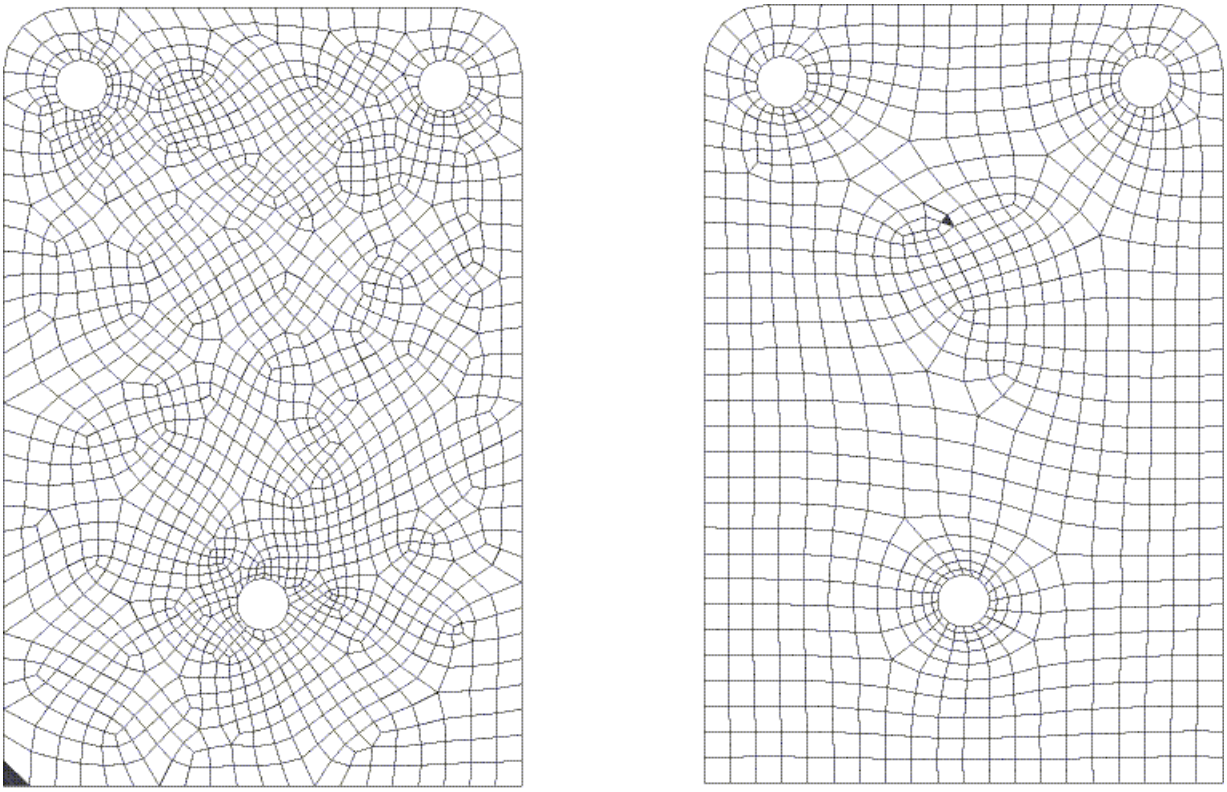
- Let the program choose which quadrilateral surface mesher to use. This is the recommended setting and the default. In most cases, the program will choose the main quadrilateral mesher, which is the Q-Morph (quad-morphing) mesher. For very coarse meshes, the program may choose the alternate quadrilateral mesher instead. If either mesher fails for any reason, the program invokes the other mesher and retries the meshing operation.

  To choose this option, issue the command **MOPT**,QMESH,DEFAULT. In the GUI, access the Mesher Options dialog box and choose Program Chooses from the Quad Mesher option menu.

- Main quadrilateral surface mesher (Q-Morph mesher). The program uses the main mesher, and it does not invoke the alternate mesher if the main mesher fails.

  In most cases, the Q-Morph mesher results in higher quality elements (see Figure 7.13: Quadrilateral and Q-Morph Mesher (p. 122)). The Q-Morph mesher is particularly beneficial to users whose applications require boundary sensitive, highly regular nodes and elements.

Figure 7.13: Quadrilateral and Q-Morph Mesher



(a) (b)

Mesh (a) shows a surface that was meshed with the alternate quadrilateral mesher; mesh (b) shows the same surface, this time meshed with the Q-Morph mesher.

Notice that although both meshes shown contain one triangle element (the triangle elements are shaded in the figure), the triangle element in Figure (a) occurs on the boundary of the area. The triangle element in Figure (b) is an internal element, which is a more desirable location for it in the mesh.

For the Q-Morph mesher to be able to generate an all-quadrilateral mesh of an area, the total number of line divisions on the boundaries of the area must be even. (In most cases, turning on SmartSizing [**SMRTSIZE**,*SIZLVL*] will result in an even total number of line divisions on the boundaries.)

A triangle element (or elements) will result in the area mesh if any of these statements is true:

1. The total number of line divisions on the boundaries of the area is odd.

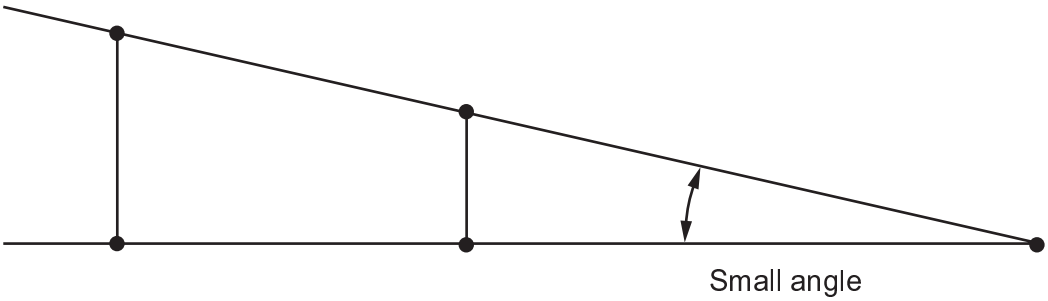
2. Quadrilateral element splitting is turned on for error elements [**MOPT**,SPLIT,ON or **MOPT**,SPLIT,ERR] and a quadrilateral element in violation of shape error limits would be created if the program *did not* split the element into triangles. (Splitting is on for error elements by default.)

3. Quadrilateral splitting is turned on for both error *and* warning elements [**MOPT**,SPLIT,WARN], and a quadrilateral element in violation of shape error and warning limits would be created if the program *did not* split the element into triangles.

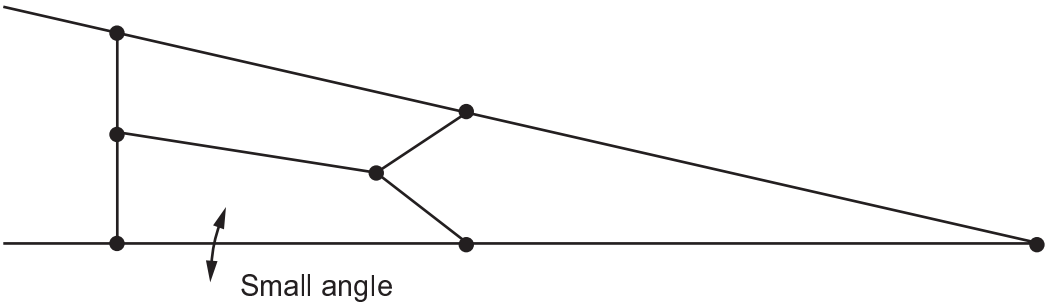
4. Quadrilateral element splitting is turned on for either a) error elements or b) error *and* warning elements, and the area contains a small angle (< 30°) between adjacent boundary intervals. See Figure 7.14: Results of Quadrilateral Splitting (p. 123).

To choose this option (Q-Morph mesher), issue the command **MOPT**,QMESH,MAIN. In the GUI, access the Mesher Options dialog box and choose Main from the Quad Mesher option menu.

Figure 7.14: Results of Quadrilateral Splitting



(a) Quad splitting on; Results in triangle element



(b) Quad splitting off; Results in quadrilateral element

- Alternate quadrilateral surface mesher. The program uses the alternate mesher, and it does not invoke the main mesher if the alternate mesher fails.

  For this mesher to be able to generate an all-quadrilateral mesh of an area, the total number of line divisions on the boundaries of the area must be even, and quadrilateral splitting must be turned off [**MOPT**,SPLIT,OFF].

  To choose this option, issue the command **MOPT**,QMESH,ALTERNATE. In the GUI, access the Mesher Options dialog box and choose Alternate in the Quad Mesher option menu. *To use this mesher, you must also select either the first alternate or the second alternate triangle surface mesher.*

7.3.8.3.2. Tetrahedral Element Meshing Options

The following options for tetrahedral element meshing are available:

- Let the program choose which tetrahedra mesher to use. This is the default. With this setting, the program uses the main tetrahedra mesher when it can; otherwise, it uses the alternate tetrahedra mesher.

  To choose this option, issue the command **MOPT**,VMESH,DEFAULT. In the GUI, access the Mesher Options dialog box and choose Program Chooses in the Tet Mesher option menu.

- Main tetrahedra mesher (Delaunay technique mesher). For most models, this mesher is significantly faster than the alternate mesher.

To choose the main tetrahedra mesher, issue the command **MOPT**,VMESH,MAIN. In the GUI, access the Mesher Options dialog box and choose Main in the Tet Mesher option menu.

- Alternate tetrahedra mesher (advancing front mesher from Revision 5.2). This mesher does not support the generation of a tetrahedral volume mesh from facets [**FVMESH**]. If this mesher is selected and you issue the **FVMESH** command, the program uses the main tetrahedra mesher to create the mesh from facets and issues a warning message to notify you.

  To choose the alternate tetrahedra mesher, issue the command **MOPT**,VMESH,ALTERNATE. In the GUI, access the Mesher Options dialog box and choose Alternate in the Tet Mesher option menu.

7.3.8.4. Controlling Tetrahedral Element Improvement

You can use the **MOPT** command to control the level of tetrahedra improvement that the program performs when the next free volume meshing operation is initiated [**VMESH**, **FVMESH**].
Command(s): MOPT,TIMP,*Value*
GUI: Main Menu> Preprocessor> Meshing> Mesher Opts

Levels for tetrahedra improvement range from 1 to 6, with level 1 offering only minimal improvement, level 5 offering the maximum amount of improvement for linear tetrahedral meshes, and level 6 offering the maximum amount of improvement for quadratic tetrahedral meshes. The minimal level of improvement [**MOPT**,TIMP,1] is supported by the main tetrahedra mesher only [**MOPT**,VMESH,MAIN]. If the alternate tetrahedra mesher [**MOPT**,VMESH,ALTERNATE] is invoked when improvement is set to level 1, the program automatically performs tetrahedra improvement at level 3 instead. You can also turn tetrahedra improvement off, but doing so is not recommended because it often leads to poorly shaped elements and meshing failures. For more details about each improvement level, see the description of the **MOPT** command in the *Command Reference*.

Note

In most cases, the default levels that the program uses for tetrahedra improvement will give you satisfactory results. However, there may be times when you want to request additional improvement of a given tetrahedral element mesh by using the **VIMP** command. See Improving the Mesh (Tetrahedral Element Meshes Only) (p. 172) for details about how to request additional improvement and when doing so would benefit you.

7.3.9. Creating Transitional Pyramid Elements

While some regions of a volume may be easy to divide into map-meshable parts, other regions may be geometrically complex. You may use hexahedral elements to fill the map-meshable regions of a volume, and tetrahedral elements to fill the remainder. In some cases, high-gradient regions may *require* hexahedral elements to capture detail, while for other, less critical regions, tetrahedral elements may be sufficient.

Unfortunately, using a mix of hexahedral and tetrahedral element shapes leads to nonconformities in a mesh, and the finite element method requires that elements within a mesh conform. You can avoid the problems that may arise from this situation by following the guidelines outlined below. By instructing the program to automatically create pyramid elements at their interface, you can easily maintain mathematical continuity between hexahedral and tetrahedral element types.

7.3.9.1. Situations in Which the Program Can Create Transitional Pyramids

Transitional pyramid elements can be created in any of these situations:

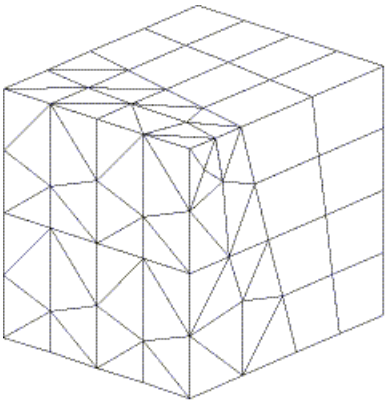
- You are ready to mesh a volume with tetrahedral elements. The volume immediately adjacent to that volume has already been meshed with hexahedral elements. The two volumes have been glued together [**VGLUE**]. (Two volumes for which you want to create transitional pyramids must share a common area; the quadrilateral faces from the hexahedral elements must be located on that common area.)

- At least one of the areas on a volume has been meshed with quadrilateral elements. In this situation you simply mesh the volume with tetrahedral elements, and the program forms the pyramids directly from the quadrilateral elements. If you want, you can then mesh any adjacent volumes with hexahedral elements.

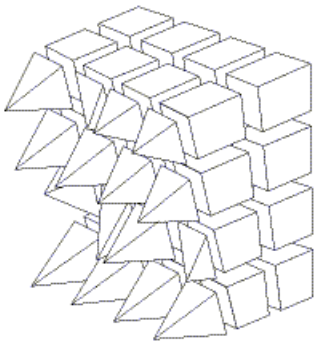
- Where detached quadrilateral-shaped elements are used as input to the **FVMESH** command.

Figure 7.15: Creation of Transitional Pyramid Elements at an Interface (p. 125) illustrates the creation of transitional pyramids at the interface of tetrahedral and hexahedral elements. In this example, a simple block is divided by an arbitrary cutting plane. The cutting plane serves as the interface between two volumes - one in which tetrahedral elements were generated, and the other in which hexahedral elements were generated Figure (a). Figure (b) provides an exploded view of the transitional pyramids; the tetrahedral elements have been removed.

Figure 7.15: Creation of Transitional Pyramid Elements at an Interface



(a) Finite element model



(b) Exploded view of pyramids

7.3.9.2. Prerequisites for Automatic Creation of Transitional Pyramid Elements

In order for transitional pyramid elements to be created when you mesh a volume with tetrahedral elements, you must meet these prerequisites:

- When setting element attributes, be sure that the element type you assign to the volume is one that can be degenerated into a pyramid shape. (See Assigning Element Attributes Before Meshing (p. 103) for information about the methods you can use to set attributes.)

- When setting meshing controls, activate transitioning and indicate that you want to degenerate 3-D elements.

To activate transitioning (the default), use one of the following methods:
Command(s): MOPT,PYRA,ON
GUI: Main Menu> Preprocessor> Meshing> Mesher Opts

To degenerate 3-D elements, use one of the following methods:
Command(s): MSHAPE,1,3D
GUI: Main Menu> Preprocessor> Meshing> Mesher Opts

If these prerequisites are met and you now mesh the volume with tetrahedral elements [**VMESH**], the program automatically:

- Determines where transitional pyramid elements are appropriate
- Combines and rearranges tetrahedral elements to create pyramid elements
- Inserts the pyramid elements into the mesh

The program creates transitional pyramid elements by default; if you prefer not to have transitional pyramid elements inserted into your mesh, issue the **MOPT**,PYRA,OFF command.

Note

For quadratic pyramid elements that are immediately adjacent to linear hexahedral elements, the program automatically drops midside nodes at the interface. This, in fact, occurs when meshing any quadratic element if linear elements are adjacent in a neighboring volume.

Also, after pyramid transitioning, if there are selected elements, the number of selected elements may be greater than the number of elements originally selected (before the transition).

7.3.10. Converting Degenerate Tetrahedral Elements to Their Nondegenerate Forms

After creating transitional pyramid elements in a model, you can convert the 20-node degenerate tetrahedral elements in the model to their 10-node non-degenerate counterparts.

7.3.10.1. Benefits of Converting Degenerate Tetrahedral Elements

The process described in Creating Transitional Pyramid Elements (p. 124) permits the formation of pyramids only when you use an element type that supports degenerate tetrahedral and pyramidal shapes. Depending on your application, you may find that this prerequisite is too limiting.

For example, if you are working on a structural application, you are limited to using SOLID186 elements wherever transitional pyramid elements are required. Solving an analysis that involves 20-node, degenerate SOLID186 elements (and storing those elements) uses more solution time and memory than would the same analysis using SOLID187 elements. (SOLID187 elements are the 10-node, non-degenerate counterpart to SOLID186 elements.)

In this example, converting SOLID186 elements to SOLID187 elements provides these benefits:

- Less random access memory (RAM) is required per element.
- When you are *not* using the Preconditioned Conjugate Gradient (PCG) equation solver, the files that the program writes during solution are considerably smaller.

- Even when you *are* using the PCG equation solver, you gain a modest speed advantage.
- If you are using the PCG solver with **MSAVE**,ON, you may be able to obtain a significant performance gain with using SOLID187 elements compared to using degenerate SOLID186 elements. The **MSAVE**,ON command can only be used when doing a static analyses or modal analyses using the PCG Lanczos method. **MSAVE**,ON results in a memory savings of up to 70% for the part of the model that meets the criteria, although the solution time may vary depending on the processor speed and the manufacturer of your computer, as well as the chosen element options (for example, reduced 2 x 2 x 2 integration for SOLID186).

7.3.10.2. Performing a Conversion

To convert 20-node degenerate tetrahedral elements to their 10-node non-degenerate counterparts:
Command(s): TCHG,*ELEM1*,*ELEM2*,*ETYPE2*
GUI: Main Menu> Preprocessor> Meshing> Modify Mesh> Change Tets

Regardless of whether you use the command or the GUI method, you are limited to converting the combinations of elements that are presented in Table 7.3: Allowable Combinations of *ELEM1* and *ELEM2* (p. 127).

Table 7.3: Allowable Combinations of *ELEM1* and *ELEM2*

Physical Properties	Value of *ELEM1*	Value of *ELEM2*
Structural Solid	SOLID186 or 186	SOLID187 or 187
Thermal Solid	SOLID90 or 90	SOLID87 or 87
Electrostatic Solid	SOLID122 or 122	SOLID123 or 123

If you are using the **TCHG** command to perform the conversion, specify values for the following arguments:

- Use the *ELEM1* argument to identify the type of element that you want to convert. For example, to convert SOLID186 elements, you must specify either SOLID186 or 186 for the value of *ELEM1*
- Use the *ELEM2* argument to identify the type of element that is the counterpart to the *ELEM1* element. For example, to convert SOLID186 elements, you must specify either SOLID187 or 187 for the value of *ELEM2*.
- Optionally, you can use the *ETYPE2* argument to specify the element TYPE number for *ELEM2*. To continue with our example, to assign element TYPE number 2 to the newly-converted SOLID187 elements, specify 2 for the value of *ETYPE2*. (An element type's TYPE number is the number assigned to that element type in the element attribute tables; it is based on the element type's location in the element attribute tables.) If you do not specify a value for *ETYPE2*, the program uses the next available location in the element attribute tables to determine the element TYPE number for *ELEM2* or, if *ELEM2* already appears in the element attribute tables, the program uses *ELEM2*'s existing element TYPE number for *ETYPE2*. To continue with our example,

Also see the description of the **TCHG** command in the *Command Reference*.

If you are using the GUI to perform the conversion, follow these steps:

1. Choose menu path **Main Menu> Preprocessor> Meshing> Modify Mesh> Change Tets**. The Change Selected Degenerate Hexes to Non-degenerate Tets dialog box appears.

2. Using the Change From option menu, select a combination of elements.

3. In the TYPE number for ELEM2 field, select the appropriate element TYPE number for *ELEM2*. (A single-selection list containing *all of the currently defined element types, along with their corresponding element TYPE numbers,* appears on the dialog box to help you make your selection.) To make your selection, you can do any one of the following:

   - Choose NEXT AVAIL TYPE# from the selection list and click on OK, and the program uses the next available location in the element attribute tables to determine the element TYPE number for *ELEM2* or, if *ELEM2* already appears in the element attribute tables, the program uses *ELEM2*'s existing element TYPE number for *ETYPE2*.

   - Choose USER SPECIFIED from the selection list and click on OK. A second dialog box appears, where you must enter an element TYPE number and click on OK. The program assigns the element TYPE number that you enter to *ELEM2*.

   - Choose a valid element TYPE number (if one is available) from the selection list. Remember that even though all of the currently defined element types and their assigned element TYPE numbers appear in the list, not all of them are valid choices (for example, if you are converting SOLID186 elements to SOLID187 elements, you must choose the TYPE number for the defined SOLID187 element from the selection list). If no SOLID187 elements are currently defined, then you have to use one of the other selection methods described above. Assuming that a valid element TYPE number is available and you select it, the program will assign that TYPE number to the newly-converted elements.

7.3.10.3. Other Characteristics of Degenerate Tetrahedral Element Conversions

Other characteristics of degenerate tetrahedral element conversions include the following:

- As a result of the conversion operation, only selected elements of type *ELEM1* are converted to type *ELEM2*. The program ignores any elements that *are* type *ELEM1* but *are not* degenerate tetrahedra; for instance, the program will ignore SOLID186 elements that have a hexahedral, pyramidal, or prism shape. For example, assume that you have a simple model that contains only SOLID186 elements. Some of these elements are hexahedral, some are tetrahedral, and some are pyramidal. If you issue the command **TCHG**,186,187,2, the program converts only the tetrahedral SOLID186 elements to SOLID187 elements; it leaves the hexahedral and pyramidal SOLID186 elements untouched. Since you specified 2 as the value of *ETYPE2*, the program assigns element TYPE number 2 to the SOLID187 elements.

- Performing a conversion is likely to create circumstances in which more than one element type is defined for a single volume. Currently, the program has no way of storing more than one element type per volume. This limitation may result in incorrect information when you perform a volume listing operation [**VLIST** command]. The output listing will fail to indicate that the element type of the converted elements has changed. Instead, it will indicate the element TYPE number that was *originally* assigned to those elements. (On the other hand, the output of an element listing operation [**ELIST** command] *will* indicate the new element TYPE number.) If you plan to perform a conversion, we recommend that the conversion be your last step in the modeling and meshing process; that is, complete any desired mesh refinement, moving or copying of nodes and elements, and any other desired modeling and meshing revision processes prior to beginning the conversion.

7.3.11. Doing Layer Meshing

The program's layer meshing feature (currently, for 2-D areas only) enables you to generate line-graded free meshes having either of the following:

- Uniform (or moderately varying) element size along the line.
- Steep transitions in element size and number in the direction normal to the line.

Such meshes are suitable for simulating CFD boundary layer effects, electromagnetic skin layer effects, etc.

7.3.12. Setting Layer Meshing Controls via the GUI

If you are using the GUI, you set layer mesh controls on a picked set of lines by choosing **Main Menu> Preprocessor> Meshing> Mesh Tool**, which displays the MeshTool panel. Pressing the Set button next to "Layer" opens a picking dialog for selecting lines, followed by the "Area Layer Mesh Controls on Picked Lines" dialog box. On it, you may specify any of the following.

- The desired element size on the line, either by setting the element size directly (SIZE), or by setting the number of line divisions (NDIV).
- The line spacing ratio (SPACE, normally 1.0 for layer meshing).
- The thickness of the inner mesh layer (LAYER1). Elements in this layer will be uniformly-sized, with edge lengths equal to the specified element size on the line. LAYER1's thickness may be specified with either a factor on the element size for the line (size factor = 2 produces two rows of uniformly-sized elements along the line; size factor = 3, three rows, etc.), or with an absolute length.
- The thickness of the outer mesh layer (LAYER2). The size of elements in this layer will gradually increase from those in LAYER1 to the global element size. LAYER2's thickness may be specified with either a mesh transition factor (transition factor = 2 produces elements which approximately double in size as the mesh front progresses normal to the line; transition factor = 3, triple in size, etc.), or with an absolute length.

Note

The thickness of LAYER1 should be greater than or equal to the specified element size for the line. If you use a size factor to specify LAYER1, it must be greater than or equal to 1.0.

Note

LAYER2's "thickness" is really the distance over which mesh transition must occur between elements of LAYER1 size and the global size. Appropriate values for LAYER2 thus depend on the magnitude of the global-to-LAYER1 size ratio. If you use a mesh transition factor to specify

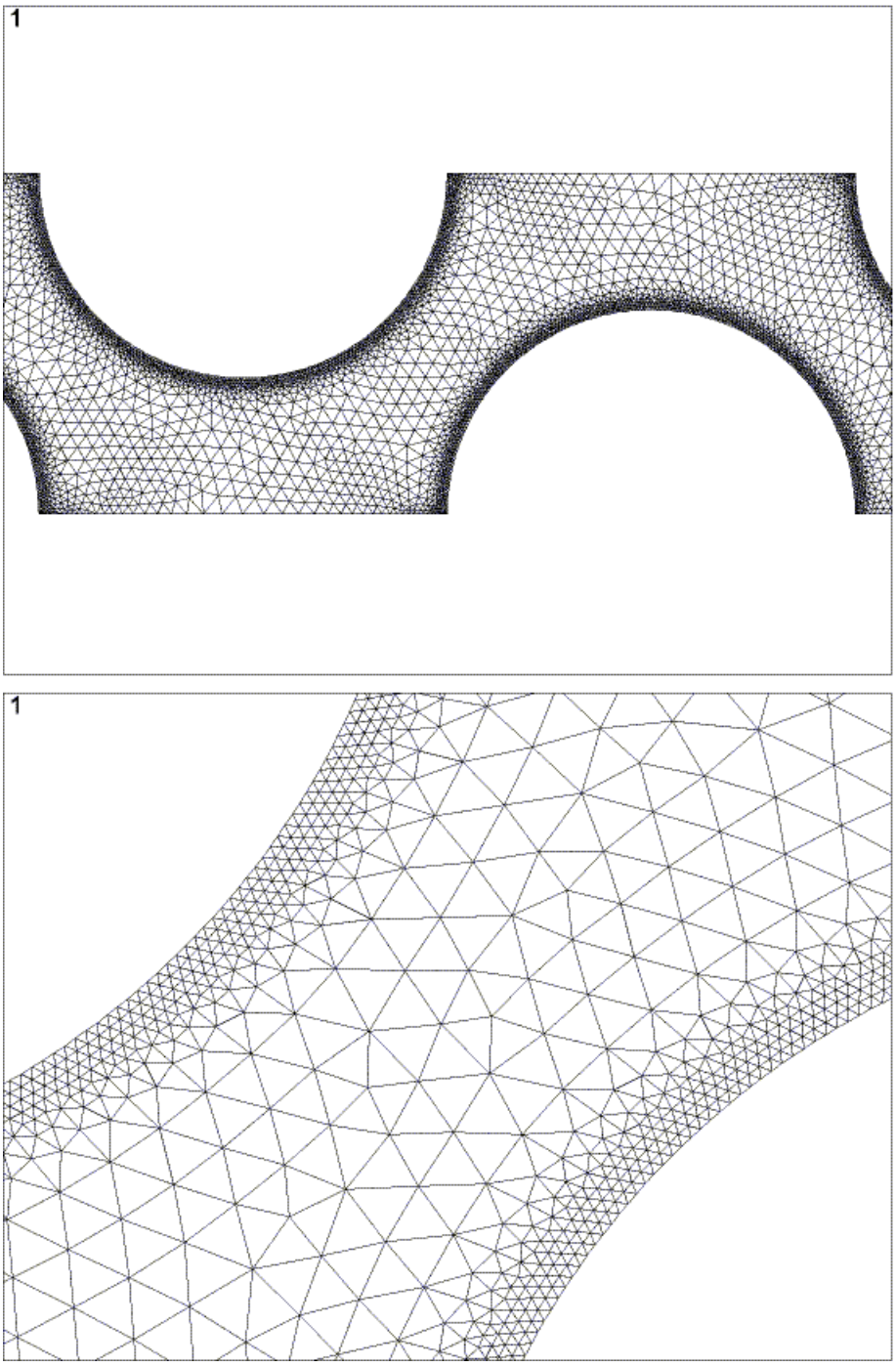
LAYER2, it must be greater than 1.0 (implying the next row's size must be larger than the previous) and, for best results, should be less than 4.0.

Note

For a picked set of lines, layer mesh controls may be set or cleared without altering the existing line divisions or spacing ratio settings for those lines. In fact, within this dialog box, blank or zero settings for SIZE/NDIV, SPACE, LAYER1, or LAYER2 will *remain the same* (that is, they will not be set to zero or default values).

The figures below illustrate a line-graded layer mesh showing uniform element size along the line and steep transitions in element size and number normal to the line

Figure 7.16: Line-Graded Layer Mesh



To delete layer mesh control specifications from a picked set of lines, choose the Clear button beside "Layer" on the MeshTool. Existing line divisions and spacing ratios for the set of lines will remain the same.

7.3.13. Setting Layer Meshing Controls via Commands

The **LESIZE** command specifies layer meshing controls and other element size characteristics. For information about this command, see the *Command Reference*.

7.3.14. Listing Layer Mesh Specifications on Lines

To view or print layer meshing size specifications on lines, use one of the following:
Command(s): LLIST
GUI: Utility Menu> List> Lines

7.4. Controls Used for Free and Mapped Meshing

In the previous sections, we have described various meshing controls that are available to you. Now we will focus on which controls are appropriate for free meshing, and which are appropriate for mapped meshing.

7.4.1. Free Meshing

In *free meshing* operations, no special requirements restrict the solid model. Any model geometry, even if it is irregular, can be meshed.

The element shapes used will depend on whether you are meshing areas or volumes. For area meshing, a free mesh can consist of only quadrilateral elements, only triangular elements, or a mixture of the two. For volume meshing, a free mesh is usually restricted to tetrahedral elements. Pyramid-shaped elements may also be introduced into the tetrahedral mesh for transitioning purposes. (See Creating Transitional Pyramid Elements (p. 124) for information about pyramid-shaped elements.)

If your chosen element type is strictly triangular or tetrahedral, the program will use only that shape during meshing. However, if the chosen element type allows more than one shape (for example, PLANE183 or SOLID186), you can specify which shape (or shapes) to use by one of the following methods:
Command(s): MSHAPE
GUI: Main Menu> Preprocessor> Meshing> Mesher Opts

You must also specify that free meshing should be used to mesh the model:
Command(s): MSHKEY,0
GUI: Main Menu> Preprocessor> Meshing> Mesher Opts

For area elements that support more than one shape, a mixed shape mesh (which is usually quad-dominant) will be produced by default. An all triangle mesh can be requested [**MSHAPE**,1,2D and **MSHKEY**,0], but is not recommended if lower-order elements are being used.

Note

There may be times when it is important to you to have an all-quadrilateral mesh. Free meshing of an area results in an all-quadrilateral mesh when the total number of line divisions on the boundaries of the area is even, and the quality of the quadrilateral elements produces no errors. You can increase the chances that the area's boundaries will have an even total number of line divisions by turning SmartSizing on and letting it determine the appropriate element divisions (rather than setting the number of element divisions on any of the boundaries manually [**LESIZE**]). You should also make sure that quadrilateral splitting is off [**MOPT**,SPLIT,OFF] to keep the program from splitting poorly shaped quadrilateral elements into triangles. (Quadrilateral splitting is turned on for error elements by default. See the description of the **MOPT** command for details.)

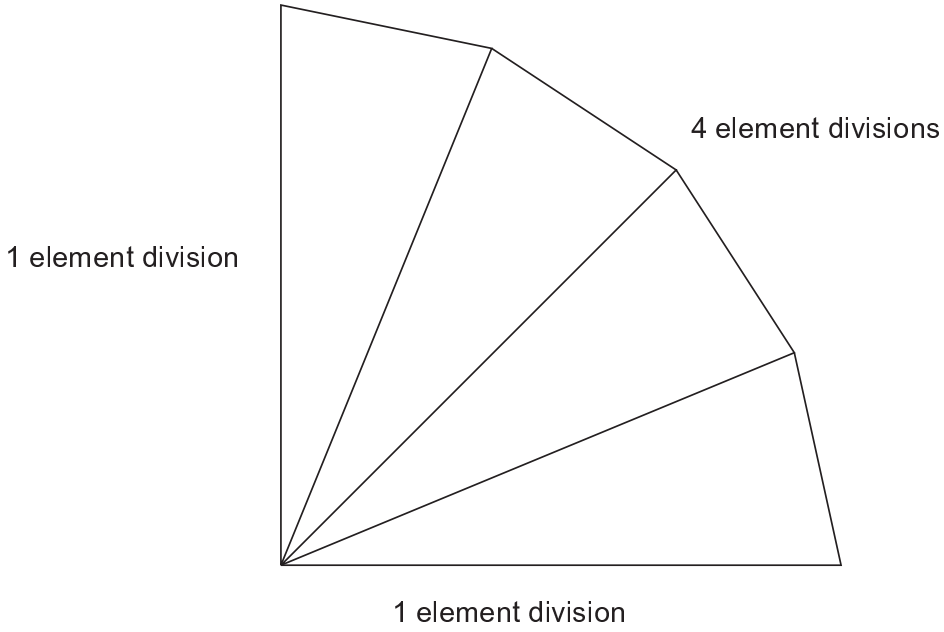
To achieve a free volume mesh, you should choose an element type that allows only a tetrahedral shape, or use an element that supports multiple shapes and set the shape option to tetrahedral only [**MSHAPE**,1,3D and **MSHKEY**,0].

For free meshing operations, element sizes are produced based on the current settings of the **DESIZE** command, along with **ESIZE**, **KESIZE**, and **LESIZE**. If SmartSizing is turned on, the element sizes will be determined by the **SMRTSIZE** command along with **ESIZE**, **KESIZE**, and **LESIZE**. (SmartSizing is recommended for free meshing.) You can find all of these meshing controls under both **Main Menu> Preprocessor> Meshing> MeshTool** and **Main Menu> Preprocessor> Meshing> Size Cntrls**.

7.4.1.1. Fan Type Meshing and the TARGE170 Element

A special type of free meshing, called fan type meshing, is available for certain contact analysis cases that involve the meshing of 3-sided areas with the TARGE170 element. When two of the three sides have only one element division, and the third side has any number of divisions, the result will be a fan type mesh. (The **LESIZE** command is used to set element divisions.) Fan type meshing ensures that the program uses the minimum number of triangles to fill the area, which is important for contact problems. Consider the example shown in Figure 7.17: Example of Fan Type Meshing (p. 133), in which two of the sides have only one element division, while the third side has four.

Figure 7.17: Example of Fan Type Meshing



7.4.1.1.1. Conditions for Fan Type Meshing

Remember that to use fan type meshing, the following conditions must be satisfied:

- You must be meshing a 3-sided area. Two of the sides must have only one element division; the third side can have any number of divisions.
- You must be meshing with the TARGE170 element.
- You must specify that free meshing be used [**MSHKEY**,0 or **MSHKEY**,2].

For more information, see the *Contact Technology Guide* and the description of the TARGE170 element in the *Element Reference*.

7.4.2. Mapped Meshing

You can specify that the program use all quadrilateral area elements, all triangle area elements, or all hexahedral (brick) volume elements to generate a *mapped mesh*. Mapped meshing requires that an area or volume be "regular", that is, it must meet certain criteria.

For mapped meshing, element sizes are produced based on the current settings of **DESIZE**, along with **ESIZE**, **KESIZE**, **LESIZE** and **AESIZE** settings (**Main Menu> Preprocessor> Meshing> Size Cntrls>** ***option***). SmartSizing [**SMRTSIZE**] cannot be used for mapped meshing.

Note

Mapped meshing is not supported when hard points are used.

7.4.2.1. Area Mapped Meshing

An area mapped mesh consists of either all quadrilateral elements or all triangular elements.

Note

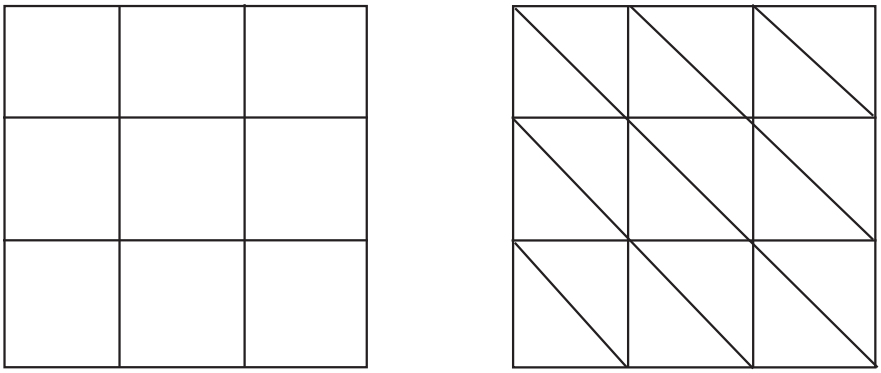
Mapped triangle meshing refers to the process in which the program takes a map-meshable area and meshes it with triangular elements, based on a pattern you specify. This type of meshing is particularly useful for analyses that involve the meshing of rigid contact elements. (See the *Contact Technology Guide* for details about contact analyses.)

