AUTOMATIC

DYNAMIC

INCREMENTAL

NONLINEAR

ANALYSIS

ADINA Handbook

ADINA 9.6

April 2020

ADINA R & D, Inc.

ADINA SYSTEM 9.6

THE ADINA HANDBOOK

ADINA R&D, INC.

Copyright © 2020 ADINA R&D, Inc.
71 Elton Avenue
Watertown, MA 02472 USA
tel. (617) 926-5199
fax (617) 926-0238
http://www.adina.com/

ADINA R&D, Inc. owns both this software program system and its documentation. Both the program system and the documentation are copyrighted with all rights reserved by ADINA R&D, Inc.

The information contained in this document is subject to change without notice. ADINA R&D, Inc. makes no warranty whatsoever, expressed or implied that the Program and its documentation including any modifications and updates are free from errors and defects. In no event shall ADINA R&D, Inc. become liable to the User or any party for any loss, including but not limited to, loss of time, money or goodwill, which may arise from the use of the Program and its documentation including any modifications and updates.

ADINA is a registered trademark of K.J. Bathe / ADINA R&D, Inc. All other product names are trademarks or registered trademarks of their respective owners.

March 2020

Table of Contents

Introduction

This handbook is written as a desktop reference for both newcomers and experienced ADINA users. Topics are organized by typical work flows, and special attention is given to common pitfalls and solutions as they would be encountered.

Currently, this handbook covers:

- Importing, creating, and manipulating geometry using the ADINA User Interface (AUI)
- A detailed chapter demonstrating ADINA's powerful meshing features
- A description of ADINA's Fast Graphics Mode
- A chapter describing the differences between solver settings for ADINA and SOL 601/701

The reader will find many simple examples demonstrating how to effectively use ADINA's commands and command options. In most cases, examples include detailed figures and illustrations. These examples can be copied and pasted directly into the AUI Command Window's input line, located just beneath the `ADINA-IN(1)>` prompt, or adapted for use into the reader's own model.

It is worth noting that this handbook is not intended to be a replacement for the ADINA system's detailed Command Reference Manuals, Theory and Modeling Guides, and Primer Problems. Rather, its aim is to more effectively bring to bear upon the reader's specific problem all of ADINA's features and documentation.

1 Geometry Definition and Manipulation

Defining the geometry is the starting point for an engineering analysis. The ADINA User Interface (AUI) offers geometric primitives in its own native geometry engine or via ADINA-M's Parasolid or OpenCascade geometry engines.

1.1 AUI Native Geometry

Points, lines, surfaces, and volumes are the AUI's native geometry primitives. Each of these entities is relatively simple[1] and may be meshed individually. More complex geometries can be created by combining simpler geometries that share common boundaries.

[1] In fact, the native AUI geometry is sometimes referred to as 'simple' geometry

Points can be created with command `COORDINATES POINT`. The `POINT` command may also be used to create points in relation to other geometry. For example, `POINT BETWEEN` creates a geometry point between two other points or nodes, and `POINT CENTER` creates a point that lies at the center of the circle passing through three other points or nodes. `POINT NODE` simply creates geometry points at nodes. Meshing a geometry point with command `GPOINT` creates a node at the point's location and associates[2] the node with the point. The command `NODE-SNAP` moves (snaps) the closest node to a geometry point and associates that node with the point.

[2] *Associating* nodes with geometry assigns fixities, constraints, loads, skew system, etc. defined on the geometry to those nodes.

Lines can be created with the `LINE` family of commands and meshed with command `GLINE`. Meshing a line generates nodes corresponding to the subdivisions of the line (created by command `SUBDIVIDE LINE`) and one-

dimensional elements (*e.g.*, trusses, beams, etc.) connecting them.

Surfaces can be created with the SURFACE family of commands in several ways:

- Connecting three or four points possibly already connected by existing lines (command SURFACE VERTEX).
- Patching 3 or 4 lines (command SURFACE PATCH).
- Extruding a line along a vector (command SURFACE EXTRUDE).
- Revolving a line about an axis (command SURFACE REVOLVED).
- Using a grid of geometry points (command SURFACE GRID).
- Transforming or copying an existing surface (command SURFACE TRANSFORMED).
- Converting a body face into a surface (command SURFACE FACE).

Volumes can be created with the VOLUME family of commands and meshed with command GVOLUME. Volumes can be formed by:

- Connecting up to eight points possibly already connected by existing lines and surfaces (command VOLUME VERTEX).
- Extruding a surface along a vector (command VOLUME EXTRUDE).
- Sweeping a surface along a line/curve (command VOLUME SWEEP).
- Revolving a surface about an axis (command VOLUME REVOLVED).
- Transforming or copying an existing body (command VOLUME TRANSFORMED).
- Converting a body into a volume (command VOLUME BODY).

1.2 *Bodies*

The AUI optionally includes ADINA-M (the ADINA Modeler). ADINA-M uses either a Parasolid (PS) or OpenCascade (OCC) kernel to provide solid modeling capabilities.

Bodies are composed of points, edges, and faces. Bodies do not share entities with other bodies or volumes, except points (body faces and edges are never shared). For example, if a body connects to another body at a face, each body has its own face even though the two body faces are geometrically identical. Sheet bodies and body faces are allowed to share both edges and points. Bodies (of any dimension) and points, lines, surfaces, and/or volumes can coexist in the AUI.

A manifold is a mathematical term for an object that locally resembles a line, a plane, or a space.[3] A manifold body is one for which each edge is associated with exactly two faces. A manifold face is one for which each point is associated with exactly two edges. Figure 1.1 helps illustrate the differences between manifold and non-manifold geometries.[4]

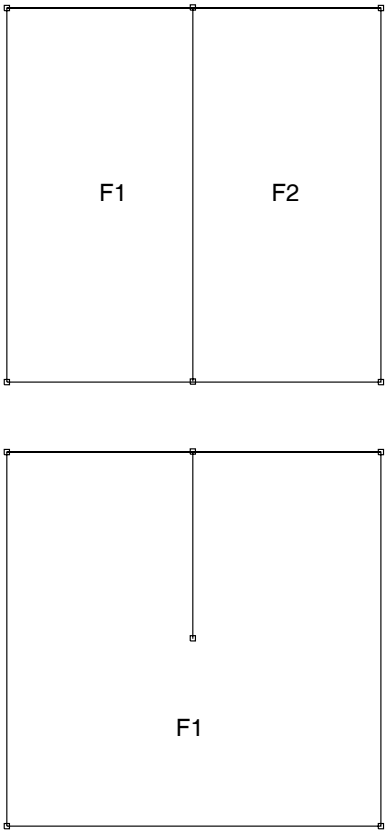


Figure 1.1: Example of manifold (top) and non-manifold (bottom) geometries. Each geometry point on the manifold face has exactly two edges associated with each face. The top-center point on the non-manifold face has three edges associated with the face.

Bodies can be imported into ADINA-M in either Parasolid or OpenCascade formats. Imported bodies may also be modified. IGES files can be imported as ADINA-M bodies (command `LOADIGES`). STEP files can be imported (using command `IMPORTIGES`) if ADINA-M is coupled with OpenCascade.

The family of `BODY` commands invokes ADINA-M and defines a body. Three-dimensional bodies can be:

- Created using primitive bodies (for example, commands `BODY BLOCK`, `BODY CYLINDER`, etc.).
- Created by sweeping a face along a line (command `BODY SWEEP`), revolving a face about an axis (command `BODY REVOLVED`), or lofting through a set of surfaces or faces (command `BODY LOFTED`).
- Created using Boolean operations on bodies (commands `BODY MERGE`, `BODY SUBTRACT`, and `BODY INTERSECT`).
- Modified (*e.g.*, commands `BODY PARTITION` or `BODY SECTION`, etc.).

[3] A mathematically rigorous definition of a manifold is beyond the scope of this handbook, but physically, *manifold* essentially means *manufacturable*. That is, it is possible to machine a manifold shape from a single block of material, whereas it is not possible to do so for a non-manifold shape.

[4] Non-manifold bodies are permitted in the AUI, but meshing non-manifold bodies is not recommended.

When using Boolean operations, users may keep or discard the original bodies and/or preserve imprinted or original edges, as desired. For example, if several adjacent bodies are to be merged, but the user wishes to preserve their original internally-shared edges (*e.g.*, for manual Subdivision – see page 22), then the user can invoke the Boolean operation via `BODY MERGE MERGE-IMPRINT = NO`, as shown in command input 1.1. Figure 1.2 shows the effect of `MERGE-IMPRINT = NO` (panel *b*) and `MERGE-IMPRINT = YES` (panel *c*).

Command Input 1.1: Boolean merge of three separate bodies (as shown in Figure 1.2, panel *a*) into one. The option `BODY MERGE MERGE-IMPRINT = NO` (Figure 1.2, panel *b*) preserves the internal edges rather than merges them.

```
feprogram program=adina
body block name=1 option=centered position=vector,
     cx1=0.5 cx2=0.25 cx3=0.0 dx1=1.0 dx2=0.5 dx3=1.0
body block name=2 option=centered position=vector,
     cx1=0.5 cx2=1.0 cx3=0.0 dx1=1.0 dx2=1.0 dx3=1.0
body block name=3 option=centered position=vector,
     cx1=1.25 cx2=0.75 cx3=0.0 dx1=0.5 dx2=1.5 dx3=1.0
*
body merge name=1 merge-imprint=no
2
3
```

It is possible to create two-dimensional bodies in ADINA-M by combining lines using the command `BODY SHEET`. Although this is a two-dimensional entity, it is still considered a body by ADINA-M.

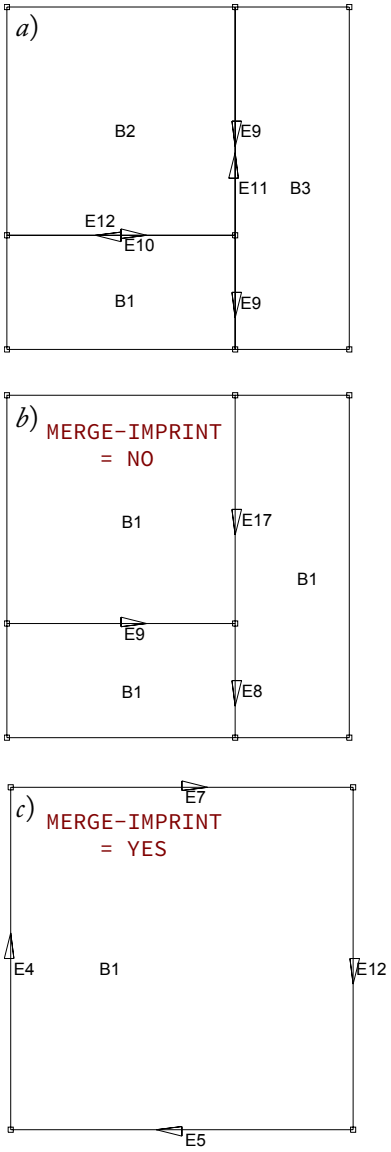


Figure 1.2: Illustration of the effects of `MERGE-IMPRINT`. Three bodies (panel *a*) are to be joined – note the overlapping internal edges. By merging the bodies using `MERGE-IMPRINT = NO` (panel *b*), the internal edges are preserved but are no longer overlapping. This is useful when internal subdivisions are desired. If the bodies are merged with `MERGE-IMPRINT = YES` (panel *c*), the internal edges vanish.

1.2.1 *Face-linking*

Although two adjoining bodies might interface at two geometrically identical faces (or a body's face exactly matches an adjoining geometry volume's surface), the meshes for those two adjacent entities may not be *congruent*[5], especially when the free-form mesher is used. In other words, at the interface, nodes from one mesh might not *coincide* with the nodes from the other mesh. For more information about nodal coincidence and equivalence, see Nodal Coincidence on page 77.

For example, command input 1.2 instructs ADINA to

[5] A congruent mesh is continuous from one region to the other.

construct two adjacent bodies and to mesh body 1 with mapped meshing and body 2 with free-form meshing.

Command Input 1.2: Creating two adjacent bodies and meshing one using free-form meshing and the other using mapped meshing.

```
feprogram program=adina
body block 1 dx1=1 dx2=1 dx3=1
body block 2 dx1=1 dx2=1 dx3=1 cx1=1
*
subdivide body 1 mode=length size=.2
subdivide body 2 mode=length size=.2
*
egroup threedsolid 1
egroup threedsolid 2
*
gbody 1 nodes=4 group=1 meshing=mapped
gbody 2 nodes=4 group=2 meshing=free-form
```

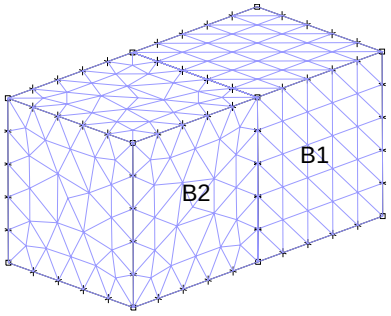


Figure 1.3: Two adjacent, meshed bodies. The body on the left was free meshed and the body of the right was map meshed. The nodes at the adjoining edges appear coincident.

Figure 1.3 shows the resulting meshes. A quick inspection might lead to the conclusion that the nodes at the connecting interface are coincident and that the two meshes are congruent. However, a more detailed visual inspection with front faces culled (see Figure 1.4) reveals that, in fact, the nodes on the planar interface are *not* coincident. For additional ways of checking for nodal coincidence, see Checking for coincidence on page 82.

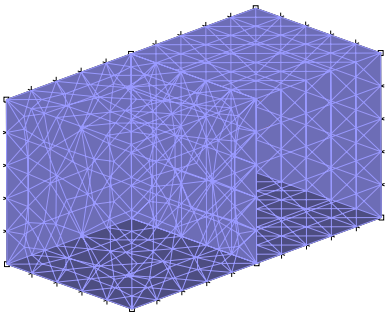


Figure 1.4: Culled rendering revealing incongruent meshes. The visible interface indicates that the nodes at the planar interface are not coincident.

This is only an issue when there are two distinct 2-dimensional entities at the interface (one corresponding to each 'side' of the interface) as when:

- Two bodies share a 2-dimensional interface (at least one body face for each body)
- A body and a geometry volume share a 2-dimensional interface (at least one body face and at least one geometry surface)

Two adjacent geometry volumes sharing a single geometry surface do not require linking.

The solution here is to link the faces using the command FACELINK prior to meshing (see command input 1.3). This command ensures nodal coincidence by instructing the mesher to use the same triangulation on the interface for both meshes. Note that face linking will work only if the adjacent faces to be linked are geometrically and topologically identical.[6]

[6] *Topologically identical* faces are of the same shape and share the same points and edges.

Command Input 1.3: Continuation of command input 1.2, deleting the mesh for body 2 and remeshing after using FACELINK.

```
eldelete body 2
*
facelink option=all
*
gbody 2 nodes=4 meshing=free-form group=2
```

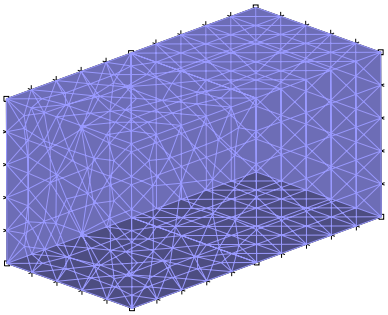


Figure 1.5: The effects of face-linking. After linking the faces between adjacent bodies, the meshes are congruent.

Figure 1.5 illustrates the effects of FACELINK. The culled rendering reveals no visible interface, indicating that the connecting nodes are now coincident and equivalent. The two meshes are congruently joined.

Command input 1.4 and Figure 1.6 show what can occur when attempting to link non topologically equivalent faces. The culled rendering in Figure 1.6 (bottom) reveals that not all nodes on the interfaces are coincident and that the resulting mesh is not congruent.

Command Input 1.4: Attempting to face link non topologically equivalent faces. Figure 1.6 shows that FACELINK fails to link the faces between body 1 (large face) and bodies 2 and 3 (geometrically smaller faces).

```
feprogram program=adina
body block name=1 option=centered position=vector,
     cx1=-0.25 cx2=0.0 cx3=0.0 dx1=0.5 dx2=1.0 dx3=1.0
body block name=2 option=centered position=vector,
     cx1=0.25 cx2=0.0 cx3=-0.25 dx1=0.5 dx2=1.0 dx3=0.5
body block name=3 option=centered position=vector,
     cx1=0.25 cx2=0.0 cx3=0.25 dx1=0.5 dx2=1.0 dx3=0.5
*
subdivide body name=1 mode=length size=0.1
subdivide body name=2 mode=length size=0.1
subdivide body name=3 mode=length size=0.1
*
egroup threedsolid
*
facelink option=all
*
gbody 1 nodes=4 meshing=free-form
gbody 2 nodes=4 meshing=free-form
gbody 3 nodes=4 meshing=free-form
```

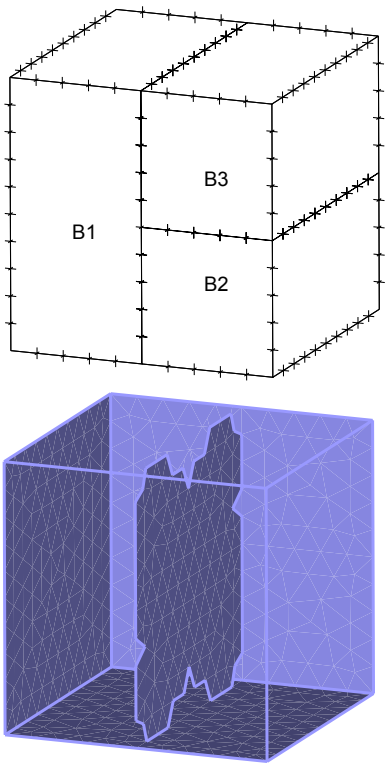


Figure 1.6: The result of attempting to link non topologically equivalent faces. FACELINK cannot link the larger face to either of the smaller faces. The culled rendering shows the resulting incongruent mesh.

Using command BODY PROJECT, the large face on body 1 can be split into two faces, which are topologically identical to the corresponding faces on bodies 2 and 3. Command in-

put 1.5 demonstrates how to use BODY PROJECT. Figure 1.7 illustrates the steps taken in command input 1.5. After face linking, the three bodies can be congruently meshed.

Command Input 1.5: Continuation of command input 1.4 demonstrating the use of BODY PROJECT for projecting a line onto Face 6 of Body 1 to create topologically identical faces for linking to corresponding faces on bodies 2 and 3. The resulting mesh is congruent across all shared interfaces (see Figure 1.7).

```
eldelete body 1
eldelete body 2
eldelete body 3
*
line straight p1=13 p2=14
*
body project 1 6
1
*
subdivide body name=1 mode=length size=0.1
*
facelink option=all
*
gbody 1 nodes=4 meshing=free-form
gbody 2 nodes=4 meshing=free-form
gbody 3 nodes=4 meshing=free-form
```

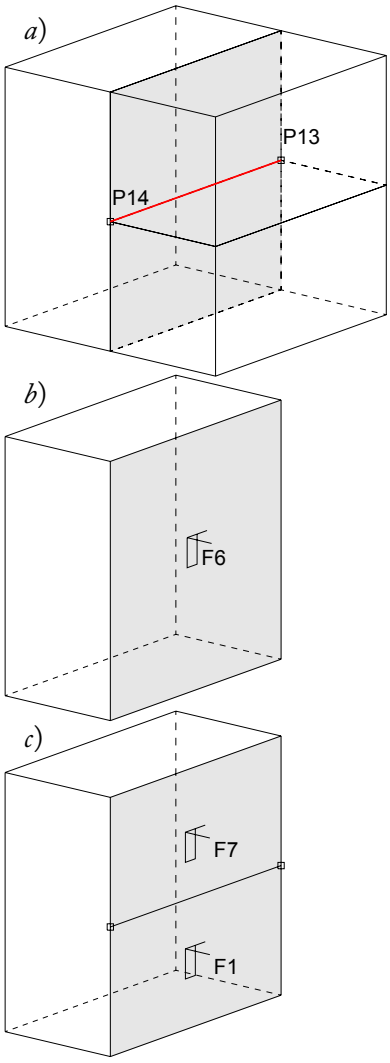


Figure 1.7: Illustration of steps taken in command input 1.5. The large shaded face is to be split into two faces along the red line (panel *a*). After defining a line connecting points 13 and 14, this large face (face 6 of body 1 - panel *b*) is split into two faces using the command BODY PROJECT. The result is shown in panel *c*.

1.2.2 *De-featuring and body cleanup*

Unnecessary detail can be removed using the command BODY DEFEATURE. The related command BODY-CLEANUP[7] de-features a body by changing its topology but without altering the actual geometry. BODY-CLEANUP relies on two removal operators:

1. Remove body edge (command REM-EDGE). Because the geometry of the body is unchanged, the body edge should be small in length.
2. Remove body face (command REM-FACE). A body face can be removed only if it has exactly two body edges, or in other words, if it is a degenerate face which became degenerate due to bounding body edges being removed by command REM-EDGE in the previous step.

The command BODY-CLEANUP removes any body edge with length below the size threshold (parameter SIZE) and any

[7] BODY-CLEANUP requires that features to be removed be small.

degenerate body face with width below the size threshold. The body's topology can be restored to its original state using BODY-RESTORE.

Command input 1.6 creates a body with a small chamfer, subdivides it, and free-form meshes it using tetrahedral elements.

Command Input 1.6: Commands for generating geometry and mesh shown in Figure 1.8.

```
feprogram program=adina
body block 1 dx1=1.0 dx2=1.0 dx3=1.0
body chamfer name=1 r1=0.01 r2=0.01 option=edge
10 0
*
subdivide body 1 mode=length size=0.2
egroup threedsolid
*
gbody 1 nodes=4
```

Figure 1.8 illustrates the chamfered body and the resulting mesh.[8] The chamfer leads to the creation of thin elements. If the chamfer is not important for the analysis, it is best to remove the detail. Command input 1.7 invokes BODY-CLEANUP to remove the chamfer from the body and free-form meshes the body. Figure 1.9 shows the result.

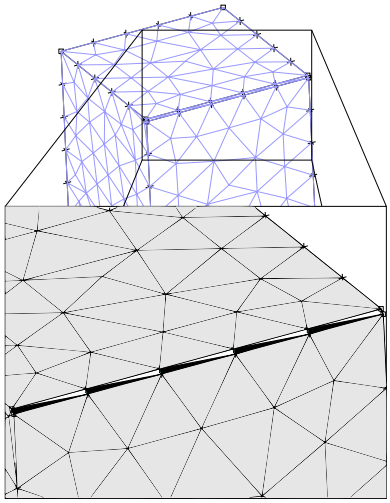


Figure 1.8: A thin chamfer results in small, thin sliver element faces (see inset: shown in alternating black and white), necessary to capture the small geometric detail.

[8] Thin body faces like this chamfer are often created as artifacts resulting from a Boolean operation on complex models.

Command Input 1.7: Commands for generating geometry and mesh shown in Figure 1.9.

```
feprogram program=adina
body block 1 dx1=1.0 dx2=1.0 dx3=1.0
body chamfer name=1 r1=0.01 r2=0.01 option=edge
10 0
*
body-cleanup 1 size=.05
*
subdivide body 1 mode=length size=0.2
egroup threedsolid
*
gbody 1 nodes=4
```

The BODY MERGE command can be used to clean up repeated edges between two sheet bodies. For more information, see command input 1.1 and Figure 1.2 on page 14.

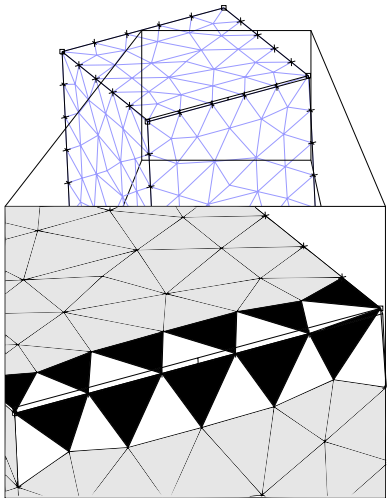


Figure 1.9: The effects of BODY-CLEANUP. Note the lack of small sliver element faces near the chamfer as were present in Figure 1.8. The inset shows the much larger element faces (shown in alternating black and white) along the edge.

1.3 STL Bodies

The STL (STereoLithography) file format describes an object's surface geometry with triangular facets; there is no topological information of the object being represented. That is, the entities which one would find in a general body, namely body edges or body faces, are not present.

The advantage of having an STL representation supplemented with a topological representation (STL body) lies in the ease in which mesh densities can be set and loads and fixities applied. In essence, whatever can be done to a general body can be done to an STL body, though STL bodies can only be meshed with tetrahedral (command GBODY) or hexahedral elements (command BHEXA). See STL Body Free-Form Meshing on page 71.

1.3.1 Creating an STL body from an STL file

The AUI command LOAD-STL loads STL files in either ASCII or binary formats and augments the purely graphical representation with a topological representation by creating a corresponding body for the object and grouping facets into body faces, segments into body edges, and having some key vertices (where body edges meet) be points. When the angle between adjacent facets is greater than a threshold (parameter RIDGEANG), ADINA assumes that common segments belong to a body edge. From there, one can create body edges, body faces, and points.[9]

The object described in the STL file should be a single 'watertight' 3D object that is *manifold*, meaning that any segment must be connected to exactly two facets. In some cases (typically when the STL file is of poor quality), the tolerance NCTOLERANCE can be adjusted if there are under-connected or over-connected segments in the STL file and if larger than expected variations in vertex coordinates need to be taken into account. Figure 1.10 shows the underlying geometry that defines an STL body.

[9] Clearly, topology that is built in this manner is not ideal, especially if the STL file originates from a CAD model which previously had topology.

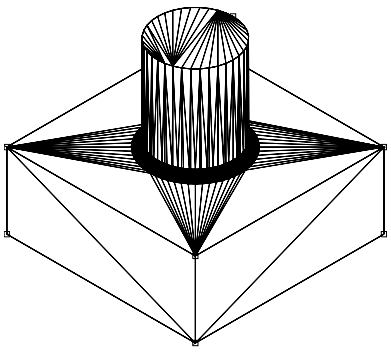


Figure 1.10: Underlying geometry of an imported STL file, visible after using command BODY-DISCREP.

1.3.2 *Converting a Parasolid body into an STL body*

It can be useful to obtain an STL body from a Parasolid body:

- A Parasolid body must be converted into an STL body if an all-hexahedral mesh is to be created using command BHEXA.
- Converting to an STL body may also be convenient for generating tetrahedral elements using command GBODY after adapting the discrete representation on the surface with command BODY-DSCADAP.
- Body edges can easily be eliminated from STL bodies. This is useful in cases when the de-feature tools cannot be used.

The conversion process (command CONVERT-STL) is straightforward since the tessellation that is used to display the body defines the set of triangular facets that make up the surface of the object. The fineness of this tessellation is controlled by parameter PCCANG. The lower the parameter, the finer the tessellation. The topology of the current Parasolid body is unchanged, which means that no topological information is lost during the conversion process. Since the topology does not change, the STL body will look the same as the original Parasolid body.

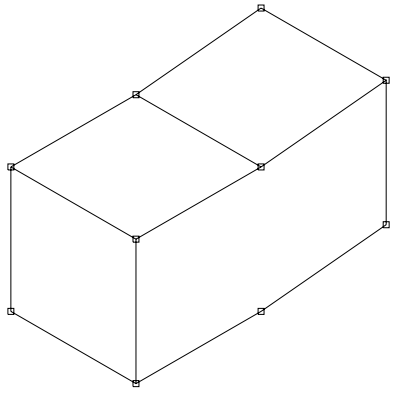


Figure 1.11: Two distinct faces on the top of an STL body.

1.3.3 *Eliminating edges from STL bodies*

Because STL bodies are composed of triangular facets, it is easy to eliminate edges from the body and merge adjacent faces. Note that when the body is regenerated from the surface mesh, the numbering of the body faces, edges, and points will change.

There are two ways to eliminate edges from an STL body:

- Eliminating specific edges with command STL ELIM-EDGE.
- Specifying an angular threshold via command STL ELIM-EDGES-ANGLE to eliminate any edge for which the two connected faces have normals that differ by less than the threshold parameter ANGLE.

Figure 1.11 shows an STL body with two connected, nearly coplanar body faces, and Figure 1.12 shows the same STL

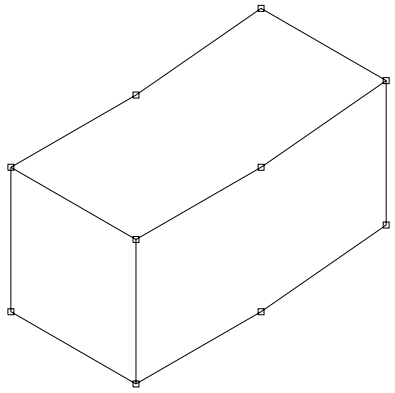


Figure 1.12: The two faces visible in Figure 1.11 have been merged following the use of command STL ELIM-EDGES-ANGLE.

body after using STL ELIM-EDGES-ANGLE. The body edge connecting the two nearly coplanar body faces has been removed and the faces merged.

1.3.4 *Boundary cells*

A volume's geometry and topology can be defined by boundary cells; these cells must all be either triangular facets (defined by three existing nodes) or quadrilateral facets (four existing nodes) with normals oriented toward the interior of the volume. The BCELL command creates a set of boundary cells which can then be meshed with either tetrahedral elements (if the boundary cells are triangular) or mixed elements (if the boundary cells are quadrilateral) using command GBCELL. Upon meshing, a node set and element face set (of the same name or label) will be created to facilitate application of fixities and/or loads.

When a Nastran file is loaded using command NASTRAN-ADINA, it is possible to create boundary cells sets from shell elements according to the Property Identification Number or PID (parameter BCELL in command NASTRAN-ADINA). Shell elements with the same PID are put into the same boundary cell set.

2 Meshing

ADINA offers powerful meshing tools to help users generate high quality meshes of various element types.

Note that much of this chapter is equally applicable to both structural and fluid meshes, but special attention is given to Boundary layer meshing on page 55 (2D) and page 68 (3D).

2.1 Mesh Size Control

Users can control local mesh density in a variety of ways. Depending on which is most convenient, the user may choose to manually subdivide geometry, specify subdivision sizes at arbitrary points, and/or make use of ADINA's more automated features. With practice, a user can effectively use the AUI to control the local mesh density over complex geometries.

2.1.1 Subdivision

Mesh densities can be set by subdividing lines (command `SUBDIVIDE LINE`), surfaces (`SUBDIVIDE SURFACE`), volumes (`SUBDIVIDE VOLUME`), body edges (`SUBDIVIDE EDGE`), body faces (`SUBDIVIDE FACE`), and/or bodies (`SUBDIVIDE BODY`) prior to meshing.

Lines can be subdivided using either number of divisions (`MODE = DIVISIONS`) or length (`MODE = LENGTH`). When using length, subdivisions are always created uniformly. When using number of divisions, subdivisions can be created non-uniformly (graded/biased subdivisions using `RATIO` and/or `CBIAS`).

Command Input 2.1: Commands for generating lines and subdivisions shown in Figure 2.1. Note the use of parameters MODE, RATIO, PROGRESS, and CBIAS.

```
feprogram program=adina
coordinates point
1 0.0 0.0 0.0 0
2 0.0 1.0 0.0 0
3 0.0 0.0 -0.2 0
4 0.0 1.0 -0.2 0
5 0.0 0.0 -0.4 0
6 0.0 1.0 -0.4 0
*
line straight name=1 p1=1 p2=2
line straight name=2 p1=3 p2=4
line straight name=3 p1=5 p2=6
*
subdivide line name=1 mode=divisions ndiv=20 ratio=1.0,
    progress=geometric cbias=no
*
subdivide line name=2 mode=divisions ndiv=20 ratio=10.0,
    progress=geometric cbias=no
*
subdivide line name=3 mode=divisions ndiv=20 ratio=10.0,
    progress=geometric cbias=yes
```

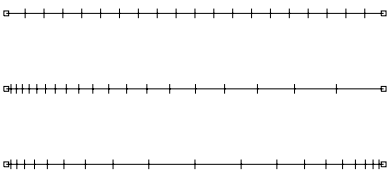


Figure 2.1: Line subdivisions using number of divisions. Top: length ratio (last/first) = 1; Middle: (last/first) = 10; Bottom: (last/first) = 10 with central biasing.

Figure 2.1 shows the various types of ratio that can be applied to line subdivisions using number of divisions as the mode of subdivisions.

Surfaces, volumes, body edges, body faces, and bodies can be subdivided in a similar fashion, with the exception that body edges, faces, and bodies cannot be subdivided with a central bias.

By default, the element size on the interior of a body face cannot exceed the greatest element size on the bounding body edges. This behavior can be changed by assigning a maximum size to the body face (with parameter MAX-SIZE in command SUBDIVIDE FACE). Note that this only applies to triangular free-form meshes generated using the Delaunay method. Command input 2.2 demonstrates using MAX-SIZE to obtain a fine mesh along the boundaries of a domain and a coarser mesh internally. A maximum size can also be applied to a three-dimensional body, with similar results (tetrahedral free-form meshing only).

Command Input 2.2: Commands for generating geometry and mesh shown in Figure 2.2.

```
feprogram program=adina
body block 1 dx1=1 dx2=1 dx3=1
*
subdivide face 1 max-size=0.1
*
subdivide edge 9 mode=length size=0.01
subdivide edge 10 mode=length size=0.01
subdivide edge 11 mode=length size=0.01
subdivide edge 12 mode=length size=0.01
*
egroup shell
*
gface 1 nodes=4 meshing=free-form,
    method=delaunay refine=along-edge density-factor=1.2
```

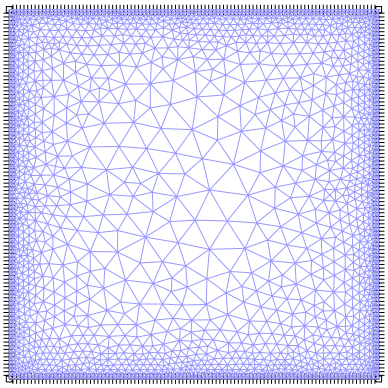


Figure 2.2: Body face with a fine mesh along the bounding edges and a coarser interior mesh.

Figure 2.2 shows the result of free-form meshing a body face when a maximum size has been imposed on the body face.

2.1.2 *Point size*

When the subdivision mode is set to point size, subdivisions along lines or body edges are computed from the 'sizes' stored at the two end points such that the subdivisions vary smoothly. The `POINT-SIZE` command guarantees that the subdivisions on all lines or body edges that connect to a point are always smooth. This is in contrast with subdividing a line or a body edge using number of divisions where the smoothness of subdivisions (assuming the ratio of lengths in not set to `1.0`) is limited only to those lines or body edges specified.

The following shows the typical work flow for using the `POINT-SIZE` command (see command input 2.3):

1. Change the subdivision mode for the complete model to "Use End-Point Sizes" (command `SUBDIVIDE MODEL` with `MODE = POINTWISE`).
2. Set the point size for the body to `0.2` (command `POINT-SIZE` with `OPTION = DIRECT` and `INPUT = BODY`). This creates a uniform subdivision for the body.
3. Set the point size for the top body face to `0.05` (com-

mand POINT-SIZE with OPTION = DIRECT and INPUT = FACE). This creates a uniform subdivision for the body face and automatically smooths out the subdivisions on the body edges that connect to the points bounding the body face.

Command Input 2.3: Commands for generating geometry and mesh shown in Figure 2.3.

```
feprogram program=adina
body block 1 dx1=1 dx2=1 dx3=1
*
subdivide model mode=pointwise ndiv=1,
     progress=geometric mincur=1
*
point-size option=direct input=body
1 0.2
*
point-size option=direct input=face body=1
1 0.05
```

Figure 2.3 shows the result of using point sizes.

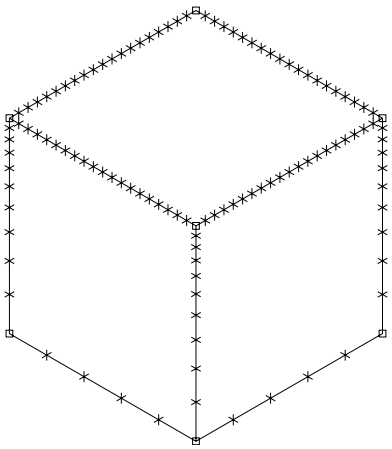


Figure 2.3: Mesh density set at vertices of the body and then at the vertices of a face using command POINT-SIZE.

2.1.3 *Automatic grading*

Automatic grading automatically adjusts the subdivisions for entities (*e.g.*, body edges and/or faces) without assigned mesh densities. Automatic grading is only available for bodies or body faces. It is enabled by setting AUTO-GRADING = YES. [1]

The automatic adjustment is performed using a fixed size variation between adjacent segments on the body edges. A segment is either as long as the adjacent segment or twice/half as long.

[1] GBODY AUTO-GRADING is not used if PYRAMIDS = ONLY or if boundary layers are used. For more information about Boundary layer meshing, see page 68.

Command Input 2.4: Commands for generating geometry and subdivisions shown in Figure 2.4.

```
feprogram program=adina
body block name=1 dx1=1.0 dx2=1.0 dx3=1.0
body sphere name=2 position=vector dimension=radius,
    cx1=0.5 cx2=0.5 cx3=0.5 radius=0.1
*
body subtract name=1 keep-too=no keep-imp=no
2
*
subdivide face 7 body=1 mode=length size=0.02
```

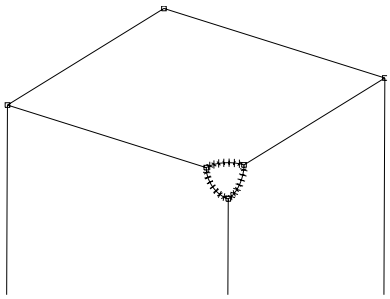


Figure 2.4: Subdivisions on a feature prior to applying automatic grading.

Command Input 2.5: Continuation of command input 2.4 generating subdivisions using `AUTO-GRADING = YES`. The result is shown in Figure 2.5. Note that because `SIMULATE = YES`, the body was not meshed – it was only subdivided.

```
egroup threedsolid
*
gbody 1 nodes=4,
    simulate=yes auto-grading=yes min-size=0.00001
```

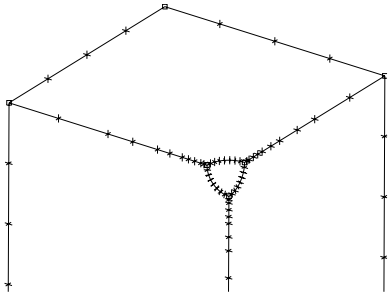


Figure 2.5: Newly subdivided edges after applying automatic grading. In this example, the parameter `SIMULATE = YES` suppressed mesh generation.

In Figure 2.4, subdivisions have been applied to one body face (the spherical face). Figure 2.5 shows the result of automatic grading. Note that because `SIMULATE = YES`, the body was not meshed – it was only subdivided.

In some cases, using automatic grading to update mesh subdivisions on body edges is not sufficient to guarantee a smoothly graded mesh. It may be necessary to actually mesh the body (free-form with tetrahedral elements) or the body face (free-form with triangular elements) using parameter `AUTO-GRADING = YES` to obtain smooth mesh densities everywhere. In this case, there is no need to grade the subdivisions prior to meshing since this will be done automatically during meshing.