For an area to accept a mapped mesh, the following conditions must be satisfied:

1. The area must be bounded by either three or four lines (with or without concatenation).
2. The area must have equal numbers of element divisions specified on opposite sides, or have divisions matching one of the transition mesh patterns (see Figure 7.24: Applicable Transition Patterns-Transition Mapped Quadrilateral Meshes (p. 138)).
3. If the area is bounded by three lines, the number of element divisions must be even and equal on all sides.
4. The meshing key must be set to mapped [**MSHKEY**,1]. This setting results in a mapped mesh of either all quadrilateral elements or all triangle elements, depending on the current element type and/or the setting of the element shape key [**MSHAPE**].
5. If your goal is a mapped triangle mesh, you can also specify the pattern the program uses to create the mesh of triangular elements [**MSHPATTERN**]. If you do not specify a pattern, the program chooses one for you. See the **MSHPATTERN** command description in the *Command Reference* for an illustration of the available patterns.

Figure 7.18: Area Mapped Meshes (p. 135) shows a basic area mapped mesh of all quadrilateral elements, and a basic area mapped mesh of all triangular elements.

Figure 7.18: Area Mapped Meshes



If an area is bounded by more than four lines, it cannot be map meshed. However, some of the lines can be combined or "concatenated" to reduce the total number of lines to four. Line concatenation is discussed later in this section.

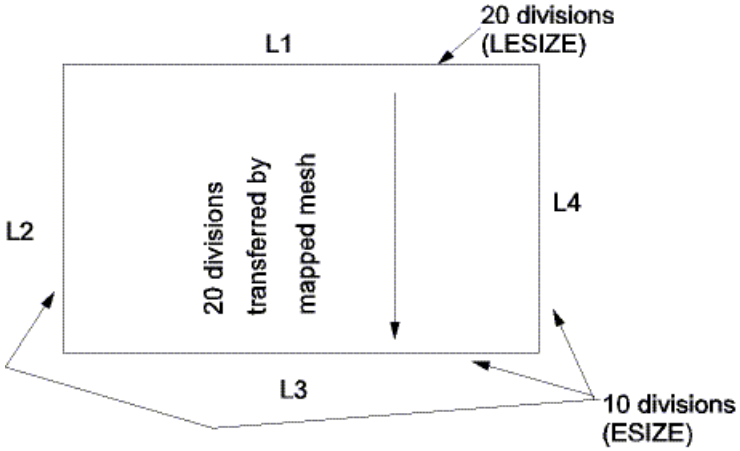
A suggested alternative to using line concatenation is to use the **AMAP** command to map mesh an area by picking three or four corners of the area. This method internally concatenates all lines between the keypoints. (Simplified area mapped meshing is described later in this section.)

7.4.2.1.1. Line Divisions for Mapped Meshing

You must specify equal numbers of line divisions on opposite edges of the area (or define line divisions to match one of the transition patterns) to achieve a mapped mesh. You do not necessarily have to specify line divisions on all lines. As long as mapped meshing has been requested [**MSHKEY**,1], the program will transfer line divisions from one line to the opposite line, and on into adjacent areas being meshed [**AMESH**]. The program will also produce matched line divisions from **AESIZE**, "soft" **LESIZE**, **KESIZE** or **ESIZE** specifications, when possible.

The same hierarchy that applied to **LESIZE**, **ESIZE**, etc. will also apply to transferred line divisions. Thus, in the example shown in Figure 7.19: Transferred Hard **LESIZE** Controls Override **ESIZE** Controls (p. 135), **LESIZE** line divisions transferred from line 1 to line 3 will override explicitly defined **ESIZE** line divisions.

Figure 7.19: Transferred Hard LESIZE Controls Override ESIZE Controls




```
MSHKEY,1! mapped mesh
ESIZE,,10! 10 divisions set by ESIZE
LESIZE,1,,,20! 20 divisions specified for line 1
AMESH,1! 20 line divisions will be transferred onto line 3
```

Please see the **MSHKEY**, **ESIZE**, **AESIZE**, **LESIZE**, and **AMESH** command descriptions for more information.

7.4.2.1.2. Line Concatenation

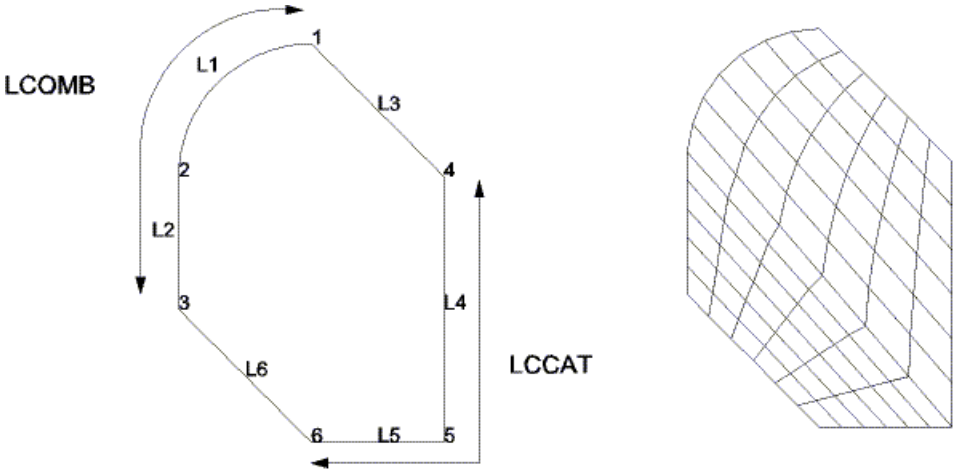
If an area is bounded by more than four lines, you can *combine* [**LCOMB**] or *concatenate* [**LCCAT**] some of the lines to reduce the total number of lines to four. Whenever **LCOMB** is permitted (that is, when lines are tangent and are attached to the same areas), it is generally preferred over **LCCAT**. **LCOMB** can also be used for non-tangent lines, but a node will *not* necessarily be generated at the kink in the line.

To concatenate lines:
Command(s): LCCAT
GUI: Main Menu> Preprocessor> Meshing> Mesh> Areas> Mapped> Concatenate> Lines

To combine lines:
Command(s): LCOMB
GUI: Main Menu> Preprocessor> Modeling> Operate> Booleans> Add> Lines

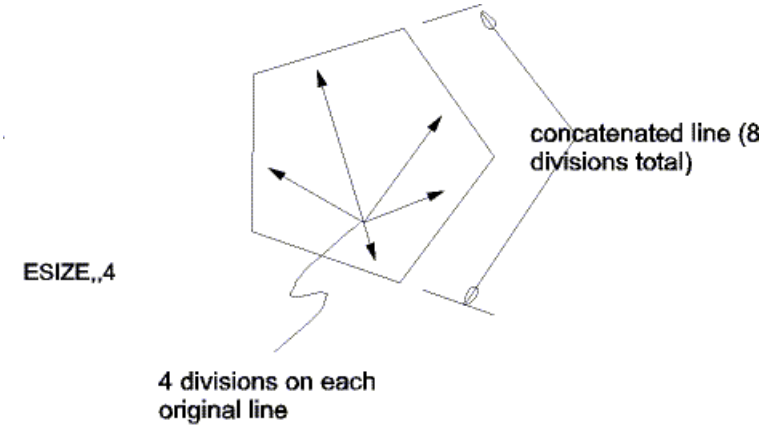
Consider the example of Figure 7.20: Line Combination and Concatenation Can Enable Mapped Meshing (p. 136), in which an area is bounded by six lines. Two of the lines can be combined, and two others concatenated, to produce an area bounded by four lines, suitable for mapped meshing.

Figure 7.20: Line Combination and Concatenation Can Enable Mapped Meshing



A node will be generated wherever there is a keypoint attached to a line, area, or volume. Therefore, a concatenated line will have at least as many divisions as are defined implicitly by the keypoints on that line. The program will not allow you to transfer a smaller number of divisions onto such a line. Also, if a global element size [**ESIZE**] is specified, it applies to your *original* lines, not to your concatenated lines.

Figure 7.21: ESIZE Applies to Original (Not Concatenated) Lines



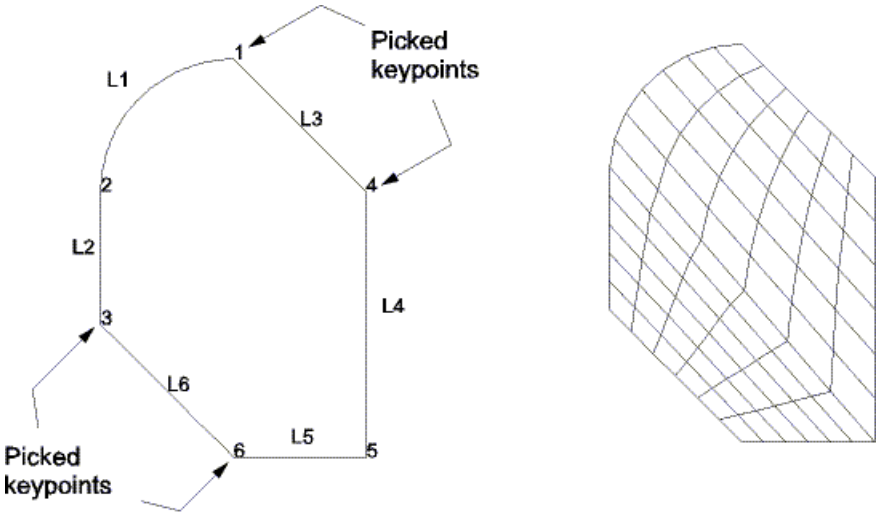
Line divisions cannot be directly assigned to concatenated lines. However, divisions can be assigned to combined lines [**LCOMB**]. Therefore, there is some advantage to using line combination instead of concatenation.

7.4.2.1.3. Simplified Area Mapped Meshing

The **AMAP** command offers the easiest way to obtain a mapped mesh. **AMAP** (**Main Menu> Preprocessor> Meshing> Mesh> Areas> Mapped> By Corners**) uses specified keypoints as corners and *internally* concatenates all lines between the keypoints. The area is automatically meshed with all quadrilateral or all triangular elements (a **MSHKEY** specification is not required). The same rules about meshing controls apply for **AMAP** as for mapped meshing by line concatenation.

Consider the example presented earlier for concatenation, but now meshed with the **AMAP** method. Notice that there are multiple lines between several of the picked keypoints. After picking the area, keypoints 1, 3, 4, and 6 can be picked in any order, and the mapped mesh is automatically created.

Figure 7.22: Simplified Mapped Meshing (AMAP)

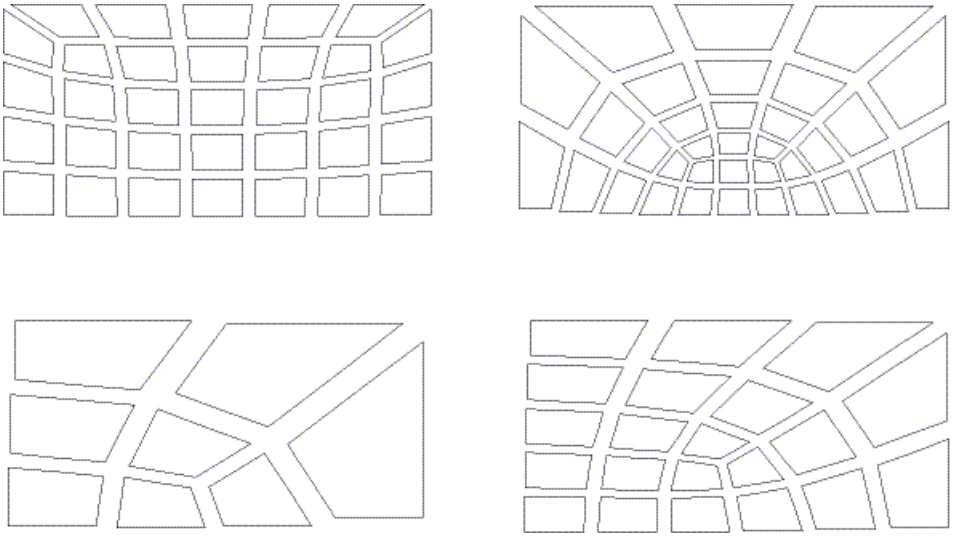


No line concatenation is needed prior to the **AMAP** operation; the concatenation is done internally and then deleted. The area's line list is left unchanged.

7.4.2.1.4. Transition Mapped Quadrilateral Meshing

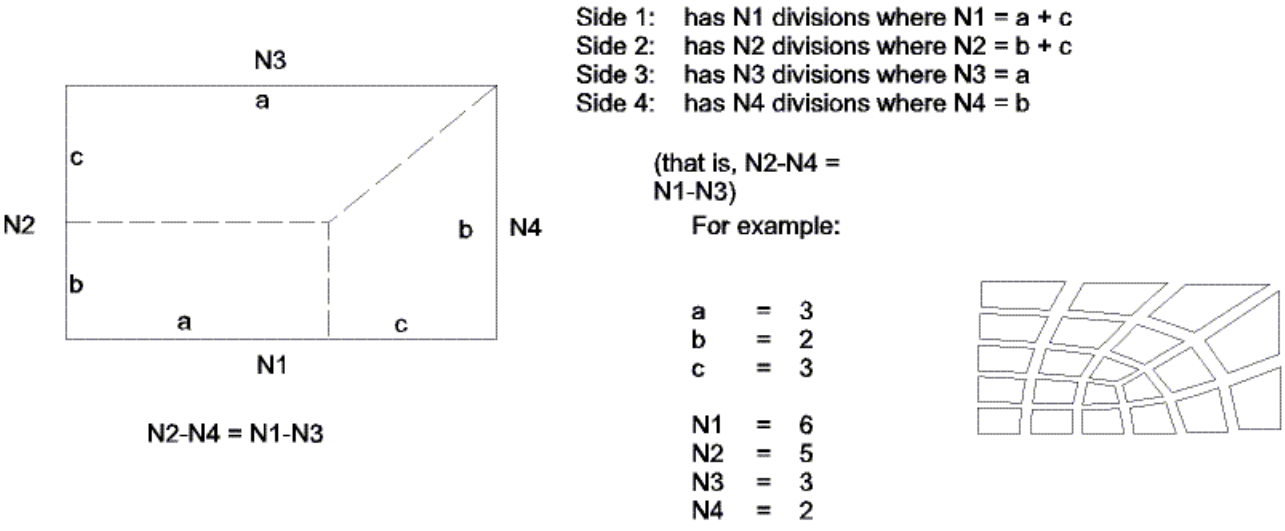
Another way to create a mapped area mesh is to specify line divisions on opposite sides of the area such that the divisions permit a transition mapped quadrilateral mesh. Transition mapped quadrilateral meshing is only applicable to four-sided areas (with or without concatenation). Some examples are shown in Figure 7.23: Examples of Transition Mapped Quadrilateral Meshes (p. 138).

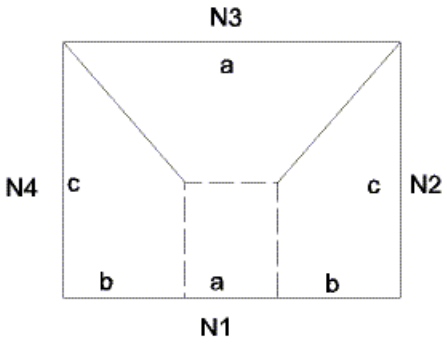
Figure 7.23: Examples of Transition Mapped Quadrilateral Meshes



To achieve a transition mapped quadrilateral mesh, you must use an element type that supports a quadrilateral shape, set the meshing key to mapped [**MSHKEY**,1], and set the shape specification to allow quadrilaterals [**MSHAPE**,0,2D]. (If you want a transition mapped triangle mesh, see the next section.) In addition, specified line divisions must match one of the patterns shown in Figure 7.24: Applicable Transition Patterns-Transition Mapped Quadrilateral Meshes (p. 138).

Figure 7.24: Applicable Transition Patterns-Transition Mapped Quadrilateral Meshes





N2=N4 and the difference
between N1 and N3 is even

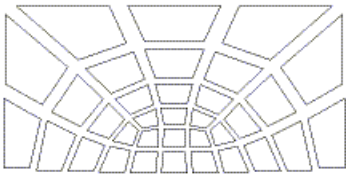
Side 1: has N1 divisions where N1 = a + 2*b
Side 2: has N2 divisions where N2 = c
Side 3: has N3 divisions where N3 = a
Side 4: has N4 divisions where N4 = c

(that is, N2=N4 and the
difference
between N1 and N3 is even)

For example:

a = 3
b = 4
c = 2

N1 = 11
N2 = 2
N3 = 3
N4 = 2



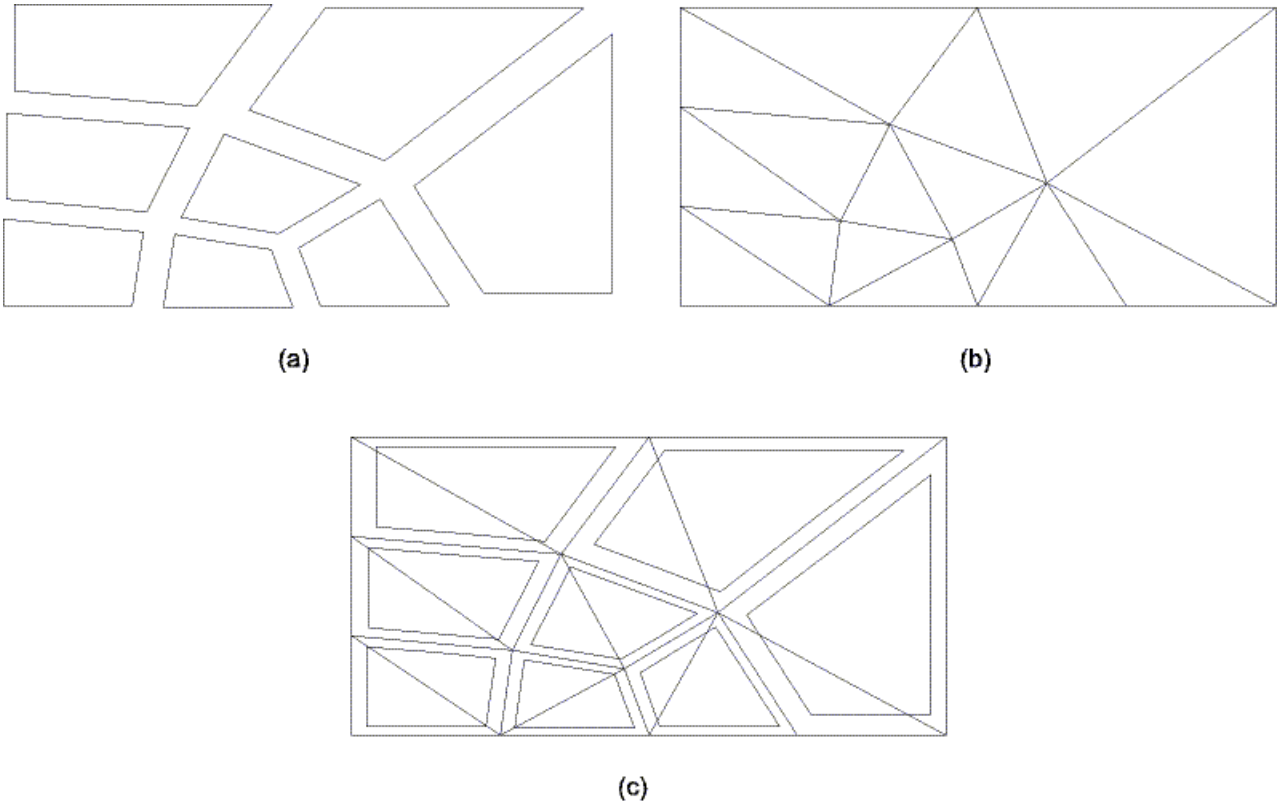
The quad-dominant free mesher [**MSHAPE**,0 and **MSHKEY**,0] automatically looks for four-sided regions that match these transition patterns. If a match is found, the area is meshed with a transition mapped quadrilateral mesh, unless the resulting elements are of poor quality (in which case a free mesh will be produced).

7.4.2.1.5. Transition Mapped Triangle Meshing

Transition mapped meshing is also valid for mapped area meshes of triangle elements. As with transition mapped quadrilateral meshing, transition mapped triangle meshing is only applicable to four-sided areas, and the specified line divisions must match one of the patterns shown in Figure 7.24: Applicable Transition Patterns-Transition Mapped Quadrilateral Meshes (p. 138). To achieve a transition mapped triangle mesh, you must also use an element type that supports a triangular shape, set the meshing key to mapped [**MSHKEY**,1], and set the shape specification to allow triangles [**MSHAPE**,1,2D].

The relationship between a transition mapped quadrilateral mesh and a transition mapped triangle mesh can be seen in Figure 7.25: Mapped Meshes (p. 140). Figure (b) illustrates a transition mapped triangle mesh. When you request a mapped triangle mesh, the program actually begins by map meshing the area with quadrilateral elements, and then it automatically splits the quadrilateral elements into triangles. Figure (a) shows the quadrilateral mesh that was used as the basis for the triangle mesh shown in Figure (b). Figure (c) illustrates the triangle mesh, with the quadrilateral elements superimposed over it. The dotted lines represent the boundaries of the quadrilateral elements that the program split into triangles.

Figure 7.25: Mapped Meshes



7.4.2.2. Volume Mapped Meshing

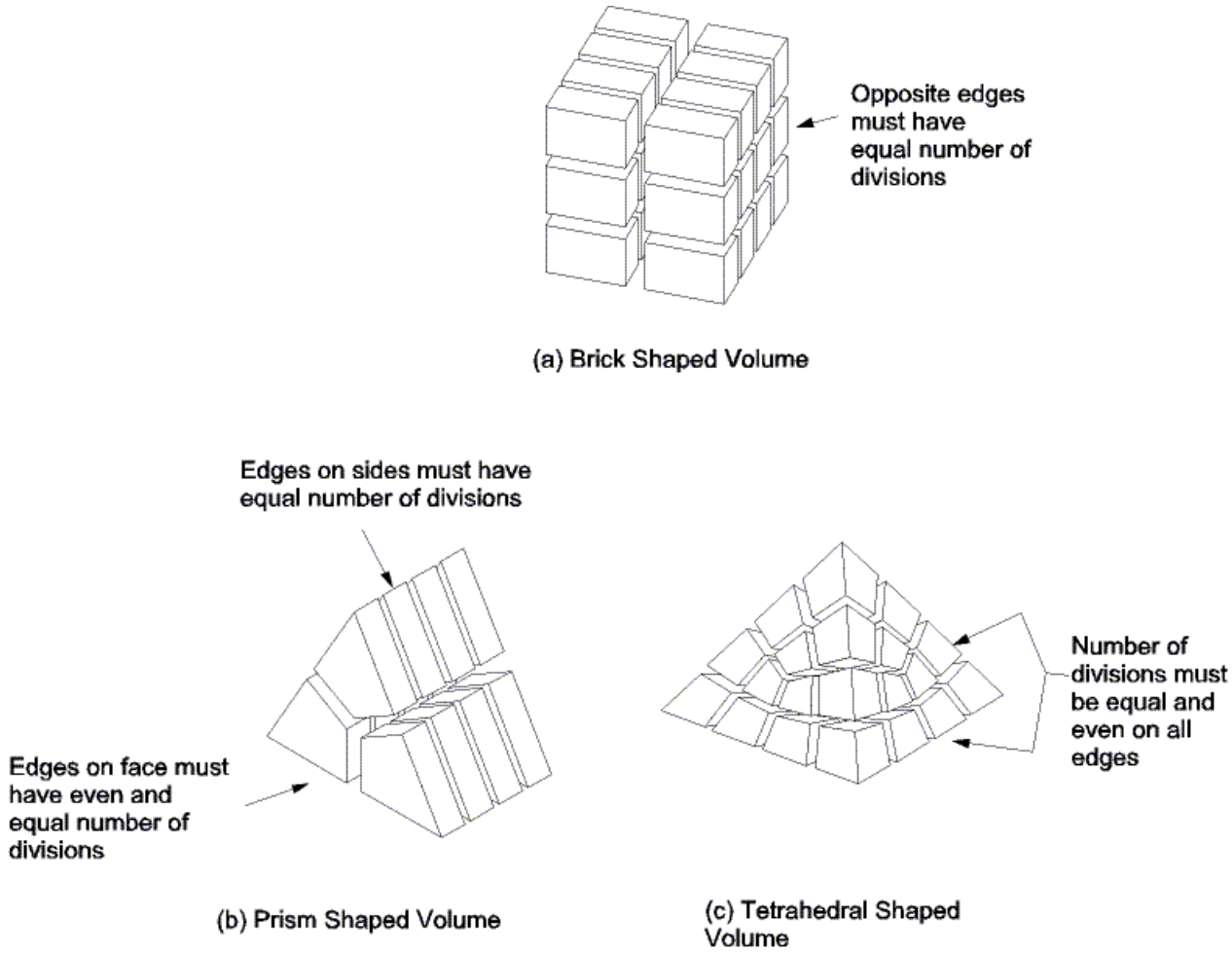
To mesh a *volume* with all hexahedron elements, the following conditions must be satisfied:

1. The volume must take the shape of a brick (bounded by six areas), wedge or prism (five areas), or tetrahedron (four areas).

2. The volume must have equal numbers of element divisions specified on opposite sides, or have divisions matching one of the transition mesh patterns for hexahedral meshes. See Figure 7.26: Examples of Element Divisions for Mapped Volume Meshing (p. 141) for examples of element divisions that will produce a mapped mesh for different volume shapes. Transition mesh patterns for hexahedral meshes are described later in this section.

3. The number of element divisions on triangular areas must be *even* if the volume is a prism or tetrahedron.

Figure 7.26: Examples of Element Divisions for Mapped Volume Meshing



7.4.2.2.1. Area Concatenation

As with lines, you can *add* [**AADD**] or *concatenate* [**ACCAT**] areas if you need to reduce the number of areas bounding a volume for mapped meshing. If there are also lines bounding the concatenated areas, the lines must be concatenated as well. You must concatenate the areas first, then follow with line concatenations. This procedure is illustrated by the sample input listing that appears below:

```
! first, concatenate areas for mapped volume meshing:
ACCAT,...
! next, concatenate lines for mapped meshing of bounding areas:
LCCAT,...
LCCAT,...
VMESH,...
```

Note

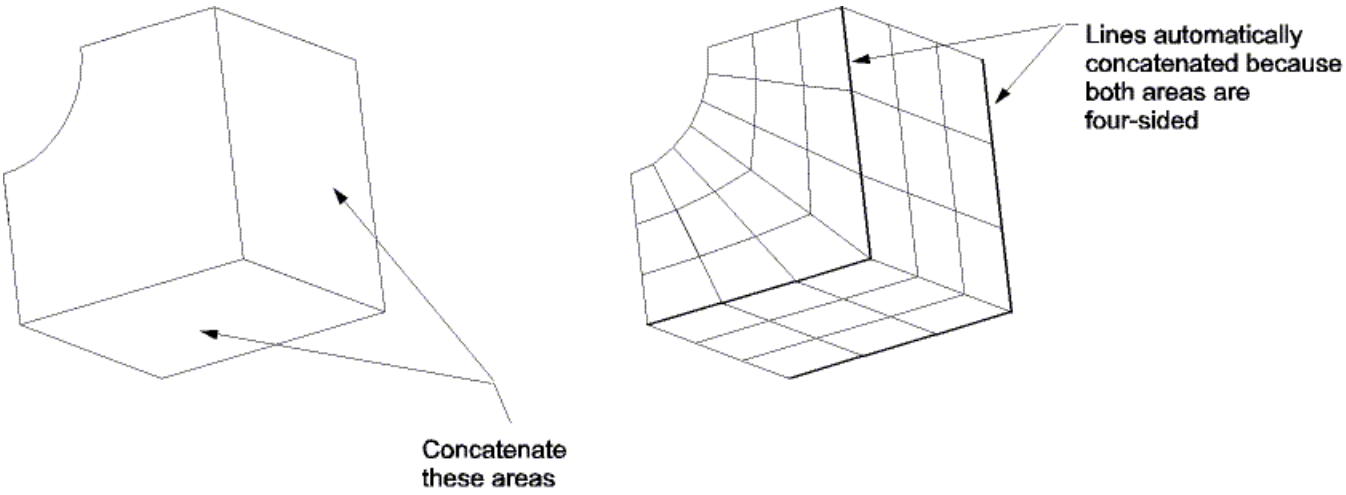
Whenever **AADD** is permitted (that is, when areas are flat and coplanar), it is generally preferred over **ACCAT**. (Line divisions will be transferred from one edge to another as described earlier.)

As shown in the sample input listing above, line concatenations [**LCCAT**] are normally required after area concatenations [**ACCAT**]. However, if both areas that are concatenated are bounded by four lines (no concatenated lines), the line concatenation operations will be done automatically. Thus, because

the areas in Figure 7.27: Area Concatenation (p. 142) are both bounded by four lines, line concatenation [**LCCAT**] is not required. Also note that deleting the concatenated area does not automatically delete the associated concatenated lines.

Area concatenation used for mapped volume meshing are shown in Figure 7.27: Area Concatenation (p. 142). The lines are automatically concatenated by the area concatenation operation [**ACCAT**] because both areas are bounded by four lines.

Figure 7.27: Area Concatenation



To concatenate areas, use one of the following methods:
Command(s): ACCAT
GUI: Main Menu> Preprocessor> Meshing> Concatenate> Areas
Main Menu> Preprocessor> Meshing> Mesh> Areas> Mapped

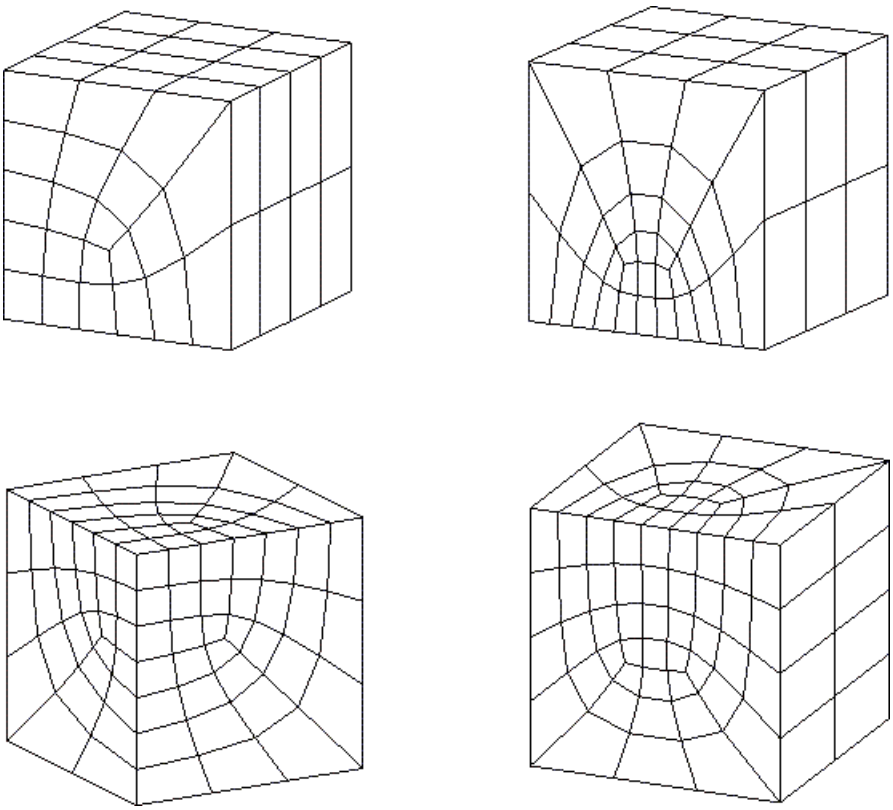
To add areas, use one of the following methods:
Command(s): AADD
GUI: Main Menu> Preprocessor> Modeling> Operate> Booleans> Add> Areas

Please see the **ACCAT**, **LCCAT**, and **VMESH** command descriptions for more information.

7.4.2.2.2. Transition Mapped Hexahedral Meshing

You can create a mapped volume mesh by specifying line divisions on opposite edges of the volume such that the divisions permit a transition mapped hexahedral mesh. Transition mapped hexahedral meshing is only applicable to six-sided volumes (with or without concatenation). Some examples are shown in Figure 7.28: Examples of Transition Mapped Hexahedral Meshes (p. 143).

Figure 7.28: Examples of Transition Mapped Hexahedral Meshes



To achieve a transition mapped hexahedral mesh, you must use an element type that supports a hexahedral shape. If you previously set the element shape specification to mesh with tetrahedral-shaped elements [**MSHAPE**,1,3D], you must now set the shape specification to allow hexahedron [**MSHAPE**,0,3D]. In addition, specified line divisions must match one of the patterns shown in Figure 7.29: Applicable Transition Patterns-Transition Mapped Hexahedral Meshes (p. 144).

Note

Even if you specify free meshing [**MSHKEY**,0], the program automatically looks for six-sided volumes that match these transition patterns. If a match is found, the volume will be meshed with a transition mapped hexahedral mesh, unless the resulting elements are of poor quality (in which case the mesh will fail).

Note

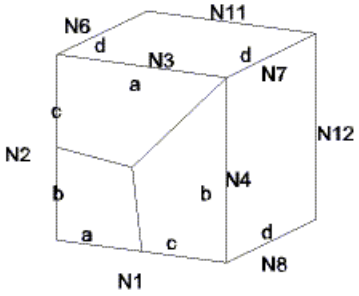
As indicated in Figure 7.29: Applicable Transition Patterns-Transition Mapped Hexahedral Meshes (p. 144), some of the edges of the volumes are hidden (edges N5, N9, and N10). Edge N5 is opposite edge N8; edge N9 is opposite edge N1; and edge N10 is opposite edge N2.

Figure 7.29: Applicable Transition Patterns-Transition Mapped Hexahedral Meshes

Note-In the volumes below, edges N5, N9, and N10 are hidden. Edge N5 is opposite edge N8; edge N9 is opposite edge N1; and edge N10 is opposite edge N2.

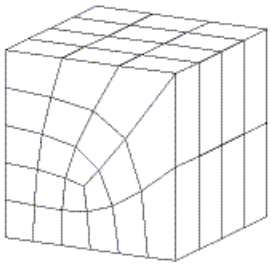
N1 = N9 (hidden)
= a + c
N2 = N10 (hidden)
= b + c
N3 = N11 = a
N4 = N12 = b

N5 (hidden) = N6
= N7 = N8
= d



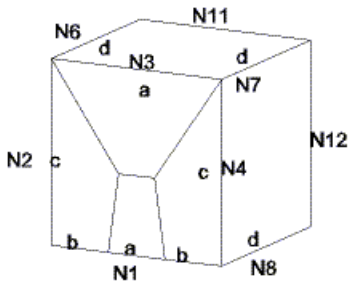
For example:
a = 3, b = 2
c = 3, d = 4

N1 = N9 = 6
N2 = N10 = 5
N3 = N11 = 3
N4 = N12 = 2
N5 = N6 = N7 = N8 = 4



N1 = N9 (hidden)
= a + 2*b
N2 = N10 (hidden)
= N4 = N12
= c
N3 = N11 = a

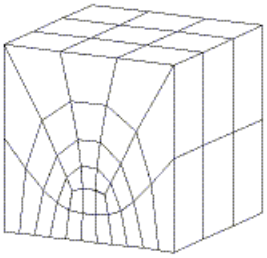
N5 (hidden) = N6
=N7 = N8
= d



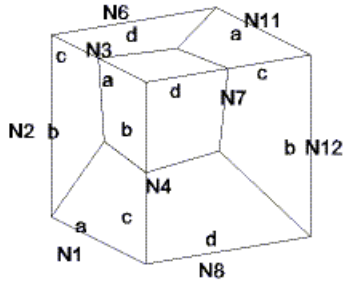
For example:
a = 3, b = 4
c = 2, d = 3

N1 = N9 = 11
N2 = N10 = N4 = N12 = 2
N3 = N11 = 3

N5 = N6 = N7 = N8 = 3

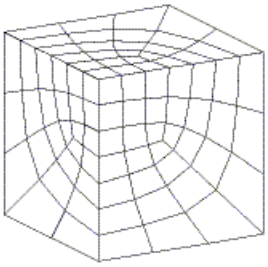


N1 = N9 (hidden)
= N11 = a
N2 = N10 (hidden)
= N12 = b
N3 = a + c
N4 = b+c
N5 (hidden) = N6
= N8 = d
N7 = c + d



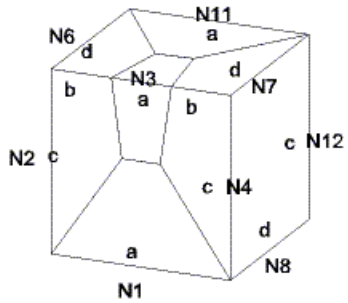
For example:
a = 2, b = 3
c = 4, d = 3

N1 = N9 = N11 = 2
N2 = N10 = N12 = 3
N3 = 6
N4 = 7
N5 = N6 = N8 =3
N7 = 7



N1 = N9 (hidden)
= N11 = a
N2 = N4 = N10
(hidden) = N12
= c
N3 = a + 2*b

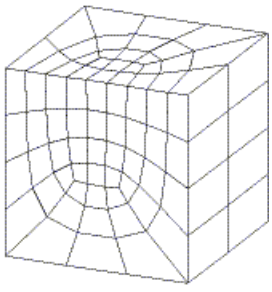
N5 (hidden) = N6
= N7 = N8
= d



For example:
a = 3, b = 3
c = 4, d = 2

N1 = N9 = N11 = 3
N2 = N4 = N10 = N12 = 4
N3 = 9

N5 = N6 = N7 = N8 = 2



7.4.2.3. Some Notes about Concatenated Lines and Areas

Concatenation is solely intended to be used as an aid to mapped meshing; it is not a Boolean "add" operation. Concatenation should be the *last* step you undertake before you execute a mapped mesh

of your solid model, because the output entity obtained from a concatenation cannot be used in *any* subsequent solid modeling operation (other than meshing, clearing, or deleting). For example, a line created by an **LCCAT** operation cannot have any solid model loads applied to it; nor can it be part of any Boolean operation; nor can it be copied, dragged, rotated [**xGEN**, **xDRAG**, **xROTAT**], etc.; nor can it be used in another concatenation.