Command Input 2.6: Commands for generating geometry and subdivisions shown in Figure 2.6. Note the finely subdivided cubic cavity.

```
feprogram program=adina
body block name=1 option=centered position=vector,
    cx1=0.0 cx2=0.0 cx3=0.0 dx1=1.0 dx2=1.0 dx3=1.0
body block name=2 option=centered position=vector,
    cx1=0.2 cx2=0.4 cx3=0.2 dx1=0.1 dx2=0.1 dx3=0.1
*
body subtract name=1 keep-too=no keep-imp=no
2
*
subdivide face 7 body=1 mode=length size=0.02
8 to 12
*
meshrendering hidden=dashed
*
egroup threedsolid
*
gbody 1 nodes=4,
    simulate=yes auto-grading=yes min-size=0.00001
```

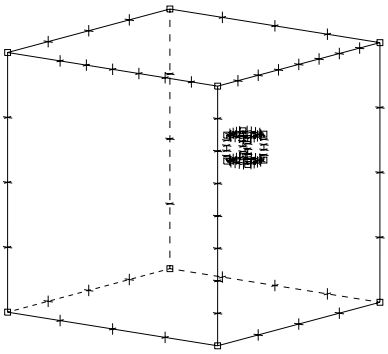


Figure 2.6: Block with internal cubic cavity. Subdivisions show the effects of AUTO-GRADING = YES. In this case, no mesh was generated (SIMULATE = YES).

Figure 2.6 shows a body with a subdivided internal cubic cavity. This internal cavity is just beneath a body face. AUTO-GRADING allows for the presence of that finely subdivided internal cavity to influence the edge subdivisions. However, the full effect of AUTO-GRADING on mesh density can not be seen when parameter AUTO-GRADING = YES. Rather, one must use SIMULATE = NO to capture the proper mesh densities on the nearby body face.

Command Input 2.7: Continuation of command input 2.6 for generating the mesh shown in Figure 2.7. Note the use of SIMULATE = NO.

```
subdivide body 1 mode=length size=1
subdivide face 7 body=1 mode=length size=0.02
8 to 12
*
gbody 1 nodes=4,
    simulate=no auto-grading=yes min-size=0.00001
```

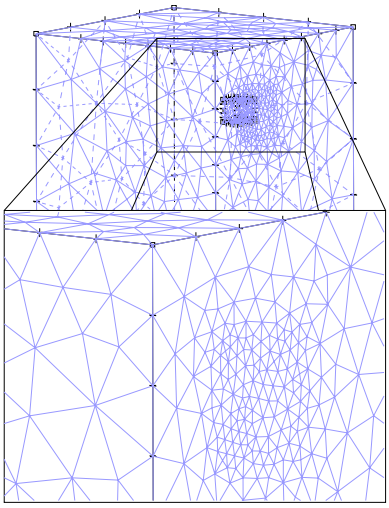


Figure 2.7: Tetrahedral mesh obtained using autograding with SIMULATE = NO. The inset shows the surface mesh with the inner cavity hidden.

Figure 2.7 shows the result of automatic grading with free-form meshing (tetrahedral elements) with SIMULATE = NO. The mesh on the body face nearest the cavity has clearly been influenced by both the cavity and the bounding edges.

2.1.4 *Curvature-based sizing*

Mesh densities can be assigned according to the curvature of body edges and body faces by specifying a maximum allowable geometric discretization error. These mesh densities are automatically graded for a smooth mesh density distribution.

Curvature-based sizing is only available for free-form tetrahedral meshing on bodies or body faces using the Delaunay method. For bodies, curvature-based sizing is enabled with the command `GBODY` using parameter `SIMULATE = YES`, `GEO-ERROR`, `SAMPLING`, and `MIN-SIZE`. Curvature-based sizing on body faces is performed using `GFACE` and the same parameters.

The parameter `GEO-ERROR` [2] controls the maximum permitted geometric discretization error. The lower the value, the finer the subdivisions and the more closely the mesh will match the geometry. Figure 2.8 illustrates the geometric interpretation of `GEO-ERROR` and its effects.

[2] `GBODY GEO-ERROR` is not used if `PYRAMIDS = ONLY` or if boundary layers are used. For more information about Boundary layer meshing, see page 68.

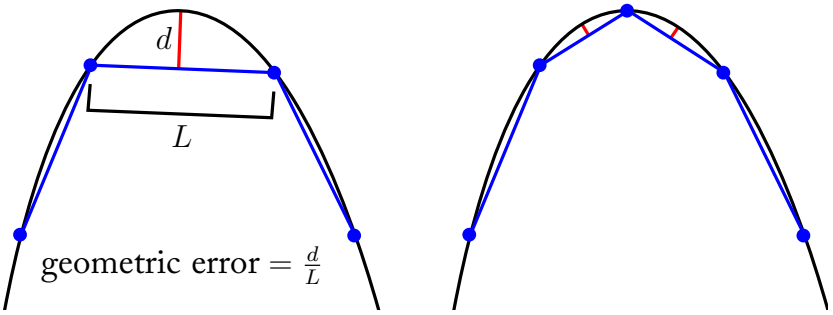


Figure 2.8: Schema illustrating the effect of reducing the maximum allowable geometric error (smaller value shown on right as compared to left). Decreasing the maximum permitted geometric discretization error results in finer subdivisions and a mesh with greater geometric fidelity.

The parameter `SAMPLING` controls the number of curvature samples taken along a body edge. The higher the sampling, the more accurate the mesh densities will be.

In areas of very strong curvature, mesh densities can become excessive unless controlled by parameter `MIN-SIZE`.

The size variation between adjacent segments on body edges is fixed. A segment is either as long as the adjacent segment or twice/half as long.

Command Input 2.8: Commands for generating bodies and subdivisions shown in Figure 2.9. Note the use of the GEO-ERROR option in the GBODY command.

```
feprogram program=adina
body block name=1 dx1=1.0 dx2=2.0 dx3=1.0
body cylinder name=2 option=centered position=vector,
    cx1=0.0 cx2=1.0 cx3=0.0 axis=zl radius=0.5 length=1.0
*
body merge name=1 keep-too=no merge-im=yes
2
*
egroup threedsolid
*
gbody 1 nodes=4,
    simulate=yes geo-error=0.01 sampling=20 min-size=0.00001
```

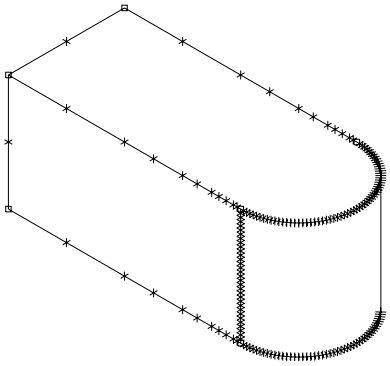


Figure 2.9: Body edge subdivisions after application of curvature-based sizing.

Figure 2.9 shows subdivisions after having applied curvature-based sizing. There were no subdivisions prior to using the command.

As with automatic grading, it may be necessary to mesh the body or the body face using parameter SIMULATE = NO, GEO-ERROR, SAMPLING, and MIN-SIZE to obtain smooth mesh densities everywhere.

2.1.5 *Size functions*

Size functions can be used to control mesh density in regions of interest. These regions can be defined by:

- A point (command `SIZE-FUNCTION POINT`).
- An axis (command `SIZE-FUNCTION AXIS`).
- A plane (command `SIZE-FUNCTION PLANE`).
- A bounding box (command `SIZE-FUNCTION BOUNDS`).
- A hexahedron (command `SIZE-FUNCTION HEX`).
- A combination of the above (command `SIZE-FUNCTION COMBINE`).

For a point, the size function is uniform (value given by parameter `SIZE`) for any location within the radial distance given by parameter `DISTANCE` of the point. The size function gradually increases as one moves away from the point. The behavior is the same for an axis and a plane.

For a hexahedron (defined by eight corners), the size function inside the hexahedron is defined by interpolating the sizes given at the corners). The size function outside the hexahedron increases gradually with distance. The behavior for the bounding box is the same.

These size functions can be used to:

- Locally refine regions of interest (*e.g.* near a stress concentration).
- Update sizes at points (command `POINT-SIZE` with `OPTION = FUNCTION`). The size at points can in turn be used to update the subdivisions on the connected lines and/or body edges if the subdivision mode for the complete model has been set to "Use End-Point Sizes."
- Update body edge subdivisions directly (without going through the points as described above). This can be performed for a body, using `GBODY` with parameters `SIMULATE = YES` and `SIZE-FUNCTION`, [3] or for a body face, using `GFACE` with the same parameters.
- Update body edge subdivisions directly and serve as additional mesh density constraints when free-form meshing is performed. This can be performed for a body, using `GBODY` with parameter `SIZE-FUNCTION`, or for a body

[3] `GBODY SIZE-FUNCTION` is not used if `PYRAMIDS = ONLY` or if boundary layers are used. For more information about Boundary layer meshing, see page 68.

face, using GFACE with the same parameter.

Command Input 2.9: Example showing how to use SIZE-FUNCTION to refine mesh near a point. Figure 2.10 shows the resulting mesh.

```
feprogram program=adina
body block 1 dx1=1 dx2=1 dx3=1
*
coordinates point
100 0.2 0.3 0.5
*
size-function point 1,
    mode=point point=100 size=0.01 distance=0.04
*
egroup threedsolid
*
gbody 1 nodes=4,
    size-function=1 boundary=delaunay brefine=edge-middle
```

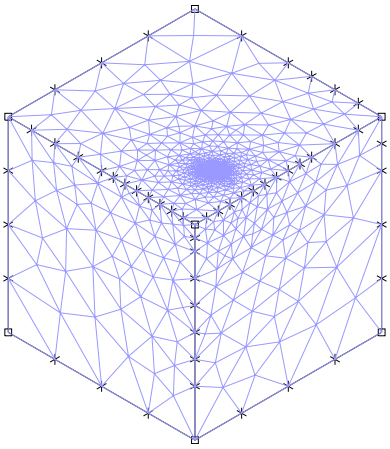


Figure 2.10: Free-form tetrahedral mesh obtained by using a size function at a point, located on the top of the body. The point's location is clearly visible by the high mesh density.

Command Input 2.10: Example showing how to use SIZE-FUNCTION to refine mesh near a line (AXIS) connecting 2 points. Figure 2.11 shows the resulting mesh.

```
feprogram program=adina
body block 1 dx1=1 dx2=1 dx3=1
*
coordinates point
100 0.5 0.2 0.5
101 -0.5 -0.2 0.5
*
size-function axis 1,
    mode=point p1=100 p2=101 size=0.01 distance=0.04
*
egroup threedsolid
*
gbody 1 nodes=4,
    size-function=1 boundary=delaunay brefine=edge-middle
```

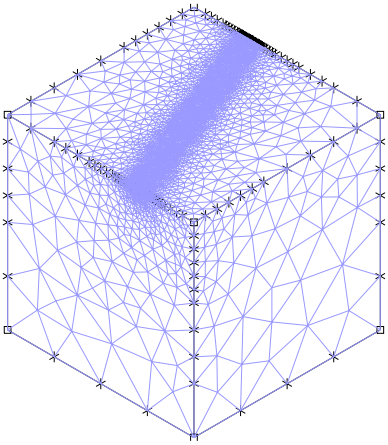


Figure 2.11: Free-form tetrahedral mesh obtained by using a size function along an axis. The axis is defined by a line connecting two points.

Figure 2.10 shows an example of using a size function at a point with free-form meshing (tetrahedral elements), and Figure 2.11 shows an example of using a size function along an axis.

2.2 *Mapped Meshing*

Mapped meshing, or *rule-based* meshing, gives the user more control over the meshing process than free-formed meshing and allows for geometrically-structured meshes. The term comes from the *mapping* operation which transforms simple shapes onto more general, though topologically-equivalent geometries (*e.g.*, squares into quadrilaterals). Mapped meshing can require more user intervention but often yields superior meshes over free-form meshing. See command input 2.11 and Figure 2.12 for an example in which mapped meshing may be preferred.

Command Input 2.11: Commands demonstrating an instance in which mapped meshing (`MESHING = MAPPED`) may be preferred over free-form meshing (`MESHING = FREE-FORM`). Figure 2.12 shows the resulting meshes.

```
feprogram program=adina
coordinates point
1 0.0 0.0 0.0 0
2 0.0 0.0 10.0 0
3 0.0 1.0 10.0 0
4 0.0 1.0 0.0 0
5 0.0 3.0 0.0 0
6 0.0 3.0 10.0 0
7 0.0 4.0 10.0 0
8 0.0 4.0 0.0 0
*
surface vertex name=1 p1=1 p2=2 p3=3 p4=4
surface vertex name=2 p1=5 p2=6 p3=7 p4=8
*
subdivide surface name=1 mode=divisions ndiv1=10 ndiv2=5
subdivide surface name=2 mode=divisions ndiv1=10 ndiv2=5
*
egroup shell name=1
*
gsurface 1 nodes=4 meshing=mapped
gsurface 2 nodes=4 meshing=free-form
```

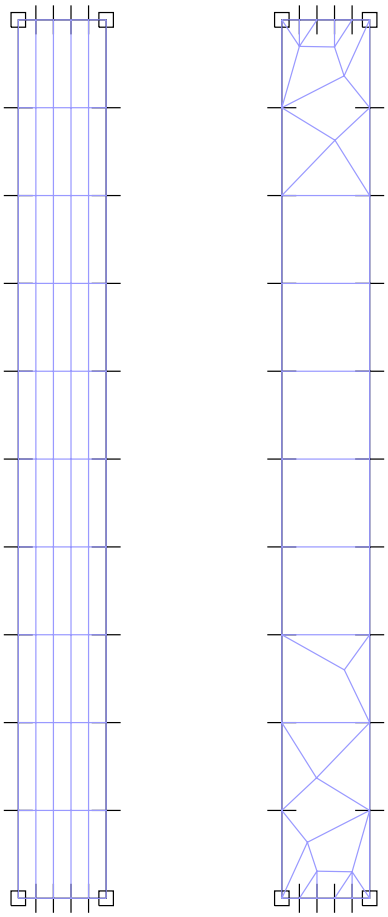


Figure 2.12: Comparison of mapped (left) vs free-form meshing (right). Clearly, mapped meshing allows the user to more effectively control the mesh.

Situations similar to that illustrated by Figure 2.12 arise in two-dimensional CFD or FSI problems wherein it is important to use quadrilateral elements. A high-quality mesh is difficult to generate with free-form meshing when the aspects ratios are large.

2.2.1 *Surface meshing*

Mapped meshes can be generated on geometry surfaces using the command GSURFACE. The type of element created (quadrilaterals or triangles) depends upon:

- Whether the surface is quadrilateral or triangular.
- Whether the subdivisions are regular or irregular. A surface is said to be *regularly subdivided* if the number of subdivisions on opposite lines is the same.
- The number of nodes for the desired element.

Command Input 2.12: Commands for generating a mapped mesh using MESHING = MAPPED and 4-node shell elements. Figure 2.13 shows the resulting mesh.

```
feprogram program=adina
coordinates point
1 0.0 0.0 0.0 0
2 0.0 1.0 0.0 0
3 0.0 1.0 1.0 0
4 0.0 0.0 1.0 0
*
surface vertex name=1 p1=1 p2=2 p3=3 p4=4
*
subdivide surface name=1 mode=divisions ndiv1=10 ndiv2=20
*
egroup shell name=1
*
gsurface 1 nodes=4 meshing=mapped
```

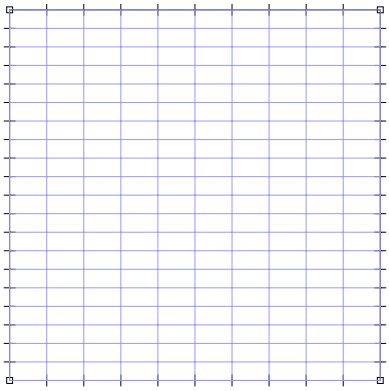


Figure 2.13: Mapped meshing on a quadrilateral surface with regular subdivisions and NODES = 4.

The surface in Figure 2.13 was created using four points and regularly subdivided. It was then meshed with SHELL elements (NODES = 4). The resulting mapped mesh is composed exclusively of quadrilateral elements.

Command Input 2.13: Continuation of command input 2.12 for generating the mapped mesh shown in Figure 2.14 using MESHING = MAPPED and 3-node 2D-solid elements.

```
eldelete surface 1
*
egroup twosolid name=2
*
gsurface 1 nodes=3 meshing=mapped pattern=automatic
```

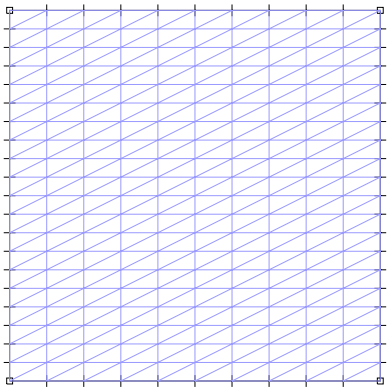


Figure 2.14: Mapped meshing on a quadrilateral surface with regular subdivisions and NODES = 3.

Figure 2.14 demonstrates the effect of selecting NODES = 3. The resulting mapped mesh is composed exclusively of triangular elements. ADINA first creates a mapped mesh composed of quadrilaterals which are then split into triangles depending on the pattern desired (parameter PATTERN).

Command Input 2.14: Continuation of command input 2.12 for the irregularly-subdivided geometry and the mapped mesh shown in Figure 2.15 using MESHING = MAPPED and 4-node shell elements.

```
eldelete surface 1
*
subdivide line name=1 mode=divisions ndiv=20
subdivide line name=2 mode=divisions ndiv=10
*
gsurface 1 nodes=4 meshing=mapped pattern=automatic
```

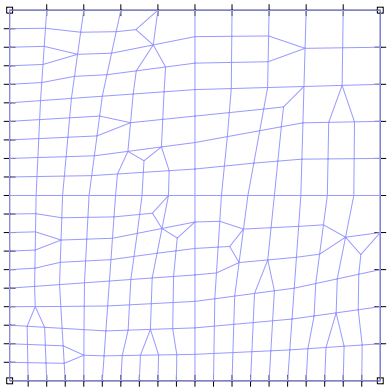


Figure 2.15: Mapped meshing on a quadrilateral surface with irregular subdivisions and NODES = 4. Note the presence of triangular and quadrilateral elements.

The geometry in Figure 2.15 was irregularly subdivided. It was then meshed with SHELL elements (NODES = 4). The resulting mapped mesh is composed primarily of quadrilaterals but with some triangles. The triangles are necessary to transition from a number of subdivisions on one side and a different number of subdivisions on the opposite side.

When a surface is triangular (vertex point 4 is point 1), the user may choose to treat the triangular surface as either *degenerate* or *not degenerate*. This choice affects the mapped meshing of the triangular surface.

If the triangular surface is not to be treated as degenerate (using DEGENERATE = NO), ADINA splits the surface into three quadrilateral sub-surfaces which are then mapped meshed. To obtain an all-quadrilateral mesh, the three sides of the triangular surface must have the same number of subdivisions.

Command Input 2.15: Commands for generating a mapped mesh on a triangular surface using DEGENERATE = NO and 4-node shell elements. Figure 2.16 shows the resulting mesh.

```
feprogram program=adina
coordinates point
1 0.0 0.0 0.0 0
2 0.0 1.0 0.0 0
3 0.0 0.5 1.0 0
*
surface vertex name=1 p1=3 p2=1 p3=2 p4=3
*
subdivide surface name=1 mode=divisions ndiv1=10 ndiv2=10
*
egroup shell name=1
*
gsurface 1 nodes=4,
    pattern=automatic meshing=mapped degenerate=no
```

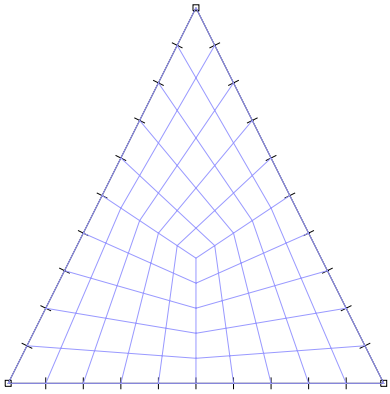


Figure 2.16: Mapped meshing on a triangular surface not treated as degenerate with regular subdivisions and NODES = 4.

The surface shown in Figure 2.16 was created from three points and not treated as degenerate. It was regularly subdivided and meshes using SHELL elements (NODES = 4). The resulting mapped mesh is composed exclusively of quadrilaterals.

Command Input 2.16: Continuation of command input 2.15 for generating the mapped mesh on the irregularly-subdivided triangular surface shown in Figure 2.17 using DEGENERATE = NO and 4-node shell elements.

```
eldelete surface 1
*
subdivide line name=2 mode=divisions ndiv=20
*
gsurface 1 nodes=4,
    pattern=automatic meshing=mapped degenerate=no
```

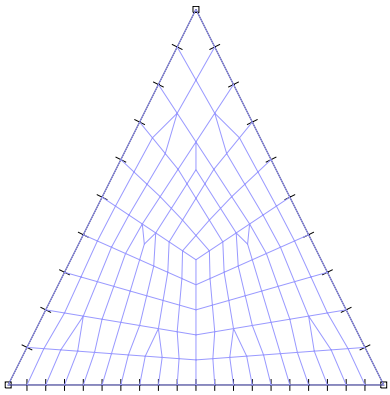


Figure 2.17: Mapped meshing on a triangular surface not treated as degenerate with irregular subdivisions and NODES = 4).

The surface shown in Figure 2.17 was irregularly subdivided and not treated as degenerate. The resulting mapped mesh is composed primarily of quadrilaterals but with some triangles.

If a triangular surface is treated as degenerate by specifying DEGENERATE = YES, it is meshed as if it were a quadrilateral surface. In this case, the subdivisions are said to be *regular* if the two lines connected to the degenerate point have an equal number of subdivisions.

Command Input 2.17: Continuation of command input 2.15 for generating a mapped mesh on the regularly-subdivided triangular surface shown in Figure 2.18 using DEGENERATE = YES and 4-node shell elements.

```
eldelete surface 1
*
gsurface 1 nodes=4,
    pattern=automatic meshing=mapped degenerate=yes
```

The triangular surface in Figure 2.18 was treated as degenerate and regularly subdivided. It was then meshed using SHELL elements (NODES = 4). The resulting mapped mesh is composed exclusively of quadrilaterals except near the top point, where the elements are all triangular.

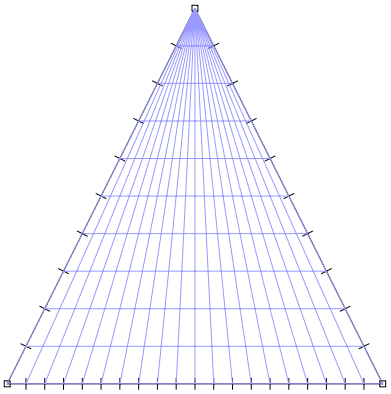


Figure 2.18: Mapped meshing on a triangular surface treated as degenerate.

Command Input 2.18: Continuation of command input 2.15 for generating a mapped mesh on the irregularly-subdivided triangular surface shown in Figure 2.19 using DEGENERATE = YES and 4-node shell elements.

```
eldelete surface 1
*
subdivide line name=3 mode=divisions ndiv=20
*
gsurface 1 nodes=4,
    pattern=automatic meshing=mapped degenerate=yes
```

The surface in Figure 2.19 was irregularly subdivided and treated as degenerate. The resulting mapped mesh is composed primarily of quadrilaterals, except near the top point and the vertical central region.

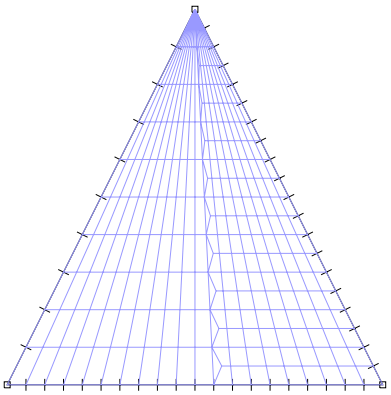


Figure 2.19: Mapped meshing on an irregularly-subdivided triangular surface treated as degenerate.

A body face may also be map meshed, provided the face is topologically identical to a surface. For instance, a quadrilateral (4-edges, not necessarily straight) body can be map meshed because it is topologically identical to a quadrilateral surface.

2.2.2 *Volume meshing*

Mapped meshing can be performed on volumes using the command GVOLUME. The type of element created (hexahedral, prismatic, pyramid, or tetrahedral) depends upon:

- whether the volume is an hexahedron, a prism, a pyramid,

or a tetrahedron.

- whether the subdivisions are regular or irregular. A cube, for example, is regularly subdivided if all parallel lines have the same number of subdivisions.
- the number of nodes for the element.

Command Input 2.19: Commands for generating a mapped mesh on an regularly-subdivided hexahedral volume using 8-node 3D elements. Figure 2.20 shows the resulting mesh.

```
feprogram program=adina
coordinates point
1 0.0 0.0 1.0 0
2 1.0 0.0 1.0 0
3 1.0 1.0 1.0 0
4 0.0 1.0 1.0 0
5 0.0 0.0 0.0 0
6 1.0 0.0 0.0 0
7 1.0 1.0 0.0 0
8 0.0 1.0 0.0 0
*
volume vertex name=1 shape=hex,
    vertex1=1 vertex2=2 vertex3=3 vertex4=4,
    vertex5=5 vertex6=6 vertex7=7 vertex8=8
*
subdivide volume name=1,
    mode=divisions ndiv1=5 ndiv2=10 ndiv3=20
*
egroup threedsolid name=1
*
gvolume 1 nodes=8 meshing=mapped
```

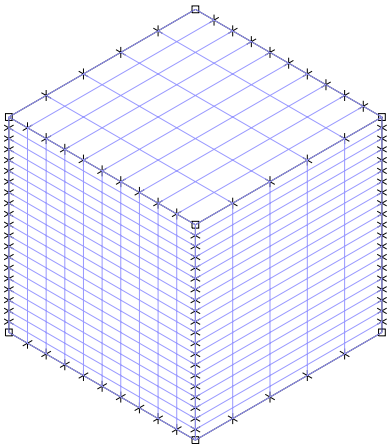


Figure 2.20: Mapped meshing of a regularly-subdivided hexahedral volume.

The hexahedral volume shown in Figure 2.20 has regular subdivisions and is meshed using `THREEDSOLID` elements (with `NODES = 8`). The resulting mapped mesh is composed exclusively of hexahedral (brick) elements.

Command Input 2.20: Continuation of command list 2.19 for regenerating a mapped mesh on an irregularly-subdivided hexahedral volume using 8-node 3D elements. Figure 2.21 shows the resulting mesh.

```
eldelete volume 1
*
subdivide line name=3 mode=divisions ndiv=10
subdivide line name=4 mode=divisions ndiv= 5
subdivide line name=11 mode=divisions ndiv=10
subdivide line name=12 mode=divisions ndiv= 5
*
gvolume 1 nodes=8 meshing=mapped
```

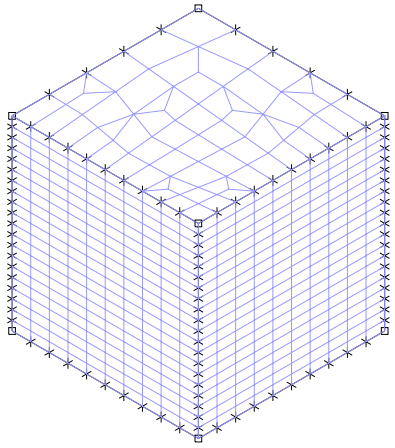


Figure 2.21: Mapped meshing of an irregularly-subdivided hexahedral volume.

The irregularly subdivided volume in Figure 2.21 is mapped meshed primarily with hexahedral elements and some prismatic elements.

When the volume is not a hexahedron, it may be treated as either degenerate or not degenerate. This choice affects the mapped meshing of the volume.

If the volume is to be treated as degenerate (`DEGENERATE = YES`), the volume is meshed as if it were a hexahedron.

Command Input 2.21: Commands for generating a mapped mesh on a regularly-subdivided prismatic volume using 8-node 3D elements and `DEGENERATE = YES`. Figure 2.22 shows the resulting mesh.

```
feprogram program=adina
coordinates point
1 0.5 1.0 1.0 0
2 0.0 0.0 1.0 0
3 1.0 0.0 1.0 0
4 0.5 1.0 0.0 0
5 0.0 0.0 0.0 0
6 1.0 0.0 0.0 0
*
volume vertex name=1 shape=prism,
    vertex1=3 vertex2=2 vertex3=5,
    vertex4=6 vertex5=1 vertex6=4
*
subdivide volume name=1,
    mode=divisions ndiv1=5 ndiv2=10 ndiv3=20
*
egroup threedsolid name=1
*
gvolume 1 nodes=8 meshing=mapped degenerate=yes
```

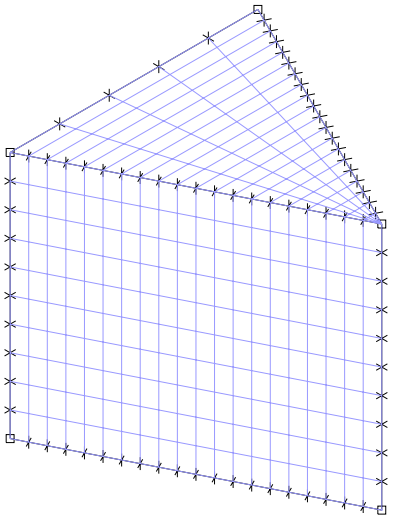


Figure 2.22: Mapped meshing of a regularly-subdivided prismatic volume with `DEGENERATE = YES`.

The regularly divided prismatic volume shown in Figure 2.22 is treated as degenerate. It is then meshed using THREEDSOLID elements (NODES = 8). The resulting mapped mesh is composed of hexahedral elements, except along the degenerate line defined by points 5 through 6, in the canonical ordering of a prismatic volume.

Command Input 2.22: Continuation of command input 2.21 for generating a mapped mesh on the regularly-subdivided prismatic volume shown in Figure 2.23 using 8-node 3D elements and DEGENERATE = NO.

```
eldelete volume 1
*
gvolume 1 nodes=8 meshing=mapped degenerate=no
```

The irregularly subdivided volume shown in Figure 2.23 is treated as non degenerate and meshes using THREEDSOLID elements (NODES = 8). The resulting mapped mesh is composed primarily of hexahedral elements, with some prismatic elements. For the subdivisions to be regular in this context (prismatic volume not treated as degenerate), the lines making up the top triangular surface should have the same number of subdivisions (and the bottom triangular surface would need to match that as well).

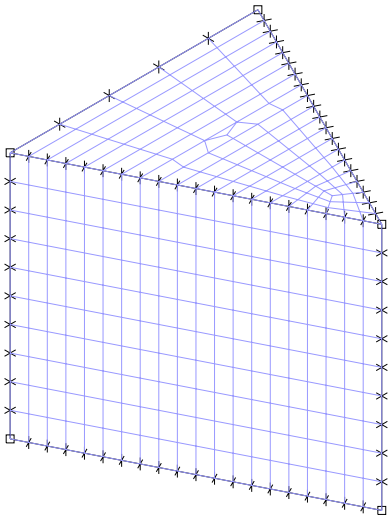


Figure 2.23: Mapped meshing of an irregularly-subdivided prismatic volume with DEGENERATE = NO.

Pyramids and tetrahedons can similarly be map meshed. Mapped meshing can also be applied to a body, provided the body is topologically identical to a volume. For instance, a body that has the shape of a hexahedron can be mapped meshed because it is topologically identical to a hexahedral volume.

2.2.3 *Body face meshing*

When generating second-order meshes, GFACE MIDNODES allows the user to select from four mid-side node placement options.

- MIDNODES = STRAIGHT: The element's mid-side nodes are placed midway on the straight line connecting the vertex nodes rather than on the curved boundary. Clearly, this option results in less geometric fidelity. See Figure 2.24, panel *a*.
- MIDNODES = PROJECT: The mid-side node is first placed on the straight line connecting the vertex nodes and then

projected onto the curved boundary. See Figure 2.24, panel *b*.

- `MIDNODES = CURVED`: The mid-side node's location on the curved boundary is determined by a mapping operation using the body's parametric space. See Figure 2.24, panel *c*.
- `MIDNODES = DEFAULT`: The default behavior is that for mapped meshing (`MESHING = MAPPED`), the `CURVED` option is used, and for free-form and mixed meshing (`MESHING = FREE-FORM` or `= MIXED`), the `= PROJECT` option is used.

Command inputs 2.23, 2.24, and 2.25 demonstrate all three mid-side node placement cases with mapped meshes. When using mapped body face meshing, it is preferrable to use the `MIDNODES = CURVED` because the mid-side nodes will follow the parametric space of the body.

Note, however, that when generating *free-form* second-order meshes, the `MIDNODES = PROJECT` option will typically generate better quality elements. See the section on Mid-side node placement for free-form body face meshing (on page 57 – see also Figure 2.44).

When connecting mapped meshes to free-form meshes (e.g., when using `GFACE MESHING = MIXED`), the `MIDNODES = PROJECT` must be used to ensure that the midside nodes are compatible at the interface between the mapped and the free-form mesh.

Command Input 2.23: Commands for generating the geometry and mapped mesh shown in Figure 2.24, panel *a* with `MIDNODES = STRAIGHT`.

```
feprogram program=adina
body block name=1 cx1=1.5 cx3=-0.5 dx1=1.0 dx2=1.0 dx3=1.0
*
body revolved name=2 mode=axis angle=-90.0 axis=yl
1 1
*
delete body first=1 last=1
*
subdivide face name=1 body=2 mode=divisions ndiv=3
1 2
*
egroup shell name=1
*
gface nodes=9 meshing=mapped midnodes=straight
1 2
*
view name=1 type=parallel xview=0 yview=-1 zview=0
nodedepiction name=1 symbolplot=yes
meshplot view=1 nodedepiction=1
```

Command Input 2.24: Continuation of command input 2.23 for generating the mapped mesh shown in Figure 2.24, panel *b* with `MIDNODES = PROJECT`.

```
eldelete face name=1 body=2
gface nodes=9 meshing=mapped midnodes=project
1 2
*
frame
meshplot view=1 nodedepiction=1
```

Command Input 2.25: Continuation of command input 2.24 for generating the mapped mesh shown in Figure 2.24, panel *c* with `MIDNODES = CURVED`.

```
eldelete face name=1 body=2
gface nodes=9 meshing=mapped midnodes=curved
1 2
*
frame
meshplot view=1 nodedepiction=1
```

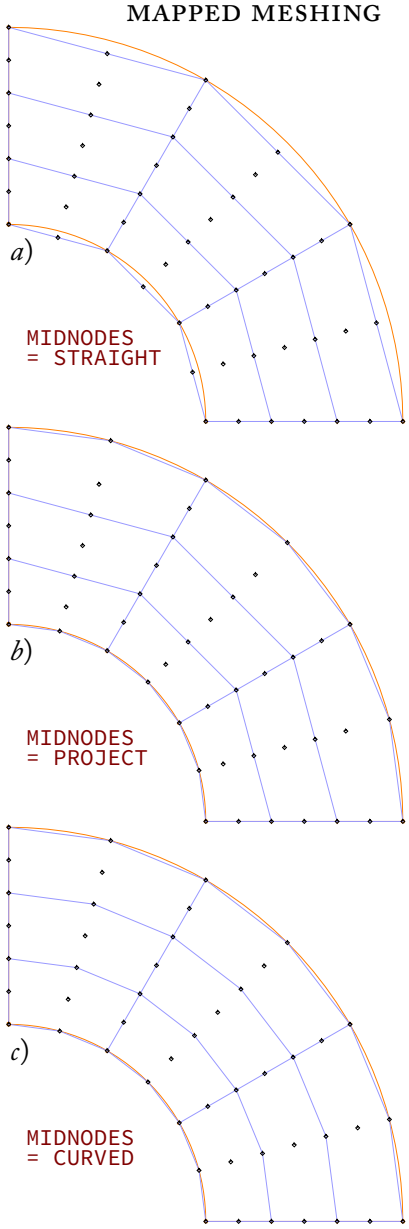


Figure 2.24: The effect of `MIDNODES` when creating mapped second-order meshes. When `MIDNODES = STRAIGHT`, mid-side nodes are always placed midway on straight lines connecting corner nodes (panel *a*). When `MIDNODES = PROJECT`, mid-side nodes are projected onto curved edges/faces from straight lines connecting corner nodes (panel *b*). When `MIDNODES = CURVED`, mid-side nodes are placed on curves defined by the body's parametric space (panel *c*).

2.2.4 *Sweep and revolved meshing*

Three-dimensional meshes can be created by *sweeping* or *revolving* meshed two dimensional body faces. Such meshes are created simultaneously with the three-dimensional body during the sweep or revolve operation. For example, if a body face has been meshed with 4-node shell elements, a `BODY SWEEP` or `BODY REVOLVED` operation will produce hexahedral elements.

Command Input 2.26: Commands for generating the geometry and mesh shown in Figure 2.25.

```
feprogram program=adina
coordinates point
1 0.0 1.00 1.0 0
2 0.0 0.00 1.0 0
3 0.0 0.00 0.0 0
4 0.0 1.00 0.0 0
5 0.0 0.50 0.5 0
6 0.0 0.75 0.5 0
*
line straight name=1 p1=1 p2=2
line straight name=2 p1=2 p2=3
line straight name=3 p1=3 p2=4
line straight name=4 p1=4 p2=1
line circle name=5 mode=1 p1=6 p3=1 center=5
line combined name=6 coupled=yes restrict=yes
1
2
3
4
*
body sheet name=1 line=6 delete-line=yes option=line
5
*
subdivide body name=1 mode=length size=0.1
*
egroup shell name=1
*
gface 1 nodes=4,
    meshing=free-form prefshap=quad-direct density=1.2
```

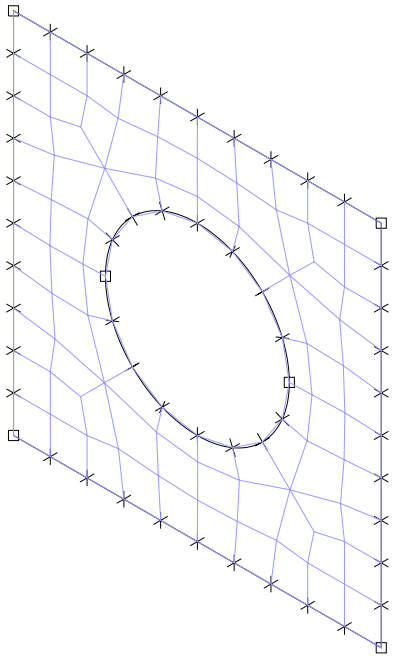


Figure 2.25: Two-dimensional mesh to be swept.

Command Input 2.27: Continuation of command input 2.26 for sweeping the two-dimensional mesh. Figure 2.26 shows the resulting swept body and three-dimensional mesh.

```
egroup threedsolid name=2
*
body sweep name=2 face=1 option=vector dx=2.0 dy=0.0 dz=0.0,
    body=1 mesh=yes 2d-egroup=1 3d-egroup=2 ndiv=10
```

Command Input 2.28: Continuation of command input 2.26 for revolving the two-dimensional mesh. Figure 2.27 shows the resulting swept body and three-dimensional mesh.

```
egroup threedsolid name=2
*
body revolved name=2 mode=vectors face=1 angle=60.0 za=1.0,
    body=1 mesh=yes 2d-egroup=1 3d-egroup=2 ndiv=10
```

Figure 2.25 shows two-dimensional mesh on a body face. Figure 2.26 shows the result of a sweep of the body face along a vector (command BODY SWEEP). Figure 2.27 shows the result of a revolution of the body face about an axis (command BODY REVOLVED).