You can readily "undo" a concatenation by simply deleting the line or area produced by the concatenation:

- The fastest way to delete concatenated lines or areas is by choosing menu path **Main Menu> Preprocessor> Modeling> Delete> Del Concats> Lines** or **Main Menu> Preprocessor> Modeling> Delete> Del Concats> Areas**.

  Caution

  When you use this method, the program automatically selects all concatenated lines (or areas) and deletes them without prompting you.

- If you want more control over which concatenated lines or areas are selected and deleted, use one of these methods:
  Command(s): LSEL,*Type*,LCCA,,,,,*KSWP* or **ASEL**,*Type*,ACCA,,,,,*KSWP*
  GUI: Utility Menu> Select> Entities

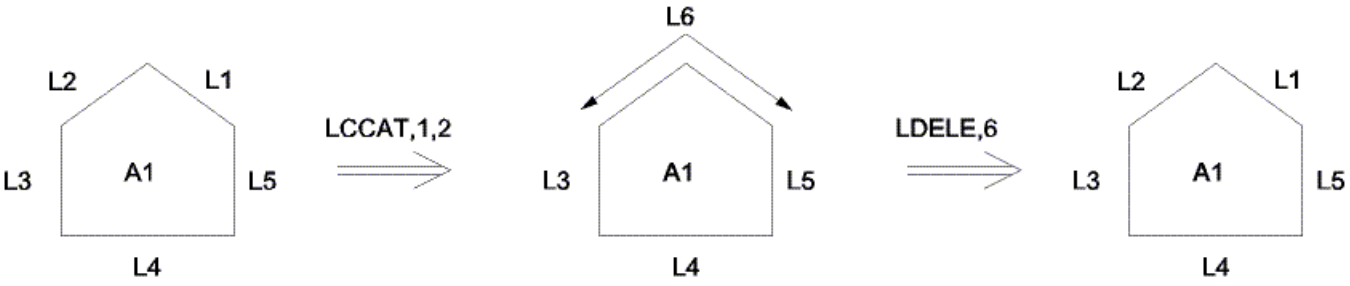
If you are using the Select Entities dialog box, choose both Lines and Concatenated to select concatenated lines. Choose both Areas and Concatenated to select concatenated areas. If desired, use the other controls in the dialog box to refine your selection.

You can then delete all of the selected lines or areas [**LDELE**,ALL or **ADELE**,ALL] as necessary.

Although you need to be aware of the restrictions on output entities listed earlier in this section, no such restrictions affect the *input* entities in a concatenation. However, the input entities will become "lost" or "detached," so far as higher-level entities are concerned. That is, if an area is bounded by five lines (L1-L5), and two of those lines are concatenated (**LCCAT**,1,2 $\geq$ L6), the program will no longer recognize lines L1 and L2 as being attached to that area. However, you can reattach L1 and L2 to the area by deleting L6 to undo the concatenation. (See Figure 7.30: Input Lines in a Concatenation (p. 145).)

Input lines in a concatenation become detached until the concatenation is undone

Figure 7.30: Input Lines in a Concatenation



If you find that concatenation becomes too restrictive for your intended modeling operations, you can usually obtain a mapped mesh by some other means, such as by subdividing an area or volume into

appropriately-bounded entities. Boolean operations will often be helpful for subdividing an entity in this fashion.

See the descriptions of the **ASEL**, **LSEL**, **ACCAT**, **LCCAT**, **ADELE**, and **LDELE** commands in the *Command Reference* for details.

7.5. Meshing Your Solid Model

Once you have built your solid model, established element attributes, and set meshing controls, you are ready to generate the finite element mesh. First, however, it is usually good practice to save your model before you initiate mesh generation:
Command(s): SAVE
GUI: Utility Menu> File> Save as Jobname.db

You may also want to turn on the "mesh accept/reject" prompt by picking **Main Menu> Preprocessor> Meshing> Mesher Opts**. This feature, which is available only through the GUI, allows you to easily discard an undesirable mesh. (For more information, see Changing the Mesh (p. 170).)

If you are meshing multiple volumes or areas at one time, you should consider using the meshing option **By Size** (or issuing the **MOPT**,ORDER,ON command) so the mesh is created in the smallest volume or area first. This helps ensure that your mesh is adequately dense in smaller volumes or areas and that the mesh is of a higher quality.

7.5.1. Generating the Mesh Using *x*MESH Commands

To mesh the model, you must use a meshing operation that is appropriate for the entity type being meshed. You can mesh keypoints, lines, areas, and volumes using the commands and GUI paths described below.

- To generate point elements (such as MASS21) at keypoints:
  Command(s): KMESH
  GUI: Main Menu> Preprocessor> Meshing> Mesh> Keypoints

- To generate line elements (such as LINK31) on lines:
  Command(s): LMESH
  GUI: Main Menu> Preprocessor> Meshing> Mesh> Lines

Also see Generating a Beam Mesh With Orientation Nodes (p. 147) for information about special beam meshing procedures.

- To generate area elements (such as PLANE183) on areas:
  Command(s): AMESH, **AMAP**
  GUI: Main Menu> Preprocessor> Meshing> Mesh> Areas> Mapped> 3 or 4 sided
  Main Menu> Preprocessor> Meshing> Mesh> Areas> Free
  Main Menu> Preprocessor> Meshing> Mesh> Areas> Target Surf
  Main Menu> Preprocessor> Meshing> Mesh> Areas> Mapped> By Corners

If you need to mesh multiple areas at one time, you should consider issuing the **MOPT**,ORDER,ON command so the mesh is created in the smallest area first. This helps ensure that your mesh is adequately dense in smaller areas and that the mesh is of a higher quality.

- To generate volume elements (such as SOLID90) in volumes:
  Command(s): VMESH
  GUI: Main Menu> Preprocessor> Meshing> Mesh> Volumes> Mapped> 4 to 6 sided

Main Menu> Preprocessor> Meshing> Mesh> Volumes> Free

Also see Generating a Volume Mesh From Facets (p. 153) and Generating a Volume Mesh By Sweeping (p. 154) for information about special volume meshing procedures.

- To generate interface elements (such as INTER192) of uniform thickness along lines or areas:
  Command(s): IMESH
  GUI: Main Menu> Preprocessor> Meshing> Mesh> Interface Mesh> 2D Interface
  Main Menu> Preprocessor> Meshing> Mesh> Interface Mesh> 3D Interface

Also see Generating an Interface Mesh for Gasket Simulations (p. 162) for information about special gasket meshing procedures.

7.5.2. Generating a Beam Mesh With Orientation Nodes

You can assign orientation keypoints as attributes of a line for beam/pipe meshing, just as you would assign a real constant set number, or a material property set number. The orientation keypoints are independent of the line that is to be meshed. Based on the location of these keypoints, the program will automatically create orientation nodes along with the beam elements. Line meshing with automatic generation of orientation nodes is supported for current-technology elements (such as BEAM161, BEAM188, BEAM189, PIPE288, PIPE289, and ELBOW290).

To assign orientation keypoints as attributes of a line:
Command(s): LATT
GUI: Main Menu> Preprocessor> Meshing> Mesh Attributes> All Lines
Main Menu> Preprocessor> Meshing> Mesh Attributes> Picked Lines

7.5.2.1. How the Program Determines the Location of Orientation Nodes

If a line is bounded by two keypoints (KP1 and KP2) and two orientation keypoints (*KB* and *KE*) have also been defined as attributes of the line, the orientation vector at the beginning of the line extends from KP1 to *KB*, and the orientation vector at the end of the line extends from KP2 to *KE*. The program computes the orientation nodes by interpolating the orientation as given by the above two orientation vectors.

Note

Although this discussion refers to them as "orientation nodes," elsewhere you may see this type of node referred to as an off-node, third node (for linear beam elements only), or fourth node (for quadratic beam elements only).

7.5.2.2. Benefits of Beam Meshing With Orientation Nodes

The direction in which beam sections are oriented will affect the beam element mesh and the analysis results. Beam meshing with orientation nodes gives you control over these effects. Examples of Beam Meshing With Orientation Nodes (p. 150) provides examples of various ways to align the beam sections.

If your analysis uses current-technology beam, pipe or elbow elements, you can use the cross-section data-definition, analysis, and visualization capabilities for these elements. You can assign a section ID number as an attribute of a line (**LATT**). The section ID number identifies the cross section used by the beam elements that will be generated when you mesh the line. The orientation nodes, generated automatically based on orientation keypoints that you specify (**LATT**), determine the section orientations

for the beam elements. For detailed information about beam analysis and cross sections, see Beam Analysis and Cross Sections in the *Structural Analysis Guide*.

7.5.2.3. Generating a Beam Mesh With Orientation Nodes

This section describes how to generate a beam mesh with orientation nodes, using either command input or the GUI. It assumes that you have already defined the geometry and element attribute tables for your model, and you are now ready to assign specific attributes to a line for beam meshing. This section does not attempt to cover other aspects of a typical beam analysis. For detailed information about beam analysis and a sample problem illustrating the generation of a beam mesh with orientation nodes, see Beam Analysis and Cross Sections in the *Structural Analysis Guide*.

If you are using the command method to generate the beam mesh, include these commands in your input:

1. Use the **LSEL** command to select the lines that you want to mesh with orientation nodes.

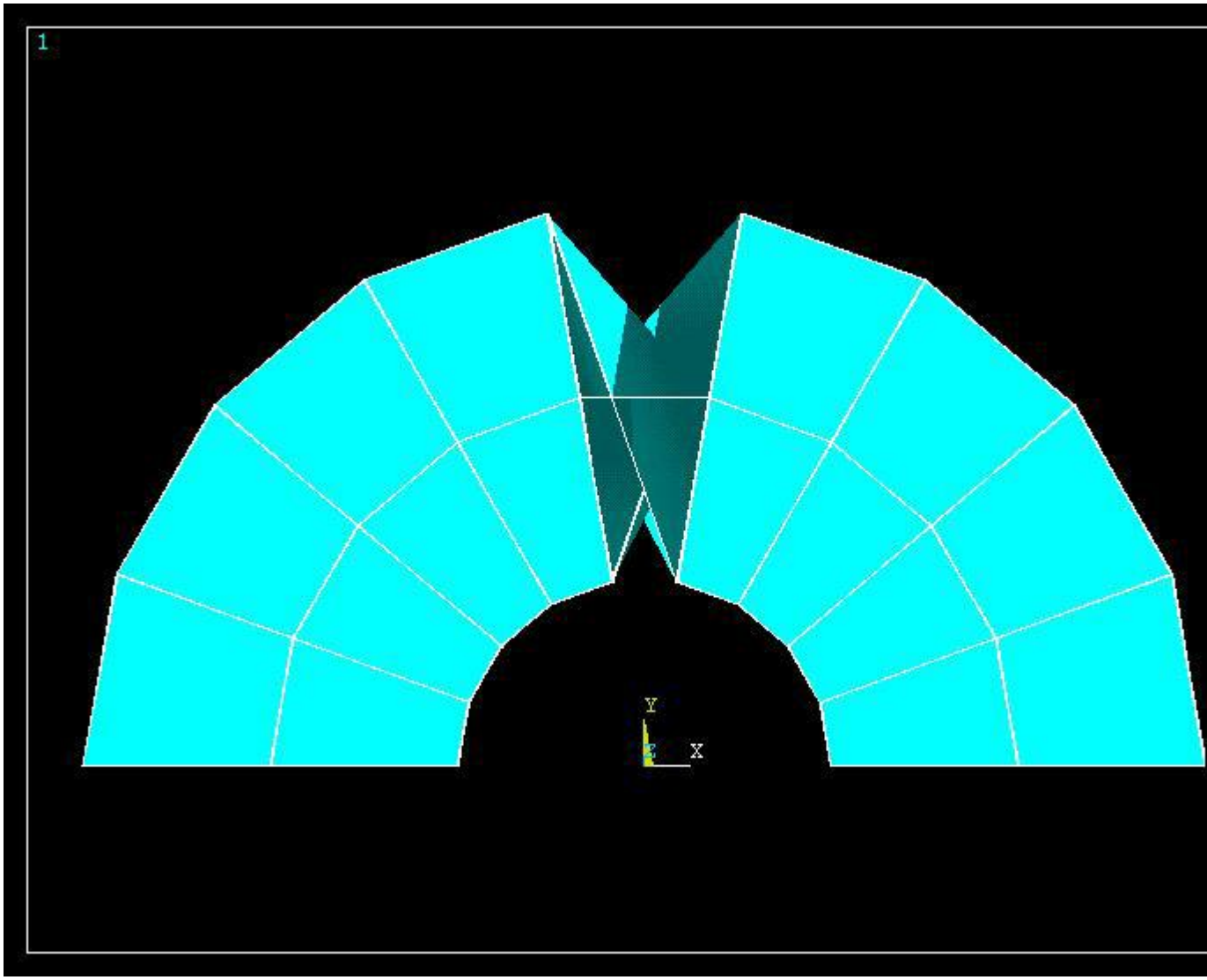
2. Use the **LATT** command to associate element attributes with the selected, unmeshed line(s). Specify values for the *MAT, REAL, TYPE, ESYS*, *KB*, *KE*, and *SECNUM* arguments.

   - When specifying a value for the *TYPE* argument on the **LATT** command, be sure that the element type you assign to the line is one that supports beam meshing with orientation (current-technology beam, pipe or elbow elements).

   - Use the *KB* and *KE* arguments on the **LATT** command to assign beginning and ending orientation keypoints. If you are meshing with current-technology beam elements, you must define at least one orientation keypoint when you set your attributes for meshing. When the mesh generates (**LMESH**), each beam element along the line will have two end nodes and one orientation node.

   - If you are meshing a circular arc with beams, and if that arc spans more than 90°, the beam orientation is twisted at the 90° point as shown in Figure 7.31:**LMESH** of an Arc (p. 149):

Figure 7.31: LMESH of an Arc



To prevent the beam orientation from twisting, set the ending orientation keypoint *(KE)* to be the same as *KB*, or split the line into 90° arcs.

- If you are using current-technology beam, pipe or elbow elements, use the *SECNUM* argument on the **LATT** command to assign a section ID number.

See Examples of Beam Meshing With Orientation Nodes (p. 150) for examples that illustrate various ways of assigning orientation keypoints. See Beam Analysis and Cross Sections in the *Structural Analysis Guide* for details about cross sections.

3. Set the number of element divisions to be generated along the line mesh [**LESIZE**].
4. Use the **LMESH** command to mesh the line(s).
5. After meshing a beam, always use the **/ESHAPE**,1 command to verify the beam's orientation graphically.
6. You can use the **LLIST**,,,,ORIENT command to list the selected line(s), along with any assigned orientation keypoints and section data.

If you are using the GUI to generate the beam mesh, follow these steps:

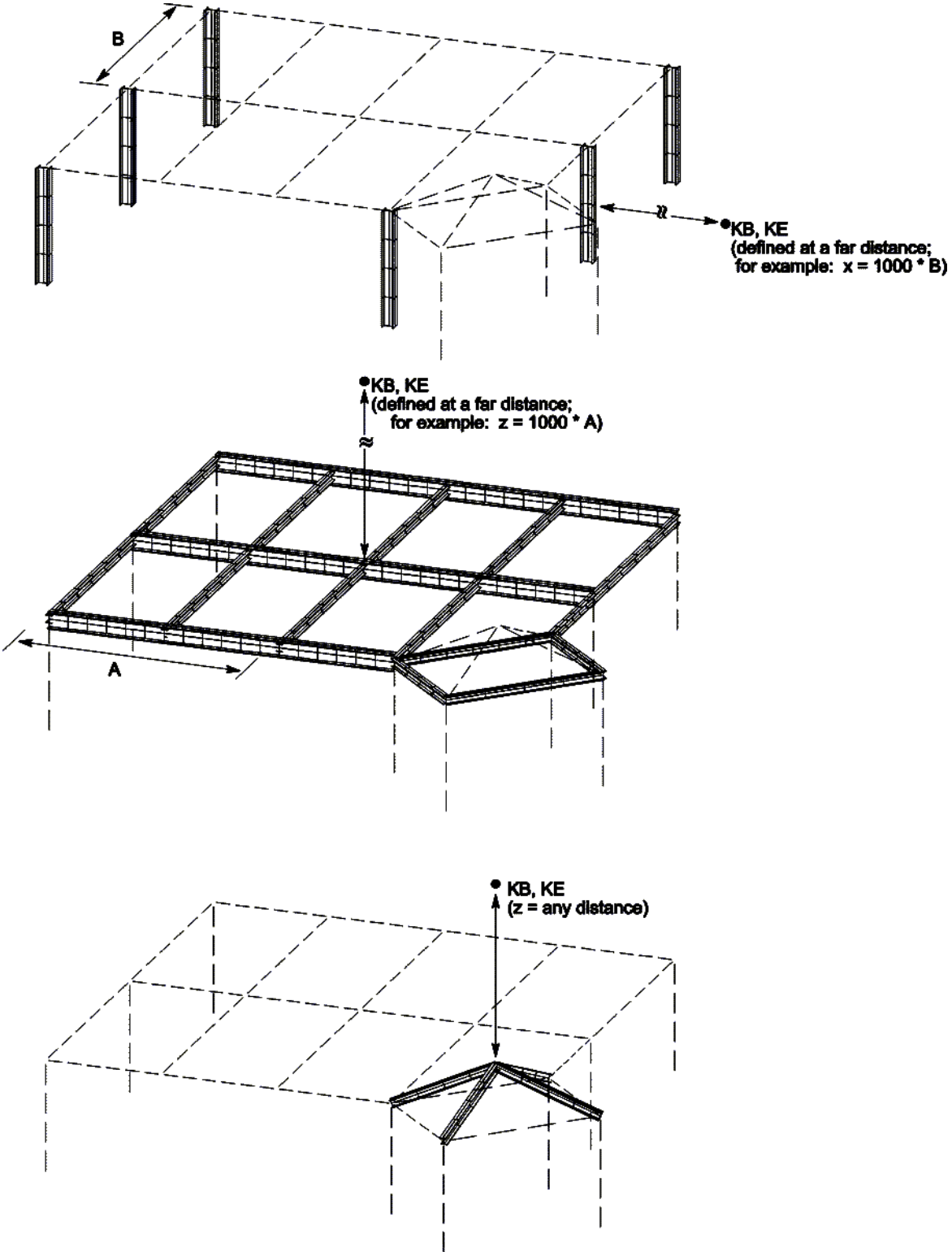
1. Choose menu path **Main Menu> Preprocessor> Meshing> MeshTool**. The MeshTool appears.
2. In the Element Attributes section of the MeshTool, select Lines from the option menu on the left and then click on Set. The Line Attributes picker appears.
3. In the Graphics window, click the line(s) to which you want to assign attributes (including orientation keypoints) and then click on OK in the Line Attributes picker. The Line Attributes dialog box appears.
4. In the Line Attributes dialog box, assign MAT, REAL, TYPE, ESYS, and/or SECT attributes as desired, click the Pick Orientation Keypoint(s) option so that Yes appears, and click on OK. The Line Attributes picker reappears.
5. In the Graphics window, pick the orientation keypoint(s) and then click on OK in the Line Attributes picker.
6. Back in the MeshTool, set any desired element size controls. Then initiate the line mesh operation by choosing Lines from the Mesh option menu and clicking on MESH. The Mesh Lines picker appears.
7. In the Graphics window, pick the line(s) that you want to mesh and then click on OK in the Mesh Lines picker. The program meshes the beam.
8. After the beam is meshed, always verify the beam's orientation graphically. Choose menu path **Utility Menu> PlotCtrls> Style> Size and Shape**. Click the **/ESHAPE** option to turn it on and click on OK. The meshed beam appears.
9. You can list the selected line(s), along with any defined orientation keypoints and section data. To do so, choose menu path **Utility Menu> List> Lines**. The **LLIST** Listing Format dialog box appears. Choose Orientation KP and then click on OK.

7.5.2.4. Examples of Beam Meshing With Orientation Nodes

You can define one orientation keypoint or two orientation keypoints as attributes of a line. If you define two, you can assign both of them to the same location in your model.

Figure 7.32: Placement of Orientation Keypoints and Element Orientation (p. 151) shows three examples. For each example, a beginning orientation keypoint and an ending orientation keypoint have been defined at the same location. The examples illustrate how you can assign different orientation keypoints to align selected beam sections within a structure in different directions.

Figure 7.32: Placement of Orientation Keypoints and Element Orientation

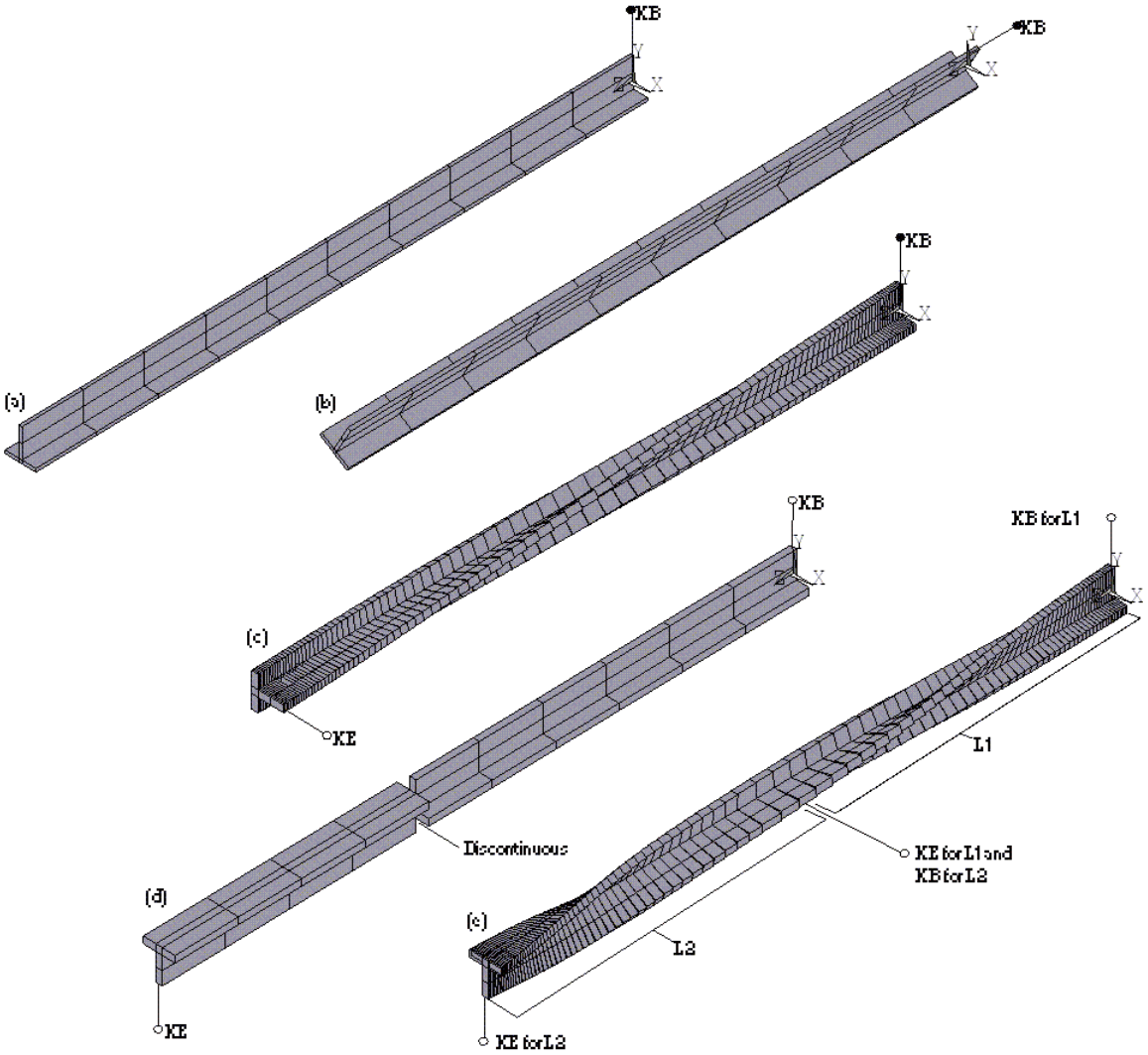


If you specify one orientation keypoint for a line, the program generates beam elements along the line with a constant orientation. If you specify different orientation keypoints at each end of the line, the program generates a pre-twisted beam.

Figure 7.33: Constant Orientation vs. Pre-Twist (p. 152) illustrates some differences between beam meshing with constant orientation as opposed to beam meshing with pre-twist.

- In Figure (a), only a beginning orientation keypoint was assigned. The keypoint is 0° from the y axis at a distance of 10 units in the y direction. The beam exhibits constant orientation.
- In Figure (b), only a beginning orientation keypoint was assigned. The keypoint is 30° from the y axis at a radius of 10 units. The beam exhibits constant orientation.
- In Figure (c), both a beginning and an ending orientation keypoint were assigned. The keypoints are 90° apart, causing a 90° twist in the beam. Due to the linear interpolation that is used to determine the location of the orientation nodes, a line biasing with small divisions at each end was used to cause the nodes to be located closer to the vectors made by the keypoints.
- In Figure (d), the orientation keypoints are 180° apart, and this time the beam flips. Assigning these keypoints causes a discontinuity because the interpolation of the two vectors is linear.
- Figure (e) provides a remedy to the problem illustrated in Figure (d). Here one line was divided into two, with the ending orientation keypoint for L1 and the beginning orientation keypoint for L2 being assigned to the same keypoint. A 180° twist is achieved.

Figure 7.33: Constant Orientation vs. Pre-Twist



7.5.2.5. Other Considerations for Beam Meshing With Orientation Nodes

Other things to consider when meshing beams with orientation nodes include the following:

- Since orientation is not required for 2-D beam elements, the beam meshing procedure described in this section does not support 2-D beam elements.
- Any operation on a line (copying the line, moving the line, and so on) will destroy the keypoint attributes.
- If an orientation keypoint is deleted, the program issues a warning message.
- If an orientation keypoint is moved, it remains an orientation keypoint. However, if an orientation keypoint is redefined (**K**,*NPT*,*X*,*Y*,*Z*), The program no longer recognizes it as an orientation keypoint.

Caution

If you issue the **CDWRITE** command after generating a beam mesh with orientation nodes, the database file will contain all of the nodes for every beam element, including the orientation nodes. However, the orientation keypoints that were specified for the line [**LATT**] are no longer associated with the line and are not written out to the geometry file. The line does not recognize that orientation keypoints were ever assigned to it, and the orientation keypoints do not "know" that they are orientation keypoints. Thus, the **CDWRITE** command does not support (for beam meshing) any operation that relies on solid model associativity. For example, meshing the areas adjacent to the meshed line, plotting the line that contains orientation nodes, or clearing the line that contains orientation nodes may not work as expected. This limitation also exists for the **IGESOUT** command. See the descriptions of the **CDWRITE** command and the **IGESOUT** command in the *Command Reference* for more information.

7.5.3. Generating a Volume Mesh From Facets

In addition to using **VMESH** to generate volume elements, you can generate a volume mesh from a set of detached exterior area elements (facets). For example, this capability is useful in situations where you cannot mesh a particular area. In such a situation, first mesh the areas that can be meshed. Next, define the remaining area elements using direct generation. (Elements that you define using direct generation are considered to be detached elements, because they have no solid model associativity.) Finally, use one of the methods below to generate nodes and tetrahedral volume elements from the detached area elements:

Command(s): FVMESH

GUI: Main Menu> Preprocessor> Meshing> Mesh> Tet Mesh From> Area Elements

---

Note

The main tetrahedra mesher [**MOPT**,VMESH,MAIN] is the only tetrahedra mesher that supports the generation of a volume mesh from facets; the alternate tetrahedra mesher [**MOPT**,VMESH,ALTERNATE] does not.

Note

The **FVMESH** command and its corresponding menu path do not support multiple "volumes." If you have multiple volumes in your model, select the surface elements for one "volume,"

while making sure that the surface elements for the other volumes are deselected. Then use **FVMESH** to generate a mesh for the first volume. Continue this procedure by selecting one volume at a time and meshing it, until all of the volumes in the model have been meshed.

7.5.4. Additional Considerations for Using *x*MESH Commands

Additional considerations for using **xMESH** commands include the following:

- Sometimes you may need to mesh the solid model with a variety of elements of different dimensionalities. For example, you may need to "reinforce" a shell model (area elements) with beams (line elements), or overlay one of the faces of a 3-D solid model (volume elements) with surface effect (area) elements. You can do so by using the appropriate meshing operations [**KMESH**, **LMESH**, **AMESH**, and **VMESH**] in any desired order. Make sure, however, that you set the appropriate element attributes (discussed earlier in this chapter) *before* meshing.

- No matter which volume mesher you choose [**MOPT**,VMESH,*Value*], it may produce different meshes on different hardware platforms when meshing volumes with tetrahedral elements [**VMESH**, **FVMESH**]. Therefore, you should be cautious when evaluating results at a specific node or element. The location of these entities may change if the input created on one platform is later run on a different platform.

- The adaptive meshing macro [**ADAPT**] is an alternative meshing method that automatically refines the mesh based on mesh discretization errors. See Adaptive Meshing in the *Advanced Analysis Guide* for more information on this feature.

7.5.5. Generating a Volume Mesh By Sweeping

Using volume sweeping, you can fill an existing unmeshed volume with elements by sweeping the mesh from a bounding area (called the "source area") throughout the volume. If the source area mesh consists of quadrilateral elements, the volume is filled with hexahedral elements. If the area consists of triangles, the volume is filled with wedges. If the area consists of a combination of quadrilateral and triangular elements, the volume is filled with a combination of hexahedral and wedge elements. The swept mesh is fully associated with the volume.

7.5.5.1. Benefits of Volume Sweeping

Volume sweeping provides these benefits:

- Unlike other methods for extruding a meshed area into a meshed volume [**VROTAT**, **VEXT**, **VOFFST**, and **VDRAG** commands], volume sweeping [**VSWEEP**] is intended for use in *existing* unmeshed volumes. Thus it is particularly useful in these situations:

  – You have imported a solid model that was created in another program, and you want to mesh it in this one.

  – You want to create a hexahedral mesh for an irregular volume. You only have to break up the volume into a series of discrete sweepable regions.

  – You either want to create a different mesh than the one that was created by one of the other extrusion methods, or you forgot to create a mesh during one of those operations.

  – If you do not mesh the source area prior to volume sweeping, the program meshes it for you when you invoke the volume sweeper. The other extrusion methods require you to mesh the area yourself

before you invoke them. If you do not, the other extrusion methods create the volume, but no area or volume mesh is generated.

7.5.5.2. What to Do Before You Sweep a Volume

Follow these steps before you invoke the volume sweeper:

1. Determine how many volumes need to be swept by **VSWEEP**. **VSWEEP** will sweep either a single volume, all selected volumes (**VSWEEP**,ALL) or a component of volumes (**VSWEEP**,CMVL where CMVL is the name of a volume component.)

2. Determine whether the volume's topology can be swept. The volume *cannot* be swept if any of these statements is true:

   - If **LESIZE** is used with the "hard" option, and the source and target areas contain hard division which are the not the same for each respective line, then the volume is not sweepable.

   - The volume contains more than one *shell*; in other words, there is an internal void within the volume. (A shell is the volumetric equivalent of an area loop - a set of entities that defines a continuous closed boundary. The SHELL column in a volume listing [**VLIST**] indicates the number of shells in the volume.)

   - The source area and the target area are not opposite one another in the volume's topology. (By definition, the target area must be opposite the source area.)

   - There is a hole in the volume that does not penetrate the source and/or target areas.

   Note

   Even if you have satisfied these requirements, there may be times when the shape of a volume causes the volume sweeper to create poorly shaped elements. For help, see Strategies for Avoiding Shape Failures During Volume Sweeping (p. 158).

   If the volume is not sweepable, try to cut the volume into sweepable sub-volumes before using the **VSWEEP** command.

3. Make sure that you have defined the appropriate 2-D and 3-D element types [**ET**]. For example, if you are going to pre-mesh the source area, and you want the swept volume to contain quadratic hexahedral elements, you should mesh the source area with quadratic 2-D elements.

4. Determine how you want to control the number of element layers that will be created during the sweeping operation; that is, the number of elements that will be created along the length of the sweep direction (see Figure 7.34: Specifying Volume Sweeping (p. 156)). You can use any of these methods to control this number:

   - Use **ESIZE**,SIZE to specify a guiding element size for the sweeper. **VSWEEP** uses this value to internally compute the number of element layers. This is the preferred method for setting this value.

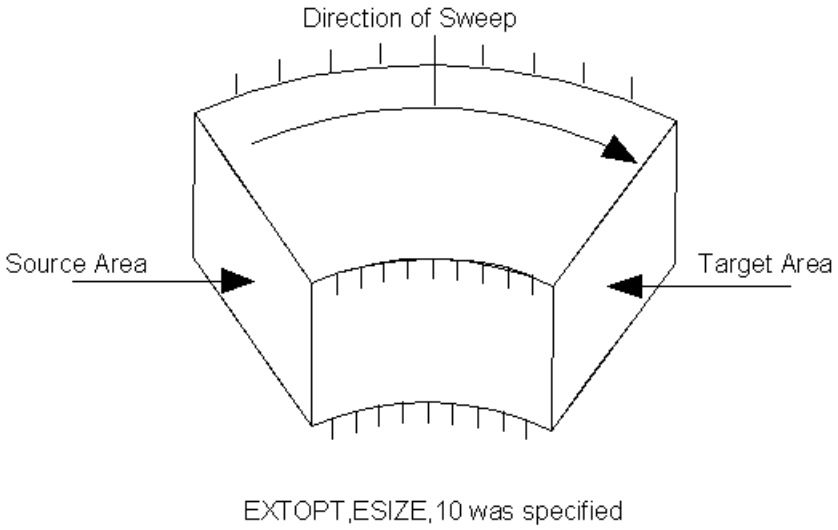
   - Use the **EXTOPT**,ESIZE,*Val1*,*Val2* command to specify the number of element layers (and if desired, the spacing ratio or bias) to be used along the volume's side lines (where *Val1* is the number of element layers and *Val2* is the bias). Note that the number of element layers and bias that you specify with **EXTOPT** will apply to all of the volume's unmeshed side lines. For any side

line that has been pre-meshed or that has other sizing specifications associated with it (via **LESIZE**), the values set by **EXTOPT** are ignored.
Command(s): EXTOPT,ESIZE,*Val1*,*Val2*
GUI: Main Menu> Preprocessor> Meshing> Mesh> Volume Sweep> Sweep Opts

- On one or more of the volume's side lines, use the **LESIZE** command to specify the number of element divisions for that particular line. This method also permits you to specify a bias that the volume sweeper will honor [**LESIZE**,,,,,*SPACE*]; however, the bias applies only to the line that you identify on the **LESIZE** command.
- Generate a mapped mesh on one or more of the side areas or within a volume or area that is adjacent to a side area or side line.
- Generate a mesh of beam elements on one or more of the side lines [**LMESH**].
- Turn on Smart sizing (via SMRT). **VSWEEP** will use smart sizing parameters to internally compute the number of element layers. Note that because of the constrained nature of Hexahedral Elements, and because smart sizing attempts to transition element sizes in different regions of the model, smart sizing does not always yield the highest quality elements. Although using the **ESIZE**,SIZE method mentioned above usually generates more elements, for Hex meshing they will normally be higher quality.