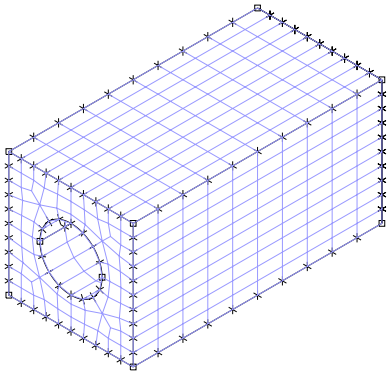


Figure 2.26: Resulting geometry and three-dimensional mesh following BODY SWEEP operation.

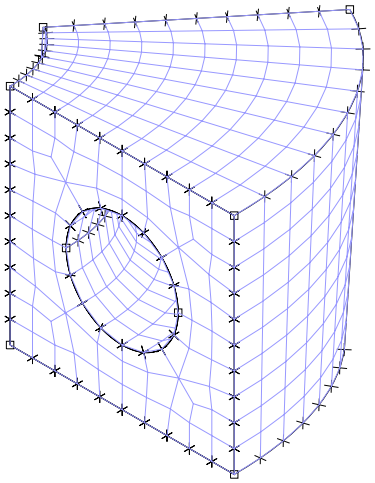


Figure 2.27: Resulting geometry and three-dimensional mesh following BODY REVOLVED operation.

When using BODY REVOLVED, the user should note that when revolving a quadrilateral mesh with only one node on the axis of revolution, the resulting mesh will contain collapsed elements (see Command input 2.29 and Figure 2.28, panel *a*).

To prevent BODY REVOLVED from generating collapsed elements in this situation, the user can make make the any of the following modifications in the 2D mesh before revolving:

- Create a boundary layer mesh of single element thickness on the axis of revolution.
- Adjust the subdivisions so that there are no quadrilateral elements with only one node on the axis of revolution
- Create a triangular mesh rather than a quadrilateral mesh, as shown in Command input 2.30 and Figure 2.28 (panel *b*).

Command Input 2.29: Commands for generating the mesh shown in Figure 2.28 (panel *a*). Note that by using GFACE NODES = 4, a single quadrilateral element is generated on the face. By using BODY REVOLVED about a single node, a single collapsed 7-node element is generated.

```
feprogram program=adina
coordinates point
1  0.0  0.0
2 -1.0  1.0
3 -2.0  0.0
4 -1.0 -1.0
*
line straight name=1 p1=1 p2=2
line straight name=2 p1=2 p2=3
line straight name=3 p1=3 p2=4
line straight name=4 p1=4 p2=1
line combined name=5 coupled=yes restrict=yes
1
2
3
4
*
body sheet name=1 line=5 delete-line=yes option=line
*
egroup twodsolid name=1 subtype=stress3
gface name=1 body=1 nodes=4
egroup threedsolid name=2
body revolved name=2 mode=axis face=1 body=1 mesh=yes,
     3d-egroup=2 axis=yl angle=-30.0 ndiv=1, delete-f=element
*
nodedepiction name=1 symbolplot=yes
meshrendering hidden=dashed
meshplot view=isoview2 nodedepiction=1
```

Command Input 2.30: Continuation of command input 2.29 for generating the mapped mesh shown in Figure 2.28 (panel *b*). Note that by using GFACE NODES = 3, two triangular elements are generated on the face. This prevents BODY REVOLVED from generating a collapsed element.

```
eldelete body name=2
*
gface name=1 body=1 group=1 nodes=3
body revolved name=3 mode=axis face=1 body=1 mesh=yes,
     3d-egroup=2 axis=yl angle=-30.0 ndiv=1
*
frame
meshplot view=isoview2 nodedepiction=1
```

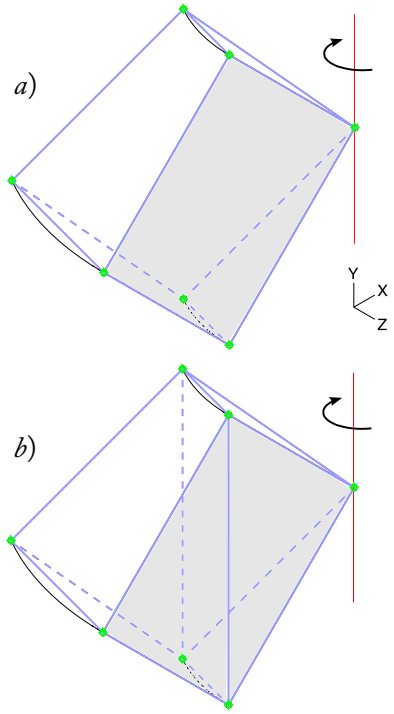


Figure 2.28: Illustration of possible outcomes when using BODY REVOLVED about a single node. The orginal meshed face is shown in grey and is being revolved about the *Y*-axis (shown by the vertical red line). In panel *a*, the face is meshed with a single 4-node quadrilateral element, and BODY REVOLVED generates a single collapsed 7-node element. Panel *b* shows that by meshing the face using two 3-node elements, the resulting revolved mesh contains of a prismatic element and a pyramid element.

2.2.5 *Lofted meshing*

The command GLOFTED creates a lofted mesh (with either prismatic or hexahedral elements) on any geometry body that can be seen to result from either a sweep, extrude, revolve, or loft of a source (body) face. Some examples of candidate bodies include

- a body that is topologically equivalent to a cube or prism and that would typically be meshed using GBODY MESHING = MAPPED,
- a body that results from a BODY SWEEP command,
- a body that results from a BODY REVOLVED command, or
- a body that results from a BODY LOFTED.

The GLOFTED command requires that the body to be meshed satisfies several topological requirements.

- There must exist topologically-equivalent[4] *source* and *target* faces. The source and target faces can have interal closed loop edges (e.g., a face with holes).
- Corresponding points on the source and target faces must be connected by a *linking edge* (either straight or curved).
- Corresponding edges on the source and target faces must be connected by a four-sided *linking face*.

In addition, there must be an equal number of subdivisions (see the Subdivision section on page 22) on corresponding edges of source and target faces, and linking edges must all have the same number of subdivisions. Figure 2.30 illustrates source/target faces, corresponding edges, linking faces, and linking edges on a lofted body. That body is used in the following examples.

When GLOFTED is executed, the subdivisions on the target face are automatically updated to be consistent with the subdivisions on the source face. Thus, it is only necessary to subdivide the source face. In addition, the subdivisions on the linking edges are automatically applied by the GLOFTED NDIV and RATIO parameters.

GLOFTED meshes the source face using either triangles (using parameter PREFSHAPE = PRISMATIC) or quadrilaterals (using PREFSHAPE = HEXAHEDRAL) and lofts the source surface mesh along the linked edges/face, thereby creating either

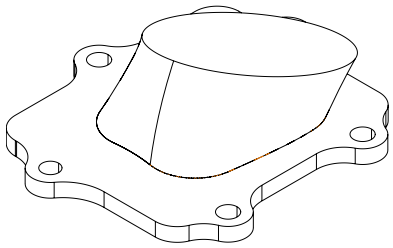


Figure 2.29: Parasolid model of a manifold adapter. The central feature is a lofted body, which is a good candidate for being meshed with GLOFTED. The bottom feature is a swept body, which may also be meshed using GLOFTED.

[4] *Topologically-equivalent* faces share a one-to-one correspondence of points and edges.

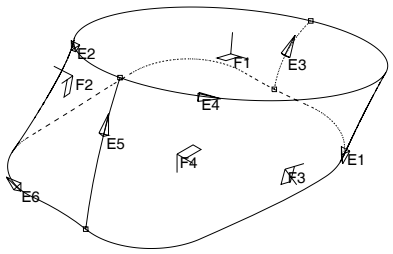


Figure 2.30: Detail of lofted body (referred to as body 1 in the following examples) from Figure 2.29 with important features labeled. Faces 1 and 4 (top and bottom, respectively) comprise a *source* and *target* face pair. Edges 1 & 4 and 2 & 6 are two pairs of *corresponding edges*. Faces 2 and 3 are *linking faces*. Edges 3 and 5 are *linking edges*.

prismatic or hexahedral elements. If the source face is to be meshed using quadrilaterals, then the sum of subdivisions of the source face bounding edges must be even.

Command input 2.31 demonstrate how to generate a lofted mesh using GLOFTED PREFSHAPE = HEXAHEDRAL. The resulting mesh is shown in Figure 2.31.

Command Input 2.31: Generating a lofted mesh for the body shown in Figure 2.30 by specifying face 1 as the source face. The resulting mesh is shown in Figure 2.31.

```
subdivide face name=1 body=1 mode=divisions ndiv=20
*
egroup threedsolid name=1
*
glofted nodes=8 group=1 prefshap=hexahedral ndiv=10
1 1
dataend
*
meshplot
```

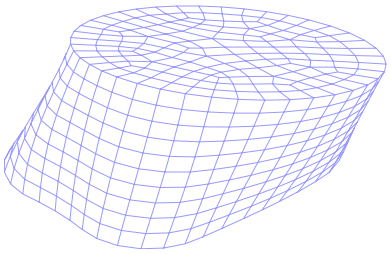


Figure 2.31: Mesh generated by GLOFTED for the body shown in Figure 2.30.

If a specific type of surface mesh is required on the source face, it should be meshed using GFACE. For example, it may be desireable to use boundary layers along the walls of the lofted manifold. For more information about Boundary layer meshing using GFACE and BLTABLE-2D, see page 55.

Command input 2.32 demonstrate how to generate a surface mesh with boundary layers and apply it to a lofted body. Figure 2.33 shows the internal structure of the resulting lofted mesh.

Command Input 2.32: Lofting a boundary layer mesh for the body shown in Figure 2.30. Face 4 is meshed using a boundary layer table and GFACE (Figure 2.32, panel *a*). GLOFTED is then used to create the final lofted mesh (Figure 2.32, panel *b*). The internal structure of this mesh is shown in Figure 2.33.

```
subdivide face name=4 body=1 mode=divisions ndiv=20
*
egroup shell name=1
*
bltable-2d name=1 progress=geometric apply-default=all,
    n-layer=6 thick-first=0.3 thick-total=5.0
*
gface name=4 body=1 nodes=4 group=1 2dbltable=1,
    prefshape=quad-direct
*
egroup threedsolid name=2
*
glofted group=2 ndiv=10
4 1
dataend
*
meshplot
```

In many cases (as in the example shown in command input 2.32), the 2D mesh used by GLOFTED is needed only temporarily. The parameter GLOFTED DELETE-FACE-ELEMENT allows the user to specify if the 2D mesh is to be deleted. For example, setting DELETE-FACE-ELEMENT = ELEMENT will delete the elements on the source face but will not delete the associated element group. DELETE-FACE-ELEMENT = ALL will delete the elements on the source face and the associated element group if it does not contain any other elements.

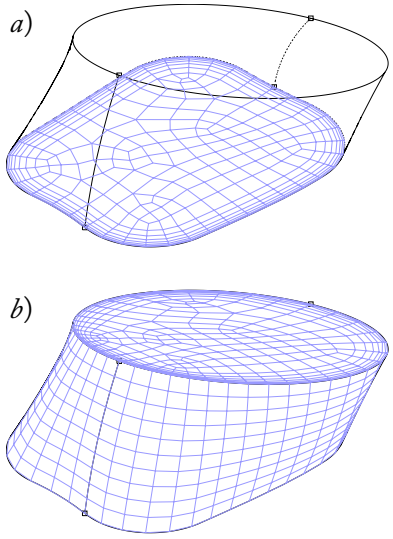


Figure 2.32: Lofting of a boundary layer mesh. The bottom face of the lofted body is meshed using BLTABLE-2D and GFACE (panel *a*). The triangulation from the resulting meshed face is then lofted to the target face, resulting in a high-quality mesh with a boundary layer along the manifold's wall (panel *b*).

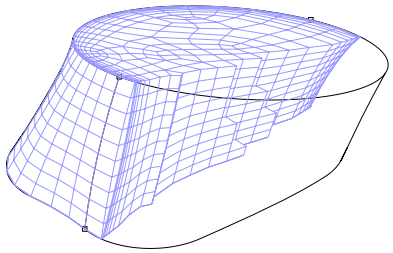


Figure 2.33: Internal structure of lofted mesh with boundary layers.

Note that in command input 2.32, it was not necessary to specify GLOFTED NODES. This parameter is only used when there is no 2D mesh or face-linked triangulation on the source face. In this case, source face 4 already has a triangulation from the boundary layer mesh shown in Figure 2.32 (panel *a*).

When the source face already has a triangulation, GLOFTED automatically assigns the number of nodes per element in the lofted mesh based on the 2D mesh or triangulation as detailed in Table 2.1.

2D Mesh	Lofted 3D Mesh
3-node triangular	6-node prismatic
6-node triangular	15-node prismatic
7-node triangular	20-node prismatic
4-node quadrilateral	8-node hexahedral
8-node quadrilateral	20-node hexahedral
9-node quadrilateral	27-node hexahedral

Table 2.1: 3D element types created when lofting the following 2D meshes.

If a *source* face is face-linked (using the FACELINK command – see the Face-linking section on page 14) with a body that has a three-dimensional mesh, the triangulation from the three-dimensional mesh is used as the surface mesh for the source face. In this case, any existing subdivisions on the source face are updated.

Similarly, if a *linking* face is face-linked with a body that has a three-dimensional mesh, the triangulation from the three-dimensional mesh is used as the surface mesh for the linking face. In this case, it is required that

- the number of subdivisions on the shared edge between the source face and the linking face is the same as the number of subdivisions used in three-dimensional mesh of the face-linked body.
- the number of subdivisions along the linking edges (as specified by the GLOFTED NDIV parameter) is the same as the number of subdivisions used in the three-dimensional mesh of the face-linked body.

2.3 *Body Face Free-Form Meshing*

The command GFACE generates triangular, quadrilateral, and boundary layer meshes on body faces. Parameter PREFSHAPE controls the shape of the elements to be generated (triangular or quadrilateral), and parameter 2DBLTABLE enables boundary layer meshing.

Note that free-form meshing produces slightly different meshes on different platforms, and meshes generated on your computer may not exactly match those shown in the figures.

2.3.1 *Triangular meshing*

Triangular meshing is selected with PREFSHAPE = TRIANGULAR.[5] There are two methodologies available to mesh a body face: advancing front and Delaunay. In either case, the starting point is the subdivisions that are present on the bounding body edges.

[5] PREFSHAPE is used only for free-form meshing when NODES = 4, 8, 9, or 16.

The advancing front method (parameter METHOD = ADVFRONT)[6] "advances" the front (initially composed of the segments joining the subdivision marks on the bounding body edges) toward the interior by creating triangular elements until the front is filled. Each time a triangular element is created, the front is modified, shrinking the domain remaining to be meshed.

[6] The advancing front method produces well-shaped elements near the boundary of the body face but may encounter difficulties closing the front. This limitation becomes more evident in three dimensions.

Command Input 2.33: Commands for generating the free-formed mesh using the advancing front method shown in Figure 2.34.

```
feprogram program=adina
body block name=1 option=centered position=vector,
        cx1=0.0 cx2=0.0 cx3=0.0 dx1=2.0 dx2=2.0 dx3=2.0
body cylinder name=2 option=centered position=vector,
        cx1=0.0 cx2=0.0 cx3=0.0 axis=zl radius=0.1 length=4.0
*
body subtract name=1 keep-too=no keep-imp=no
2
*
subdivide face 1 mode=length size=0.2
*
subdivide edge 15 mode=length size=0.01
subdivide edge 17 mode=length size=0.01
*
egroup shell
*
gface 1 nodes=4 prefshape=triangular method=advfront
```

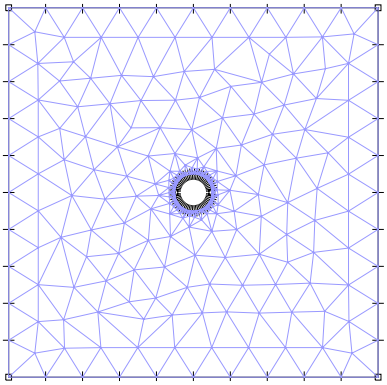


Figure 2.34: Free-form mesh generated using `METHOD = ADVFRONT`.

Figure 2.34 shows a free-form triangular mesh obtained using the advancing front method when the subdivisions on the bounding body edges are non-uniform, which is the general case. As can be seen, the mesh gradation is quite steep, which may be desirable only if the number of elements generated must be kept at a minimum.[7]

[7] The growth rate (mesh density gradient) cannot be modified using the advancing front method.

Command Input 2.34: Continuation of command input 2.33 for regenerating the mesh using uniform subdivisions shown in Figure 2.35.

```
eldelete face 1 body=1
*
subdivide face 1 mode=length size=0.05
*
subdivide edge 15 mode=length size=0.05
subdivide edge 17 mode=length size=0.05
*
gface 1 nodes=4 prefshape=triangular method=advfront
```

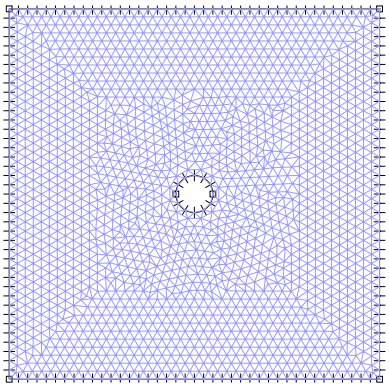


Figure 2.35: Free-form triangular mesh obtained using the advancing front method. Uniform subdivisions were used in this case.

The advancing front method generates good meshes when the subdivisions are uniform or nearly uniform. Figure 2.35 shows a free-form triangular mesh obtained using the advancing front method with uniform subdivisions on the bounding body edges.

The Delaunay method (invoked by setting `METHOD = DELAUNAY`)

proceeds as follows:

1. Create an initial mesh (typically composed of two triangles) that fully contains the body face.
2. Insert the boundary vertices into the mesh.
3. Recover the boundary segments if they are not present in the mesh. Note that, in two dimensions, the initial boundary segments are likely to be in the mesh.
4. Refine the mesh until it satisfies the mesh density requirements.
5. Optimize the mesh quality.

When generating non-uniform meshes, the *growth rate*, or mesh density gradient, is controlled by parameter DENSITY-FACTOR. The greater the parameter, the greater the mesh density gradient. This growth rate parameter may be from 1.0 to 3.0.

During free-form meshing, the mesh densities specified along edges are always satisfied. Between edges, however, it is computationally more difficult to satisfy the requested growth rate. Parameter REFINE affects how closely the requested growth rate is satisfied within the domain:

- **Meshing Speed:** If REFINE = EDGE-MIDDLE, a mesh edge that is too long is split at the middle. This procedure is used to optimize for meshing speed at the expense of growth rate accuracy. Typically, fewer elements are generated and the mesh density gradients will be less smooth.
- **Subdivision Accuracy:** If REFINE = ALONG-EDGE, a mesh edge that is too long is split along its length so that the resulting sub-edges satisfy the mesh density requirements along the edge. This procedure is used to optimize for subdivision accuracy, at the expense of computational time. Resulting meshes will more closely and smoothly match the requested growth rate (DENSITY-FACTOR).

Command inputs 2.35 and 2.36 generate meshes using the Delaunay method and both options for parameter REFINE. Both examples use DENSITY-FACTOR = 1.2.

Command Input 2.35: Free-form triangular mesh obtained using the Delaunay method and `REFINE = EDGE-MIDDLE`. Figure 2.36 shows the resulting mesh.

```
feprogram program=adina
body block name=1 option=centered position=vector,
    cx1=0.0 cx2=0.0 cx3=0.0 dx1=2.0 dx2=2.0 dx3=2.0
body cylinder name=2 option=centered position=vector,
    cx1=0.0 cx2=0.0 cx3=0.0 axis=zl radius=0.1 length=4.0
*
body subtract name=1 keep-too=no keep-imp=no
2
*
subdivide face 1 mode=length size=0.2
*
subdivide edge 15 mode=length size=0.01
subdivide edge 17 mode=length size=0.01
*
egroup shell
*
gface 1 nodes=4 preshape=triangular method=delaunay,
        refine=edge-middle density-factor=1.2
```

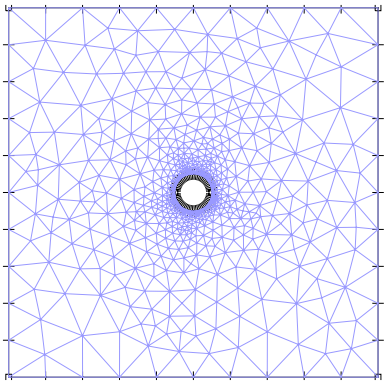


Figure 2.36: Delaunay mesh using `REFINE = EDGE-MIDDLE`.

Command Input 2.36: Continuation of command input 2.35 to remesh using `REFINE = ALONG-EDGE`. See Figure 2.37.

```
eldelete face 1 body=1
*
gface 1 nodes=4 preshape=triangular method=delaunay,
        refine=along-edge density-factor=1.2
```

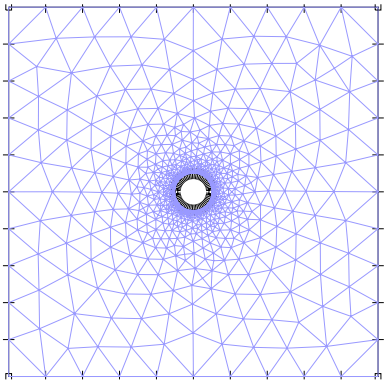


Figure 2.37: Delaunay mesh using `REFINE = ALONG-EDGE`. Note the smoother, more even growth rate (mesh density graduation) as compared to Figure 2.36.

Figures 2.36 and 2.37 show the effect of parameter `REFINE` with constant `DENSITY-FACTOR = 1.2`, which results in a gentle growth rate, or mesh density gradient.

If fewer elements are required, but the mesh densities at the outer edges and around the hole must be preserved, the mesh density gradient can be made steeper by increasing `DENSITY-FACTOR`. Command input 2.37 demonstrates its use.

Command Input 2.37: Free-form triangular mesh obtained using the Delaunay method and REFINE = ALONG-EDGE. Now, the mesh density gradient has been steepened by setting DENSITY-FACTOR = 1.8. Figure 2.38 shows the resulting mesh.

```
feprogram program=adina
body block name=1 option=centered position=vector,
    cx1=0.0 cx2=0.0 cx3=0.0 dx1=2.0 dx2=2.0 dx3=2.0
body cylinder name=2 option=centered position=vector,
    cx1=0.0 cx2=0.0 cx3=0.0 axis=zl radius=0.1 length=4.0
*
body subtract name=1 keep-too=no keep-imp=no
2
*
subdivide face 1 mode=length size=0.2
*
subdivide edge 15 mode=length size=0.01
subdivide edge 17 mode=length size=0.01
*
egroup shell
*
gface 1 nodes=4 preshape=triangular method=delaunay,
        refine=along-edge density-factor=1.8
```

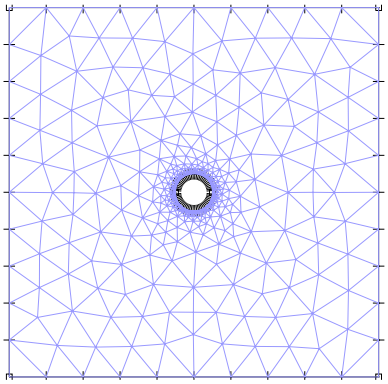


Figure 2.38: Free-form triangular mesh using the Delaunay method, parameters REFINE = ALONG-EDGE, and DENSITY-FACTOR = 1.8. Compare to Figure 2.37.

Figure 2.38 also illustrates the impact of increasing parameter DENSITY-FACTOR. The mesh densities near the circular hole and the outside edges are preserved, but fewer elements have been generated, as compared to Figure 2.37.

Curvature-based sizing meshing (parameters GEO-ERROR, SAMPLING, and MIN-SIZE), automatic grading (AUTO-GRADING), and size functions (SIZE-FUNCTION) are available when using the Delaunay method to general triangular meshes.[8] For more information, see Mesh Size Control on page 22. These options will alter the subdivisions on the bounding body edges and influence the resulting face mesh.[9]

Command input 2.38 demonstrates the use of curvature-based meshing with the Delaunay method. Figure 2.39 shows the resulting geometry and mesh.

[8] Note that the parameter REFINE is ignored when using curvature-based sizing and/or automatic grading.

[9] Setting SIMULATE = YES will suppress the creation of the mesh, thus only the subdivisions on the bounding body edges will be affected.

Command Input 2.38: Curvature based meshing with the Delaunay method and triangular elements. Figure 2.39 shows the resulting mesh. Note that no subdivisions are set.

```
feprogram program=adina
body block name=1 option=centered position=vector,
    cx1=0.0 cx2=0.0 cx3=0.0 dx1=2.0 dx2=2.0 dx3=2.0
body cylinder name=2 option=centered position=vector,
    cx1=0.9 cx2=0.0 cx3=0.0 axis=zl radius=0.05 length=4.0
*
body subtract name=1 keep-too=no keep-imp=no
2
*
egroup shell
*
gface 1 nodes=4,
    prefshape=triangular method=delaunay,
    geo-error=0.02 sampling=20 min-size=0.00001
```

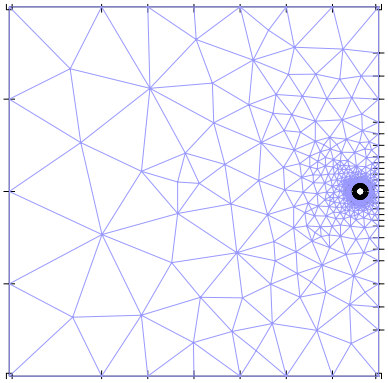


Figure 2.39: Free-form triangular mesh using the Delaunay method and curvature-based sizing.

2.3.2 *Quadrilateral meshing*

To obtain an all-quadrilateral mesh, parameter PREFSHAPE must be set to QUAD-DIRECT.

Generating an all-quadrilateral mesh requires that the total number of subdivisions on the bounding body edges be even. When parameter EVEN = SUM, the sum of all subdivisions on the bounding body edges is made even prior to meshing. When EVEN = ALL, the number of subdivisions on *all* bounding body edges is made even prior to meshing. When parameter SIMULATE = YES, the subdivision adjustment is made (if necessary) without meshing.

Command Input 2.39: Generating an all-quadrilateral mesh using PREFSHAPE = QUAD-DIRECT. Figure 2.40 shows the resulting mesh.

```
feprogram program=adina
body block name=1 dx1=2.0 dx2=2.0 dx3=2.0
body cylinder name=2 option=centered position=vector,
    cx1=0.0 cx2=0.0 cx3=0.0 axis=zl radius=0.1 length=4.0
*
body subtract name=1 keep-too=no keep-imp=no
2
*
subdivide face 1 mode=length size=0.2
subdivide edge 15 mode=length size=0.01
subdivide edge 17 mode=length size=0.01
*
egroup shell
*
gface 1 nodes=4,
    prefshape=quad-direct density-factor=1.2 even=sum
```

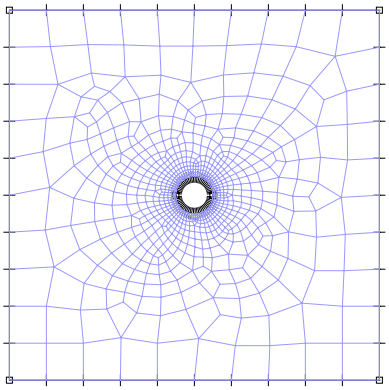


Figure 2.40: Free-form quadrilateral mesh generated using PREFSHAPE = QUAD-DIRECT.

Command Input 2.40: Continuation of command input 2.40 to remesh using DENSITY-FACTOR = 1.8. Figure 2.41 shows the resulting mesh.

```
eldelete face 1 body=1
*
gface 1 nodes=4,
    prefshape=quad-direct density-factor=1.8 even=sum
```

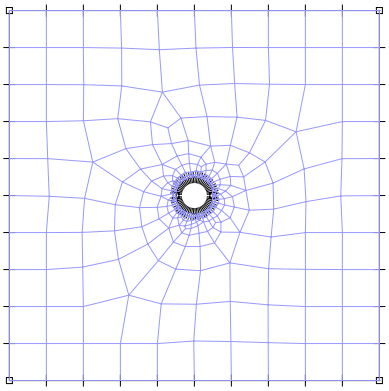


Figure 2.41: All-quadrilateral mesh generated using DENSITY-FACTOR = 1.8.

Figure 2.40 shows an example of free-form all-quadrilateral meshing with DENSITY-FACTOR = 1.2. Figure 2.41 shows the same body face and subdivisions but meshed with DENSITY-FACTOR = 1.8.[10]

It is also possible to generated mixed meshes using parameters METHOD = ADVFRONT and PREFSHAPE = QUADRILATERAL. However, mixed meshes are not recommended, because all-quadrilateral meshes can be obtained as readily.

[10] For all-quadrilateral meshing, DENSITY-FACTOR = 1.2 is recommended.

2.3.3 *Boundary layer meshing*

Boundary layers (sometimes called *inflation layers*) can be created by specifying a boundary layer table with the BLTABLE-2D command and using GFACE 2DBLTABLE. The command BLTABLE-2D has options for specifying defaults for the number of layers (N-LAYER), and layer thicknesses (THICK-FIRST and THICK-TOTAL). Row entries for

BLTABLE-2D are used to specify edges for which the defaults do not apply. [11]

The boundary layer elements are quadrilateral. Elements in the remaining domain may be either triangular (parameter PREFSHAPE = TRIANGULAR) or quadrilateral (PREFSHAPE = QUAD-DIRECT). PREFSHAPE = QUADRILATERAL generates both triangular and quadrilaterial elements, but quadrilateral elements are preferred. PREFSHAPE is only used in free-form meshes with NODES is 4, 8, 9, or 16.[12]

Command inputs 2.41 and 2.42 demonstrate how to use BLTABLE-2D in two dimensions.

Command Input 2.41: Generating a boundary layer mesh using 2DBLTABLE and PREFSHAPE = TRIANGULAR. Figure 2.42 shows the resulting mesh. Note the single row with edge number 12 specified as having zero boundary layer thickness. All other edges have the default boundary layer parameters applied.

```
feprogram program=adina
body block name=1 option=centered position=vector,
    cx1=0.0 cx2=0.0 cx3=0.0 dx1=2.0 dx2=2.0 dx3=2.0
body cylinder name=2 option=centered position=vector,
    cx1=0.0 cx2=0.0 cx3=0.0 axis=zl radius=0.6 length=4.0
*
body subtract name=1 keep-too=no keep-imp=no
2
*
subdivide face 1 body=1 mode=length size=0.1
*
bltable-2d name=1 progress=geometric apply-default=all,
     n-layer=6 thick-first=0.01 thick-total=0.16
12 1 0.0 0.0
*
egroup shell
*
gface 1 nodes=4 2dbltable=1 prefshape=triangular
```

Command Input 2.42: Continuation of command input 2.41 to remesh using PREFSHAPE = QUAD-DIRECT. Figure 2.43 shows the resulting mesh.

```
eldelete face 1 body=1
*
gface 1 nodes=4 2dbltable=1 prefshape=quad-direct
```

[11] This feature simplifies user input when there are many edges which should have boundary layers applied.

[12] The NODES parameter specifies the maximum number of nodes per element outside of the boundary layer.

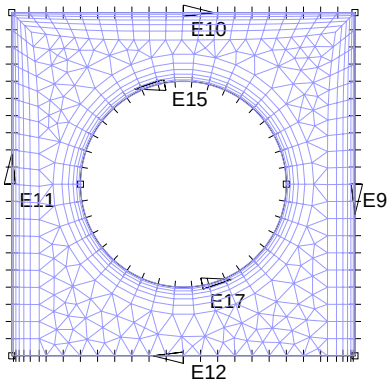


Figure 2.42: Boundary layer meshing with triangular elements between layers. (PREFSHAPE = TRIANGULAR). Note that the defaults in BLTABLE-2D are applied to all edges except for edge 12, which is specified to have zero boundary layer thickness.

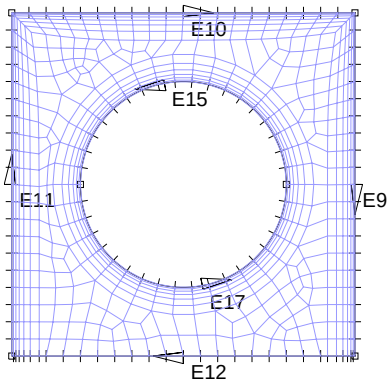


Figure 2.43: Boundary layer meshing with quadrilateral elements between layers (PREFSHAPE = QUAD-DIRECT).

2.3.4 *Mid-side node placement*

Users interested in *mapped* Body face meshing with higher order elements should use the MIDNODES = CURVED option to map mid-side nodes to curves following the parametric space of the body face (see Figure 2.24). However, when *free-form* meshing is desired, the MIDNODES = CURVED option may yield distorted second-order elements near regions where the parametric space becomes singular (e.g., the pole of a sphere). Thus, it is preferred to generate the free-form second-order meshes using MIDNODES = PROJECT (see Figure 2.44).

For complex geometries, both the MIDNODES = CURVED and MIDNODES = PROJECT options can yield distorted elements. In this case, it may be necessary to use MIDNODES = STRAIGHT to ensure that mid-side nodes are placed midway on the straight line between vertex nodes.

Command inputs 2.43 and 2.44 illustrate the effects of free-form body face meshing with MIDNODES = CURVED and MIDNODES = PROJECT on a the faces of spherical body.

Command Input 2.43: Commands for generating the geometry and free-form mesh shown in Figure 2.44, panel *a* with MIDNODES = CURVED.

```
feprogram program=adina
body sphere radius=1.0
*
subdivide body name=1 mode=length size=0.3
*
egroup shell
*
gface nodes=8 prefshape=triangular midnodes=curved
1 1
2 1
*
view name=1 type=parallel xview=0 yview=0 zview=-1
nodedepiction name=1 symbolplot=yes
meshplot view=1 nodedepiction=1
```

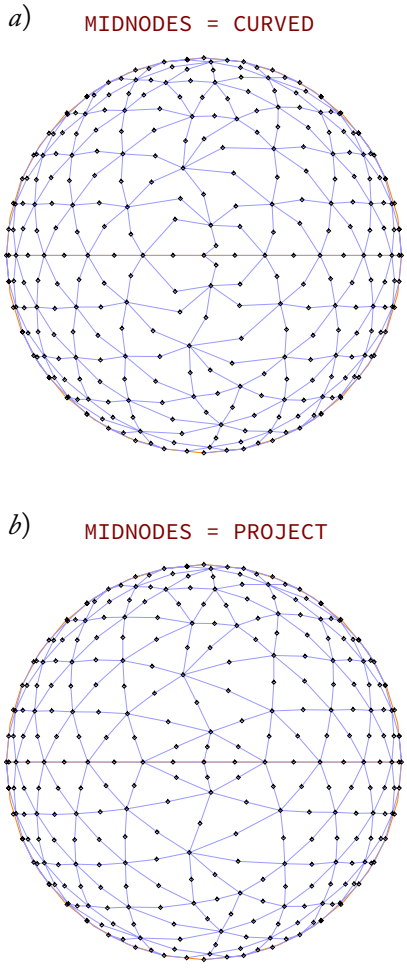


Figure 2.44: The effect of MIDNODES when creating free-form second-order meshes. A spherical body's faces are meshed and viewed from above a pole. When MIDNODES = CURVED, mid-side nodes are always mapped onto curves defined by the body's parametric space, which in this case leads to distorted elements near the pole (center, panel *a*). Because MIDNODES = PROJECT does not map mid-side nodes onto the body's parametric space, the elements near the pole are of higher quality (center, panel *b*)

Command Input 2.44: Continuation of command input 2.43 for generating the free-form mesh shown in Figure 2.44, panel *b* with `MIDNODES = PROJECT`.

```
eldelete face name=1 body=1
eldelete face name=2 body=1
*
gface nodes=8 prefshape=triangular midnodes=project
1 1
2 1
*
frame
meshplot view=1 nodedepiction=1
```

2.4 *Body Free-Form Meshing*

Bodies are free-form meshed using the command `GBODY MESHING = FREE-FORM`. If the number of `NODES` is set to `4`, `10`, or `11`, `GBODY` will generate tetrahedral elements. If `NODES` is set to `8`, `20`, or `27`, `GBODY` will generate a mixed mesh of hexahedral, pyramid, and tetrahedral elements. If boundary layers are desired, either all-prismatic (wedge) elements or all-hexahedral (brick) elements will be generated in the layers.

Note that free-form meshing produces slightly different meshes on different platforms, and meshes generated on your computer may not exactly match those shown in the figures.

2.4.1 *Tetrahedral meshing*

Tetrahedral meshing is enabled when `NODES` is `4`, `10`, or `11`. There are three free-form tetrahedral meshing schemes available: advancing front (`METHOD = ADVFRONT`), Delaunay (`METHOD = DELAUNAY`), and hybrid advancing front/Delaunay (`METHOD = ADVFDELA`).

The advancing front method begins from the triangular meshes on bounding faces and works its way inward. As the mesher advances, it attempts to fill the remaining domain by connecting a triangular mesh face on the advancing front to a new or existing mesh vertex. This process repeats incrementally, thereby filling the domain with elements.

The advancing front method creates well-shaped elements near boundaries. Unfortunately however, it does not always succeed in closing the front, except for the simplest of geometries. Thus, the advancing front method is not recommended, in general.

The Delaunay free-form tetrahedral mesh generator is more robust. It performs the following operations:

1. Build an initial mesh for a box bounding the body. When parameter `GRID = YES`, this initial mesh is constructed by uniformly subdividing the box into sub-boxes which are then meshed with five tetrahedral elements (mesh regions) each, otherwise, the box is meshed directly with five mesh regions.
2. Insert the boundary vertices (coming from the meshing of the body faces) into this initial mesh.
3. Recover the boundary edges and the boundary faces. After insertion into the mesh of the boundary vertices, not all boundary edges and faces will actually be present in the mesh. They therefore need to be recovered by performing topological operations on the mesh.
4. Refine the mesh until all mesh density requirements are met. When `REFINE = EDGE-MIDDLE`, if a mesh edge is too long, a new mesh vertex is inserted at its middle. When `REFINE = ALONG-EDGE`, if a mesh edge is too long, new mesh vertices are inserted along the edge according to the mesh densities along the edge. Parameter `DENSITY-FACTOR` controls the *growth rate*, or the mesh density gradient when the subdivisions are not uniform.
5. Optimize the quality of the mesh. The number of optimization passes is controlled by parameter `NOPTI`.

Typically, one would control the mesh refinement process using either `REFINE = EDGE-MIDDLE` (for faster performance) or `REFINE = ALONG-EDGE` (for better subdivision accuracy).

For large meshes, `REFINE = EDGE-MIDDLE` will usually run faster and create fewer elements, though it will less accurately satisfy internal mesh density requirements. `REFINE = ALONG-EDGE` will typically generate more elements and require more CPU time, but it will more accurately satisfy the mesh density requirements away from edges.[13] If the surface mesh has small triangles resulting

[13] The mesh density requirements on edges are always satisfied.

from the presence of small body edges and/or faces, parameter DENSITY-FACTOR may need to be increased (and parameter REFINE set to EDGE-MIDDLE) to prevent over-refining.

Command input 2.45 demonstrates the use of GBODY using the Delaunay method to generate a free-form tetrahedral mesh.