If the number of element layers is not specified with one of the methods listed above, **VSWEEP** will use **DESIZE** parameters to internally compute it.

Figure 7.34: Specifying Volume Sweeping



5. Optionally, determine which of the areas bounding the volume will be the source area, and which will be the target area. The program uses the pattern of the area elements on the source area (which can be quadrilateral and/or triangular elements) to fill the volume with hexahedral and/or wedge elements. (If you have not pre-meshed the area prior to volume sweeping, the program automatically generates "temporary" area elements. It does not save these area elements in the database; they are discarded as soon as the pattern for the swept volume is determined.) The target area is simply the area that is opposite the source area. See Figure 7.34: Specifying Volume Sweeping (p. 156) above, which illustrates one way that a user might set the number of element layers, source area, and target area for a volume sweeping operation. If more than one volume is being swept any user specified source and targets are ignored.

If no source and/or target area(s) are specified by the user, **VSWEEP** will attempt to automatically determine which bounding areas should be used as the source or target. If **VSWEEP** cannot determine the source/target automatically, **VSWEEP** will stop. When using the GUI, you can turn on auto source and target detection with the **EXTOPT**,VSWE,AUTO command.

6. Optionally, mesh the source, target, and/or side area(s).

   The results of a volume sweeping operation will differ depending on whether you have meshed any of the areas (source, target, and/or side areas) in your model prior to sweeping.

   Typically, you will generate a mesh for the source area yourself, before you sweep the volume. If you have not meshed the source area, the program will mesh it internally when you invoke the volume sweeping operation (as described above in Step 4.

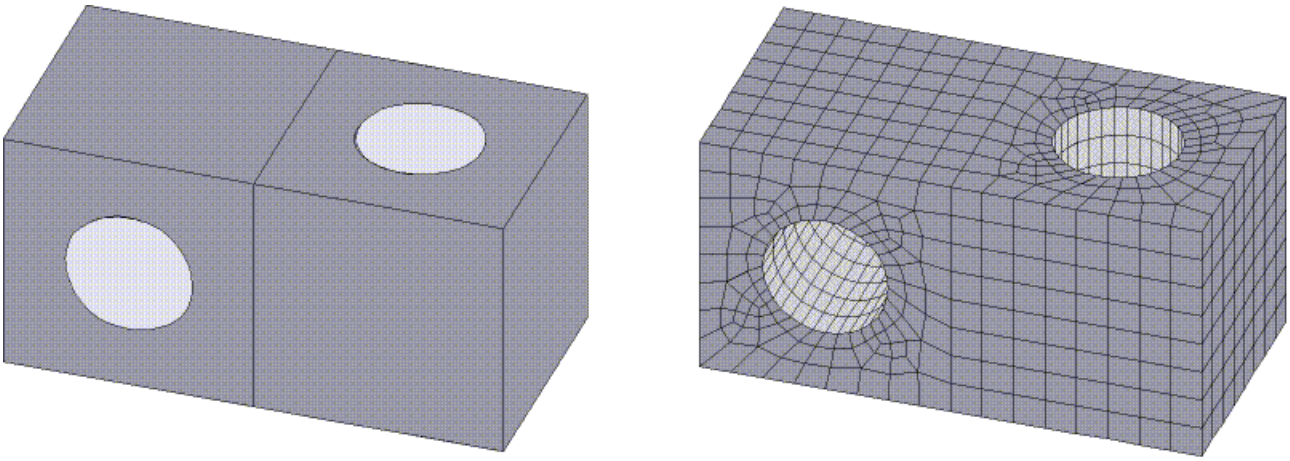
Consider the following information when deciding whether to "pre-mesh" before sweeping:

- If you do not pre-mesh, the program uses the element shape setting [**MSHAPE**] to determine the element shape it uses to mesh the source area (**MSHAPE**,0,2D results in quadrilateral elements; **MSHAPE**,1,2D results in triangular elements). The exception to this is if you sweep multiple volumes with a single call to **VSWEEP**. In this case the source area is always meshed with quadrilaterals.
- If you want the source area meshed with a **KSCON** specification, pre-mesh the source area.
- The sweeping operation will fail if hard points are present on an area or line that is associated with the volume, *unless* you pre-mesh the area or line containing the hard point.
- If you pre-mesh both the source area and the target area, the area meshes must match. However, the source area mesh and the target area mesh do not have to be mapped meshes.

See Other Characteristics of Volume Sweeping (p. 161) for more information about the characteristics of the volume sweeping feature.

Figure 7.35: Sweeping Adjacent Volumes (p. 157) (a) shows an example of a model that contains two volumes adjacent to one another. Because of the model's geometry, it is necessary to sweep the volumes in different directions, as shown in Figure (b).

Figure 7.35: Sweeping Adjacent Volumes



7.5.5.3. Invoking the Volume Sweeper

To invoke the volume sweeper:
Command(s): VSWEEP,*VNUM*,*SRCA*,*TRGA*,*LSMO*
GUI: Main Menu> Preprocessor> Meshing> Mesh> Volume Sweep> Sweep

If you are using the **VSWEEP** command to sweep a volume, specify values for the following arguments:

- Use the *VNUM* argument to identify the volume or volumes that you want to sweep.
- Optionally, use the *SRCA* argument to identify the source area.
- Optionally, use the *TRGA* argument to identify the target area.
- Optionally, use the *LSMO* argument to specify whether the program should perform line smoothing during the sweeping operation.

See the description of the **VSWEEP** command in the *Command Reference* for details about these arguments.

If you are using the GUI to call **VSWEEP**, the process depends on the setting of **EXTOPT**,VSWE,AUTO. If **EXTOPT**,VSWE,AUTO is ON (the default), follow these steps:

1. Choose menu path **Main Menu> Preprocessor> Meshing> Mesh> Volume Sweep> Sweep**. The Volume Sweeping picker appears.
2. Pick the volume or volumes to sweep and click OK to close the picker.

If **EXTOPT**,**VSWEEP**,**/AUTO** is OFF, follow these steps:

1. Choose menu path **Main Menu> Preprocessor> Meshing> Mesh> Volume Sweep> Sweep**. The Volume Sweeping picker appears.
2. Pick the volume you want to sweep and click Apply.
3. Pick the source area and click Apply.
4. Pick the target area. Click OK to close the picker.

Note

When using the GUI to sweep a volume, you cannot control whether line smoothing occurs. The program does not perform line smoothing when volume sweeping is invoked from the GUI.

7.5.5.4. Strategies for Avoiding Shape Failures During Volume Sweeping

If a volume sweeping operation fails due to bad element shapes, try one or more of the strategies listed below. We recommend that you try these strategies in the order in which they are listed. Hereafter in this section, all references to a figure are specific to Figure 7.36: Strategies for Avoiding Stretched Elements (p. 160).

1. If you did not specify a source and target, specify them and try the volume sweeper again.

2. Switch the source and target areas and re-invoke the volume sweeper. For example, if you specify area A1 as your source area and area A2 as your target area, and the sweep operation fails, try again using A2 as the source area and A1 as the target area.

3. Choose an entirely different set of source and target areas and re-invoke the volume sweeper. (Some volumes can be swept in more than one direction.) For example, if area A1 and area A2 do not work, try using A5 and A6.

4. Use shape checking as a diagnostic tool to determine which region of the model is causing the sweep failure. To do this, reduce the shape checking level to warning mode [**SHPP**,WARN], so that elements that violate error limits result in warning messages rather than element failures. Then re-invoke the sweeping operation. Use the resulting warning messages to identify the region of the model that contains the bad elements, and then clear the bad element mesh [**VCLEAR**]. Turn shape checking back on [**SHPP**,ON]. Next, modify the region of the model that contained the bad elements. Finally, mesh the volume again with a subsequent sweep operation. Here are some suggestions for modifying the model:

   - If one of the lines in the region of poor elements appears to have too many or too few divisions, manually specify a different number of divisions for that line using **LESIZE**.

   - Try using a smaller element size specification (**ESIZE**,SIZE) or a different number of element layers.

   - Divide the volume into two or more volumes [**VSBA**,**VSBW**], which will shorten the length of the sweep direction. Try dividing the volume near the region where the poorly shaped elements occur. Afterwards, invoke **VSWEEP** for each of the resulting volumes.

   - If the elements flagged by **SHPP**,WARN appear to be stretched within thin sections of the target area as in Figure (c), try dividing the side areas in that region along the direction of the sweep. Use these steps:

     a. Clear the mesh [**VCLEAR**].

     b. Divide one of the lines on the source area and one of the lines on the target area by adding keypoints at the desired division locations [**LDIV**]. See Figure (e).

     c. Copy the line divisions from the new lines on the source area to the corresponding new lines on the target area as described in Figure (e). (The "new lines" are those that were created by Step 2.) You can copy line divisions easily via the MeshTool. Choose menu path **Main Menu> Preprocessor> Meshing> MeshTool**. On the MeshTool, press the Copy button to open the picker. Use the picker to copy the line divisions - including spacing ratios - from one line to the other.

     d. Manually map mesh the side area that was affected by Step 2. See Figure (f).

     e. Re-invoke the volume sweeper.

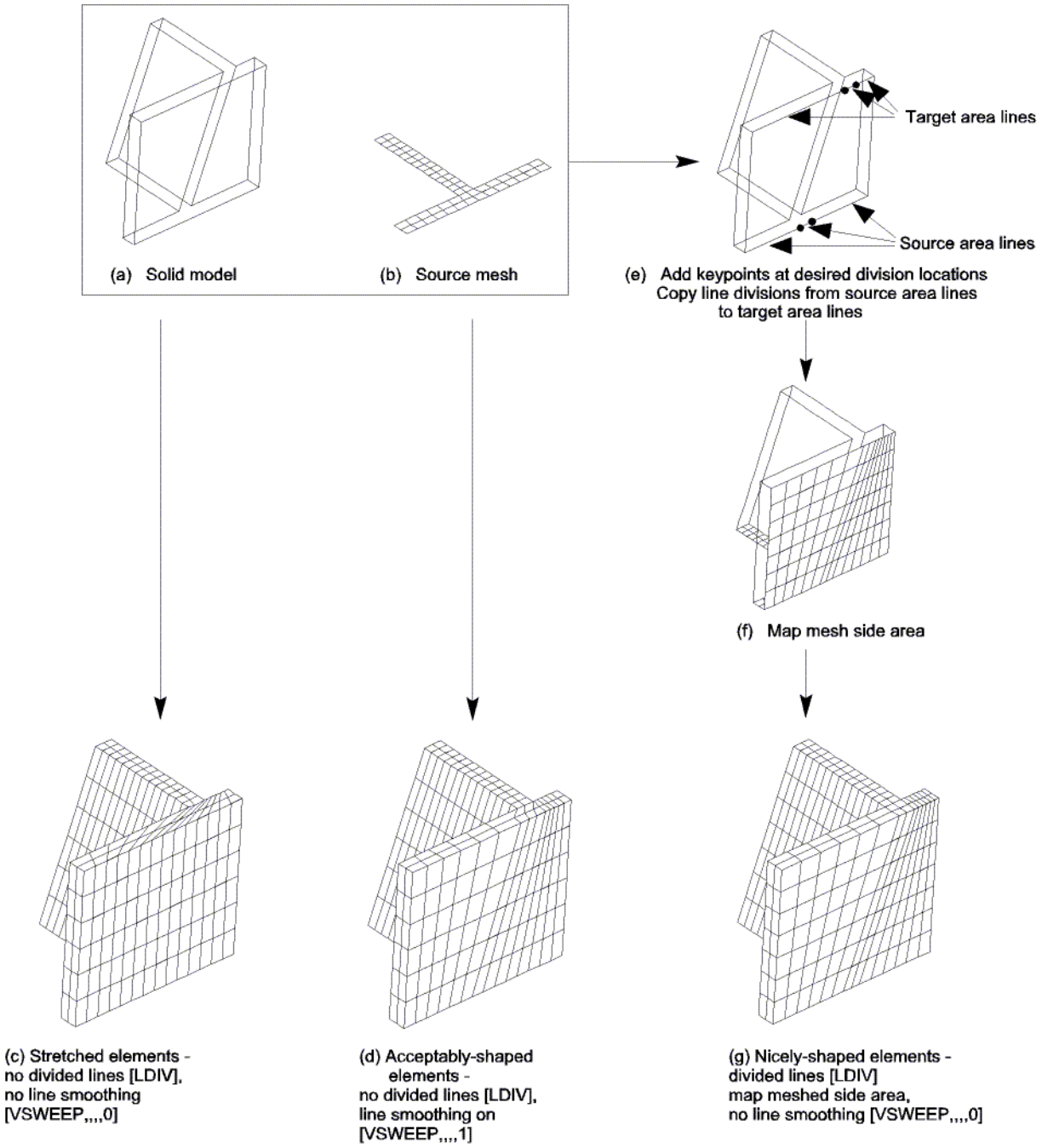
5. If the elements flagged by **SHPP**,WARN are stretched within thin sections of the target area, but the previous strategy does not work, clear the mesh and then re-invoke the volume sweeper with line smoothing turned on [**VSWEEP**,,,,1]. See Figure (d). (This setting is not recommended for large models due to speed considerations.)

Figure (c), (d), and (g) show the results of three different sweeping operations, and illustrate how you can use some of the strategies described above to affect the quality of a swept mesh. In all three cases, the user started with the same volume, which is shown in Figure (a). Figure (b) illustrates the source

mesh that was used during the sweep. Again, in all three cases, the user generated this source mesh prior to invoking volume sweeping.

The differences in the results are due to the additional actions (if any) that the user took prior to sweeping. To get the results shown in Figure (c), the user invoked volume sweeping without using any of the strategies described above. Notice the stretched elements that appear on the target area. For the results shown in Figure (d), the user invoked volume sweeping with line smoothing turned on [**VSWEEP**,,,,1]. In this case, the element shapes are better than those shown in Figure (c); however, they are not as good as those shown in Figure (g). For the results in Figure (g), the user divided lines [**LDIV**] on the source and target area and map meshed the affected side area prior to sweeping. Notice the significant improvement in the shape of the elements on the target area.

Figure 7.36: Strategies for Avoiding Stretched Elements

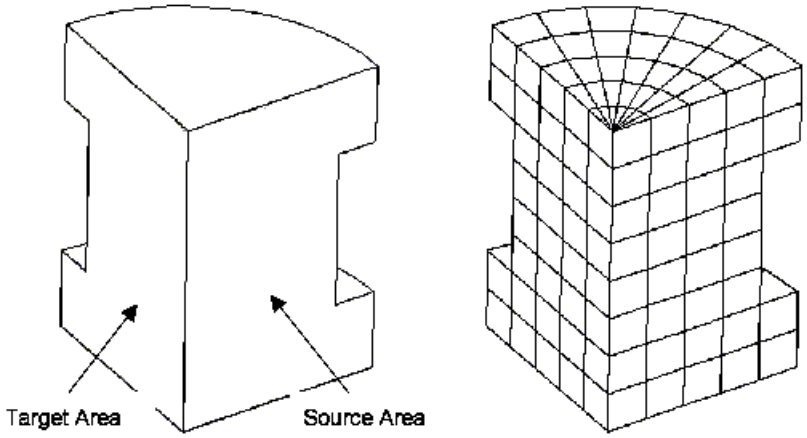


7.5.5.5. Other Characteristics of Volume Sweeping

Other characteristics of volume sweeping include the following:

- The source area and the target area do not have to be flat or parallel.
- If the topology of the source area and the topology of the target area are the same, the sweeping operation will often succeed even if the shape of the source area is different from the shape of the target area. However, drastically different shapes can cause element shape failures.
- During volume sweeping, the program can create either linear or quadratic elements. It can sweep quadratic area elements into linear volume elements, and it can sweep linear area elements into quadratic volume elements. (However, when the program sweeps linear area elements into quadratic volume elements, mid-nodes are not added to the edges of the source area. This will result in an element shape failure if you are using quadratic volume elements that do not support dropped mid-nodes.)
- If no source and target is provided, any values specified **EXTOPT**, **ESIZE**, NDIV, BIAS are ignored because they are direction dependent.
- If you have pre-meshed the source, target, and/or side areas, you can issue the **EXTOPT**,ACLEAR,1 command prior to sweeping and the program will automatically clear the area elements from any selected source, target, or side area after the swept volume mesh is created. (In the GUI, choose menu path **Main Menu> Preprocessor> Meshing> Mesh> Volume Sweep> Sweep Opts** to access a dialog box where you can toggle this clearing option on and off.) *The areas that you want to clear must be selected for clearing of the area meshes to occur.*
- Volume sweeping does not require your model to have a constant cross section. However, if the cross section varies, it should vary linearly from one end of the sweep to the other for best results.
- During volume sweeping, if you do not pre-mesh the source area, the program *will* use the **SMRTSIZE**, **KESIZE**, **ESIZE**, or **DESIZE** command setting (as appropriate) to mesh the source area. Also note the following information about the **ESIZE** command. The **ESIZE**,,*NDIV* setting *may be used* to determine the number of element layers that will be created along the volume's side lines during sweeping. However, this is *not* the preferred method because if the source area is not pre-meshed, the number of element divisions specified by *NDIV* will apply to all of the source area lines as well. If you want a specified number of element layers, the preferred method is to use the **EXTOPT** command (as described earlier).
- **VSWEEP** can sweep about a zero radius axis (i.e. the source and target areas are adjacent) if the user specifies which areas are to be used as the source and target (see figure below).

Figure 7.37: Sweep About Zero-Radius Axis

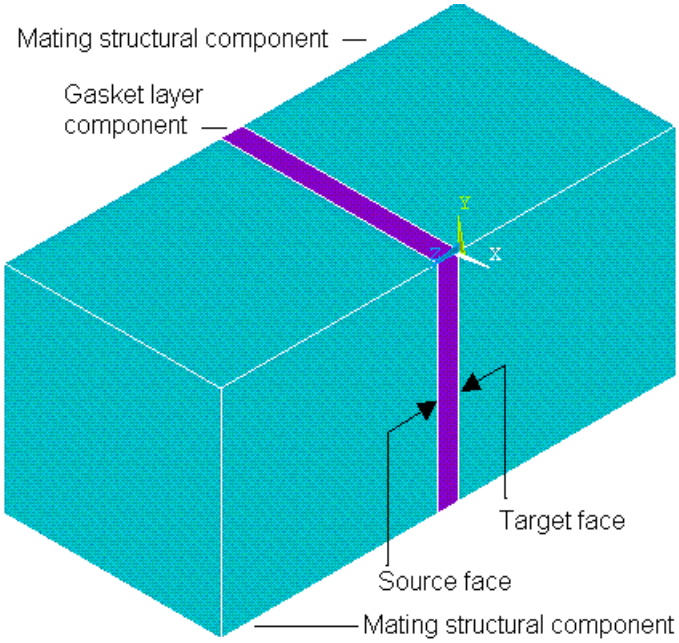


If the user does not specify a source and target on a volume requiring a zero radius axis, **VSWEEP** will not succeed. In addition, the element type used by **VSWEEP** must support wedges if sweeping a quad mesh and pyramids and tets if sweeping a tri mesh. You cannot perform a sweep about a zero-radius axis on defeaturable models.

7.5.6. Generating an Interface Mesh for Gasket Simulations

You must generate an interface mesh as part of the overall meshing procedure for simulating a gasket joint. This type of simulation involves using one of the 2-D or 3-D linear or quadratic interface elements layered between structural elements that have the same characteristics. Meshing of the interface element is done through the **IMESH** command for the gasket layer, and meshing of the structural elements is done through either the **AMESH** or **VMESH** command. The specific order of the meshing procedure is shown below, in reference to the following figure:

Figure 7.38: Components of an Interface Mesh



1. Define interface and structural elements that have corresponding characteristics. See Element Selection in the *Structural Analysis Guide* for element selection guidelines.

2. Mesh the structural component element that contains the *source* face using either the **AMESH** or **VMESH** command.

3. Mesh the gasket layer component element using either **IMESH**,LINE or **IMESH**,AREA; or **VDRAG**.

4. Mesh the structural component element that contains the *target* face using either the **AMESH** or **VMESH** command.

See the Notes sections of the **IMESH**, **VDRAG**, and **EGEN** command descriptions for special gasket meshing requirements.

Also, see Meshing Interface Elements in the *Structural Analysis Guide* for a sample input listing that includes the **IMESH** command, and graphics representing various meshes of interface and structural elements.

7.5.7. Aborting a Mesh Operation

When meshing is initiated, a status window appears. The window displays a message concerning the current status of the meshing operation, and also displays a scale showing the percentage of the meshing operation that is complete. Both the message and the percentage scale are updated periodically as the operation proceeds.

A STOP button is located at the bottom of the status window. Picking the STOP button aborts the mesh operation and causes incomplete meshes to be discarded. Areas or volumes that are completely meshed before STOP is picked will be retained. The solid model and finite element model will be left as they were before meshing was initiated.

You will see the meshing status window only when working in GUI mode. (Status windows will appear by default, but can be turned off by issuing **/UIS**,ABORT,OFF.) In non-GUI mode, a mesh abort is triggered by the system "break" function (CTRL-C or CTRL-P on most systems).

Note

If a session log file (`Jobname.LOG`) from an interactive session that included an intentional mesh abort is used as input for another session, the results will not likely be the same as they were for the interactive session since the abort will not be reproduced in the subsequent runs.

7.5.8. Element Shape Checking

"Badly shaped" elements can, on occasion, cause very poor analytical results. For this reason, the program performs element shape checking to warn you whenever any operation creates an element having a poor shape. Unfortunately, however, there are few *universal* criteria that can be used to identify a "poorly shaped" element. In other words, an element that gives poor results in one analysis might give perfectly acceptable results in another analysis. Thus, you must realize that the criteria that the program uses to identify poor element shapes are *somewhat arbitrary*. The fact that you receive even hundreds of element shape warnings does not necessarily mean that element shapes will cause any inaccuracy of results. (Conversely, if you do not receive any warnings about element shapes, that does not guarantee accurate results.) As in so many aspects of finite element analysis, the final determination of whether or not your element shapes are acceptable for your application remains your responsibility.

The program detects and flags *all* element shape warning and error conditions at the time of element creation, *before* storing each element.

Although the program performs element shape checking by default, a number of options for controlling element shape checking are available. Although most of the options are described in the sections that follow, you should refer to the **SHPP** command description in the *Command Reference* for additional information. Use either of these methods to modify shape checking:
Command(s): SHPP
GUI: Main Menu> Preprocessor> Checking Ctrls> Shape Checking
Main Menu> Preprocessor> Checking Ctrls> Toggle Checks

The sections that follow cover how to:

- Turn element shape checking off entirely or to warning-only mode
- Turn individual shape tests off and on
- View a summary of shape test results
- View current shape parameter limits
- Change shape parameter limits
- Retrieve element shape parameter data
- Understand the circumstances under which the program retests existing elements, and why doing so is necessary
- Decide whether element shapes are acceptable

Caution

The existence of badly shaped elements in a model may lead to certain computational errors that can cause your system to abort during solution. Therefore, you run the risk of a system abort during solution any time that you turn element shape checking off entirely, run shape checking in warning-only mode, turn off individual shape checks, or loosen shape parameter limits.

Note

The *Mechanical APDL Theory Reference* provides detailed information about the shape tests that the program performs and explains the logic that was used to determine each test's default warning and error limits.

7.5.8.1. Turning Element Shape Checking Off Entirely or to Warning-Only Mode

As stated above, the program performs element shape checking by default. When element shape checking occurs, any new element - regardless of how it was created - is tested against existing shape parameter warning and error limits. If the element violates any of the error limits, it not only produces an error message, but also either (a) causes a meshing failure, or (b) for element creation other than **AMESH** or **VMESH**, is not stored.

In certain cases, it may be desirable to turn element shape checking off, or to turn it on in warning-only mode. Turning element shape checking off [**SHPP**,OFF,ALL] deactivates shape checking entirely. When element shape checking is turned on in warning-only mode [**SHPP**,WARN], shape checking occurs, but

elements that violate error limits now only give warnings and do not cause either a meshing or element storage failure.

In the GUI, you can run shape checking in warning-only mode or turn it off entirely by choosing menu path **Main Menu> Preprocessor> Checking Ctrls> Shape Checking**. When the Shape Checking Controls dialog box appears, choose either "On w/Warning msg" or "Off"; then click on OK.

Situations in which we recommend that you turn shape checking off or run it in warning-only mode include:

- When you are generating an area mesh [**AMESH**], but your ultimate intention is to generate a volume mesh [**VMESH**] of quadratic tetrahedrons with that area as one of the volume's faces. Note that the tetrahedra mesher can fix meshes in which area elements have poor Jacobian ratios. Thus, if you are generating an area mesh for an area that will be a face on a volume in a subsequent volume meshing operation, it may make sense to turn element shape checking to warning-only mode, mesh the area, turn element shape checking on, and then mesh the volume.
- When you are importing a mesh [**CDREAD**]. If "bad" elements exist in a mesh that you want to import and element shape checking is turned on, the program may bring the mesh into the database with "holes" where the bad elements should be (or it may not import the mesh at all). Since neither of these outcomes is desirable, you may want to turn element shape checking either off or to warning-only mode prior to importing a mesh. After you import, we suggest that you turn shape checking back on and recheck the elements [**CHECK**,ESEL,WARN or **CHECK**,ESEL,ERR].

  Note

  Once elements are in the database, performing element shape checking will not delete them. If any elements in violation of error limits are selected when you initiate a solution [**SOLVE**], the program issues an error message and does not process the solution.

- When you are using direct generation and you are creating elements that you *know* will be temporarily invalid. For example, you may be creating a wedge-shaped element that has coincident nodes. You know that you need to merge the coincident nodes [**NUMMRG**] in order to get valid elements. In this case, it would make sense to turn off element shape checking, complete the desired operations (such as merging nodes in this example), turn element shape checking on, and then check the elements for completeness [**CHECK**].

7.5.8.2. Turning Individual Shape Tests Off and On

Rather than turn off shape checking entirely, you can selectively control which tests are off and which are on.

To use the command method to toggle the tests off and on, issue the command **SHPP**,*Lab*,*VALUE1*:

- Use the *Lab* argument to indicate whether you want to turn tests off or on. Specify OFF to turn tests off; specify ON to turn tests on.
- Use the *VALUE1* argument to indicate which tests you want to turn off or on. You can specify ANGD (SHELL28 corner angle deviation tests), ASPECT (aspect ratio tests), PARAL (deviation from parallelism of opposite edges tests), MAXANG (maximum corner angle tests), JACRAT (Jacobian ratio tests), or WARP (warping factor tests). You can also specify ALL to turn all tests off or on.

For example, the command **SHPP**,OFF,WARP turns off all warping factor tests.

In the GUI, you can toggle the tests off and on by choosing menu path **Main Menu> Preprocessor> Checking Ctrls> Toggle Checks**. When the Toggle Shape Checks dialog box appears, click the individual tests off or on as desired; then click on OK.

7.5.8.3. Viewing a Summary of Shape Test Results

The output below, which is from the **SHPP**,SUMMARY command, provides a summary of shape test results for all selected elements.

In the GUI, you can view a summary listing by choosing menu path **Main Menu> Preprocessor> Checking Ctrls> Shape Checking**. When the Shape Checking Controls dialog box appears, choose "Summary" in the option menu; then click on OK.

```
 SUMMARIZE SHAPE TESTING FOR ALL SELECTED ELEMENTS

  ---------------------------------------------------------------------------
            <<<<<<          SHAPE TESTING SUMMARY           >>>>>>
            <<<<<<        FOR ALL SELECTED ELEMENTS         >>>>>>
  ---------------------------------------------------------------------------
                    --------------------------------------
                    |  Element count       214 PLANE183   |
                    --------------------------------------

   Test               Number tested  Warning count  Error count    Warn+Err %
   ----               -------------  -------------  -----------    ----------
   Aspect Ratio               214              0             0         0.00 %
   Maximum Angle              214             59             0        27.57 %
   Jacobian Ratio             214              0             0         0.00 %

   Any                        214             59             0        27.57 %
  ---------------------------------------------------------------------------
```

7.5.8.4. Viewing Current Shape Parameter Limits

The output below, which is from the **SHPP**,STATUS command, lists the element shape parameters and default shape parameter limits. By default, when an element's shape falls outside of these limits, a warning or error condition occurs. See Changing Shape Parameter Limits (p. 167) for information about how to change the limits.

In the GUI, you can view a status listing by choosing menu path **Main Menu> Preprocessor> Checking Ctrls> Shape Checking**. When the Shape Checking Controls dialog box appears, choose "Status" in the option menu; then click on OK.

As stated above, this output shows the *default* shape parameter limits. If you modify any of these limits or turn off any of the individual shape tests, your output will differ accordingly.

In most cases in the output below, "FACE" also means "cross-section of solid element." For example, the ASPECT RATIO limits apply to both faces and cross-sections of tetrahedra, hexahedra (bricks), pyramids, and wedges.

```
 ASPECT RATIO (EXCEPT EMAG)
    QUAD OR TRIANGLE ELEMENT OR FACE
         WARNING TOLERANCE ( 1) =   20.00000
         ERROR TOLERANCE   ( 2) =   1000000.
 DEVIATION FROM 90° CORNER ANGLE
    SHELL28 SHEAR/TWIST PANEL
         WARNING TOLERANCE ( 7) =   5.000000
         ERROR TOLERANCE   ( 8) =   30.00000
 DEVIATION FROM PARALLEL OPPOSITE EDGES IN DEGREES    (EXCEPT EMAG)
    QUAD ELEMENT OR FACE WITHOUT MIDSIDE NODES
```

```
         WARNING TOLERANCE (11) =   70.00000
         ERROR TOLERANCE   (12) =   150.0000
   QUAD OR QUAD FACE WITH MIDSIDE NODES
         WARNING TOLERANCE (13) =   100.0000
         ERROR TOLERANCE   (14) =   170.0000
MAXIMUM CORNER ANGLE IN DEGREES (EXCEPT EMAG)
   TRIANGLE ELEMENT OR FACE
         WARNING TOLERANCE (15) =   165.0000
         ERROR TOLERANCE   (16) =   179.9000
   QUAD ELEMENT OR FACE WITHOUT MIDSIDE NODES
         WARNING TOLERANCE (17) =   155.0000
         ERROR TOLERANCE   (18) =   179.9000
   QUAD ELEMENT OR FACE WITH MIDSIDE NODES
         WARNING TOLERANCE (19) =   165.0000
         ERROR TOLERANCE   (20) =   179.9000
JACOBIAN RATIO
   h-METHOD ELEMENT
         WARNING TOLERANCE (31) =   30.00000
         ERROR TOLERANCE   (32) =   1000.000
   QUAD ELEMENT OR FACE WARPING FACTOR
   SHELL163, SHELL181
         WARNING TOLERANCE (51) =   1.000000
         ERROR TOLERANCE   (52) =   5.000000
   INFIN47, INTER115, SHELL131, SHELL157,
   SHELL181 WITH NLGEOM OFF AND KEYOPT1 NOT = 1
         WARNING TOLERANCE (53) =  0.1000000
         ERROR TOLERANCE   (54) =   1.000000
   SHELL41, OR SHELL181 WITH KEYOPT1=1
         WARNING TOLERANCE (55) =  0.4000000E-04
         ERROR TOLERANCE   (56) =  0.4000000E-01
   SHELL28
         WARNING TOLERANCE (57) =  0.1000000
         ERROR TOLERANCE   (58) =   1.000000
   SHELL181 WITH NLGEOM ON AND KEYOPT1 NOT = 1
         WARNING TOLERANCE (59) =  0.1000000E-04
         ERROR TOLERANCE   (60) =  0.1000000E-01
   3D SOLID ELEMENT FACE
         WARNING TOLERANCE (67) =  0.2000000
         ERROR TOLERANCE   (68) =  0.4000000

ELEMENT SHAPE CHECKING IS ON WITH DEFAULT LIMITS
```

7.5.8.5. Changing Shape Parameter Limits

If the program's default shape parameter limits do not suit your purposes, you can change them by using either the command method [**SHPP**,MODIFY,*VALUE1*,*VALUE2*] or the GUI.

For information about how to use the command method, see the description of the **SHPP** command in the *Command Reference*.

The GUI method is the simplest, and thus preferred, method for changing shape parameter limits. Follow these steps:

1. Choose menu path **Main Menu> Preprocessor> Checking Ctrls> Shape Checking**. The Shape Checking Controls dialog box appears.

2. Click the Change Settings option so that Yes appears.

3. Click on OK. The Change Shape Check Settings dialog box appears.

4. For any limit that you wish to change, enter a new limit. Use the scroll bar to move up and down within the list of limits.

5. When finished entering new limits, click on OK.

7.5.8.5.1. Examples of Changing Shape Parameter Limits

The element shape checking controls provide the flexibility to fit varied analysis needs. For example, perhaps you are not particularly concerned about aspect ratio measures. You can turn off all aspect ratio testing by including the command **SHPP**,OFF,ASPECT in your start150.ans file. If that seems too drastic for your situation, you may choose to specify **SHPP**,MODIFY,1,1000 instead. Doing so radically loosens the warning limit for aspect ratio tests, without turning off the tests entirely:

7.5.8.6. Retrieving Element Shape Parameter Data

You can use the ***GET** and ***VGET** commands to retrieve element shape parameter data:

Command(s):

***GET**, *Par*, ELEM, *ENTNUM*, SHPAR, *IT1NUM*
***VGET**, *ParR*, ELEM, *ENTNUM*, SHPAR, *IT1NUM*,,, *KLOOP*

For example, the command ***GET**,A,ELEM,3,SHPAR,ASPE returns the calculated aspect ratio of element number 3 and stores it in a parameter named A. The command ***VGET**,A(1),ELEM,3,SHPAR,ASPE returns the aspect ratio of element number 3 and stores it in the first location of A. Retrieval continues with elements numbered 4, 5, 6, and so on, until successive array locations are filled.

See the descriptions of the ***GET** and ***VGET** commands in the *Command Reference* for more information.

7.5.8.7. Understanding Circumstances Under Which the Program Retests Existing Elements

Certain types of changes that you make to defined elements can invalidate prior element shape testing. The program generally traps these types of changes and retest the affected elements automatically. Circumstances under which the program retests existing elements include:

- When you change the element type [**ET**,,*Ename* or **ETCHG**,*Cnv*] or one of its key options [**KEYOPT**].
- When you change the element TYPE number of an element [**EMODIF**].
- When you change the large deformation key [**NLGEOM**,*Key*] for SHELL181 elements.
- When you define a shell thickness [**R**] after you define an element, or you change an existing shell thickness [**RMODIF**] or the REAL number of a shell element [**EMODIF**].

Element type and TYPE number There is a distinction between the element type and that element type's TYPE number. The element *type* is the true name of the element (for example, SHELL181, sometimes shortened to simply 181). The element type's TYPE number is an arbitrary number that is locally assigned to a particular element type; you use the TYPE number to reference the element type when assigning attributes to your model.

7.5.8.8. Deciding Whether Element Shapes Are Acceptable

Here are some suggestions to help you decide whether you should be concerned about an element shape warning:

- Never ignore an element shape warning. Always investigate the effect that a "poorly" shaped element might have on your analysis results.

- Recognize that structural stress analyses that have the goal of determining stress at particular locations will typically suffer more severe effects from "poorly" shaped elements than will other types of analyses (deflection or nominal stress, modal, thermal, magnetic, etc.).
- If any "poorly" shaped elements are located in a critical region (such as near a critical stress point), their effect is more likely to be detrimental.
- "Poorly" shaped higher-order (midside-node) elements will generally produce better results than will similarly shaped linear elements. The default shape parameter limits are more restrictive on linear elements than on higher-order elements.
- Verification of analysis results by comparison with other analyses, test data, or hand calculations is essential regardless of whether elements produce shape warnings. If such verification indicates that you have high-quality results, there is little reason to worry about shape warnings.
- Some of the best quantitative measures of an element's acceptability are the error measures based on the element-to-element discontinuity in a stress or thermal gradient field. (See The General Post-processor (POST1) in the *Basic Analysis Guide*. *An element that produces shape warnings and shows a high error measure compared to its neighbors is suspect.*)

To check element shapes in an existing mesh (an internally created mesh or an imported mesh), use the **CHECK** command (**Main Menu> Preprocessor> Meshing> Check Mesh> Connectivity> Sel Bad Elems**).