Command Input 2.45: Generating a tetrahedral mesh using GBODY and METHOD = DELAUNAY. Figure 2.45 shows the resulting mesh. Note the additional commands for exposing the internal mesh structure.

```
feprogram program=adina
body block name=1 option=centered position=vector,
    cx1=0.0 cx2=0.0 cx3=0.0 dx1=2.0 dx2=2.0 dx3=2.0
body cylinder name=2 option=centered position=vector,
    cx1=0.0 cx2=0.0 cx3=0.0 axis=xl radius=0.2 length=4.0
*
body subtract name=1 keep-too=no keep-imp=no
2
*
subdivide body 1 mode=length size=0.2
subdivide face 7 body=1 mode=length size=0.02
subdivide face 8 body=1 mode=length size=0.02
*
egroup threedsolid
*
gbody 1 nodes=4 method=delaunay
*
* COMMANDS FOR EXPOSING THE INTERNAL MESH
boxzone b
-10 0 -10 10 -10 10
modeldepic geometry=no
frame/meshplot zone=b
```

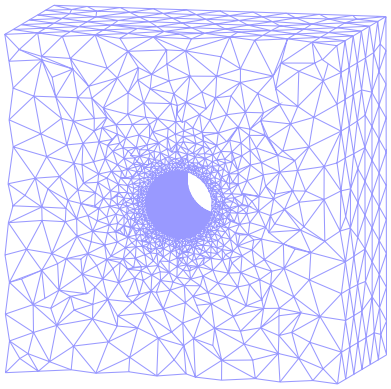


Figure 2.45: Free-form tetrahedral mesh generated using METHOD = DELAUNAY. Internal mesh structure exposed.

The hybrid advancing front/Delaunay method (selected by setting METHOD = ADVFDELA – see command input 2.46) is useful for cases in which the advancing front method fails. It can mesh relatively complex bodies with an advancing front type method.

Command Input 2.46: Continuation of command input 2.45 to regenerate the mesh shown in Figure 2.46 using METHOD = ADVFDELA.

```
eldelete body 1
*
egroup threedsolid
*
gbody 1 nodes=4 method=advfdela
*
* COMMANDS FOR EXPOSING THE INTERNAL MESH
boxzone b
-10 0 -10 10 -10 10
modeldepic geometry=no
frame/meshplot zone=b
```

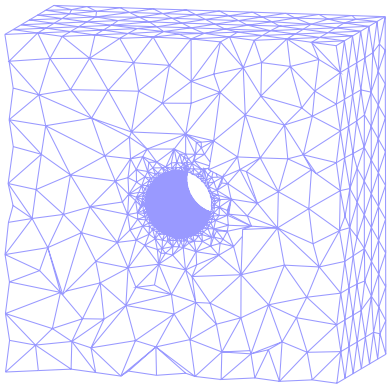


Figure 2.46: Free-form tetrahedral mesh generated using METHOD = ADVFDELA. Internal mesh structure exposed.

The meshing algorithm used for the bounding body faces is controlled by parameter BOUNDARY-METHOD. If BOUNDARY-METHOD = ADVFRONT, the advancing front method is used. If BOUNDARY-METHOD = DELAUNAY, the Delaunay method is used. Just as the REFINE controls how the tetrahedral mesh is refined when BOUNDARY-METHOD = DELAUNAY, BREFINE controls how the triangular mesh on the bounding body faces are refined when BOUNDARY-METHOD = DELAUNAY.[14]

[14] REFINE and BREFINE are not used if PYRAMIDS = ONLY.

As with body face free-form triangular meshing, size functions (parameter SIZE-FUNCTION), curvature-based sizing (parameters GEO-ERROR, SAMPLING, and MIN-SIZE), and automatic grading (parameter AUTO-GRADING) can be used to control mesh densities while meshing. These are available only if options METHOD and BOUNDARY-METHOD are both set to DELAUNAY.[15] The effect of these mesh density parameters can be limited to the bounding body edges by setting SIMULATE = YES, as detailed in Mesh Size Control on page 22.

[15] In fact, if any of these size controls is specified with METHOD and/or BOUNDARY-METHOD set to ADVFRONT, ADINA will ignore the erroneous request to use the advancing front method and override with DELAUNAY.

Command input 2.47 demonstrates how to generate a free-form tetrahedral mesh using curvature-based sizing.

Command Input 2.47: Generating a tetrahedral mesh using curvature-based sizing (GEO-ERROR). Figure 2.47 shows the resulting mesh.

```
feprogram program=adina
body block name=1 option=centered position=vector,
    cx1=0.0 cx2=0.0 cx3=0.0 dx1=2.0 dx2=2.0 dx3=2.0
body cylinder name=2 option=centered position=vector,
    cx1=0.0 cx2=0.7 cx3=0.0 axis=xl radius=0.2 length=4.0
*
body subtract name=1 keep-too=no keep-imp=no
2
*
egroup threedsolid
*
gbody 1 nodes=4 method=delaunay boundary-method=delaunay,
        geo-error=0.02 sampling=20 min-size=0.00001
*
boxzone b
-10 0 -10 10 -10 10
modeldepic geometry=no
frame/meshplot zone=b
```

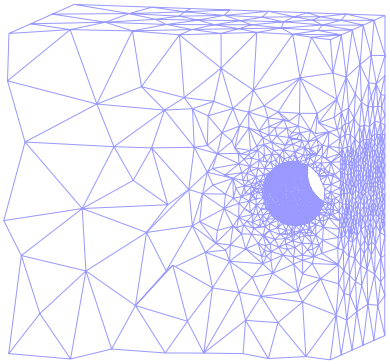


Figure 2.47: Free-form tetrahedral mesh generated using curvature-based sizing (GEO-ERROR).

Figure 2.47 shows a free-form tetrahedral mesh with curvature-based sizing enabled (parameter GEO-ERROR).[16] There were no subdivisions prior to meshing. The effect is most evident on the triangular mesh on the face nearest the cylindrical wall.

[16] Note that automatic grading is always turned on when curvature-based sizing is requested.

The parameter NLAYER [17] specifies the minimum number of element layers through the section. This option is useful for bodies with thin regions (*e.g.* locations likely to experience strong stress gradients). Command inputs 2.48 and 2.49 demonstrate the effects of varying NLAYER.

[17] GBODY NLAYER is not used if PYRAMIDS = ONLY or if boundary layers are used. For more information about Boundary layer meshing, see page 68.

Command Input 2.48: Thin-walled geometry being meshed with `NLAYER = 1`. Figure 2.48, (panel *a*) shows the resulting mesh.

```
feprogram program=adina
body block name=1 option=centered position=vector,
    cx1=0.0 cx2=0.0 cx3=0.0 dx1=2.0 dx2=2.0 dx3=2.0
body cylinder name=2 option=centered position=vector,
    cx1=0.0 cx2=0.0 cx3=0.0 axis=xl radius=0.9 length=4.0
*
body subtract name=1 keep-too=no keep-imp=no
2
*
subdivide body 1 mode=length size=0.2
subdivide face 7 body=1 mode=length size=0.2
subdivide face 8 body=1 mode=length size=0.2
*
egroup threedsolid
*
gbody 1 nodes=4 nlayer=1 method=delaunay
```

Command Input 2.49: Continuation of command input 2.48 regenerating the mesh using `NLAYER = 3`. Figure 2.48, (panel *b*) shows the resulting mesh.

```
eldelete body 1
*
egroup threedsolid
*
gbody 1 nodes=4 nlayer=3 method=delaunay
```

Consider, for example, the body shown in Figure 2.48. Meshing with `NLAYER = 1` will generate a mesh with a single 'layer' of elements in the thin regions. Increasing `NLAYER = 3` ensures that those thin regions are meshed using at least 3 layers of elements, as shown in Figure 2.48, (panel *b*). Clearly, this second mesh can better resolve strong stress gradients in the thin walls.

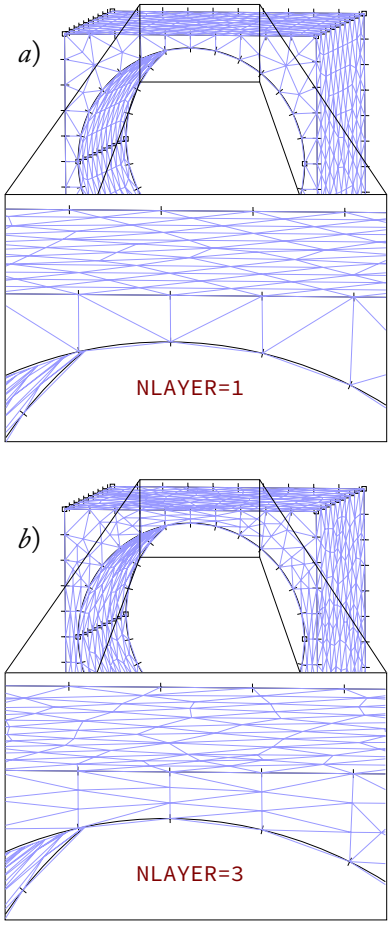


Figure 2.48: Free-form tetrahedral mesh generated using `NLAYER = 1` (panel *a*) and `NLAYER = 3` (panel *b*).

2.4.2 *Mixed meshing*

Mixed meshing is a powerful and versatile technique, and its versatility is most effectively used with an understanding of some basic behaviors. More complex meshing situations, however, require a more detailed understanding of the associated meshing options and how those options interact to affect the final mesh.

By default (with `GBODY MESHING = MIXED`), mixed meshing can be performed. In the special case that the body to be meshed is topologically-equivalent to a hexahedron (and that the subdivisions allow for it), `GBODY MESHING = MIXED` will generate a mapped mesh. However, in all other cases, `GBODY MESHING = MIXED` will generate a free-form mesh.

Free-form mixed meshing is enabled when `NODES` is set to `8`, `20`, or `27` and `MESHING = FREE-FORM`. The approach is based on the advancing front method, where the initial front is the quadrilateral surface mesh on the bounding body faces. Hexahedral elements are generated from those quadrilateral facets until it becomes impossible to do so.

Once it becomes impossible to advance the front with hexahedral elements, the mesher must either transition to tetrahedrons using pyramids or simply create tetrahedrons without the transitional pyramids. In free-form meshing, the user can control this behavior using the parameter `PYRAMIDS`.

- If `PYRAMIDS = NO`, the leading part of the front is meshed with tetrahedral elements and no pyramid elements. This option is used only when `NODES` is set to `8`, `20`, or `27`.
- If `PYRAMIDS = YES`, pyramid elements are created from the quadrilateral facets on the front, and the remaining domain is meshed with tetrahedral elements. This option is used only when `NODES` is set to `8`, `20`, or `27`.
- If `PYRAMIDS = ONLY`, pyramid elements are created off the quadrilateral facets coming from the quadrilateral surface mesh and tetrahedral elements are used to fill the rest of the domain. No hexahedral elements are generated. This option is useful when some of the bounding body faces are already meshed with quadrilaterals (possibly because of facelinks) but tetrahedral elements are preferred.

Command input 2.50 demonstrates how to generate a free-form mixed mesh with PYRAMIDS = NO.

Command Input 2.50: Generating a mixed mesh using PYRAMIDS = NO. Figure 2.49 shows the resulting mesh.

```
feprogram program=adina
body block name=1 option=centered position=vector,
    cx1=0.0 cx2=0.0 cx3=0.0 dx1=2.0 dx2=2.0 dx3=2.0
body cylinder name=2 option=centered position=vector,
    cx1=0.0 cx2=0.0 cx3=0.0 axis=xl radius=0.2 length=4.0
*
body subtract name=1 keep-too=no keep-imp=no
2
*
subdivide body 1 mode=length size=0.1
subdivide face 7 body=1 mode=length size=0.04
subdivide face 8 body=1 mode=length size=0.04
*
egroup threedsolid
*
gbody 1 nodes=8 pyramids=no
*
boxzone b
-10 0 -10 10 -10 10
modeldepic geometry=no
frame/meshplot zone=b
```

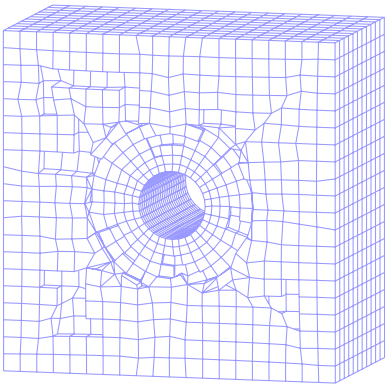


Figure 2.49: Free-form mixed mesh generated using PYRAMIDS = NO.

Setting NODES = 20 forces PYRAMIDS = YES. This is because it is impossible to have a higher order tetrahedral element adjacent to a 20-node hexahedral element (the tetrahedral element brings an extra mid-side node).

Similarly, in ADINA-F, parameter PYRAMIDS is always forced to be YES, no matter the number of nodes requested. This is because ADINA-F requires transitional pyramid elements between hexahedral and tetrahedral elements.

Because the initial front on the bounding body faces for mixed meshes is composed of all-quadrilateral boundary cells, the total number of bounding body edges must be even. The EVEN parameter allows users to control how the subdivisions should be adjusted to satisfy this requirement.

- Setting EVEN = SUM, the sum of all subdivisions on the bounding body edges for each body face is made even prior to meshing.

- Face-linking on a free-formed mixed mesh boundary requires that the face(s) to be linked have an even number of subdivisions on each bounding edge. When `EVEN = LINK`, this condition is satisfied before meshing. Also, the sum of the subdivisions of the remaining body edges is made even, if necessary.
- Setting `EVEN = ALL`, the number of subdivisions on all bounding body edges is made even prior to meshing.

The quality of the hexahedral elements produced is controlled by parameters `DANGMAXB`, `DANGMAXC`, and `DANGMAXD`:

- `DANGMAXB`, or *Internal Angle Deviation*, controls the maximum allowed deviation from 90 degrees for the internal angle at the corners of quadrilateral facets.
- `DANGMAXC` controls the maximum allowed deviation from 180 degrees for the dihedral angle at the diagonals of quadrilateral facets.
- `DANGMAXD` controls the maximum allowed deviation from 90 degrees for the dihedral angle at the edges of hexahedral elements.

The higher the quality requested, the fewer hexahedral elements are generated.

It might not be possible to create an all-hexahedral mesh. In that case, it may help to decrease parameter `DANGMAXC`, which makes quadrilateral facets more planar.

Command input 2.51 demonstrates the use of transitional pyramids between face-linked mapped hexahedral and free-form tetrahedral meshes.

Command Input 2.51: Meshing two face-linked bodies. Body 1 is map meshed using 8 node bricks, and body 2 is free-form meshed. Figure 2.50 shows the resulting mesh.

```
feprogram program=adina
body block name=1 option=centered position=vector,
    cx1=0.0 cx2=0.0 cx3=0.0 dx1=1.0 dx2=1.0 dx3=1.0
body block name=2 option=centered position=vector,
    cx1=0.0 cx2=1.0 cx3=0.0 dx1=1.0 dx2=1.0 dx3=1.0
*
facelink option=all
*
subdivide body 1 mode=length size=0.1
subdivide body 2 mode=length size=0.1
*
egroup name=1 type=threedsolid
egroup name=2 type=threedsolid
*
gbody 1 group=1 nodes=8
gbody 2 group=2 nodes=4 meshing=free-form pyramids=only
*
boxzone b
-10 0 -10 10 -10 10
modeldepic geometry=no
frame/meshplot zone=b
```

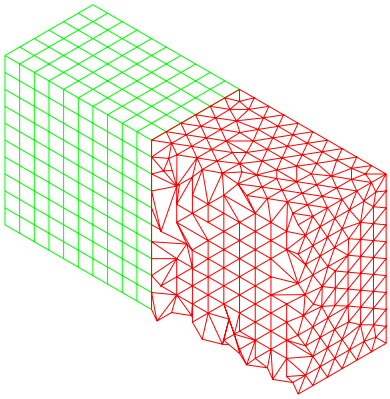


Figure 2.50: Free-form mesh (in red) generated using tetrahedral elements and transitional pyramids.

Figure 2.50 shows the results of Command input 2.51. Body 1 has been map meshed with hexahedral elements. Body 2 is face linked (see page 14) to body 1 and is meshed using tetrahedral elements and transitional pyramids.

To generate the red mesh containing mostly tetrahedral elements but transition pyramids on the interface with the hexahedral mesh, as shown, the user must:

1. Specify `NODES = 4` to mesh the body using tetrahedral elements.
2. Invoke free-form mixed meshing using `MESHING = FREE-FORM`.
3. Specify `PYRAMIDS = ONLY`.

The `PYRAMIDS = ONLY` option generates transitional elements only on the linked body faces that have quadrilaterals. Transitional pyramids are required for CFD analysis to ensure a compatible mesh. In structural analysis, the `PYRAMIDS = ONLY` option is not required, and tetrahedral elements can be directly attached to the hexahedral elements without transitional pyramids.

When generating higher-order mixed meshes, the user should be aware of two related node-placement options.

GBODY MIDNODES specifies where mid-side nodes are to be located when at least one edge of the geometry body is curved. This GBODY[18] parameter works in the same way as its GFACE equivalent. For more information about mid-side node placement, see the sections on Body face meshing on page 39 and Mid-side node placement on page 57. Figure 2.24 illustrates the different mid-side node placement options.

GBODY MIDFACENODES determines where the mid-face node on a quadrilateral facet should be placed.

- MIDFACENODES = TRIA places the mid-face node midway on the diagonal of the two triangles making up the quad face (see Figure 2.51, panel *a*).
- MIDFACENODES = QUAD places the mid-face node on the quad face's centroid (see Figure 2.51, panel *b*).

The parameter MIDFACENODES is especially important when generating a higher-order mixed mesh which is to be face-linked with a higher-order tetrahedral mesh. In that case, MIDFACENODES = TRIA must be used; the mid-face node will coincide with the tetrahedral element's mid-side node, and face-linking will be successful. If MIDFACENODES = QUAD, the mid-face node will *not*, in general, coincide with the tetrahedral element's mid-side node (as shown in Figure 2.51, panel *b*), and face-linking will fail.

If face-linking is not required, MIDFACENODES = QUAD is recommended and will lead to better quality elements.

By default (MIDFACENODES = DEFAULT), if the body has at least one linked face, the MIDFACENODES = TRIA option is used. Otherwise, MIDFACENODES = QUAD is used.

[18] GBODY MIDNODES works in the same way for both free-form and mapped body meshing.

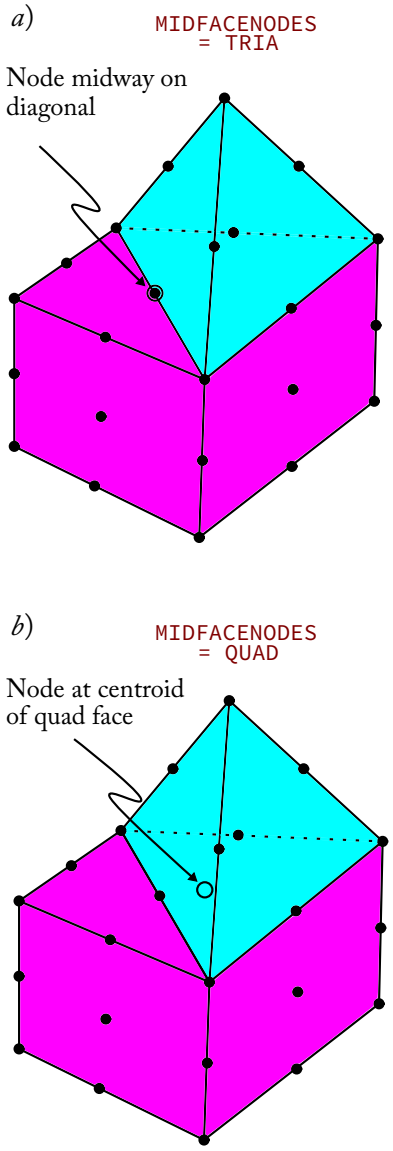


Figure 2.51: The effect of MIDFACENODES. The mid-face node on the quadrilateral face is denoted by the open circle. When MIDFACENODES = TRIA, the mid-face node is placed midway on the diagonal (panel *a*). When MIDFACENODES = QUAD, the mid-face node is placed at the quadrilateral facet's centroid (panel *b*).