Refer to the description of the **SHPP** command in the *Command Reference* for more information about element shape checking.

7.5.9. Mesh Validity Checking

There are times when the **CHECK** command will fail to detect potential problems in a mesh. The **CHECK** command is designed to examine each individual selected element and report warnings or errors based on specified shape criteria. It will not detect overlapping or poorly-connected elements. The **MCHECK** command is designed to detect potential problems with how elements are connected to each other. Figure 7.39: Flawed Mesh (elements “folded”) (p. 170) shows an example of a flawed mesh that would not be detected by the **CHECK** command. All elements in this mesh are of acceptable quality. The **MCHECK** command, however, would detect the potential connectivity problem and report an error.

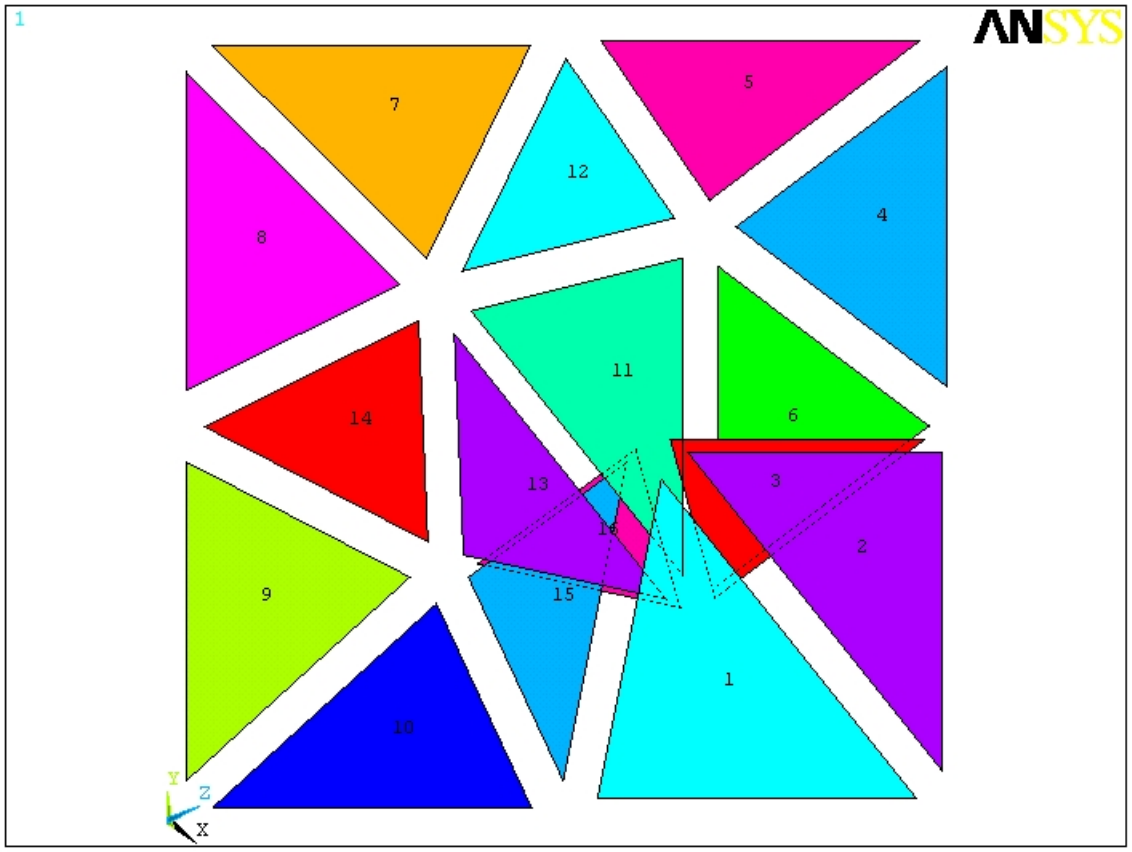
The **MCHECK** command will perform a number of validity checks on the selected elements, including

1. Normal check: Wherever two area elements share a common edge, **MCHECK** verifies that the ordering of the nodes on each element is consistent with their relative normals.
2. Volume check: Wherever two volume elements share a common face, **MCHECK** verifies that the sign of the integrated volume of each element is consistent.
3. Closed surface check: **MCHECK** verifies that the element exterior faces form simply-connected closed surfaces (this may detect unintended cracks in a mesh).
4. Check for holes in the mesh: **MCHECK** warns if the number of element faces surrounding an interior void in the mesh is small enough to suggest one or more accidentally omitted elements, rather than a deliberately formed hole. For this test, the number of faces around the void is compared to the smaller of a) three times the number of faces on one element, or b) one-tenth the total number of element faces in the model.

Similar to the CHECK command, **MCHECK** provides the option of unselecting all valid elements, so that the potential problem elements can be identified visually. Use *Lab*=ESEL to unselect valid elements.

To check mesh connectivity:
Command(s): MCHECK,*Lab*
GUI: Main Menu> Preprocessor> Meshing> Check Mesh> Ck Connectivity

Figure 7.39: Flawed Mesh (elements "folded")



7.6. Changing the Mesh

If you decide that the generated mesh is not appropriate, you can easily change the mesh by one of the following methods:

- Remesh with new element size specifications.
- Use the accept/reject prompt to discard the mesh, then remesh.
- Clear the mesh, redefine mesh controls, and remesh.
- Refine the mesh locally.
- Improve the mesh (for tetrahedral element meshes only).

Details of these methods are discussed below.

7.6.1. Remeshing the Model

You can remesh a meshed model by resetting element size controls and initiating the meshing operation [**AMESH** or **VMESH**]. This is the simplest way to change your mesh. The accept/reject prompt is not required, and the mesh does not need to be cleared in order to remesh it.

However, there are some restrictions to using this method. You can change element size specifications controlled by the **KESIZE**, **ESIZE**, **SMRTSIZE**, and **DESIZE** commands, but you cannot change size specifications assigned directly to lines [**LESIZE**]. If you want the option of changing **LESIZE** settings before remeshing, use the mesh accept/reject prompt instead of this method.

This remesh option is available only when meshing is performed interactively through the GUI. If you are using command input, you must first clear the mesh before remeshing (see Clearing the Mesh (p. 171) for more information).

7.6.2. Using the Mesh Accept/Reject Prompt

As mentioned earlier, you can activate the mesh accept/reject prompt in the GUI by picking **Main Menu> Preprocessor> Meshing> Mesher Opts** before meshing. (The prompt is turned off by default.) When activated, the prompt appears after each meshing operation and allows you to either accept or reject the generated mesh. If the mesh is rejected, all nodes and elements will be cleared from the meshed entities. You can then reset any of the meshing controls and remesh the model.

The accept/reject prompt is available for area and volume meshing. The advantage of using the prompt is that you do not have to manually clear the mesh [**ACLEAR** and **VCLEAR**].

7.6.3. Clearing the Mesh

Clearing the mesh of nodes and elements is not always required before remeshing. However, you do have to clear the mesh in order to respecify **LESIZE** settings. You also have to clear the mesh if you want to change the underlying solid model.

To clear the mesh from keypoints [**KCLEAR**], lines [**LCLEAR**], areas [**ACLEAR**], or volumes [**VCLEAR**], pick **Main Menu> Preprocessor> Meshing> Clear>** ***entity type*** in the GUI. (For more information on the clearing operation, see Clearing a Mesh (p. 195) of this manual.)

7.6.4. Refining the Mesh Locally

If you are generally satisfied with a mesh but would like to have more elements in a particular region, you can refine the mesh locally around selected nodes [**NREFINE**], elements [**EREFINE**], keypoints [**KREFINE**], lines [**LREFINE**], or areas [**AREFINE**]. The elements surrounding the chosen entities will be split to create new elements. You control the refinement process by specifying:

- The level of refinement to be done (in other words, the desired size of the elements in the refinement region, relative to the size of the original elements)
- The depth of surrounding elements that will be remeshed, in terms of the number of elements outward from the selected entity
- The type of postprocessing to be done after the original elements are split (smoothing and cleaning, smoothing only, or neither)

- Whether triangles can be introduced into the mesh during the refinement of an otherwise all-quadrilateral mesh

You can access local mesh refinement in the GUI by picking **Main Menu> Preprocessor> Meshing> Modify Mesh> Refine At>** ***entity***. You can also do overall refinement by using the command **ESEL**,ALL or by picking the menu path **Main Menu> Preprocessor> Meshing> Modify Mesh> Refine At> All**. See Revising Your Model (p. 181) of this manual for details on refining a mesh locally.

7.6.5. Improving the Mesh (Tetrahedral Element Meshes Only)

The tetrahedral mesh improvement feature enables you to improve a given tetrahedral mesh. The program performs this improvement through face swapping, node smoothing, and other techniques that it uses to reduce the number of poorly-shaped tetrahedral elements (in particular, the number of sliver tetrahedral elements) - as well as the overall number of elements - in the mesh. It also improves the overall quality of the mesh.

7.6.5.1. Automatic Invocation of Tetrahedral Mesh Improvement

In many cases, you won't need to take any action to obtain the benefits offered by the tetrahedral mesh improvement feature. As described earlier in Controlling Tetrahedral Element Improvement (p. 124), the program invokes the feature automatically as a postprocessing step of its volume meshers. Tetrahedral mesh improvement also occurs automatically during the creation of transitional pyramid elements (described in Creating Transitional Pyramid Elements (p. 124)) and the refinement of tetrahedral element meshes (described in Revising Your Model (p. 181)).

7.6.5.2. User Invocation of Tetrahedral Mesh Improvement

Although tetrahedral mesh improvement often occurs automatically, there are certain situations in which you'll find it useful to request additional improvement for a given tetrahedral mesh:

- When tetrahedra improvement is invoked automatically during a volume meshing operation [**VMESH**], the program uses a linear tetrahedral shape metric for improvement. This means that the program ignores midside nodes that may be present within the elements. However, when you request tetrahedra improvement of a given mesh as documented below, the program takes the midside nodes into account. Thus, for meshes of quadratic (midside-node) tetrahedral elements, requesting additional tetrahedra improvement [**VIMP**] after the mesh is created [**VMESH**] can help to remove, or at least reduce, the number of element shape test warning messages that are produced and to improve the overall quality of the mesh.

- Since imported tetrahedral meshes have not received the benefits of the tetrahedral mesh improvement that the program often performs automatically, imported tetrahedral meshes are likely candidates for user-invoked improvement.

Tetrahedral mesh improvement is an iterative process. Each time that processing completes, a special window appears to report the improvement statistics from that iteration, along with diagnostic messages. If you want to try to improve the mesh further, you can reissue your request repeatedly, until either the statistics indicate a satisfactory mesh, or until it converges and no more noticeable improvement is made.

You can request improvement of two "types" of tetrahedral elements:

- You can request improvement of tetrahedral elements that are not associated with a volume. (Typically, this option is useful for an imported tetrahedral mesh for which no geometry information is attached.) Use one of these methods:
  Command(s): TIMP
  GUI: Main Menu> Preprocessor> Meshing> Modify Mesh> Improve Tets> Detached Elems

- You can request improvement of tetrahedral elements that are in a selected volume or volumes. (You might want to use this option to further improve a volume mesh created in the program [**VMESH**].) Use one of these methods:
  Command(s): VIMP
  GUI: Main Menu> Preprocessor> Meshing> Modify Mesh> Improve Tets> Volumes

7.6.5.3. Restrictions on Tetrahedral Mesh Improvement

The following restrictions apply to tetrahedral mesh improvement:

- The mesh must consist of either all linear elements or all quadratic elements.

- For all of the elements in the mesh to be eligible for tetrahedral mesh improvement, they must all have the same attributes, including element type. (The element type must be tetrahedral, but the tetrahedral elements may be the degenerated form of hexahedral elements.) After tetrahedral mesh improvement, the program reassigns the attributes from the old set of elements to the new set of elements.

  Note

  Tetrahedral mesh improvement is possible in a mesh of mixed element shapes (as opposed to types). For example, as stated earlier, improvement occurs automatically during the creation of transitional pyramid elements at the interface between hexahedral and tetrahedral element types. However, in a mixed mesh, only the tetrahedral elements are improved.

- Loading has an effect on whether tetrahedral mesh improvement is possible. Tetrahedral mesh improvement is possible when loading occurs in either of these ways:

  – When loads have been applied to the element faces or nodes on the boundary of the volume only

  – When loads have been applied to the solid model (and have been transferred to the finite element mesh)

  Tetrahedral mesh improvement is not possible when loading occurs in either of these ways:

  – When loads have been applied to the element faces or nodes within the interior of a volume

  – When loads have been applied to the solid model (and have been transferred to the finite element mesh), but have also been applied to the element faces or nodes within the interior of a volume

  Note

  In the last two loading situations, the program issues a warning message to notify you that you must remove the loads if you want tetrahedral mesh improvement to occur.

- If node or element components are defined, you will be asked whether you want to proceed with mesh improvement. If you choose to proceed, you must update any affected components.

7.6.5.4. Other Characteristics of Tetrahedral Mesh Improvement

Other characteristics of tetrahedral mesh improvement include:

- Element numbering and node numbering are modified.
- Generally, if the program encounters an error or a user abort occurs, it leaves the mesh unchanged. However, the program may save a partially improved mesh after an abort if you verify the save when the program prompts you. If you have requested improvement for multiple volumes [**VIMP**], the abort applies only to the current volume mesh that is being improved; all previously improved volume meshes are already saved. (The same applies when an error occurs after the first of multiple volumes is improved.)

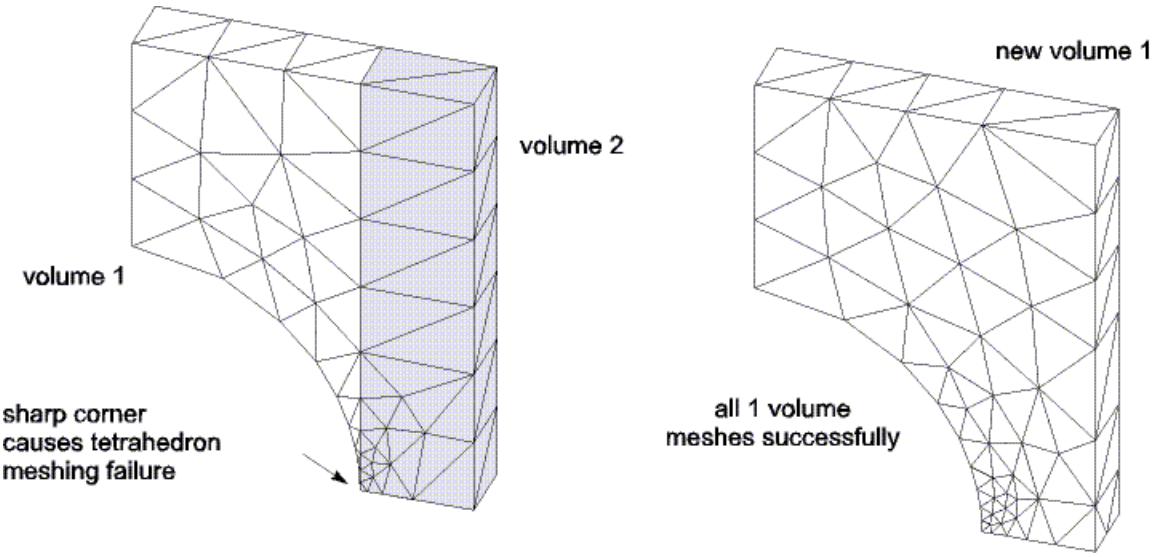
Please see the **TIMP** and **VIMP** command descriptions for more information.

7.7. Meshing Hints

Consider the hints and cautions in this section if you are having difficulty obtaining a good mesh.

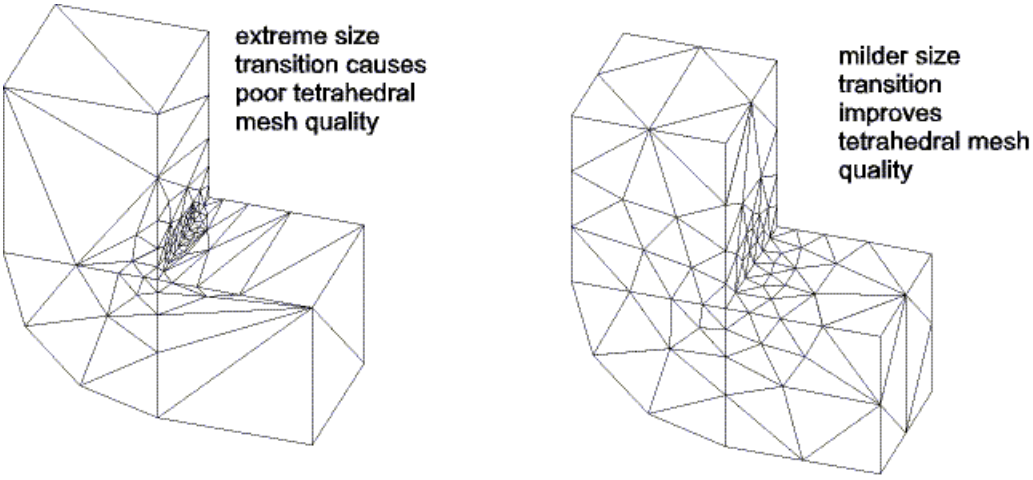
Flattened regions or regions having excessively sharp corners: Areas or volumes that are flattened or have a sharp interior corner can commonly experience a meshing failure.

Figure 7.40: Avoiding Sharp Corners



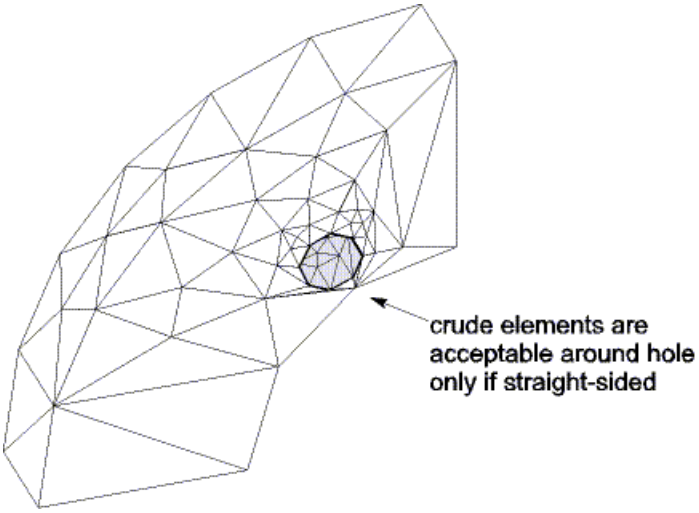
Extreme element size transition: Poor element quality will often occur if you specify too extreme a transition in element sizes.

Figure 7.41: Avoiding Extreme Element Size Transitions



Excessive element curvature: When using midside-node structural elements to model a curved boundary, you should usually make sure that you make your mesh dense enough that no single element spans more than 15° of arc per element length. If you do not need detailed stress results in the vicinity of a curved boundary, you can force the creation of straight-sided elements [**MSHMID**,1] in a coarse mesh along curved edges and faces. In cases where a curved-sided element will create an inverted element, the tetrahedra mesher automatically changes it to a straight-sided element and outputs a warning.

Figure 7.42: Use of MSHMID,1 to Force Straight-Sided Elements



Tetrahedron meshing failure: A tetrahedron meshing failure can be quite time-consuming. One relatively quick way that you can make a preliminary check for a possible tetrahedron meshing failure is by meshing the surfaces of a volume with 6-node triangles. If this surface triangle mesh contains no sudden size transitions (admittedly often a difficult judgement) and produces no curvature or aspect ratio warnings, tetrahedron meshing failure is much less likely than if these conditions are not met. (Be sure to clear or deactivate the triangle elements before using the analysis model.)

Subtracting meshed entities: Avoid subtracting meshed entities from your model whenever possible. However, if you subtract entities that have been meshed and an undesirable mesh mismatch results, you can recover by clearing the mesh and remeshing the entities.

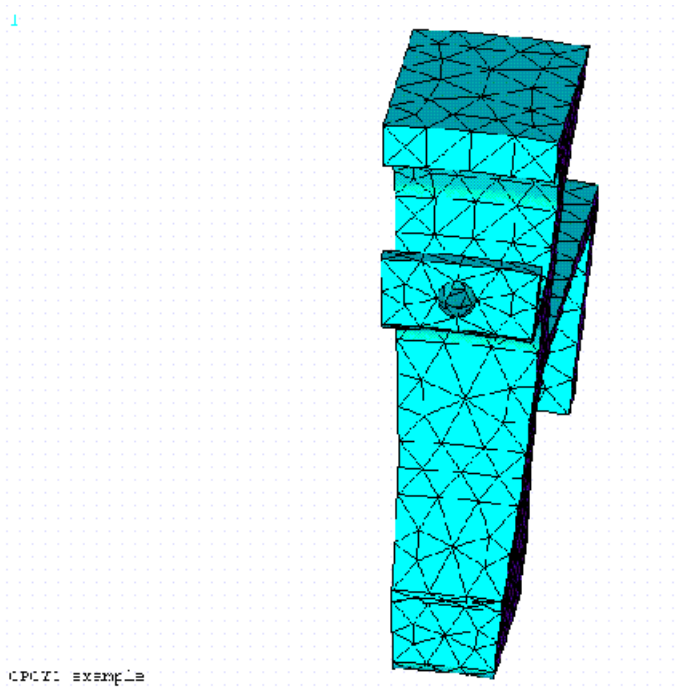
High- and low-order elements in adjacent entities: When you use high- and low-order elements in adjacent entities (for example, high-order elements in one volume and low-order elements in an adjacent volume, or high-order elements in an area and low-order elements in a volume attached to the area), ANSYS, Inc. recommends that you perform the low-order mesh first, followed by the high-order mesh. Reversing this order can cause database corruption when the entity containing the high-order mesh is cleared without first clearing the low-order mesh.

7.8. Using CPCYC and MSHCOPY Commands

7.8.1. CPCYC Example

The cyclic sector model shown below represents 1/20 (18° of arc) of a disk. For loadings which are the same on every sector, we can obtain a correct solution by coupling the degrees of freedom on the "low" sector boundary and the corresponding DOF on the "high" boundary. *Cyclic sector models are typically referred to the global cylindrical coordinate system. Moving from the "low" boundary in the $+\theta$ direction passes through the sector toward the "high" boundary. Cyclic coupling requires identical node and element patterns on the low and high sector boundaries. The* ***MSHCOPY*** *operation allows convenient generation of such a matching mesh.*

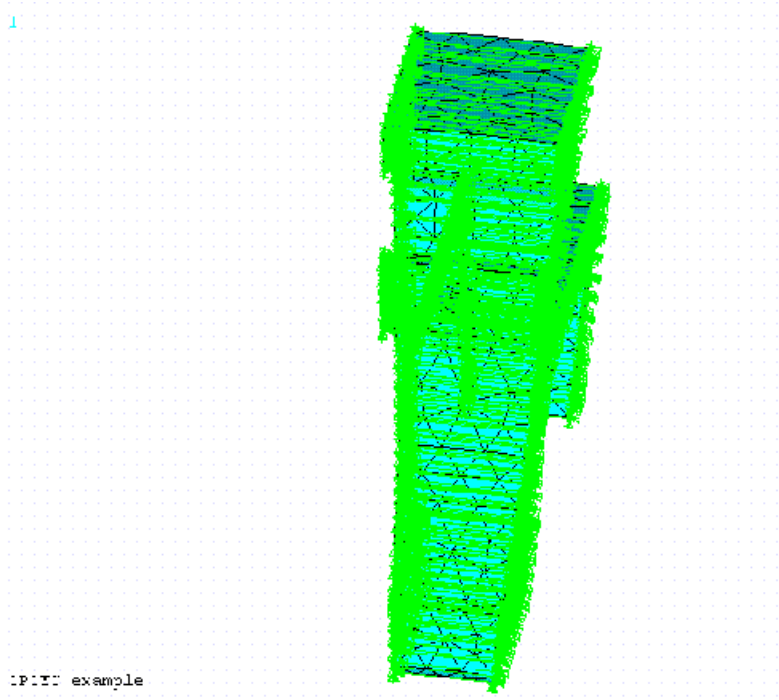
Figure 7.43: Prior to Coupling



7.8.2. CPCYC Results

The **CPCYC** operation will couple the DOF on the low boundary to the corresponding DOF on the high boundary. It will also rotate the nodal DOF into the cylindrical system you use to specify the angle spanned by the sector. (This is needed to couple radial deflection to radial deflection, axial to axial, and circumferential to circumferential.)

Figure 7.44: After Coupling



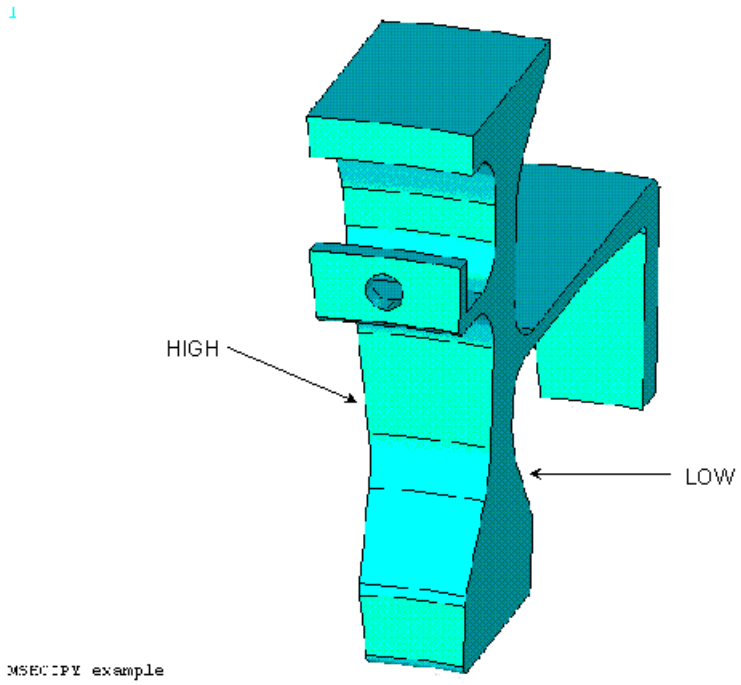
7.8.3. MSHCOPY Example

The cyclic sector model shown below represents 1/20 (18° of arc) of a disk.

If loads are symmetric about the cut planes, the left and right faces can be treated as symmetry surfaces. In such cases, equally good results could be obtained by slicing the model in half through the center of the hole, including only 9° of arc.

For more complex loadings, or for cyclic sector models having non-flat cut surfaces, symmetry boundary conditions are not correct, and may produce misleading results. There are several approaches to obtaining correct solutions for such cases, all of which require establishing a relationship between solution degrees of freedom on the "low" sector boundary and corresponding DOF on the "high" boundary. (Cyclic sector models are typically referred to the global cylindrical coordinate system. Moving from the "low" boundary in the $+\theta$ direction passes through the sector toward the "high" boundary. For example, see **CPCYC**.) *These methods all require identical node and element patterns on the low and high sector boundaries.* The **MSHCOPY** operation allows convenient generation of such a matching mesh.

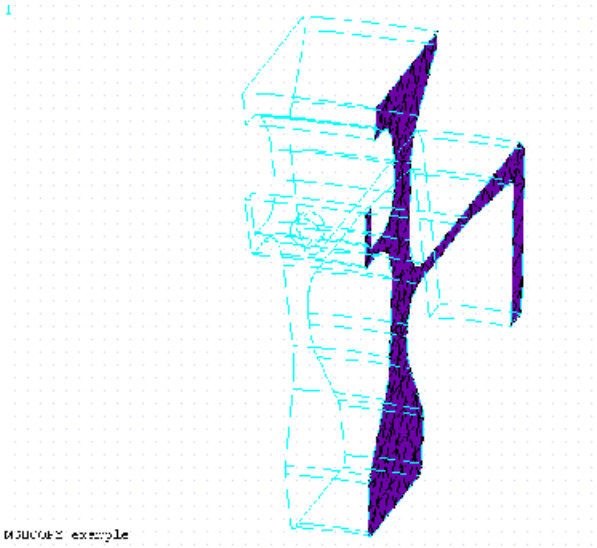
Figure 7.45: Illustration of High and Low Boundary Sector Boundaries



7.8.4. Low Sector Boundary

The first step is to mesh the low sector boundary areas. MESH200 is a good choice.

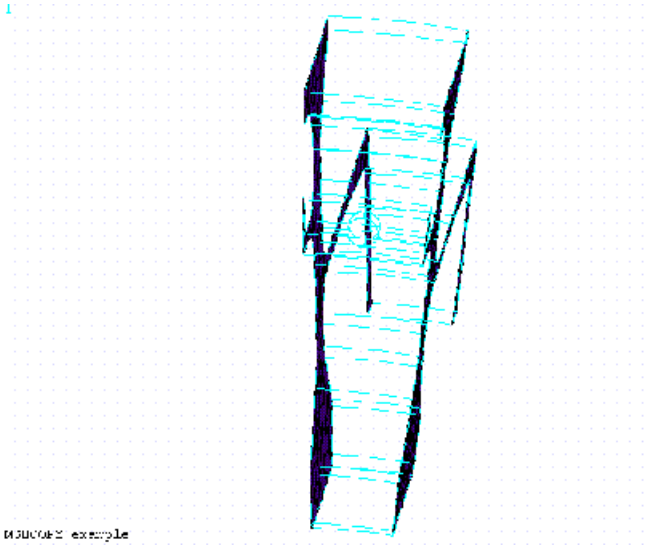
Figure 7.46: Illustration of Low Sector Boundary



7.8.5. Area Elements from MSHCOPY and AMESH

The second step is to use **MSHCOPY** to duplicate the low sector boundary elements onto the high sector boundary. If you intend to perform a modal cyclic symmetry analysis, be sure to allow **MSHCOPY** to create the low and high node components.

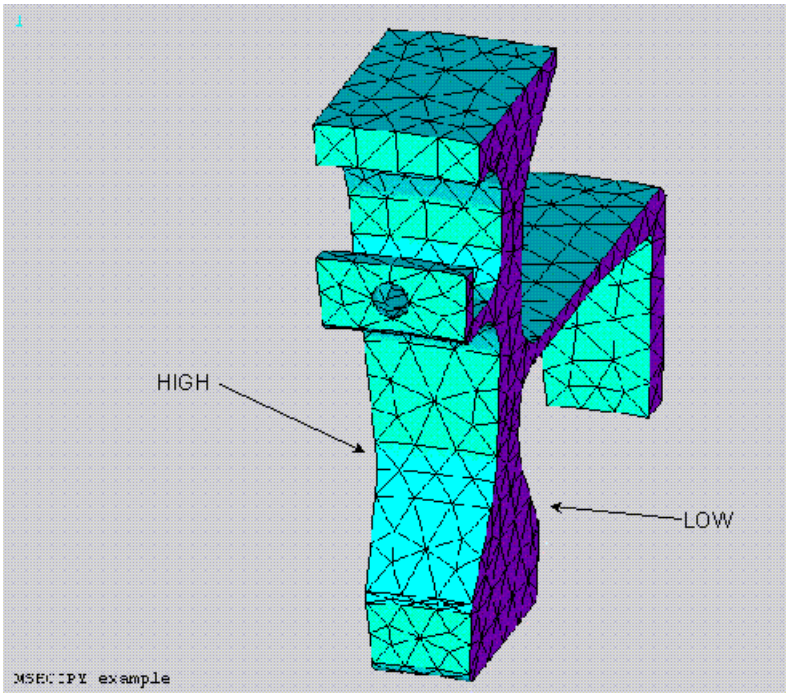
Figure 7.47: Low and High Sector Boundaries Selected



7.8.6. Meshing the Sector Volume(s)

The third step is to mesh the sector volume(s). The volume elements will use the area element connectivity already in place on the sector boundaries.

Figure 7.48: Illustration of Meshed Volume(s)



The final step is to make sure that the area elements do not affect the solution, by one of the following methods:

– Use MESH200 elements to mesh the sector boundaries. They have no DOF, and cannot affect a solution, or

– Clear all area elements after creating the volume mesh, or

– Unselect all area elements before solution. This method will cause a warning at solution about the unselected elements. You also run the risk that someone may inadvertently activate them during a later solution using the same analysis model.

Chapter 8: Revising Your Model

ANSYS offers various methods for revising and refining your model. The following topics are available:

8.1. Refining a Mesh Locally

There are generally two situations in which you may want to refine a mesh in a local region:

- You have meshed a model and you would like a finer mesh in specific regions of the model, or
- You have completed the analysis and, based on the results, would like a more detailed solution in a region of interest.

For all area meshes and for volume meshes composed of tetrahedra, the ANSYS program allows you to refine the mesh locally around specified nodes, elements, keypoints, lines, or areas. Meshes composed of volume elements other than tetrahedra (for example, hexahedra, wedges, and pyramids) cannot be locally refined.

The following mesh-refinement topics are available:

8.1.1. How to Refine a Mesh

You must follow these two steps to refine a mesh:

1. Select the entity (or set of entities) around which refinement will be done.
2. Specify the level of refinement to be done (in other words, the desired size of the elements in the refinement region, relative to the size of the original elements). The refined elements will always be smaller than the original elements; the local mesh refinement process does not provide mesh coarsening (*LEVEL*).

8.1.1.1. Advanced Controls

If you prefer to have more control over the refinement process, you can also set values for any of the following advanced options:

- You can specify the depth of the mesh refinement region in terms of the number of elements outward from the indicated entities (*DEPTH*).

- You can specify the type of postprocessing to be done after the original elements are split. Postprocessing may include both smoothing and cleaning, smoothing only, or neither one (*POST*).
- You can specify whether triangles can be introduced into the mesh during the refinement of an otherwise all-quadrilateral mesh. In other words, you can indicate whether quadrilateral elements must be retained (*RETAIN*).

8.1.2. Refinement Commands and Menu Paths

Use the following **xREFINE** commands and menu paths to select entities for refinement and to set refinement controls. (The refinement controls are described in detail below.)

- To refine around a selected set of nodes, use one of the following methods:
  Command(s): NREFINE
  GUI: Main Menu> Preprocessor> Meshing> Modify Mesh> Refine At> Nodes
- To refine around a selected set of elements, use one of the following methods:
  Command(s): EREFINE
  GUI: Main Menu> Preprocessor> Meshing> Modify Mesh> Refine At> Elements
  Main Menu> Preprocessor> Meshing> Modify Mesh> Refine At> All
- To refine around a selected set of keypoints, use one of the following methods:
  Command(s): KREFINE
  GUI: Main Menu> Preprocessor> Meshing> Modify Mesh> Refine At> Keypoints
- To refine around a selected set of lines, use one of the following methods:
  Command(s): LREFINE
  GUI: Main Menu> Preprocessor> Meshing> Modify Mesh> Refine At> Lines
- To refine around a selected set of areas, use one of the following methods:
  Command(s): AREFINE
  GUI: Main Menu> Preprocessor> Meshing> Modify Mesh> Refine At> Areas

Figure 8.1: Examples of Local Mesh Refinement (p. 183) shows examples of mesh refinement around a node [**NREFINE**], element [**EREFINE**], keypoint [**KREFINE**], and line [**LREFINE**].

Figure 8.1: Examples of Local Mesh Refinement

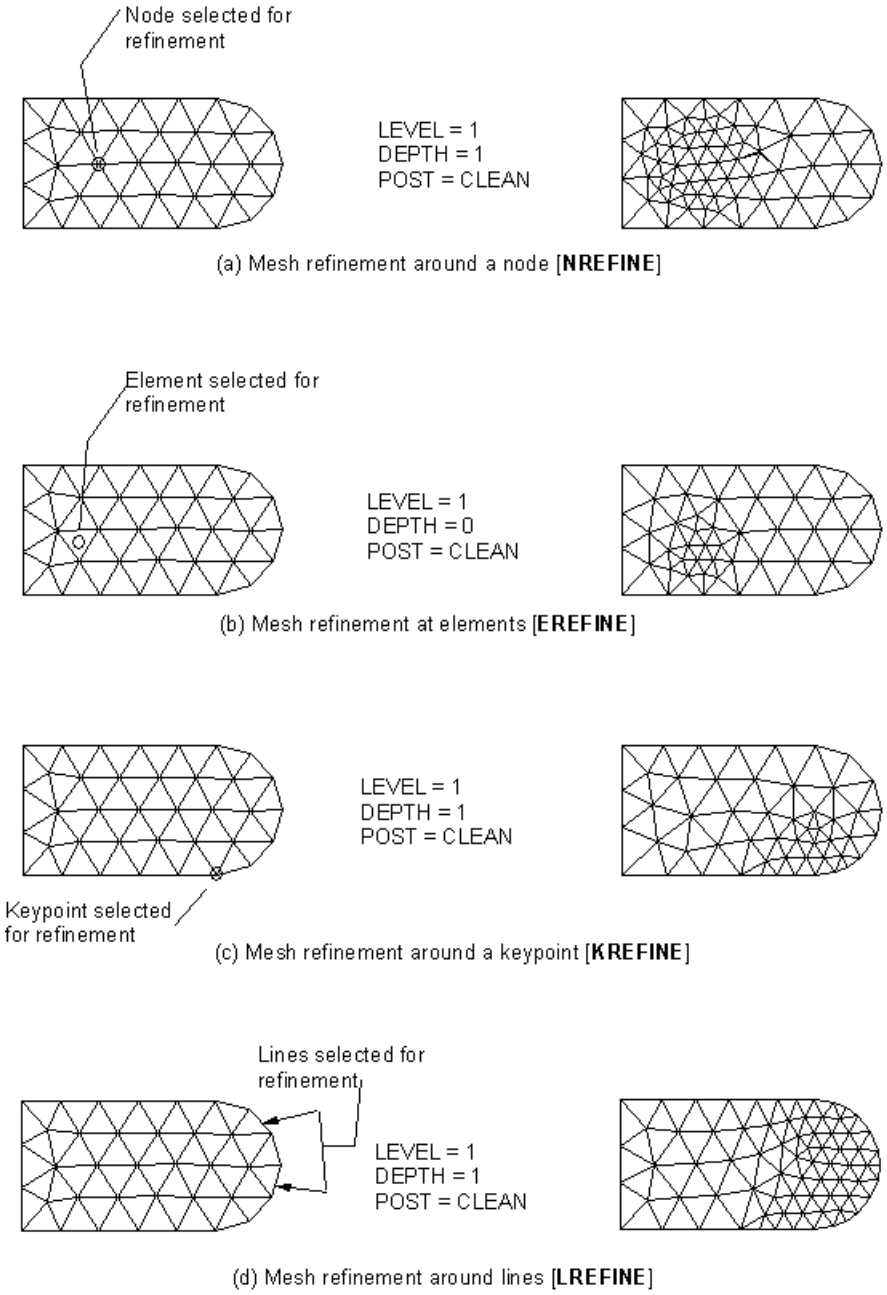
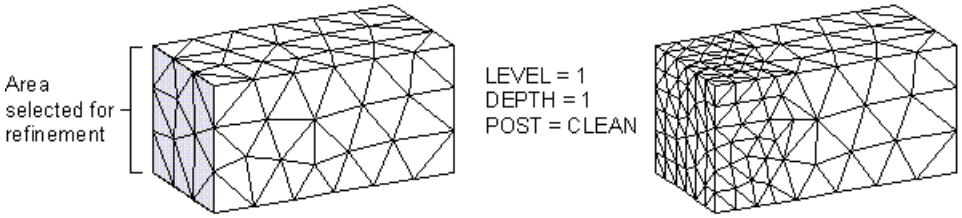


Figure 8.2: Tetrahedral Mesh Refinement Around an Area (p. 183) illustrates the use of the **AREFINE** command to perform tetrahedral mesh refinement around an area.

Figure 8.2: Tetrahedral Mesh Refinement Around an Area



8.1.2.1. Specifying the Level of Refinement

Use the *LEVEL* argument to specify how much refinement is to be done. The value of *LEVEL* must be an integer from 1 to 5, where a value of 1 provides minimal refinement, and a value of 5 provides maximum refinement. When *LEVEL* = 1, the resulting element edges in the refined region are approximately 1/2 the original edge lengths; when *LEVEL* = 5, the resulting element edges are approximately 1/9 the original edge lengths. The table below lists all of the possible settings of *LEVEL*, along with the approximate resulting edge length for each setting.

Value of LEVEL Argument	Approximate Edge Length
1	1/2
2	1/3
3	1/4
4	1/8
5	1/9

Values of *LEVEL* from 1 to 5 attempt to provide gradually decreasing element edge lengths. However, be aware that when *RETAIN* = ON, different values of *LEVEL* may provide the same refinement. (For more information, see the explanation of the *RETAIN* argument below.) Elements in the layer just outside of the refinement region (that is, beyond the specified *DEPTH*) may also be split in order to transition to the refinement elements.

Note

All values of *LEVEL* result in smaller elements in the refinement region. The local mesh refinement process does not provide mesh coarsening.

8.1.2.2. Specifying the Depth of Refinement

By default, the mesh is refined one element outward from the chosen entities (except for element refinement, which uses *DEPTH* = 0 as the default), and the elements are split one time (that is, the element edges are divided in half, since *LEVEL* = 1 by default).

8.1.2.3. Specifying Postprocessing for the Refinement Region: Smoothing and Cleanup

As part of the refinement process, you can specify the type of postprocessing ANSYS does after the original elements are split. You can choose smoothing and cleaning (the default), smoothing only, or neither one.

- If you want ANSYS to do smoothing and cleaning, set *POST* = CLEAN (or choose Cleanup & Smooth in the GUI).
- If you want ANSYS to do only smoothing, set *POST* = SMOOTH (or choose Smooth in the GUI).
- If you do not want ANSYS to do either type of postprocessing, set *POST* = OFF (or choose Off in the GUI).

Smoothing: By default, nodes in the refinement region are smoothed (that is, their locations are adjusted) to improve the element shapes. Node locations will be adjusted subject to the following constraints:

- Nodes at keypoints will not move.

- Nodes on lines will move only on the line.
- Nodes within areas will remain on the surface.
- If the mesh has been detached from the solid model (**MODMSH**,DETACH or menu path **Main Menu> Preprocessor> Checking Ctrls> Model Checking**), smoothing will not be done.

You can turn node smoothing off for all nodes by setting *POST* = OFF for the refinement command being used. (Doing so turns cleanup off as well.)

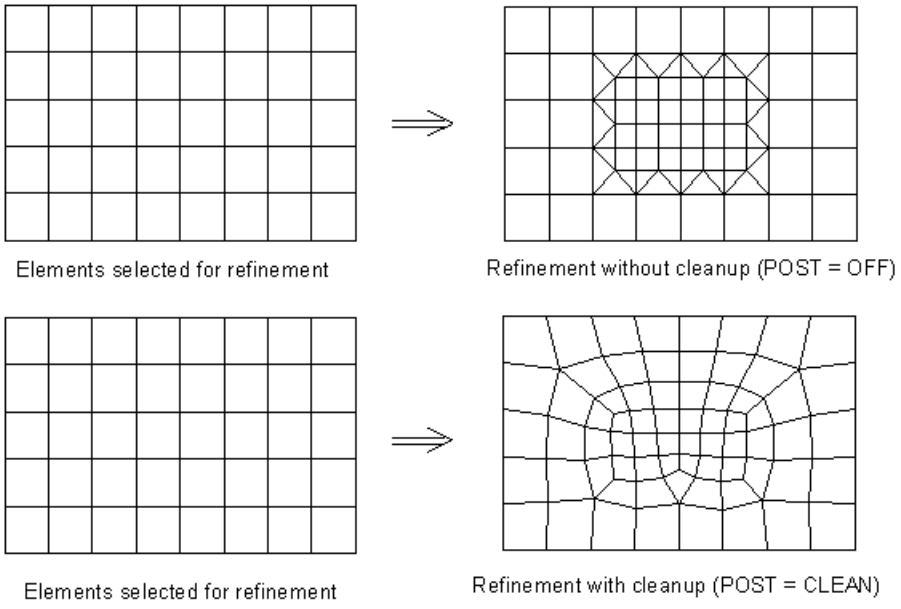
Cleanup: When cleanup is turned on (*POST* = CLEAN), the ANSYS program performs cleanup operations (in 2-D models) on all of the elements that are associated with any affected geometric entities. In 3-D models, ANSYS performs cleanup only on those elements that are in or directly adjacent to the refinement region. Cleanup operations improve the quality of the elements. If the mesh has been detached from the solid model, (**MODMSH**,DETACH or menu path **Main Menu> Preprocessor> Checking Ctrls> Model Checking**), cleanup will not be done on area meshes, but it will be done on tetrahedral volume meshes.

When you are refining a quadrilateral mesh, cleanup operations attempt to eliminate triangles from the refinement transition region. If poorly-shaped quadrilateral elements remain after the cleanup operations have optimized element quality, ANSYS splits the elements into triangles. You can prevent this splitting of quadrilateral elements by setting *RETAIN* = ON (the default). Figure 8.3: All-Quadrilateral Mesh (p. 185) illustrates the cleanup of an all-quadrilateral mesh.

Note

You can turn cleanup off by setting *POST* = OFF or *POST* = SMOOTH for the refinement command being used.

Figure 8.3: All-Quadrilateral Mesh



8.1.2.4. Specifying Whether Quadrilateral Elements Should Be Retained

Note

The ANSYS program ignores the *RETAIN* argument when you are refining anything other than a quadrilateral mesh.

By default, *RETAIN* = ON, which means that the refinement process will not introduce triangular elements into an otherwise all-quadrilateral mesh. When *RETAIN* = OFF and *POST* = SMOOTH or OFF, the resulting refinement region may contain triangles in order to maintain transitioning. When *RETAIN* = OFF and *POST* = CLEAN, triangles are minimized; however, they may not be completely eliminated - a minimal number of triangles may remain in the transition region in order to provide good element quality.

Note

If an area is meshed with a mixture of quadrilateral and triangular elements, the quadrilateral elements will not be maintained in the refinement region even when *RETAIN* = ON.

Since quadrilateral meshes are much more constrained than triangular meshes, increasing or decreasing the value of the *LEVEL* argument may not provide the desired increase or decrease in the level of refinement when *RETAIN* = ON. In addition, even when quadrilateral elements can be maintained, the quality of some of them may be poor, especially with a higher value of *LEVEL*. By setting *RETAIN* = OFF, however, some triangles may be introduced into the mesh. This may not be desirable, especially if you are using lower-order elements. You can keep these triangles out of the point of interest by doing one of the following:

- Refine with a larger *DEPTH*; that is, at a larger radius from the point of interest.
- Refine with *POST* = CLEAN. This setting for the *POST* argument minimizes the number of triangles as much as possible.
- Refine using another method (for example, use local mesh controls and remesh).

8.1.3. Transfer of Attributes and Loads

Element attributes associated with the "parent" element are automatically transferred to all of the "child" elements. These attributes include element type, material properties, real constants, and element coordinate systems (see Generating the Mesh (p. 101) for details on element attributes).

Loads and boundary conditions applied to the solid model are transferred to nodes and elements when the solution is initiated (or when loads are manually transferred with the **SBCTRAN** or **DTRAN** commands). Therefore, solid model loads will be correctly applied to the new nodes and elements created during refinement. However, loads and boundary conditions applied at the node and element level (finite element loads) cannot be transferred to new nodes and elements created during refinement. If you have such loads in a region selected for refinement, the program will not allow refinement to take place

unless the loads are first deleted. Therefore, it is recommended that you apply loads only to the solid model rather than directly to nodes and elements if you anticipate using mesh refinement.

Note

Since solid model loading is not applicable for an explicit dynamics analysis (that is, the ANSYS LS-DYNA product), mesh refinement must be performed before the application of loads in this type of analysis.

8.1.4. Other Characteristics of Mesh Refinement

Other characteristics of mesh refinement include the following:

- The new elements and nodes created during refinement (including midside nodes) are projected to the underlying solid model geometry. (See Figure 8.4: Nodes and Elements are Projected to Underlying Geometry (p. 188).)
- When using the option to refine around nodes [**NREFINE**], midside nodes that are included in the selected set are ignored.
- Mesh refinement will not cross area and volume boundaries. That is, if the specified *DEPTH* goes beyond the edges of the meshed area or volume, the adjacent area and/or volume meshes won't change (see Figure 8.5: Mesh Refinement Will Not Cross Area Boundaries (p. 188)). However, if an entity picked for refinement (node, element, keypoint, or line) is *on* the boundary, or entities are picked on both sides of the boundary, then refinement will extend into the adjacent area or volume.
- Mesh refinement will take place only in elements that are currently selected (see Figure 8.6: Only Selected Elements are Refined (p. 188)).
- Refinement can be used on a mesh that has been detached from the solid model (**MODMSH**,DETACH or menu path **Main Menu> Preprocessor> Checking Ctrls> Model Checking**). In this case, the refinement will not be stopped by area boundaries. Also, the nodes and elements will not be projected to the solid model and none of the postprocessing specified with the *POST* argument will be done.
- When cleanup is turned on (*POST* = CLEAN) during the refinement of a tetrahedral mesh, ANSYS automatically performs a high level of cleanup operations (that is, a level equivalent to **VIMP**,,,2) in the refinement region only. If you receive shape error messages during refinement, turn off shape checking [**SHPP**,OFF], perform refinement again [**xREFINE**], and then request tetrahedral element improvement at the highest level [**VIMP**,,,3].
- If you use the **LESIZE** command to specify divisions for lines and those lines are later affected by the refinement process, ANSYS will change the line divisions for the affected lines (that is, the numbers of divisions on the lines not only increase, but also show up as *negative* numbers in a subsequent line listing [**LLIST**]).

Note

The presence of the negative sign affects how ANSYS treats the line divisions if you clear the mesh later (**ACLEAR**, **VCLEAR**, etc., or menu path **Main Menu> Preprocessor> Meshing> Clear> *entity***). If the number of line divisions is positive, ANSYS does not remove the line divisions during the clearing operation; if the number is negative, ANSYS removes the line divisions (which will then show up as zeros in a subsequent line listing).

Figure 8.4: Nodes and Elements are Projected to Underlying Geometry

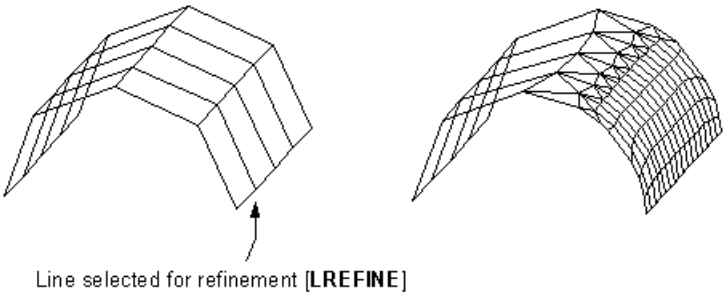


Figure 8.5: Mesh Refinement Will Not Cross Area Boundaries

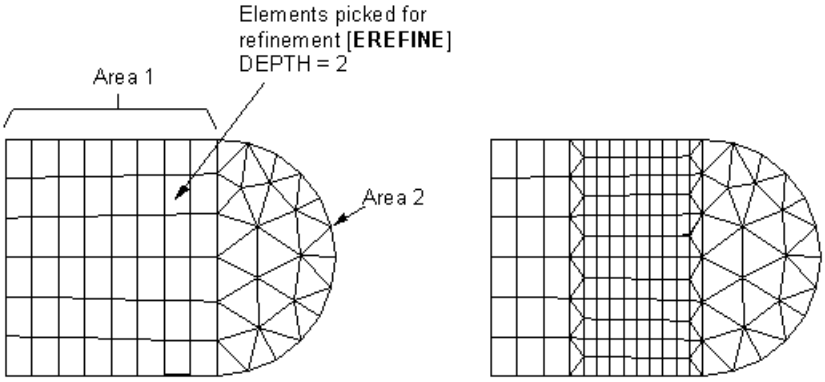
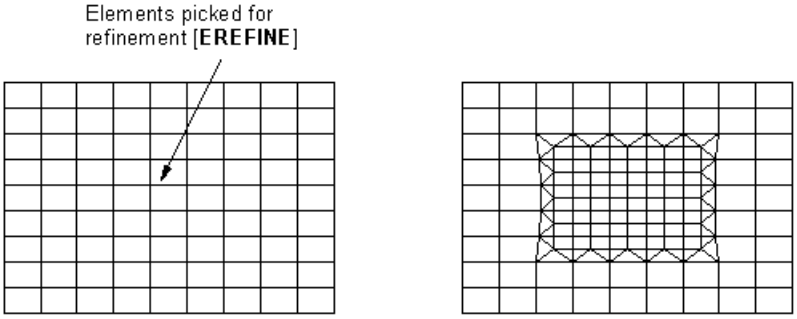
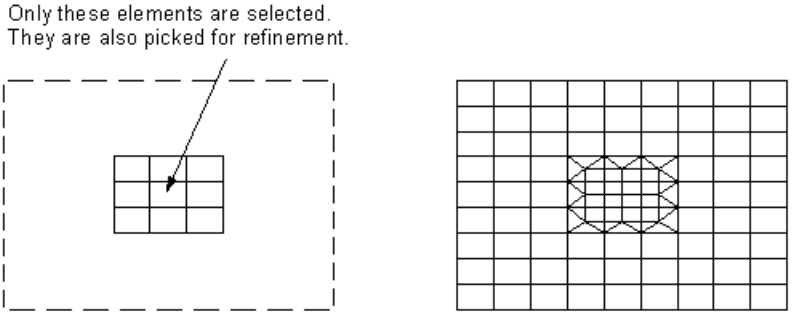


Figure 8.6: Only Selected Elements are Refined



(a) Refining with all elements selected



(b) Refining a selected subset of elements

8.1.5. Restrictions on Mesh Refinement

The following restrictions apply to mesh refinement:

- Although local refinement can be used for all area meshes, it can be used for only those volume meshes that are composed of tetrahedra. Meshes composed of volume elements other than tetrahedra (for example, hexahedra, wedges, and pyramids) cannot be locally refined.

- If the model contains contact elements in the region selected to refine, you cannot use local mesh refinement. In this case, refine the mesh before defining contact elements (or, delete the contact elements, refine the mesh, then reapply the contact elements).

- Local mesh refinement does not support elements that are generated on the free faces of existing elements [**ESURF**]. Instead of using refinement on these elements, delete the surface elements, refine the underlying elements, and then generate the surface elements again.

- If beam elements exist adjacent to the refinement region, refinement cannot be done. In order to refine in this area, the beam elements should be deleted and redefined after refinement is performed.

- If loads are applied directly to the nodes and elements in a model, refinement cannot be done. In this case, you must delete the loads in order to refine the mesh. (To avoid this situation, it is recommended that you apply loads to the solid model instead of the finite element model.)

- Local mesh refinement cannot be done if initial conditions at nodes [**IC**], coupled nodes [**CP** family of commands], or constraint equations [**CE** family of commands] exist in the model. If any of these exist, you must remove them before refinement can be done.

- Local mesh refinement is not recommended for an explicit dynamics model (that is, when using the ANSYS LS-DYNA product) since the small elements resulting from refinement may dramatically reduce the time step size.

- The **KSCON** command is not supported. For any area that was meshed with the **KSCON** command, midside nodes will be placed at the middle of the edges when refinement is done.

- If element or node components are defined, you will be asked whether you want to proceed with the refinement. If you choose to proceed, you must update any affected components.

8.2. Moving and Copying Nodes and Elements

In the normal solid modeling procedure, you will ordinarily complete your entire solid model before generating your finite element mesh. However, if repetitive geometric features appear in your model, you might sometimes find it more efficient to model and mesh just one representative portion of your model, and then copy that meshed region as many times as needed to complete the model. (It is usually much less expensive to copy an existing mesh than to generate a new mesh.) A certain amount of forethought is generally required if you are to accomplish this procedure successfully.

The general procedure for copying a meshed region is to use the commands for generating and transferring areas and volumes, which are described below. When a meshed solid model entity is copied using one of these commands, all the attached lower-order entities, as well as the node and element mesh, are copied along with that entity.

- To generate additional areas from a pattern of areas, use one of the following methods:
  Command(s): AGEN
  GUI: Main Menu> Preprocessor> Modeling> Copy> Areas
  Main Menu> Preprocessor> Modeling> Move/Modify> Areas> Areas

- To generate additional volumes from a pattern of volumes, use one of the following methods:
  Command(s): VGEN

GUI: Main Menu> Preprocessor> Modeling> Copy> Volumes
Main Menu> Preprocessor> Modeling> Move/Modify> Volumes

- To generate areas from an area pattern by symmetry reflection, use one of the following methods:
  Command(s): ARSYM
  GUI: Main Menu> Preprocessor> Modeling> Reflect> Areas

- To generate volumes from a volume pattern by symmetry reflection, use one of the following methods:
  Command(s): VSYMM
  GUI: Main Menu> Preprocessor> Modeling> Reflect> Volumes

- To transfer a pattern of areas to another coordinate system, use one of the following methods:
  Command(s): ATRAN
  GUI: Main Menu> Preprocessor> Modeling> Move/Modify> Transfer Coord> Areas

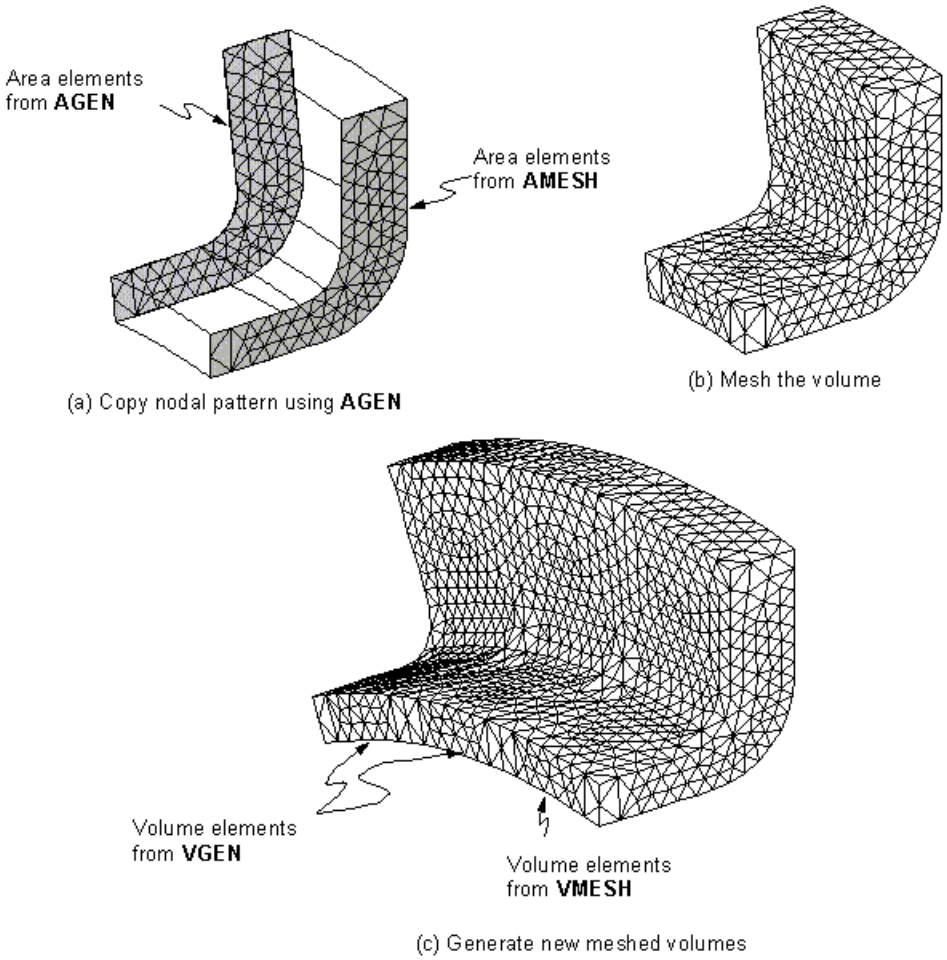
- To transfer a pattern of volumes to another coordinate system, use one of the following methods:
  Command(s): VTRAN
  GUI: Main Menu> Preprocessor> Modeling> Move/Modify> Transfer Coord> Volumes

You must plan ahead to ensure that the interfaces between copied regions will match up node for node. For example, if you freely meshed a volume, the pattern of nodes on the right end would not necessarily match the pattern of nodes on the left end. If the original part and its copy were to be joined such that the right end of one part interfaced with the left end of the other part, a seam of discontinuity would be created where the two mismatched faces touched.

It is relatively easy to create matching node patterns along the line edges of meshed areas: simply specify the same number of line divisions and division spacings on both sides of the original part. Volumes are not so straightforward, however. You will need to use a trick to force matching node patterns on two faces of a meshed volume. Before meshing with volume elements, mesh one of the matching faces with dummy area elements, then copy that meshed area to the other matching face. (Depending on how you originally created your volume, you might or might not have some cleaning up to do at this point. If you wind up with duplicate coincident areas, you should redefine your volume in terms of the new meshed area, and delete the original volume.) The volume can then be meshed with solid elements. After the volume meshing is complete, you should delete the dummy area elements. (You can do this fairly cleanly using selecting and either the **ACLEAR** command or menu path **Main Menu> Preprocessor> Meshing> Clear> Areas**.)

Having created meshed regions which will match up at their interfaces, you can now copy the part, such that the repeated regions just touch. Even though these regions will have matching nodes at the interfaces, the degrees of freedom at these nodes will remain independent; that is, a seam of discontinuity will exist in your model at the interface. You should execute **NUMMRG**,ALL to eliminate this discontinuity. It is usually good practice to follow these operations with a **NUMCMP** command (**Main Menu> Preprocessor> Numbering Ctrls> Compress Numbers**).

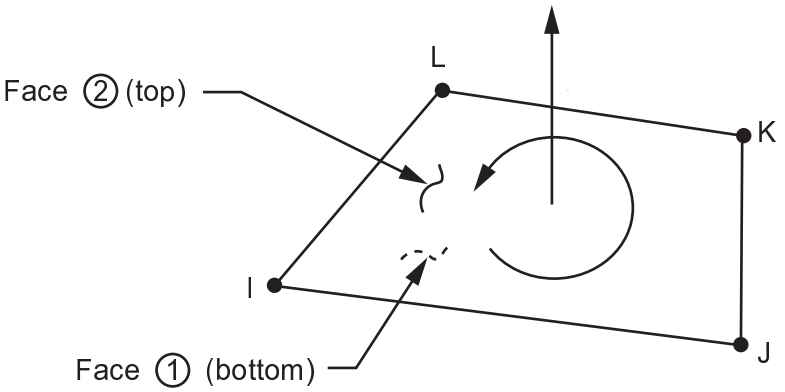
Figure 8.7: Generating Meshed Volumes With Matching Node Patterns at Interfaces



8.3. Keeping Track of Element Faces and Orientations

If your model contains shell elements, and if you apply surface loads, you will need to keep track of the element faces in order to be able to define the proper direction for your loads. In general, shell surface loads will be applied to element face one, and will be positive in accordance with the right-hand rule (following the I, J, K, L nodal sequence, as illustrated below). If you create your shell elements by meshing a solid model area, the normal direction of the elements will be consistent with the normal direction of the area (the area's normal direction can be determined by issuing the **ALIST** command or by executing menu path **Utility Menu> List> Areas**; the direction of the sequence of lines defining that area will define the normal direction by the right-hand rule).

Figure 8.8: Positive Normal Direction as Defined by the Right-Hand Rule



There are several ways that you can conduct graphical checks:

- You can conduct a quick graphical check of the positive normal direction for shell elements by issuing a **/NORMAL** command (**Utility Menu> PlotCtrls> Style> Shell Normals**), followed by an **EPLOT** command (**Utility Menu> Plot> Elements**).
- You can turn PowerGraphics ON. PowerGraphics displays the "top" and "bottom" of shells with different colors.
- You can apply your surface loads with the assumed correct sign, and then verify their direction by turning on the surface load symbol [**/PSF**,Item,Comp,2] before executing **EPLOT**.

8.3.1. Controlling Area, Line, and Element Normals

Inconsistent normal directions can lead to problems in your model. For example, if adjacent shell elements have inconsistent normal directions, you could encounter difficulties in postprocessing of stress and strain results. Clearly, if a surface of your model contains *both* top and bottom shell element faces, averaged nodal stresses and strains could be incorrect. However, PowerGraphics [**/GRAPHICS**,POWER] accounts for the mismatched normal directions and can produce proper nodal stress plots. (PowerGraphics is the default when the GUI is on.)

ANSYS provides various tools that you can use to control area, line, and element normals:
Command(s): ENORM, **ANORM**, **ENSYM**, **LREVERSE**, **AREVERSE**
GUI: Main Menu> Preprocessor> Modeling> Move/Modify> Elements> Shell Normals
Main Menu> Preprocessor> Modeling> Move/Modify> Areas> Area Normals
Main Menu> Preprocessor> Modeling> Move/Modify> Reverse Normals> of Shell Elems
Main Menu> Preprocessor> Modeling> Move/Modify> Reverse Normals> of Lines
Main Menu> Preprocessor> Modeling> Move/Modify> Reverse Normals> of Areas

The sections that follow describe how you can use these tools to:

- Reorient shell element normals so that they are consistent with that of a specified element [**ENORM**]
- Reorient area normals so that they are consistent with that of a specified area [**ANORM**]
- Reverse the normals of existing shell elements [**ENSYM**]
- Reverse the normal of a line [**LREVERSE**]

- Reverse the normal of an area [**AREVERSE**]

Note

You cannot use the tools described in this section to change the normal direction of any element that has a body or surface load. We recommend that you apply all of your loads only *after* ensuring that the element normal directions are acceptable.

Caution

Real constants (such as nonuniform shell thickness and tapered beam constants) may be invalidated by an element reversal.

8.3.1.1. Reorienting Shell Element Normals

If you find that your elements have inconsistent positive normal directions, you can reorient them uniformly to match a specified element. (Element coordinate systems, if defined by the I, J, K nodes, might also be reoriented by this operation.)

To use the command method to reorient shell element normals, issue the command **ENORM**,*ENUM*:

- Use the *ENUM* argument to identify the number of the element having the normal direction that the reoriented elements are to match.

For example, the command **ENORM**,3 reorients the normals of all selected shell elements so that they are consistent with the normal direction of element number 3. See the description of the **ENORM** command in the *Command Reference* for details.

In the GUI, you can reorient shell element normals by choosing menu path **Main Menu> Preprocessor> Modeling> Move/Modify> Elements> Shell Normals**. When the Reorient Shell Normals picker appears, pick the element having the normal direction that the reoriented elements are to match and click on OK.

8.3.1.2. Reorienting Area Normals

If a group of areas has inconsistent normal directions, you can reorient them uniformly to match the normal direction of a specified area.

To use the command method to reorient area normals, issue the command **ANORM**,*ANUM*,*NOEFLIP*:

- Use the *ANUM* argument to identify the number of the area having the normal direction that the reoriented areas are to match.
- Use the *NOEFLIP* argument to indicate whether you want to change the normal direction of any existing elements on the reoriented area(s) so that they are consistent with the new normal direction. Specify 0 if you want to make the normals consistent; specify 1 if you do not.

For example, the command **ANORM**,5,0 reorients the normals of all selected areas so that they are consistent with the normal direction of area number 5, and also makes any area elements on the areas consistent with that normal direction. See the description of the **ANORM** command in the *Command Reference* for details.

In the GUI, you can reorient area normals by choosing menu path **Main Menu> Preprocessor> Modeling> Move/Modify> Areas> Area Normals**. When the Reorient Area Normals picker appears, pick the area having the normal direction that the reoriented areas are to match and click on OK in the picker. Then, in the Make area normals consistent dialog box, indicate whether you want any existing area elements to be consistent with the new normal direction and click on OK in the dialog box.

8.3.1.3. Reversing the Normals of Existing Shell Elements

To use the command method to reverse the normals of existing shell elements, issue the command **ENSYM**,,,,*IEL1*,*IEL2*,*IEINC*:

- Use the *IEL1*, *IEL2*, and *IEINC* arguments to reverse the normals of elements from *IEL1* to *IEL2* (defaults to *IEL1*) in steps of *IEINC* (defaults to 1).

For example, the command **ENSYM**,,,,1,50 reverses the normals of shell elements numbered 1 through 50.

In the GUI, you can reverse the normals of existing shell elements by choosing menu path **Main Menu> Preprocessor> Modeling> Move/Modify> Reverse Normals> of Shell Elems**. When the Reverse Shell Normals picker appears, pick the shell elements whose normals are to be reversed and click on OK.

8.3.1.4. Reversing the Normal of a Line

To use the command method to reverse the normal of a line, issue the command **LREVERSE**,*LNUM*,*NOEFLIP*:

- Use the *LNUM* argument to identify the number of the line whose normal is to be reversed.
- Use the *NOEFLIP* argument to indicate whether you want to change the normal direction of the existing elements on the line so that they are consistent with the reversed line's new normal direction. Specify 0 if you want to make the normals consistent; specify 1 if you do not.

For example, the command **LREVERSE**,1,1 reverses the direction of line 1, but does not make any line elements on the line consistent with the new direction. See the description of the **LREVERSE** command in the *Command Reference* for details.

In the GUI, you can reverse the normal of a line by choosing menu path **Main Menu> Preprocessor> Modeling> Move/Modify> Reverse Normals> of Lines**. When the Reverse Line Normals picker appears, pick the line whose normal is to be reversed and click on OK in the picker. Then, in the Make line normals consistent dialog box, indicate whether you want any existing line elements to be consistent with the new normal direction and click on OK in the dialog box.

8.3.1.5. Reversing the Normal of an Area

To use the command method to reverse the normal of an area, issue the command **AREVERSE**,*ANUM*,*NOEFLIP*:

- Use the *ANUM* argument to identify the number of the area whose normal is to be reversed.
- Use the *NOEFLIP* argument to indicate whether you want to change the normal direction of the existing elements on the area so that they are consistent with the reversed area's new normal direction. Specify 0 if you want to make the normals consistent; specify 1 if you do not.

For example, the command **AREVERSE**,7,0 reverses the normal direction of area 7 and makes any existing area elements on it consistent with the new normal direction. See the description of the **AREVERSE** command in the *Command Reference* for details.

In the GUI, you can reverse the normal of an area by choosing menu path **Main Menu> Preprocessor> Modeling> Move/Modify> Reverse Normals> of Areas**. When the Reverse Area Normals picker appears, pick the area whose normal is to be reversed and click on OK in the picker. Then, in the Reverse area normals dialog box, indicate whether you want any existing area elements to be consistent with the new normal direction and click on OK in the dialog box.

8.4. Revising a Meshed Model: Clearing and Deleting

Because of the solid modeling cross-reference checking that the ANSYS program performs, you cannot delete meshed solid model entities, nor can you use **EDELE** or **NDELE** to delete elements and nodes that are associated with solid model entities. In order to revise your model, you generally need to clear solid model entities of their meshes by using the mesh clearing commands. These clearing commands can be thought of as the inverse of the meshing commands. After clearing your model, you can proceed to modify your solid model as desired.

8.4.1. Clearing a Mesh

The mesh clearing commands delete the nodes and elements associated with the corresponding solid model entity. When you clear a higher level entity, all lower level entities will be automatically cleared, unless those lower entities are themselves meshed with elements. Nodes on the boundary of an entity shared by an adjoining meshed entity are *not* deleted as a result of clearing.

- To delete nodes and point elements associated with selected keypoints, use one of the following methods:
  Command(s): KCLEAR
  GUI: Main Menu> Preprocessor> Meshing> Clear> Keypoints

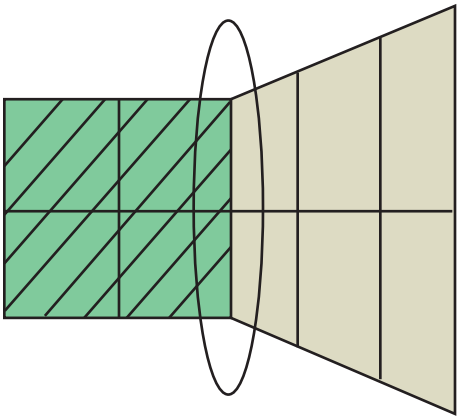
- To delete nodes and line elements associated with selected lines, use one of the following methods:
  Command(s): LCLEAR
  GUI: Main Menu> Preprocessor> Meshing> Clear> Lines

- To delete nodes and area elements associated with selected areas, use one of the following methods:
  Command(s): ACLEAR
  GUI: Main Menu> Preprocessor> Meshing> Clear> Areas

- To delete nodes and volume elements associated with selected volumes, use one of the following methods:
  Command(s): VCLEAR
  GUI: Main Menu> Preprocessor> Meshing> Clear> Volumes

The program will report how many of each kind of entity have been cleared after a mesh clearing operation. An entity is considered to have been "cleared" if either its elements or its nodes have been cleared.

Figure 8.9: Nodes at Boundary of Two Areas



If the elements/nodes being cleared are at the end of the element/node lists, then the next available element/node ID is reset accordingly. (You can suppress this resetting with a **MOPT**,CLEAR,OFF command.)

As was discussed earlier, element attributes that were assigned to the solid model by **TYPE**, **REAL**, **MAT**, and **ESYS** commands followed by a meshing command [**AMESH**, **VMESH**, etc.] will be cleared by a mesh clearing command. These "clearable" attributes are designated by *negative* attribute numbers in the output from listing commands [**ALIST**, **VLIST**, etc.]. The mesh clearing commands do not affect attributes that were assigned by an attribute association command [**AATT**, **VATT**, etc.]. In either case, issuing new attribute association commands will overwrite whatever element attributes were previously associated with the cleared solid model.

8.4.1.1. Modifying Element Attributes

There are several reasons why you might want to modify element attributes after meshing: you might have simply committed an error in assigning attributes, you might need to change your design, or you might be converting your model from one analysis discipline to another (such as in a sequential thermal-stress analysis). The techniques available for modifying element attributes include the following:

8.4.1.2. The Brute Force Method

You can clear your mesh using mesh clearing commands; set new attributes using either attribute association commands or commands such as **TYPE**, **REAL**, etc.; and then remesh using meshing commands. Because remeshing can sometimes be expensive, this approach should probably be avoided if the mesh itself is acceptable. Note what happens when a mesh clearing command is executed: solid model attributes set by a meshing command (identified by *negative* attribute numbers in listings produced by **ALIST**, **VLIST**, etc.) will be deleted; solid model attributes set by an attribute association command [**AATT**, **VATT**, etc.] will not be changed. Thus, because the attribute association commands override the **TYPE**, **REAL**, **MAT**, and **ESYS** commands, you will not be able to reassign solid model attributes with the **TYPE**, **REAL**, **MAT**, and **ESYS** commands if you initially assigned attributes by using attribute association commands. (You will need to issue a new attribute association command.) Upon remeshing, the attributes associated with the solid model entities will be assigned to the elements generated on those entities.

Direct Element Modification: Element attributes can also be modified without the expense of remeshing: you can select the elements to be modified; reset attributes (using, in this procedure, the **TYPE**, **REAL**, **MAT**, and **ESYS** commands); and execute **EMODIF**,ALL or menu path **Main Menu> Preprocessor> Modeling> Move/Modify> Elements> Modify Attrib**. This procedure modifies the element attributes directly, without affecting the corresponding solid model attributes. This procedure, although convenient, can be dangerous, because the element attributes in your finite element model will no

longer match the element attributes in your solid model. Also, you could conceivably change element attributes to inappropriate values, without receiving any kind of warning. For these reasons, you must proceed cautiously if you decide to attempt to change element attributes by direct element modification.

Another way of directly modifying the material number of specified elements is by using the **MPCHG** command or menu path **Main Menu> Preprocessor> Material Props> Change Mat Num**. (Unlike other element-modification commands, which are valid only within PREP7, **MPCHG** is valid within both PREP7 and SOLUTION. Thus, this command can be used to change element properties between solutions.)

Attribute Table Modification: Another possibility would be to change entries in the attribute tables after meshing, but before entering SOLUTION. A warning will be issued if the REAL set or the MAT set contain unused entries (such as could happen if a REAL property set for a beam were assigned to a spar element). No remeshing is required with this procedure.

A Note About Adding and Deleting Midside Nodes: For any of these procedures, if you change the element TYPE attribute to substitute midside-node elements for non-midside-node elements, you will also need to use one of the following methods to add the extra midside nodes as required:
Command(s): EMID
GUI: Main Menu> Preprocessor> Modeling> Move/Modify> Elements> Add Mid Nodes

EMID must be preceded by execution of a **MODMSH**,DETACH command or menu path **Main Menu> Preprocessor> Checking Ctrls> Model Checking**. Also, in order to delete midside nodes, you must first remove them from the midside-node elements by issuing **EMID**,-1.

8.4.2. Deleting Solid Model Entities

You can delete solid model entities with the entity deletion commands described below. Lower level entities cannot be independently deleted if they are attached to a higher level entity. Thus, if you have created a block using a geometric primitive command, you cannot selectively delete a keypoint that is associated with that block, unless you first delete, in descending hierarchical order, all the higher level entities (lines, areas, and volumes) that are attached to that keypoint.

- To delete unmeshed areas, use one of these methods:
  Command(s): ADELE
  GUI: Main Menu> Preprocessor> Modeling> Delete> Area and Below
  Main Menu> Preprocessor> Modeling> Delete> Areas Only

- To delete unmeshed keypoints, use one of these methods:
  Command(s): KDELE
  GUI: Main Menu> Preprocessor> Modeling> Delete> Keypoints

- To delete unmeshed lines, use one of these methods:
  Command(s): LDELE
  GUI: Main Menu> Preprocessor> Modeling> Delete> Line and Below
  Main Menu> Preprocessor> Modeling> Delete> Lines Only

- To delete unmeshed volumes, use one of these methods:
  Command(s): VDELE
  GUI: Main Menu> Preprocessor> Modeling> Delete> Volume and Below
  Main Menu> Preprocessor> Modeling> Delete> Volumes Only

Conversely, by activating the "sweep" option (that is, setting *KSWP* = 1) on the **LDELE**, **ADELE**, or **VDELE** commands, you can direct the program to delete all the associated lower level entities automatically. (Such lower level entities will not be deleted if they are attached to another higher level entity, however.)

For example, if you decide to delete an unmeshed sphere volume, you can issue a single **VDELE** command, with *KSWP* set to 1, to delete the volume and all its associated areas, lines, and keypoints.

8.4.3. Modifying Solid Model Entities

You can modify the geometry of a solid model by changing the position of its keypoints using one of the following methods:
Command(s): KMODIF
GUI: Main Menu> Preprocessor> Modeling> Move/Modify> Keypoints> Set of KPs
Main Menu> Preprocessor> Modeling> Move/Modify> Keypoints> Single KP

Any meshed regions attached to modified keypoints will be automatically cleared of nodes and elements. All lines, areas, and volumes attached to the modified keypoint will then be automatically redefined using the active coordinate system.

Unmeshed solid model entities may also be redefined by reissuing the commands that originally defined them. For example, consider the following sequence, in which a second **K** command is used to modify a keypoint:

```
CSYS,0
K,1,5.0,6.0,7.0         ! Create KP 1 at X=5.0, Y=6.0, Z=7.0
CSYS,1
K,1,5.0,6.0,7.0         ! Redefine KP 1 at R=5.0,  θ=6.0, Z=7.0
```

Keypoint 1 could only be redefined in this way if it was not attached to any higher level entities. Lines, areas, and volumes can be similarly redefined, but only if they are not attached to any higher level entities.

You can modify unmeshed lines using the operations described below. These operations will also update attached unmeshed areas, even if these areas are attached to volumes.

- To divide a single line into two or more lines, use one of the following methods:
  Command(s): LDIV
  GUI: Main Menu> Preprocessor> Modeling> Operate> Booleans> Divide> Line into 2 Ln's
  Main Menu> Preprocessor> Modeling> Operate> Booleans> Divide> Line into N Ln's
  Main Menu> Preprocessor> Modeling> Operate> Booleans> Divide> Lines w/ Options

- To combine adjacent lines into one line, use one of the following methods:
  Command(s): LCOMB
  GUI: Main Menu> Preprocessor> Modeling> Operate> Booleans> Add> Lines

- To generate a fillet line between two intersecting lines, use one of the following methods:
  Command(s): LFILLT
  GUI: Main Menu> Preprocessor> Modeling> Create> Lines> Lines> Line Fillet

8.5. Understanding Solid Model Cross-Reference Checking

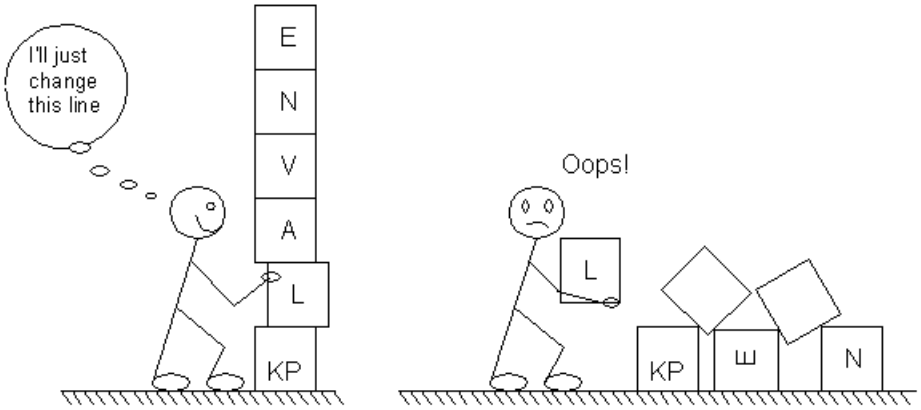
The previous sections alluded to several conditions that restrict you as you modify your meshed solid model. These restrictions arise due to the cross-reference checking that has been incorporated into the ANSYS program to prevent contamination of the solid model and finite element model data. They may be summarized as follows:

- Meshed keypoints, lines, areas, or volumes may not be deleted or moved.

- Nodes or elements associated with keypoints, lines, areas, or volumes may not be moved. They can be deleted only with the mesh clearing commands.
- Areas contained in volumes may not be deleted or changed.
- Lines contained in areas may not be deleted or changed (except by **LDIV**, **LCOMB**, or **LFILLT**, as discussed previously).
- Keypoints contained in lines may not be deleted. They can be moved only with the **KMODIF** command, which revises and clears (if meshed) attached lines, areas, and volumes.

The basic reasoning behind these rules might be visualized from the following sketch. Schematically, your completed model can be thought of as a stack of blocks, in which the bottommost block represents your keypoints, the next block represents lines, and so forth. If you were to change a lower entity (a line, for instance), you would disturb the entities that are stacked on top of it. (Admittedly, this illustration somewhat oversimplifies the dependence of higher level entities on lower entities.)

Figure 8.10: Why We Have Solid Model Cross-Reference Checking



These rules are not as restrictive as they might appear. Furthermore, for those few instances when they absolutely prevent you from performing an operation that you need, you can deactivate them, as is discussed in the next few paragraphs. Be aware, however, that in deactivating these rules, you lose the protections that they provide, and thereby increase the chances of irrevocably damaging your model's database.

8.5.1. Circumventing Cross-Reference Checking (A Risky Activity)

Cross-reference checking of the solid model usually serves only to help you avoid corrupting your model's database. There are occasions, however, when you might have good reason to attempt a "forbidden" operation. The **MODMSH** command (**Main Menu> Preprocessor> Checking Ctrls> Model Checking**) exists for this purpose. **MODMSH** has three options: DETACH, NOCHECK, and CHECK.

MODMSH,DETACH detaches the finite element model from the solid model, which will allow the finite element model to be modified by node and element commands. This detachment keeps the database "clean," in that there will be no recognized database conflicts. Consider, for instance, a single keypoint and its associated node. Following a **MODMSH**,DETACH, the program will no longer recognize the associativity between these two entities, so that you can now move the node to a new location without creating a conflict in the database. Once you have detached your model, you cannot perform such operations as selecting or defining the finite element model in terms of the solid model, clearing the mesh, or transferring solid model boundary conditions to the finite element model.

MODMSH,NOCHECK is very dangerous. It deactivates all cross-reference checking, and makes it startlingly easy to get the database so fouled up that hardly any solid modeling operations are possible. Its utility is that it allows you to use commands such as **EMODIF**, **NMODIF**, **EDELE**, **NDELE**, etc. to modify elements and nodes that were generated with the mesh commands. Activating this option will cause the program to warn you that cross-reference checking is bypassed whenever you initiate a solution or issue a **PFACT** or **SOLVE** command. *Use this option only if you are completely sure of what you are doing*, for the havoc you can inflict on your database with the NOCHECK option on will often make it impossible for the ANSYS customer support staff to help you recover from any trouble you get yourself into.

MODMSH,CHECK restores cross-reference checking after it has been inhibited. However, since the database integrity could have been compromised while the checking was off, the program will continue to warn you whenever you initiate a solution or issue a **PFACT** or **SOLVE** command. The only way to get rid of this warning message is to start over with a "clean" database.

Chapter 9: Direct Generation

Direct generation is the approach in which you define the nodes and elements of a model directly. Despite the many convenience commands that allow you to copy, reflect, scale, etc. a given pattern of nodes or elements, direct generation can commonly require about ten times as many data entries to define a model as compared to solid modeling.

This documentations's earlier discussions concerning planning (Planning Your Approach (p. 5)), coordinate systems (Coordinate Systems (p. 15)), and working planes (Using Working Planes (p. 25)) apply to direct generation as well as to solid modeling.

A model assembled by direct generation is defined strictly in terms of nodes and elements. Even though node and element generation operations can be interspersed, no one element can be defined until after all of its nodes have been created.

The following topics related to direct generation are available:

9.1. Nodes

This section describes various tasks related to the direct generation of nodes. Topics include:

- Defining nodes
- Generating additional nodes from existing nodes
- Maintaining (list, view and delete) nodes
- Moving nodes
- Rotating a node's coordinate system
- Reading and writing text files that contain nodal data

You can use any of the methods described in the following table to define nodes.

Table 9.1: Defining Nodes

Define	Command	GUI
individual nodes in the active coordinate system [1]	**N**	**Main Menu> Preprocessor> Modeling> Create> Nodes> In Active CS** or **> On Working Plane**
a node at an existing keypoint	**NKPT**	**Main Menu> Preprocessor> Modeling> Create> Nodes> On Keypoint**

1. If using ANSYS interactively, you can define a working plane snap increment and use picking [**N**, P] to generate nodes graphically. (For more information on the working plane, see Using Working Planes (p. 25).)

Once you have created an initial pattern of nodes, you can generate additional nodes using the methods described in the following table.

Table 9.2: Generating Additional Nodes from Existing Nodes

	Com-mand	GUI
generate a line of nodes between two existing nodes	**FILL**	**Main Menu> Preprocessor> Modeling> Create> Nodes> Fill between Nds**
generate additional nodes from a pattern of nodes	**NGEN**	**Main Menu> Preprocessor> Modeling> Copy> Nodes> Copy**
generate a set of nodes from a pattern of nodes	**NSCALE**	**Main Menu> Preprocessor> Modeling> Copy> Nodes> Scale & Copy** or **> Scale & Move** **Main Menu> Preprocessor> Modeling> Operate> Scale> Nodes> Scale & Copy** or **> Scale Move**
generate a quadratic line of nodes from three nodes	**QUAD**	**Main Menu> Preprocessor> Modeling> Create> Nodes> Quadratic Fill**
generate a reflected set of nodes	**NSYM**	**Main Menu> Preprocessor> Modeling> Reflect> Nodes**
transfer a pattern of nodes to another co-ordinate system	**TRANS-FER**	**Main Menu> Preprocessor> Modeling> Move/Modify> Transfer Coord> Nodes**
define a node at the center of a curvature of an arc of nodes [1]	**CENTER**	**Main Menu> Preprocessor> Modeling> Create> Nodes> At Curvature Ctr**

1. If a local cylindrical coordinate system is defined [**CS**] at the center of curvature, you can use the **FILL** command (**Main Menu> Preprocessor> Modeling> Create> Nodes> Fill between Nds**) to generate additional nodes on the arc. If a radius of curvature is given, the center of curvature is automatically calculated to be along the perpendicular bisector of the *NODE1*-*NODE2* line in the plane of *NODE1*, *NODE2*, and *NODE3*.

You can use any of the methods described in the following table to maintain nodes.

Table 9.3: Maintaining Nodes

Maintenance	Com-mand	GUI
list nodes	**NLIST**	**Utility Menu> List> Nodes** **Utility Menu> List> Picked Entities> Nodes**
display nodes [1]	**NPLOT**	**Utility Menu> Plot> Nodes**
delete nodes [2]	**NDELE**	**Main Menu> Preprocessor> Modeling> Delete> Nodes**

1. Node numbers will also be displayed in **EPLOT** displays (**Utility Menu> Plot> Elements**) for nodes attached to elements, if you have issued the proper **/PNUM** command (**Utility Menu> PlotCtrls> Numbering**).

2. Deleting a node also deletes any boundary conditions (such as displacements, forces, etc.) as well as any coupling or constraint equations containing the deleted node.

You can move a node by overwriting it with the **N** command (or any other node-generating command) or by using one of the methods in the following table.

Table 9.4: Moving Nodes

	Command	GUI
modify one or all of the coordinates defining a node	**NMODIF**	**Main Menu> Modeling> Preprocessor> Create> Nodes> Rotate Node CS> By Angles** **Main Menu> Preprocessor> Modeling> Move/Modify> Rotate Node CS> By Angles** or **> Set of Nodes** or **> Single Node**
move a node to an intersection of coordinate system surfaces	**MOVE**	**Main Menu> Preprocessor> Modeling> Move/Modify> Nodes> To Intersect**

Use the **NDIST** (**Main Menu> Preprocessor> Modeling> Check Geom> ND distances**) command to calculate the distance between two nodes.

You can use any of the methods described in the following table to rotate a node's coordinate system. The nodal coordinate system is parallel to the global Cartesian coordinate system by default. See Nodal Coordinate Systems (p. 21) for more information.

Table 9.5: Rotating a Node's Coordinate System

Rotate a Nodal Coordinate System	Command	GUI
into the active coordinate system	**NROTAT**	**Main Menu> Preprocessor> Modeling> Create> Nodes> Rotate Node CS> To Active CS** **Main Menu> Preprocessor> Modeling> Move/Modify> Rotate Node CS> To Active CS**
by direction cosines	**NANG**	**Main Menu> Preprocessor> Modeling> Create> Nodes> Rotate Node CS> By Vectors** **Main Menu> Preprocessor> Modeling> Move/Modify> Rotate Node CS> By Vectors**
by angles	**N**, **NMODIF**	**Main Menu> Preprocessor> Modeling> Create> Nodes> In Active CS** or **> On Working Plane** **Main Menu> Modeling> Preprocessor> Create> Nodes> Rotate Node CS> By Angles** **Main Menu> Preprocessor> Modeling> Move/Modify> Rotate Node CS> By Angles** or **> Set of Nodes** or **> Single Node**

9.1.1. Reading and Writing Text Files That Contain Nodal Data

You can read a text file containing nodal data. This ability could be useful if you are importing ASCII nodal data from another mesh generator, a CAD/CAM program, or another ANSYS session. You can also write such an ASCII file for export to another program (which must be able to read this ANSYS file) or to another ANSYS session. You will not normally need to read or write nodal data in a standard ANSYS model generation session. If you do need to read or write nodal data you can use any of the methods described in the following table.

Table 9.6: Reading and Writing Files Containing Nodal Data

	Com-mand	GUI
specify a range of nodes to be read from a node file	**NRRANG**	**Main Menu> Preprocessor> Modeling> Create> Nodes> Read Node File**
read nodes from a file	**NREAD**	**Main Menu> Preprocessor> Modeling> Create> Nodes> Read Node File**
write nodes to a file	**NWRITE**	**Main Menu> Preprocessor> Modeling> Create> Nodes> Write Node File**

9.2. Elements

This section describes various tasks related to the direct generation of elements. Topics include:

- Prerequisites for defining elements
- Assembling element tables
- Pointing to entries in element tables
- Reviewing the contents of element tables
- Defining elements
- Maintaining (list, view, and delete) elements
- Generating additional elements from existing elements
- Using special methods for generating elements
- Reading and writing text files that contain element data
- Modifying elements by changing nodes
- Modifying elements by changing element attributes

9.2.1. Prerequisites for Defining Element Attributes

Before you directly generate elements, you must have already defined the minimum number of nodes required for that element and must have specified the proper element attributes.

You assemble tables of element attributes using the methods described in the following table and various coordinate system commands. See Generating the Mesh (p. 101) in the *Modeling and Meshing Guide* for more information on creating element attribute tables.

Table 9.7: Assembling Element Tables

Define	Command	GUI
an element type from the element library	**ET**	**Main Menu> Preprocessor> Element Type> Add/Edit/Delete**
the element real constants	**R**	**Main Menu> Preprocessor> Real Constants**
a linear material property	**MP**, **MPDATA**, **MPTEMP**	**Main Menu> Preprocessor> Material Props> Material Models>** ***analysis type***

Once the element attribute tables are in place, you can "point" to various entries in the element tables. The values of these pointers that are in effect at the time that you create your elements are used by the program to assign attributes from the tables to the elements. Use one of the methods shown in the following table to set attribute pointers.

Table 9.8: Pointing to Entries in Element Tables

Set	Command	GUI
element type attribute pointer	**TYPE**	**Main Menu> Preprocessor> Modeling> Create> Elements> Elem Attributes**
element real constant set attribute pointer	**REAL**	**Main Menu> Preprocessor> Modeling> Create> Elements> Elem Attributes**
element material attribute pointer	**MAT**	**Main Menu> Preprocessor> Modeling> Create> Elements> Elem Attributes**
element coordinate system attribute pointer	**ESYS**	**Main Menu> Preprocessor> Modeling> Create> Elements> Elem Attributes**

You can review the contents of element table using one of the methods described in the following table.

Table 9.9: Reviewing the Contents of Element Tables

List	Command	GUI
currently defined element types	**ETLIST**	**Utility Menu> List> Properties> Element Types**
real constant sets	**RLIST**	**Utility Menu> List> Properties> All Real Constants** or **> Specified Real Constants**
linear material properties	**MPLIST**	**Utility Menu> List> Properties> All Materials** or **> All Matls, All Temps** or **> All Matls, Specified Temp** or **> Specified Matl, All Temps**
data tables	**TBLIST**	**Main Menu> Preprocessor> Material Props> Material Models**

List	Command	GUI
		Utility Menu> List> Properties> Data Tables
coordinate systems	**CSLIST**	**Utility Menu> List> Other> Local Coord Sys**

9.2.2. Defining Elements

After you have defined the necessary nodes and set the element attributes, you can proceed to define your elements. You can define an element by identifying its nodes using the **E** command (**Main Menu> Preprocessor> Modeling> Create> Elements> Auto Numbered> Thru Nodes** or **Main Menu> Preprocessor> Modeling> Create> Elements> User Numbered> Thru Nodes**). The number of nodes required for each element and the order in which they must be input are determined by the element type. The order in which nodes are defined determines the element normal direction. See Keeping Track of Element Faces and Orientations (p. 191) in the *Modeling and Meshing Guide* for more information.

If you are working interactively, you can use graphical picking (that is, pick nodes) to generate the elements by choosing one of the GUI paths associated with the **E** command.

If you are using command input, only eight nodes can be input on the **E** command. For element types that require more than eight nodes, use the **EMORE** command to define the additional nodes. For example, SOLID186, a 3-D 20-node structural solid element, will require two **EMORE** commands in addition to the **E** command. (The **EMORE** command is not necessary if graphical picking is used to create the elements.)

Caution

If you create overlapping elements (that is, elements attached to the same nodes and occupying the same space), various ANSYS features such as graphics, surface loads, selecting logic, etc. might not function as expected. It is best to avoid the use of overlapping elements altogether; if this is not possible, use extreme caution whenever you employ overlapping elements.

Once you have created your elements, use the methods described in the following table to maintain elements.

Table 9.10: Maintaining Elements

Maintenance	Command	GUI
list elements	**ELIST**	**Utility Menu> List> Elements** **Utility Menu> List> Picked Entities> Elements**
display elements [1]	**EPLOT**	**Utility Menu> Plot> Elements**
delete elements [2]	**EDELE**	**Main Menu> Preprocessor> Modeling> Delete> Elements**

1. Element numbers will be shown in your **EPLOT** display if you turn them on with the **/PNUM** command (menu path **Utility Menu> PlotCtrls> Numbering**). In most instances, the program will automatically assign element numbers, using the next available unused number. (The first **E** command defines element number 1, the second **E** command defines element number 2, and so on.)

2. Deleting elements creates "blanks" in your element-numbering sequence. The automatic numbering procedure will not reuse these blank numbers, even if they are at the end of your sequence. (If you define 10 elements, then delete them all, the next **E** command will define element number 11. Numbers 1-10 will remain blank.) You can control element numbering through the *number control* commands (see Number Control and Element Reordering (p. 211) in the *Modeling and Meshing Guide*), or through the **EN** command **Main Menu> Preprocessor> Modeling> Create> Elements> User Numbered> Thru Nodes**), which allows you to define an element's number directly.

Once you have created an initial pattern of elements, you can generate additional elements using the methods described in the following table. Please note these commands do *not* generate nodes; you must generate the necessary nodes before you generate additional elements. Also, the element attributes (MAT, TYPE, REAL, and ESYS) for the generated elements are based upon the elements in the original pattern and *not* upon the current specification settings.

Table 9.11: Generating Additional Elements From Existing Elements

Generate Elements	Command	GUI
from an existing pattern	**EGEN**	**Main Menu> Preprocessor> Modeling> Copy> Elements> Auto Numbered**
from an existing pattern with explicitly incremented element numbers	**ENGEN**	**Main Menu> Preprocessor> Modeling> Copy> Elements> User Numbered**
from a pattern by symmetry reflection	**ESYM**	**Main Menu> Preprocessor> Modeling> Reflect> Elements> Auto Numbered**
by symmetry reflection with explicitly assigned element numbers	**ENSYM**	**Main Menu> Preprocessor> Modeling> Reflect> Elements> User Numbered** **Main Menu> Preprocessor> Modeling> Move/Modify> Reverse Normals> of Shell Elements**

Some elements can be generated using the special methods described in the following table.

Table 9.12: Using Special Methods for Generating Elements

Generate	Command	GUI
"surface" elements over the exterior faces of existing elements	**ESURF** [1]	**Main Menu> Preprocessor> Modeling> Create> Elements> Surf/Contact> *option***
"surface" elements overlaid on the edges of existing plane elements and assign the extra node as the closest fluid element node	**LFSURF** [2]	**Main Menu> Preprocessor> Modeling> Create> Elements> Surf/Contact> Surface Effect> Attach to Fluid> Line to Fluid**
"surface" elements overlaid on the surface of existing solid elements and assign the extra	**AFSURF** [3]	**Main Menu> Preprocessor> Modeling> Create> Elements> Surf/Contact> Surf Effect> Attach to Fluid> Area to Fluid**

Generate	Command	GUI
node as the closest fluid element node		
"surface" elements overlaid on the surface of the existing elements and assign the extra node as the closest fluid element node	**NDSURF** [4]	**Main Menu> Preprocessor> Modeling> Create> Elements> Surf/Contact> Surf Effect> Attach to Fluid> Node to Fluid**
two-node elements between coincident or offset nodes	**EINTF**	**Main Menu> Preprocessor> Modeling> Create> Elements> Auto Numbered> At Coincid Nd**

1. Use **ESURF**,*XNODE* to generate SURF151or SURF152, elements with the "optional" node used in some thermal analyses.
2. Use **LFSURF** to generate SURF151 elements with the "optional" node used in some thermal analyses.
3. Use **AFSURF** to generate SURF152 elements with the "optional" node used in some thermal analyses.
4. Use **NDSURF** to generate SURF151 or SURF152 elements with the "optional" node used in some thermal analyses.

9.2.3. Reading and Writing Text Files That Contain Element Data

You can read or write a text file that contains element data. These capabilities can be useful for transferring data to and from another program (or another ANSYS session). You will not normally need to use these capabilities in a standard ANSYS model generating session. If you do need to read or write element data you can use any of the methods described in the following table.

	Command	GUI
specify a range of elements to be read from an element file	**ERRANG**	**Main Menu> Preprocessor> Modeling> Create> Elements> Read Elem File**
read elements from a file	**EREAD**	**Main Menu> Preprocessor> Modeling> Create> Elements> Read Elem File**
write elements to a file	**EWRITE**	**Main Menu> Preprocessor> Modeling> Create> Elements> Write Elem File**

9.2.4. A Note About Overlapping Elements

Be advised that if you create overlapping elements (that is, elements attached to the same nodes and occupying the same space), various ANSYS features such as graphics, surface loads, selecting logic, etc. might not function as expected. It is best to avoid the use of overlapping elements altogether; if this is not possible, use extreme caution whenever you employ overlapping elements.

9.2.5. Modifying Elements By Changing Nodes

You can redefine an element in terms of different nodes, taking care that the element attribute pointers are set to the appropriate values. The element attribute settings that are in place when you execute these commands or GUI paths will control the element type, real constants, material properties, and for some element types, the element coordinate system that are assigned to the redefined elements.

To modify a previously defined element, use the **EMODIF** command (**Main Menu> Preprocessor> Modeling> Move/Modify> Elements> Modify Nodes**).

To redefine an element by its number and node connectivity, use the **EN** command (**Main Menu> Preprocessor> Modeling> Create> Elements> User Numbered> Thru Nodes**).

You can also use the **ENGEN** and **ENSYM** commands and GUI paths, which are described earlier in this chapter, to overwrite and redefine groups of elements.

9.2.6. Modifying Elements By Changing Element Attributes

There are several ways of changing the attributes of an element after it has been created.

To change the material number of a specified element within either PREP7 or SOLUTION, use the **MPCHG** command. (See the **MPCHG** command description in the *Command Reference* for a list of menu paths associated with this command.)

The combination of the **EMODIF** and ***REPEAT** commands provide another versatile method of redefining the attributes of existing elements (within PREP7 only). You cannot access the ***REPEAT** command directly in the GUI. The following program listing illustrates the use of the **EGEN**, **EMODIF**, and ***REPEAT** commands. See their individual command descriptions in the *Command Reference* for more information.

```
E,1,2                   ! Element 1
REAL,3                  ! REAL set pointer = 3
E,2,3                   ! Element 2 (REAL=3)
EGEN,40,1,2             ! Generate 40 elements from el. 2 (all with REAL=3)
EMODIF,5,REAL,4         ! Redefine element 5 with REAL set 4
*REPEAT,18,2            ! Redefine els. 7-39 in steps of 2 (with REAL=4)
```

Alternatively, you can change the entries in the attribute tables after creating an element, but before entering SOLUTION. A warning will be issued if the REAL set or MAT set contain unused entries (such as could happen if a REAL property set for a beam were assigned to a spar element).

Another way of changing your element attributes is by deleting your elements using the **EDELE** command (**Main Menu> Preprocessor> Modeling> Delete> Elements**), redefining your pointers, and recreating your elements using the **EN** command (**Main Menu> Preprocessor> Modeling> Create> Elements> User Numbered> Thru Nodes**).

9.2.7. A Note About Adding and Deleting Midside Nodes

For any of these procedures, if you change the element TYPE attribute to substitute midside-node elements for non-midside-node elements, you will also need to use the **EMID** command (**Main Menu> Preprocessor> Modeling> Move/Modify> Elements> Add Mid Nodes** or **> Remove Mid Nd**) to add the extra midside nodes as required. Also, in order to delete midside nodes, you must first remove them from the midside-node elements by issuing de-node elements for non-midside-node elements, you will also need to use **EMID**,-1.

When defining midside-node elements using the direct generation method (that is, the **E**, **EN**, and similar commands), midside nodes are created and located according to the following scheme:

- Some higher-order elements permit the removal of midside nodes. For such elements, if a zero value (or blank) is used for a midside node when a higher-order element is defined, the corresponding midside node is *removed* from the element. This results in some or all of the quadratic terms (depending on the number of removed midside nodes) in the element's shape functions being ignored, thus forcing the element edge(s) to be and remain straight. In the extreme case of an element with all of its midside nodes removed, the element will use linear shape functions thus producing results similar to the analogous lower-order (non-midside-node) element type.

- When defining a higher-order element, if a node number is used for a midside node and that node has not yet been defined (**N**, **NGEN**, **FILL**, **NSYM**, and similar commands), then the node will be automatically defined and given a geometric location that is the calculated midpoint (linearly interpolated in Cartesian coordinates) between its respective corner nodes. Nodal rotations for such nodes will also be automatically calculated by linearly interpolating between the nodal rotation angles of the corner nodes. This allows for the convenience of creating midside-node elements without the need to explicitly define the geometric locations for midside nodes located midway between the corner nodes.

Note that this behavior applies only to the direct model generation method. Controls regarding midside nodes in meshed models are provided in the ANSYS meshing controls.

Chapter 10: Number Control and Element Reordering

You can control the numbering of keypoints, lines, areas, volumes, elements, nodes, element types, real constant sets, materials, coupled DOF sets, constraint equations, and coordinate systems. Control of the numbering is useful when combining separately modeled portions of a model into one. Likewise, allowing the ANSYS program to reorder elements automatically at the start of a sparse direct solution is typically the preferred approach; however, other methods are available for controlling element re-ordering. The following topics are available:

10.1. Number Control

The commands and GUI paths described here give you control of the numbering of keypoints, lines, areas, volumes, elements, nodes, element types, real constant sets, materials, coupled DOF sets, constraint equations, and coordinate systems. Control of the numbering is useful, and sometimes essential, when combining separately modeled portions of a model into one.

Caution

The numbering of Boolean output entities is not 100% predictable. That is, sometimes the same Boolean operation, when performed on different computer systems, will assign different numbers to output entities. If you plan to generate an input stream file interactively on one machine and rerun that input stream on another system, you should avoid using entity identification numbers in your command stream. Instead, use selecting logic to identify specific entities as needed.

10.1.1. Merging Coincident Items

If two separate entities have the same location, you can *merge* these entities together into a single entity using one of these methods:

Command(s): NUMMRG
GUI: Main Menu> Preprocessor> Numbering Ctrls> Merge Items

Note

Use of the **NUMMRG** command or its equivalent GUI path does not cause changes to a model's geometry; only the topology is affected.

For example, suppose that you have two separate but coincident nodes. If you use the command **NUMMRG**,NODE (or the equivalent GUI path) to merge the nodes, the higher numbered node will be deleted and will be replaced with the lower numbered coincident node. Two coincident nodes will thus be replaced by a single node.

If two regions that have already been meshed are to be joined, you need to execute three separate merge operations. If, for example, you are using the command input method, you would first issue a merge command for nodes [**NUMMRG**,NODE], then one for elements [**NUMMRG**,ELEM], and last, one for all solid model entities [**NUMMRG**,KP]. (Or, you could simply use **NUMMRG**,ALL to merge all selected coincident items in the proper sequence. **NUMMRG**,ALL will also merge non-geometric items such as MAT, CE, etc.)

Caution

When merging entities in a model that has already been meshed, the order in which you issue multiple **NUMMRG** commands is significant. If you want to merge two adjacent meshed regions that have coincident nodes and keypoints, always merge nodes [**NUMMRG**,NODE] before merging keypoints [**NUMMRG**,KP] (as described above). Merging keypoints before nodes can result in some of the nodes becoming "orphaned"; that is, the nodes lose their association with the solid model. The orphaned nodes can cause certain operations (such as boundary condition transfers, surface load transfers, and so on) to fail.

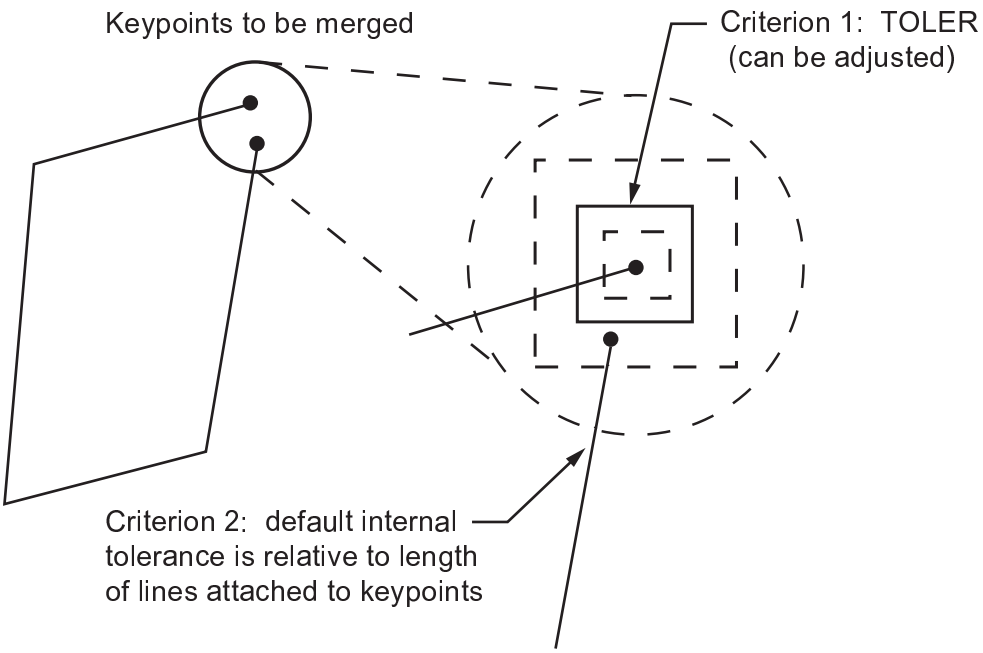
Many solid modeling operations create coincident keypoints, lines, and/or areas. You can use **NUMMRG**,KP (or the equivalent GUI path) to merge such coincident items. Keypoint locations are used as the basis for merging. Once coincident keypoints are merged, any higher order solid model entities (lines, areas, and volumes) attached to those keypoints are automatically considered for merging. The definition of coincident changes depending on the tolerances used. By default, merging of keypoints attached to lines is done when the distance between keypoints falls within the following criteria:

1. 1E-4 units of each other (see Figure 10.1: Default Merge Tolerances (p. 213)), *and*

2. 1E-5 times the length of the longest line connected to the keypoints in consideration

Criterion 1 above describes the consideration tolerance field (*TOLER*) on **NUMMRG**. *TOLER* is a *consideration* tolerance. If a keypoint is within *TOLER* of another keypoint, then those two keypoints are *candidates* to be merged. If when "moving" the higher numbered keypoint, the distance exceeds the internal relative solid model tolerance (criterion 2 above), the keypoints will *not* be merged. Lines, areas, and volumes are considered for merging in a similar manner. Criterion 2 describes the default internal relative solid model tolerance, a tolerance designed to protect you from eliminating tiny lines in the model by the merge operation. *Both* criteria must be satisfied for keypoints to be merged.

Figure 10.1: Default Merge Tolerances



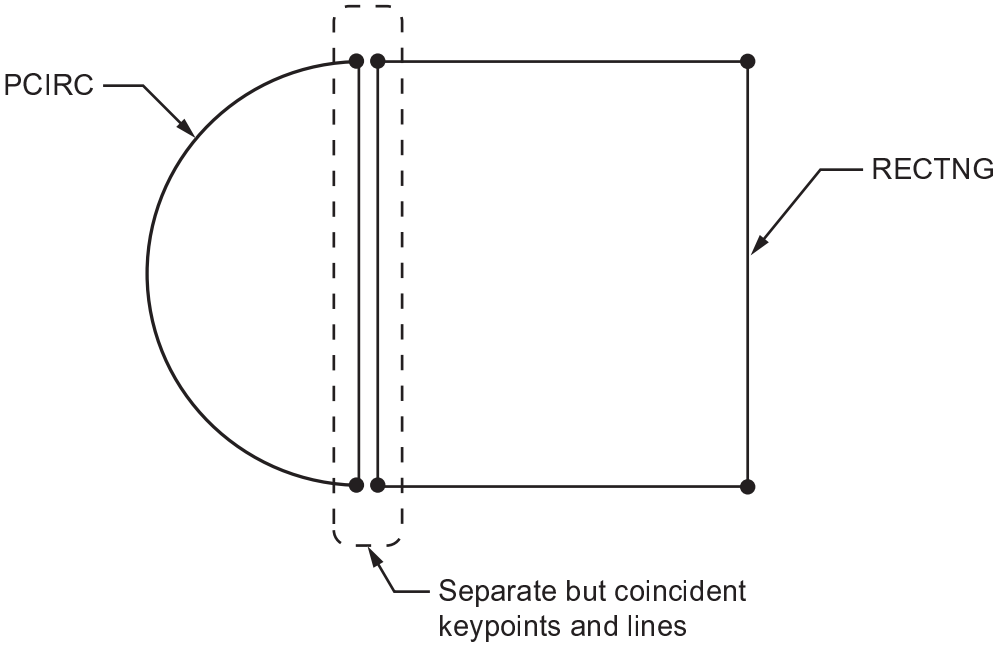
The internal relative tolerance (criterion 2) can be overridden by an option to specify a global solid model tolerance (*GTOLER*) on **NUMMRG**. *GTOLER* is a global, *absolute* tolerance, rather than a relative tolerance. If *GTOLER* is used, the size of the lines attached to keypoints is no longer considered and it is fairly easy to defeature your model by using too large a value for *GTOLER*. You should save your database before attempting to merge, especially when using the *GTOLER* option.

The following example, which corresponds to Figure 10.2: Example of NUMMRG Application (p. 214), illustrates the use of **NUMMRG** to merge entities:

```
PCIRC,... ! Create a partial circle
RECTNG,... ! Create a rectangle
NUMMRG,KP ! Default merge tolerances used
```

For a model having coincident keypoints, the **NUMMRG** operation will probably be more economical (that is, faster) than **AGLUE** (**Main Menu> Preprocessor> Modeling> Operate> Booleans> Glue> Areas**).

Figure 10.2: Example of NUMMRG Application



If you merge keypoints that are very nearly coincident, any very short line connecting those keypoints will be deleted. If the keypoints are too far apart to be merged, you can use the **LCOMB** command (**Main Menu> Preprocessor> Modeling> Operate> Booleans> Add> Lines**) to eliminate the very short line. **LCOMB** will produce a continuous (but not necessarily smooth) line. If the resulting line is kinked, it cannot be used as a drag path [**ADRAG**, **VDRAG**], nor can it be used in any Boolean operations.

10.1.2. Compressing Item Numbers

As you build your model, you might, by deleting, clearing, merging, or performing other operations, create unused slots in the numbering sequence for various items. These slots will remain empty for some items (such as elements) but will be filled in for other items (such as keypoints) as new items are created. To save data storage space (by eliminating otherwise empty numbers) or to preserve desired sequencing (by forcing newly-created items to be assigned numbers greater than those of existing items), you can eliminate these gaps by "compressing" your numbering, using one of these methods:
Command(s): NUMCMP
GUI: Main Menu> Preprocessor> Numbering Ctrls> Compress Numbers

The compression operation may be selectively re-executed for chosen item groups (elements, keypoints, etc.), or may be simultaneously applied to all valid items (using **NUMCMP**,ALL).

The following example demonstrates one application of the **NUMCMP** command:

```
VMESH,...
VCLEAR,...    ! Node and element numbers will not be reused.
...
! Change meshing controls, element attributes, etc.
...
VMESH,...     ! Node and element numbering will contain "gaps"
NUMCMP,NODE   ! Optional step - NUMCMP can free up some computer memory
NUMCMP,ELEM   ! by eliminating gaps in numbering sequences.
```

Please see the description of the **NUMCMP** command in the *Command Reference* for more information.

10.1.3. Setting Starting Numbers

When creating new, automatically-numbered items, you might want to set the starting number of your new series of items higher than the greatest number used by existing items. By doing so, you will ensure that your newly created entities will be consecutively numbered, and will thus be prevented from occupying gaps in the existing numbering sequence. Another reason for specifying a set of starting numbers would be if you were creating portions of your model independently of one another, and you wanted to avoid numbering conflicts when they were combined together into one model. You can specify such user-defined starting numbers using one of these methods:
Command(s): NUMSTR
GUI: Main Menu> Preprocessor> Numbering Ctrls> Set Start Number

If using command input, you must issue this command for each separate class of items (nodes, elements, keypoints, etc.).

This example demonstrates the command input method:

```
! Create one portion of model:
...
...
! Create a separate, distinctly-numbered portion of your model:
NUMSTR,KP,100
NUMSTR,LINE,100
NUMSTR,AREA,100
NUMSTR,VOLU,100
```

For more information, see the description of the **NUMSTR** command in the *Command Reference*.

10.1.4. Adding Number Offsets

If you wish to combine two independently created portions of a model and want to prevent numbering conflicts, you can renumber all selected items by *adding* an offset value to their existing numbers. Use one of these methods:
Command(s): NUMOFF
GUI: Main Menu> Preprocessor> Numbering Ctrls> Add Num Offset

You must re-execute this operation for each item group (nodes, elements, keypoints, etc.) that is to be renumbered.

The **CDWRITE** command (**Main Menu> Preprocessor> Archive Model> Write**), which writes all selected model data to a text file, automatically puts a series of **NUMOFF** commands at the beginning of the file to push any existing data out of the way when the file is read. Gaps in numbering that may result from these **NUMOFF** commands can be removed with the **NUMCMP** command.

Solid model data (in IGES format) can be written to a text file in PREP7. You can also transfer solid model data from an externally-generated IGES file into the ANSYS database.

- To write solid model data to a file, use one of these methods:
  Command(s): IGESOUT
  GUI: Utility Menu> File> Export

- To transfer IGES data from a file into ANSYS PREP7 data, use one of these methods:
  Command(s): IGESIN
  GUI: Utility Menu> File> Import

New solid model entities created (in the AUX15 processor) by the **IGESIN** command (or GUI path) will automatically be numbered to avoid conflicts with other solid model entities that already exist in the database. If an IGES file is read into an empty database, ANSYS solid modeling entity numbering will not necessarily start at 1. This is because intermediate entities might be created (and later deleted) in the translation process. See Importing Solid Models from IGES Files (p. 97) for more information on the IGES interface.

10.2. Element Reordering

At the start of a solution (**/SOLU**), the ANSYS program will automatically reorder your elements so that they can be accessed in close proximity to each other. This reordering scheme could improve the memory usage (locality) and, hence, the speed of the global assembly. This operation does not affect the element numbers used to identify elements in pre- and postprocessing of your database. In most cases, allowing the ANSYS program to reorder elements automatically is the preferred approach. If you do not want the program to perform automatic reordering, you can re-establish the original element ordering using one of these methods:
Command(s): NOORDER
GUI: Main Menu> Preprocessor> Numbering Ctrls> Element Reorder> Reset Elem Order

For more hands-on control of element reordering, use any of the methods listed below to initiate re-ordering within PREP7. (For any of these operations, the program knows not to overwrite your "manual" reordering with an automatic reordering.)

- To initiate element reordering based upon a geometric sort, use one of these methods:
  Command(s): WSORT
  GUI: Main Menu> Preprocessor> Numbering Ctrls> Element Reorder> Reorder by XYZ

- To define a starting wave list, use one of these methods:
  Command(s): WSTART
  GUI: Main Menu> Preprocessor> Numbering Ctrls> Element Reorder> Define Wave List

- To add more nodes to the starting wave list, use one of these methods:
  Command(s): WMORE
  GUI: Main Menu> Preprocessor> Numbering Ctrls> Element Reorder> Extend Wave List

- To erase all reordering wave lists, use one of these methods:
  Command(s): WERASE
  GUI: Main Menu> Preprocessor> Numbering Ctrls> Element Reorder> Erase Wave List

- To initiate reordering, use one of these methods:
  Command(s): WAVES
  GUI: Main Menu> Preprocessor> Numbering Ctrls> Element Reorder> Reorder by List

- To obtain an estimate of the wavefront statistics for your model, as currently ordered, use one of these methods:
  Command(s): WFRONT
  GUI: Main Menu> Preprocessor> Numbering Ctrls> Element Reorder> Est. Wavefront

You should avoid placing point elements (for example, MASS21 and MASS71) on element midside nodes if you intend to reorder your elements using the **WAVES** command; otherwise, the **WAVES** command may fail.

Chapter 11: Coupling and Constraint Equations

When generating your model, you typically define the relationships among different degrees of freedom by using *elements* to connect the nodes. However, you sometimes need to be able to model distinctive features (rigid regions, pinned structural joints, sliding symmetry boundaries, periodic conditions, and other special internodal connections) which cannot be adequately described with elements. You can establish such special associations among nodal degrees of freedom by using *coupling* and *constraint equations* (CEs). Using these techniques enables you to link degrees of freedom in ways that elements cannot.

The following topics concerning coupling and CEs are available:

11.1. What Is Coupling?

When you need to force two or more degrees of freedom (DOFs) to take on the same (but unknown) value, you can *couple* these DOFs together. A set of coupled DOFs contains a *prime* DOF, and one or more other DOFs. Coupling will cause only the prime DOF to be retained in your analysis' matrix equations, and will cause all the other DOFs in a coupled set to be eliminated. The value calculated for the prime DOF will then be assigned to all the other DOFs in a coupled set.

Typical applications for coupled DOFs include: 1) maintaining symmetry on partial models, 2) forming pin, hinge, universal, and slider joints between two coincident nodes, and 3) forcing portions of your model to behave as rigid bodies (see this chapter's discussion of constraint equations for more general rigid region capability).

11.2. How to Create Coupled Degree of Freedom Sets

You can use several different methods to create a coupled degree-of-freedom set. These include the **CP** command and other commands such as **CPNGEN**, **CPINTF**, **CPLGEN**, and **CPSGEN**. These methods of generating coupled sets are discussed in the following sections.

In addition to the methods discussed here, you can use the internal multipoint constraint (MPC) feature of certain contact elements (CONTA171, CONTA172, CONTA173, CONTA174, CONTA175, CONTA176, and CONTA177) to model coupling constraints. By this method, the program builds MPC equations internally based on the contact kinematics. See Multipoint Constraints and Assemblies in the *Contact Technology Guide* for more information on how to use this feature.

11.2.1. Creating and Modifying Coupled Sets at Specified Nodes

To define (or modify) a set of coupled degrees of freedom, use one of these methods:
Command(s): CP

GUI: Main Menu> Preprocessor> Coupling/Ceqn> Couple DOFs

After creating a coupled set of nodes, you can include more nodes in that set by simply performing an additional coupling operation (be sure to use the same set reference number). You can also use selecting logic to couple "ALL" of the selected nodes. Nodes can be deleted from a coupled set by inputting them as *negative* node numbers on the **CP** command. To modify a coupled DOF set (that is, add or delete nodes, or change the DOF label), use the **CPNGEN** command. (You cannot access the **CPNGEN** command directly in the GUI.)

11.2.2. Coupling Coincident Nodes

The **CPINTF** command couples coincident nodes in a model by generating one coupled set for each specified DOF label at every pair of coincident nodes. This operation is useful for "buttoning" together several pairs of nodes (such as at a seam).
Command(s): CPINTF
GUI: Main Menu> Preprocessor> Coupling/Ceqn> Coincident Nodes

Instead of coupling coincident nodes, you can use one of these alternative methods to force the nodes to behave in the same way:

- If *all* DOFs are to be coupled for coincident nodes, it is usually more efficient to simply merge those nodes together by using the **NUMMRG** command (**Main Menu> Preprocessor> Numbering Ctrls> Merge Items**).
- You can connect coincident pairs of nodes by creating 2-node elements between them by using the **EINTF** command (**Main Menu> Preprocessor> Modeling> Create> Elements> Auto Numbered> At Coincid Nd**).
- To tie together two regions having dissimilar mesh patterns, use the **CEINTF** command (**Main Menu> Preprocessor> Coupling/Ceqn> Adjacent Regions**). This operation generates constraint equations that connect the selected nodes of one region to the selected elements of the other region.

11.2.3. Generating More Coupled Sets

Once one or more coupled sets exist, additional sets can be generated using these methods:

- To generate new coupled sets using the same node numbers, but different DOF labels, as an existing "pattern" set, use one of these methods:
  Command(s): CPLGEN
  GUI: Main Menu> Preprocessor> Coupling/Ceqn> Gen w/Same Nodes
- To generate new coupled sets using different (uniformly incremented) node numbers, but the same DOF labels as an existing set, use one of these methods:
  Command(s): CPSGEN
  GUI: Main Menu> Preprocessor> Coupling/Ceqn> Gen w/Same DOF

11.2.4. Listing and Deleting Coupled Sets

You can perform two other operations to help you manage your coupled sets:

- To list coupled DOF sets, use one of these methods:
  Command(s): CPLIST
  GUI: Utility Menu> List> Other> Coupled Sets> All CP nodes selected
  Utility Menu> List> Other> Coupled Sets> Any CP node selected

- To delete coupled DOF sets, use one of these methods:
  Command(s): CPDELE
  GUI: Main Menu> Preprocessor> Coupling/Ceqn> Del Coupled Sets

This operation deletes entire sets; you must use the **CPNGEN** command or the **CP** command (or its GUI path) to delete specific *nodes* from coupled sets.

11.3. Additional Considerations for Coupling

Coupling operates in the nodal coordinate system of each node coupled. You should usually keep your nodal coordinate systems consistent.

Degrees of freedom are coupled within a set but are not coupled between sets. You must not allow a degree of freedom to appear in more than one coupled set.

"Grounded" degrees of freedom (that is, DOFs with values specified by **D** or other constraint commands) must not be included in the coupled set.

In a structural analysis, coupling DOFs to create a rigid region can sometimes cause apparent violations of equilibrium. A set of coupled nodes which are not coincident or are not in line with the coupled displacement direction may produce an applied moment that will not appear in the reaction forces.

11.4. What Are Constraint Equations?

Linear *constraint equations* provide a more general means of relating degree of freedom values than is possible with simple coupling. Constraint equations must have the form:

$$\text{Constant} = \sum_{I=1}^{N} (\text{Coefficient}(I) * U(I))$$

where U(I) is the degree of freedom of term (I), and N is the number of terms in the equation.

11.5. How to Create Constraint Equations

You can use several different methods to create constraint equations. These include the **CE** command (the direct method) and other commands such as **CEINTF**, **CERIG**, and **RBE3**. These methods of generating constraint equations are discussed in the following sections.

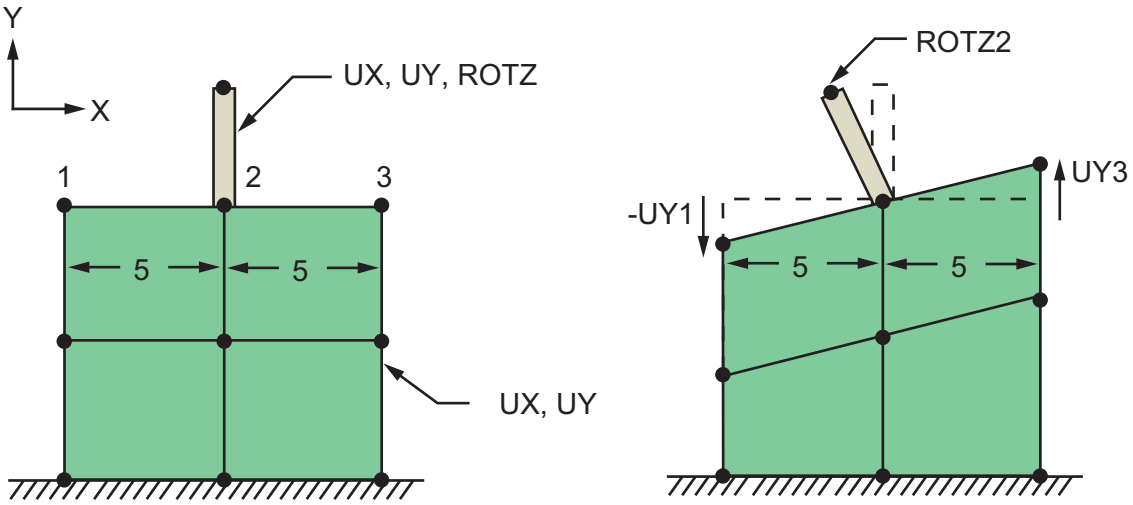
In addition to the methods discussed here, you can use the internal multipoint constraint (MPC) feature of certain contact elements (CONTA171, CONTA172, CONTA173, CONTA174, CONTA175, CONTA176, and CONTA177) to model contact assemblies and kinematic constraints. By this method, the program builds MPC equations internally based on the contact kinematics. See Multipoint Constraints and Assemblies in the *Contact Technology Guide* for more information on how to use this feature.

11.5.1. The Direct Method

You can create constraint equations directly, using one of these methods:
Command(s): CE
GUI: Main Menu> Preprocessor> Coupling/Ceqn> Constraint Eqn

The following example illustrates a typical application of a constraint equation, in which moment transfer capability is created for a connection between a beam element and plane elements:

Figure 11.1: Establishing Relationships Between Rotational and Translational DOF



In this example, node 2 acts as a hinge if no constraint equations are used. To transfer moment between the beam and the plane-stress elements, you can use the following equation:

$$ROTZ_2 = (UY_3 - UY_1)/10$$

This equation would be rewritten in the required format and entered into the program as:

$$0 = UY_3 - UY_1 - 10*ROTZ_2$$

```
CE,1,0,3,UY,1,1,UY,-1,2,ROTZ,-10
```

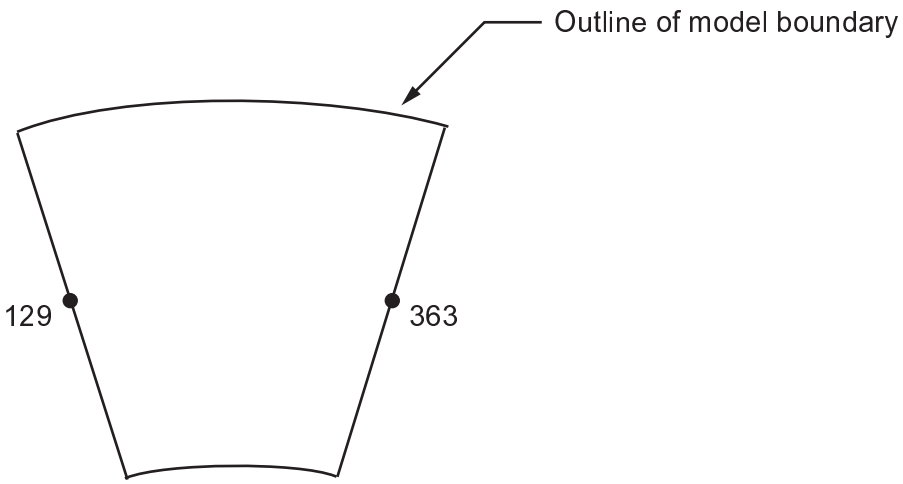
The first unique degree of freedom in the equation is eliminated in terms of all other degrees of freedom in the equation. A unique degree of freedom is one which is not specified in any other constraint equation, coupled node set, specified displacement set, or master degree of freedom set. You should make the first term of the equation be the degree of freedom to be eliminated. Although you may, in theory, specify the same degree of freedom in more than one equation, you must be careful to avoid over-specification. You must also take care to ensure that each node and degree of freedom exists in the model. (Remember that for a degree of freedom to be present at a node, that node must be connected to an element which supplies the necessary degree of freedom.)

11.5.1.1. Periodic Conditions

Often in field analysis, it is desirable to take advantage of antisymmetric or periodic field variation to limit the model size. For static magnetic analyses, this can be accomplished by using ANSYS cyclic symmetry capabilities. See Solving a Magnetic Cyclic Symmetry Analysis for more information. For harmonic or transient magnetic analyses, use the constraint equation method explained here.

A periodic condition is a boundary for which neither the flux-parallel nor flux-normal conditions hold, but rather the potential at one point is of equal magnitude but of opposite sign to that of a point in another location. This condition arises in the analysis of symmetry sectors of motors, for example, where the potentials one pole pitch apart are equal but opposite in sign. In Figure 11.2: Example of Specifying a Periodic Condition (p. 221), suppose node 129 in the outlined symmetry sector is to be constrained as described above with node 363 on the opposite pole pitch.

Figure 11.2: Example of Specifying a Periodic Condition



The constraint equation would read:

$$A_{129} = - A_{363}$$

$$0 = A_{129} + A_{363}$$

The **CE** command used to input this constraint equation would appear as follows:

```
CE,1,0,129,MAG,1,363,MAG,1
```

To automatically apply groups of periodic boundary conditions (**CP** and **CE** commands) for 2-D magnetic analyses, use the **PERBC2D** command macro (refer to Electric and Magnetic Macros in the *Low-Frequency Electromagnetic Analysis Guide* for a discussion of this modeling aide):
Command(s): PERBC2D
GUI: Main Menu> Preprocessor> Loads> Define Loads> Apply> Magnetic> Boundary> Flux Normal> Periodic BCs
Main Menu> Solution> Define Loads> Apply> Magnetic> Boundary> Flux Normal> Periodic BCs

---

Note

Periodic boundary conditions can also be represented in a structural analysis (for example, in a turbine blade model) using **CP** commands on nodes rotated into the cylindrical coordinate system.

11.5.2. Modifying Constraint Equations

To change the constant term of a constraint equation in either PREP7 or SOLUTION, use one of these methods:
Command(s): CECMOD
GUI: Main Menu> Preprocessor> Coupling/Ceqn> Modify ConstrEqn
Main Menu> Preprocessor> Loads> Load Step Opts> Other> Modify ConstrEqn
Main Menu> Solution> Load Step Opts> Other> Modify ConstrEqn

If you need to change any of the other terms of a constraint equation, you must use the **CE** command (or the corresponding GUI path) in PREP7, before you start your solution.

11.5.3. Direct vs. Automatic Constraint Equation Generation

An example that appeared earlier in this chapter illustrated how you can use the **CE** command to create constraint equations directly, one at a time.

Three other operations, described below, automatically generate multiple constraint equations for you.

11.5.3.1. Creating a Rigid Region

The **CERIG** command defines a "rigid region" by writing constraint equations to define rigid lines linking a designated retained (or "master") node to a number of removed (or "slave") nodes.
Command(s): CERIG
GUI: Main Menu> Preprocessor> Coupling/Ceqn> Rigid Region

By setting *Ldof* to ALL on the **CERIG** command (default), this operation will generate three equations for each pair of constrained nodes in 2-D space. These equations define the three rigid body motions in global Cartesian space (UX, UY, ROTZ). In order to create a rigid region on a 2-D model, you must make sure that the X-Y plane is the rigid plane and that UX, UY, and ROTZ degrees of freedom are available at each constrained node. This operation will similarly generate six equations for each pair of constrained nodes in 3-D space. All six degrees of freedom (UX, UY, UZ, ROTX, ROTY, and ROTZ) must be available at each constrained node.

Entering other labels in the *Ldof* field will create different effects. If this field is set to UXYZ, the program will write two constraint equations in 2-D (X, Y) space and three constraint equations in 3-D (X, Y, Z) space. These equations will be written in terms of the slave nodes' translational degrees of freedom, and in terms of the master node's translational and rotational degrees of freedom. Similarly, the RXYZ label allows you to generate a partial set of equations that omit the slave nodes' translational degrees of freedom. The other available *Ldof* labels will generate other types of constraint equations.

In general, your slave nodes need have only the degrees of freedom called for by *Ldof*, but your master node must have all applicable translational and rotational degrees of freedom (that is, UX, UY, ROTZ for 2-D; UX, UY, UZ, ROTX, ROTY, ROTZ for 3-D). For models that are made up of elements having no rotational degree of freedom, you might consider adding a dummy beam element to provide rotational degrees of freedom at the master node.

11.5.3.2. Tying Dissimilarly Meshed Regions Together

You can tie dissimilarly meshed regions together via the **CEINTF** command, or you can use contact elements with the internal multipoint constraint (MPC) algorithm.

11.5.3.2.1. Using the CEINTF Command

You can generate constraint equations connecting the selected nodes of one region to the selected elements of another region via the **CEINTF** command (**Main Menu> Preprocessor> Coupling/Ceqn> Adjacent Regions**). This operation ties together regions with dissimilar mesh patterns. At the interface location between two regions, select the *nodes* from the denser mesh region, A, and select the *elements* from the sparser mesh region, B. The degrees of freedom of region A nodes are interpolated with the corresponding degrees of freedom of the nodes on the region B elements, using the shape functions of the region B elements. Constraint equations are then written that relate region A and B nodes at the interface. ANSYS allows two tolerances on the location of these nodes. Nodes which are outside the element by more than the first tolerance are not accepted as being on the interface. Nodes that are closer than the second tolerance to an element surface are moved to that surface.

Certain limitations affect the **CEINTF** command: stress or thermal flux might not be continuous across the interface. Nodes in the interface region must not have specified displacements.

11.5.3.2.2. Using Contact Elements

To learn more about tying dissimilarly meshed regions together via contact elements and the internal MPC algorithm, see Multipoint Constraints and Assemblies.

11.5.3.3. Generating Sets of Constraint Equations from Existing Sets

You can issue the **CESGEN** command to generate sets of constraint equations from existing sets. All node numbers within the existing sets are then incremented to generate the additional sets. The labels and coefficients of the additional sets remain the same as those of the original sets.
Command(s): CESGEN
GUI: Main Menu> Preprocessor> Coupling/Ceqn > Gen w/same DOF

11.5.4. Listing and Deleting Constraint Equations

You can list and delete your constraint equations.

- To list constraint equations, use one of these methods:
  Command(s): CELIST
  GUI: Utility Menu> List> Other> Constraint Eqns> All CE nodes selected
  Utility Menu> List> Other> Constraint Eqns> Any CE node selected

- To delete constraint equations, use one of these methods:
  Command(s): CEDELE
  GUI: Main Menu> Preprocessor> Coupling/Ceqn> Del Constr Eqn

11.5.5. Program Modification of Constraint Equations

During the solution, the user-defined constraint equations (CEs) may be modified as follows:

- CEs that are applied to DOFs which are not active (for example, a CE relating rotational DOFs on nodes with only translational DOFs) are deleted.

- CEs for which all DOFs are constrained (**D** command) are deleted.

- CEs that have DOFs in another equation have their terms reordered so that they all have a common retained DOF.

- CEs which are chained together are merged into a single CE.

- CEs which are internal to the solution process are generated by MPC contact and by cyclic symmetry. These CEs cannot be listed or deleted.

11.5.6. Troubleshooting Problems with Constraint Equations

Overconstrained problems for which there is no unique DOF that can be solved typically generate an error message similar to one of the following:

- `Constraint equation set is defective.`

- `Constraint equation is circular.`

- `Constraint equation has no unique degree of freedom.`
- `Contact overconstraint may occur.`

In addition to CE-specific error messages, you may also notice "small (or zero) pivot" messages from the solver. Such overconstraints may be caused by one of the following conditions:

- **Duplicate CEs are specified for the same DOFs.**

  Work-around: Delete any duplicate specifications, or issue an **NUMMRG**,CE command to compress them out.

- **DOFs in a CE are also present in a coupled (CP) set.**

  Work-around: Delete the CP set and include the DOF in the CE to obtain the desired response.

- **CEs are chained together in such a way that they form a "circular" set.**

  Typically, this condition occurs when you define **CEINTF** and/or MPC contact on adjacent surfaces.

  Work-around: Perform the **CEINTF** operation, or specify the contact region encompassing both surfaces simultaneously rather than individually.

11.6. Additional Considerations for Constraint Equations

All constraint equations are based on small rotation theory. Therefore, their use in large-rotation analyses [**NLGEOM**] should be restricted to cases where the directions of the DOF included in constraint equations do not change significantly.

The presence of constraint equations can produce unexpected reaction and nodal force results. Please see The General Postprocessor (POST1) in the *Basic Analysis Guide* for a related discussion.

The constant terms in the constraint equations are considered as boundary conditions. Hence, they result in harmonically varying loads when included in a harmonic analysis.

Chapter 12: Combining and Archiving Models

It is sometimes necessary to combine two or more separate models, and it is necessary to know how to archive the models that you have made or the analyses you have performed. The following related topics are available:

12.1. Combining Models

You may need to combine two or more separate models if you are working on a portion of a model while another person works on a different portion of the same model. Or perhaps you have subdivided a large modeling task into smaller separate tasks, which created several separate models. One way to combine these models would be to merge all the input (commands) together, if you have kept copies of the input files. Conflicts may result with this method, however, because entities on different files may share the same number, material properties may overlap, etc.

One alternative method is to use the **CDWRITE** command to write out ASCII files that you can combine with the **CDREAD** command:

- To write data to an ASCII file, use one of these methods:
  Command(s): CDWRITE
  GUI: Main Menu> Preprocessor> Archive Model> Write

- To read the data back in, use one of these methods:
  Command(s): CDREAD
  GUI: Main Menu> Preprocessor> Archive Model> Read

The advantage of this method is that the combination of the write and read operations takes care of conflicting data by automatically writing appropriate **NUMOFF** commands to each file that is produced. When these files are read in, the **NUMOFF** commands prevent conflicts in data numbers by offsetting existing data numbers. The data that is being read in retains its original numbering. You can use the **NUMCMP** command (**Main Menu> Preprocessor> Numbering Ctrls> Compress Numbers**) to remove gaps in numbering that can result from these **NUMOFF** operations.

On large models, you can save time by writing out only the portion of the models that you wish to combine. For instance, with the **CDWRITE** operation, you have the capability of writing out only the solid model information or only the database information. The database information consists of the finite element model without any solid model or solid model loading information. If you are planning on combining two unmeshed solid models, consider saving only the solid model information with the **CDWRITE** operation. You can then use the **CDREAD** operation to read in the solid model files.

12.2. Archiving Models

What is the best way to keep, or archive the models that you have made or the analyses you have performed? You can save your model , a single load case, and a set of solution option settings by saving the log file, the database file, or the file(s) produced by the write operation [**CDWRITE**]. Multiple solutions

and postprocessing steps can be archived by saving the log file. There are advantages and disadvantages to each type of file; they are described below.

12.2.1. Log File (File.LOG)

12.2.1.1. Pros

This file is probably the best file to save in terms of compactness. Also, this file is a record of the commands you used to create your model, so you will probably recognize the steps that you used, and why you did operations in a particular order. Also, since the file is saved in text file format (normally ASCII), you can transfer it from one machine to another through such methods as E-mail, etc. In addition, since you can modify these files with any text editor, you can change your model by changing this file, and you can add **/COM** commands (comment lines) to describe your input. This will help you understand the input at a later date. Parametric models (and thus models used in optimization analyses) can be archived using the log file. Finally, this is the best file to give to other people (perhaps your ASD) who are helping you with questions or problems.

Note

File.LOG is automatically created during an ANSYS session. If this file is lost or corrupted, you can write a command log file for the current ANSYS database using the **LGWRITE** command (**Utility Menu> File> Write DB Log File**). See the *Operations Guide* for more information on using **LGWRITE**.

12.2.1.2. Cons

You will have to rerun the input in this file in order to get a database. Because such things as entity numbering, meshes, etc. can change between different hardware systems or between releases of ANSYS, you'll probably need to rerun the input at the same release of the program that you originally used to create the model. Also, if you created your model interactively, you may have difficulty running the input on this file in batch mode to recreate the model. If there are any errors in the input, they can terminate a batch run, which means that your model will not be built completely. If you encounter this problem, run your input interactively with the **/INPUT** command (**Utility Menu> File> Read Input from**).

12.2.2. Database File (File.DB)

12.2.2.1. Pros

The database file can be resumed [**RESUME**] within the ANSYS version that the file was created in. As long as you are resuming the file into the same ANSYS version that it was created in, you do not need to manipulate or modify the file in any way.

Note

Although not guaranteed, you can also usually resume a database file created in the previous version of ANSYS into the current version. For example, you can probably resume an ANSYS 5.2 database file into ANSYS 5.3 without encountering problems. However, ANSYS is not expected to resume an ANSYS 5.2 database file into ANSYS 5.4 or later.

12.2.2.2. Cons

Large models can produce large database files, which can soon cause you to run out of disk space. Also, because this file is saved in binary format (IEEE), transfer from one machine to another is sometimes more difficult than with a text file.

12.2.3. CDWRITE File(s)

12.2.3.1. Pros

Relatively compact text file(s) (with the .cdb extension) are produced by the **CDWRITE** operation. Since **CDWRITE** saves the current model in terms of geometric and finite element entities (rather than the commands used to create the model), nearly all hardware platforms and or program release dependencies are eliminated. **CDWRITE** files can typically be used to recreate a model on any hardware platform and in any upward compatible release of the program. Also, because these files can be modified with a text editor, you can put descriptive comments in these files to help you identify and understand them at a later date. Imported files used to create the model need not be saved.

12.2.3.2. Cons

Files produced by **CDWRITE** are essentially a text file format dump of the database. While ANSYS commands (and possibly IGES information) are contained in the files, they may not be the same commands that you used to create your model, and they will not be in the same order as the commands that you used. For this reason, it is difficult (and not recommended) to modify these files when you are trying to change a model. Parametric model information is not saved, so changing the model by modifying parameter values is not possible. For this reason, it is also not possible to save a model to be used for an optimization analysis using **CDWRITE**. Also, to recreate your model, you must read these files in with the **CDREAD** command, which can take a moderate amount of time for large models.

Note that, if a set of data exists prior to the **CDREAD** operation, that data set is offset upward to allow the new data to fit without overlap. The **NOOFFSET** command allows this offset to be ignored on a set by set basis, causing the existing data set to be overwritten with the new data set

Chapter 13: Interfaces With Other Programs

Interface software allows you to exchange data between the ANSYS program and other application programs. The software may be made available as an independent package or built into the application programs. The following topics are available:

13.1. Interfacing With Computer Aided Design (CAD) Products

Many CAD programs have direct interfaces with the ANSYS program through software written by ANSYS, Inc. or by the CAD vendors. See the *Connection User's Guide* for information on ANSYS connection functionality. For other programs, consult your CAD software vendor.

13.2. Other Interfaces

Interface software is also available for software in the following categories:

- Kinematics
- Computational Fluid Dynamics (CFD).
- Graphics and publishing
- Preprocessing and postprocessing
- Injection molding analysis
- Acoustics and noise reduction
- Experimental modal analysis
- Fracture and fatigue
- Multibody systems
- Crash analysis and metal forming
- Dynamics

Typically, the interface software for such applications is available from the respective vendors.

Index

D

E

F

G

H

I

K

N

O

T

U

V

W