2.4.3 *Boundary layer meshing*

If boundary layers are required, a boundary layer table must be specified using BLTABLE-3D and parameter 3DBLTABLE must be passed to GBODY.

The element shapes inside and outside the boundary layers depend upon both NODES and PREFSHAPE.

- NODES = 4, 10, or 11: the boundary layer elements are prismatic and the rest of the mesh is tetrahedral.
- NODES = 8, 20, or 27 and PREFSHAPE = PRISMATIC: the boundary layer elements are prismatic and the rest of the mesh is tetrahedral.
- NODES = 8, 20, or 27 and PREFSHAPE = HEXAHEDRAL: the boundary layer elements are hexahedral and the rest of the mesh is a mixture of hexahedral, tetrahedral, and pyramid elements.

Command inputs 2.52 and 2.53 demonstrate how to use BLTABLE-3D.

Command Input 2.52: Generating a boundary layer mesh using prismatic elements. Figure 2.52 shows the resulting mesh.

```
feprogram program=adina
body block name=1 option=centered position=vector,
    cx1=0.0 cx2=0.0 cx3=0.0 dx1=2.0 dx2=2.0 dx3=2.0
body cylinder name=2 option=centered position=vector,
    cx1=0.0 cx2=0.0 cx3=0.0 axis=xl radius=0.6 length=4.0
*
body subtract name=1 keep-too=no keep-imp=no
2
*
subdivide body 1 mode=length size=0.1
*
bltable-3d name=1 progress=geometric n-layer=6
1 1 0.01 0.16
2 1 0.01 0.16
7 1 0.01 0.16
8 1 0.01 0.16
*
egroup threedsolid
*
gbody 1 nodes=8 3dbltable=1 prefshape=prismatic
*
boxzone b
-10 0 -10 10 -10 10
modeldepic geometry=no
frame/meshplot zone=b
```

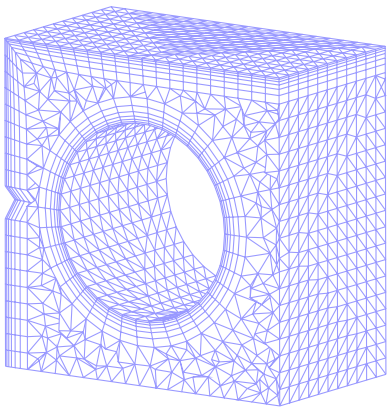


Figure 2.52: Boundary layer meshing using 3DBLTABLE with prismatic elements (PREFSHAPE = PRISMATIC).

Figure 2.52 shows an example of boundary layers composed of prismatic (wedge) elements. In this case, the rest of the domain is meshed with tetrahedral elements.

Command Input 2.53: Continuation of command input 2.52 for remeshing a boundary layer mesh using hexahedral elements. Figure 2.53 shows the resulting mesh.

```
eldelete body 1
gbody 1 nodes=8 3dbltable=1 prefshape=hexahedral
*
boxzone c
-10 0 -10 10 -10 10
modeldepic geometry=no
frame/meshplot zone=c
```

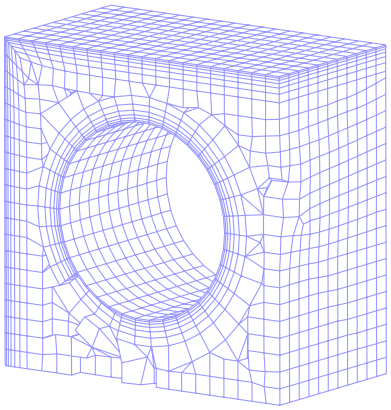


Figure 2.53: Boundary layer meshing using 3DBLTABLE with hexahedral elements (PREFSHAPE = HEXAHEDRAL).

Figure 2.53 shows an example of boundary layers composed of hexahedral elements. In this case, the rest of the domain is meshed with a mix of hexahedral, pyramid, and tetrahedral elements. The creation of pyramid elements is controlled by parameter PYRAMIDS, as with free-form mixed meshing.

Command input 2.54 shows how to use those options to transition from a region with a boundary layer to a region without.

Command Input 2.54: Face linking two adjacent bodies and generating a boundary layer mesh on body 1. Body 2 has no boundary layer. Figure 2.54 shows the resulting mesh.

```
feprogram program=adina
body block name=1 cx2=0.0 dx1=1.0 dx2=1.0 dx3=1.0
body block name=2 cx2=1.0 dx1=1.0 dx2=1.0 dx3=1.0
*
facelink option=all
*
subdivide body 1 mode=length size=0.1
subdivide body 2 mode=length size=0.1
*
bltable-3d name=1 progress=geometric n-layer=6
1 1 0.01 0.16
*
egroup threedsolid
gbody 1 nodes=8 3dbltable=1
gbody 2 nodes=4 meshing=free-form
*
boxzone b
-10 0 -10 10 -10 10
modeldepic geometry=no
frame/meshplot zone=b
```

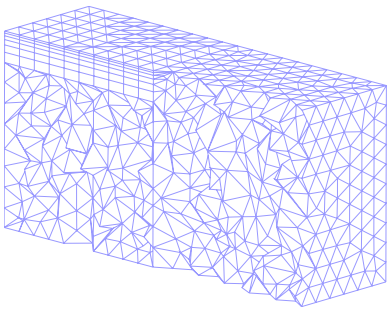


Figure 2.54: Free-form mixed mesh using 4-node tetrahedral and pyramid elements. Note the transition from the boundary layer mesh from the left body to the mesh on the right.

Figure 2.54 illustrates two adjacent, face-linked the bodies. The body on the left has been meshed with a boundary layer. Meshing the body on the right with mostly tetrahedral elements and transitional pyramids requires that `NODES = 4` and `MESHING = FREE-FORM`. Because body 2 has at least one face meshed using triangles, `PYRAMIDS = ONLY` is forced, and transitional pyramids are automatically generated.

2.4.4 *Skin of elements on 3D-solid mesh*

When a body is meshed with a 3D-solid mesh, the triangulation is fixed on the body faces. Thus, if a shell mesh or a 3D plane stress (membrane) mesh is generated on the body faces, the elements will be automatically matched with the 3D-solid element faces. This can be useful when it is important to obtain accurate surface stresses[19] or to model surface treatments, such as carburization.

If the skin of shell or membrane elements is to be created on a face of a separate body, the `COPY-TRIANGULATION` command (see Copying meshes, page 84) can be used to use the same triangulation as that used for the 3D-solid mesh.

[19] A skin places integration points on the surface of the body, thereby resulting in more accurate surface stresses. This is important when modeling fatigue, for example.

2.5 *STL Body Free-Form Meshing*

For more information about the STL format and how to convert Parasolid bodies to STL bodies, see Converting a Parasolid body into an STL body on page 20.

2.5.1 *Tetrahedral meshing*

STL bodies are composed of triangular facets. Before meshing the inside of a STL body with tetrahedral elements, the surface mesh must be adapted according to the mesh densities given using the `SUBDIVIDE BODY`, `SUBDIVIDE FACE`, and/or `SUBDIVIDE EDGE` commands.

The starting point for this adaptation process is the set of triangular facets associated with the STL body. Since the geometry associated with the STL body cannot be changed (it defines the body), it must be copied to a dis-

crete representation using command BODY-DISCREP. This discrete representation is then "adapted" (via command BODY-DSCADAP) according to the previously defined mesh densities.

Once the discrete representation has been adapted, it can then be meshed with tetrahedral elements using GBODY, as usual. After meshing, the discrete representation may be discarded using command DELETE BODY-DISCREP.

To summarize, the steps needed to mesh a STL body with tetrahedral elements are:

1. Create an STL body (command LOAD-STL or CONVERT-STL).
2. Create a discrete representation for that STL body (command BODY-DISCREP).
3. Subdivide the body, faces, and/or edges (command SUBDIVIDE).
4. Adapt the discrete representation (command BODY-DSCADAP).
5. Mesh the discrete representation (command GBODY).
6. Delete the discrete representation (command DELETE BODY-DISCREP).

In the case of an STL body, tetrahedral meshing is limited to the creation of tetrahedral elements (NODES = 4, 10, or 11) without access to the options that are available in the case of a general body (Parasolid or OpenCascade).

Command inputs 2.55, 2.56, and 2.57 detail the above process.

Command Input 2.55: Converting an STL body with CONVERT-STL and generating its discrete representation with BODY-DISCREP. Figure 2.55 shows the resulting discrete representation.

```
feprogram program=adina
body block name=1 option=centered position=vector,
    cx1=0.0 cx2=0.0 cx3=0.0 dx1=1.0 dx2=1.0 dx3=0.2
body cylinder name=2 option=centered position=vector,
    cx1=0.0 cx2=0.0 cx3=0.15 axis=yl radius=0.1 length=1.0
*
body subtract name=1 keep-too=no keep-imp=no
2
*
convert-stl 1
*
body-discrep 1
```

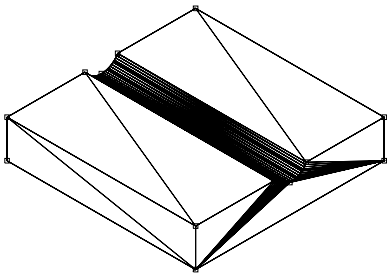


Figure 2.55: Discrete representation after BODY-DISCREP but prior to adaptation.

Command Input 2.56: Continuation of command input 2.55. The STL body is subdivided and adapted using BODY-DSCADAP. Figure 2.56 shows the resulting discrete adaptation.

```
subdivide body 1 mode=length size=.1
subdivide face 8 body=1 mode=length size=.02
subdivide face 9 body=1 mode=length size=.02
*
body-dscadap 1
```

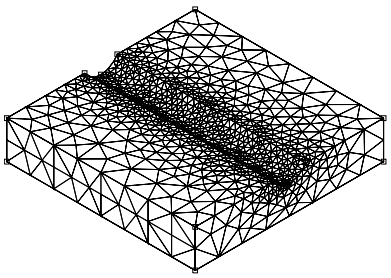


Figure 2.56: Discrete representation after adaptation with BODY-DSCADAP.

Command Input 2.57: Continuation of command input 2.56. The STL body is meshed, and the discrete representation is discarded. Figure 2.57 shows the resulting mesh.

```
egroup threedsolid
*
gbody 1 nodes=4
*
delete body-discrep 1
```

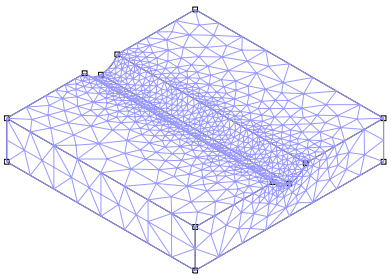


Figure 2.57: Tetrahedral mesh following GBODY.

2.5.2 *All-hexahedral meshing*

An STL body can be meshed exclusively with hexahedral elements using command BHEXA. The hexahedral mesh may be uniform (parameter SIZE set to a non-zero value) or non-uniform (parameter SIZE set to zero and mesh densities applied beforehand with the SUBDIVIDE BODY, FACE, and/or EDGE commands).

The basic steps performed by ADINA during the creation of an all-hexahedral mesh are briefly summarized:

1. Creation of a root octant surrounding the body.
2. Subdivision of the tree according to prescribed mesh densities.
3. Creation of mesh regions (hexahedral elements) using the tree structure.
4. Deletion of mesh regions that are outside or on the boundary of the body.
5. Classification of mesh vertices on points, mesh edges on body edges, and mesh faces on body faces.
6. Projection/smoothing of mesh vertices classified on the body's boundary.

If the STL body does not initially meet the meshing requirements or there is a need to simplify the body (*e.g.*, thin faces, small features, etc.), it is possible to eliminate edges from the STL body using commands `STL ELIM-EDGE` and `STL ELIM-EDGES-ANGLE`. For more information, see Eliminating edges from STL bodies on page 20

To summarize, the steps needed to mesh an STL body using only hexahedral elements are:

1. Create an STL body (command `LOAD-STL` or `CONVERT-STL`) that satisfies the hexahedral meshing requirements.
2. Create a discrete representation for that STL body (command `BODY-DISCREP`).
3. Subdivide the body, faces, and/or edges (command `SUBDIVIDE`). This step should be omitted if the mesh density is uniform (specified with parameter `SIZE` in command `BHEXA`).
4. Mesh with hexahedral elements (command `BHEXA`).
5. Delete the discrete representation (command `DELETE BODY-DISCREP`).

Command inputs 2.58 and 2.59 detail the above process.

Command Input 2.58: Importing a Parasolid file using LOADSOLID and converting it to an STL body using CONVERT-STL. Figure 2.58 shows the STL body.

```
loadsolid 'Hinge Bracket.x_t'
*
convert-stl 1
```

Command Input 2.59: Continuation of command input 2.58. The command STL ELIM-EDGE is used to remove one body edge, and the discrete representation is generated with BODY-DISCREP. The hexahedral mesh is then generated using BHEXA. Finally, the discrete representation is discarded. Figure 2.59 shows the resulting hexahedral mesh.

```
stl elim-edge 1
14
*
body-discrep 1
*
egroup threedsolid
*
bhexa 1 nodes=8 size=1.0
*
delete body-discrep 1
```

There are some limitations on what type of STL body can be meshed with hexahedral elements:

- Any body edge should have two distinct end points and should be connected to two distinct body faces.
- Any point should be connected to two, three, or four distinct body edges. A point connected to four body edges may pose a problem during meshing depending on the arrangement of the connected body edges.

All-hexahedral meshing is an inside-out method, which means that the interior mesh and the boundary mesh are created at the same time. This is unlike tetrahedral meshing on a general body where the bounding body faces are meshed first. Consequently, the all-hexahedral mesher cannot account for an existing mesh on a bounding body face (as may occur, for example, with face-linking).

During its classification phase, BHEXA introduces a substantial number of constraints to determine the locations of boundary vertices. Thus, there is no guarantee that all boundary vertices will be located on the body's boundary.

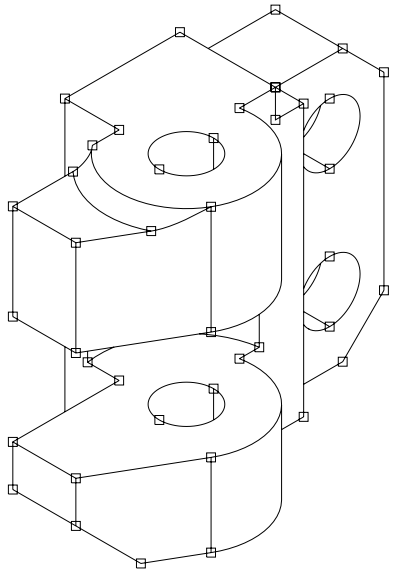


Figure 2.58: STL body converted from Parasolid file.

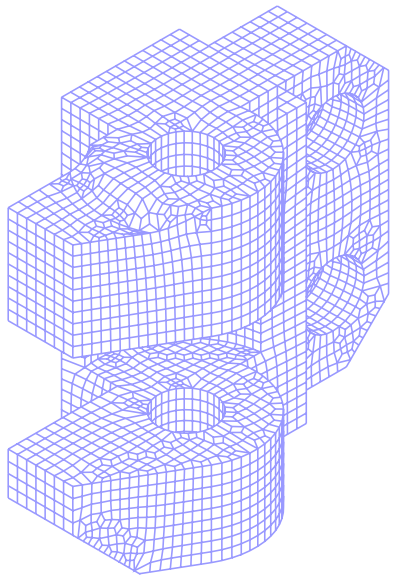


Figure 2.59: Hexahedral mesh following BHEXA.

Also, the all-hexahedral mesher may encounter difficulties when there are thin body faces in the body. In principle, requesting a finer mesh density should be able to resolve thin faces. However, this would likely result in an excessive number of hexahedral elements. Difficulties may also arise where there are sharp angles in the body between connected body faces at a body edge or between connected edges at a point on a body face.

2.6 *Nodal Coincidence*

Finite element models typically involve intricate domains (*e.g.*, seperate structures, fluid regions, etc.), each composed of multiple shapes (*e.g.*, volumes and/or bodies in 3D and faces and/or surfaces in 2D, etc). It is crucially important that those individual meshes be properly joined into a single congruent mesh representing the domain. Further, this mesh must not erroneously include nodes or elements from other domains.

A basic requirement for joining two meshes into a single congruent mesh is that those meshes must, at their interface, have *coincident* nodes.[20] ADINA offers several methods of checking for coincident nodes and *equivalencing* them.[21] Further, ADINA can attach or detach meshes, as desired.

Face-linking is typically used to ensure nodal coincidence at geometric interfaces. For more information about the command FACELINK, see page 14. However, it is also possible to ensure nodal coincidence between meshes by manually subdividing the interfaces and using Mapped Meshing (see 32).

[20] 'Nodal coincidence' is simply a shorter way of saying 'multiple nodes at the same location.'

[21] *Equivalencing* two coincident nodes simply collapses them into a single node. This action effectively joins the meshes which contained those nodes.

2.6.1 *Nodal coincidence checking during meshing*

Coincidence checking is used to determine whether to place a new node at a geometric location where (within the nodal coincidence tolerance NCTOLERANCE [22]) The parameter NCOINCIDE can be used to perform coincidence checking while meshing bodies. Parameter NCOINCIDE is available for all meshing commands (*e.g.*, GBODY, GFACE, GVOLUME, GSURFACE, etc.).

The most often-used options for parameter NCOINCIDE are:

- NCOINCIDE = BOUNDARIES: Checks nodes on the boundaries of the entity being meshed for coincidence against all other existing nodes. This is the default option.
- NCOINCIDE = GROUP: Checks nodes in the entity being meshed for coincidence against all other existing nodes *of the same element group*. This is useful when adjacent shapes modeled within the same element group (but not

[22] If NCTOLERANCE = 0, then the tolerance distance is set by the TOLERANCES GEOMETRIC command. If NCTOLERANCE > 0, the absolute tolerance distance *in each global direction* is specified by the value of NCTOLERANCE. For example, if coincidence checking is used against an existing node and NCTOLERANCE = 1E-3, then a new node will not be generated within the cubic region (with sides of length 2E-3) centered about that existing node.

any other element group) are intended to be a single, joined mesh.

- `NCOINCIDE = NO`: Performs no coincidence checking while meshing. This is used when the entity being meshed is to remain separate from all other meshed entities (though it may be in close contact with other meshed parts).

Command input 2.60 demonstrates `NCOINCIDE = GROUP` to join two meshes of the same element group, but not meshes of different element groups.

Command Input 2.60: Example demonstrating coincidence checking by element group (`NCOINCIDE = GROUP`). Three seperate bodies are created and meshed using two element groups. The two meshes of element group 1 (shown in green) are joined, but the mesh for element group 2 (red) is not joined, as shown in Figure 2.60.

```
feprogram program=adina
body block name=1,
     cx1=0.0 cx2=0.0 cx3=0.0 dx1=1.0 dx2=1.0 dx3=1.0
body block name=2,
     cx1=0.0 cx2=0.0 cx3=-1.0 dx1=1.0 dx2=1.0 dx3=1.0
body block name=3,
     cx1=0.0 cx2=0.0 cx3=-2.0 dx1=1.0 dx2=1.0 dx3=1.0
*
subdivide body name=1 mode=length size=0.1
subdivide body name=2 mode=length size=0.1
subdivide body name=3 mode=length size=0.1
*
egroup threedsolid name=1
egroup threedsolid name=2
*
gbody 1 nodes=8 group=1 ncoincide=group
gbody 2 nodes=8 group=1 ncoincide=group
gbody 3 nodes=8 group=2 ncoincide=group
```

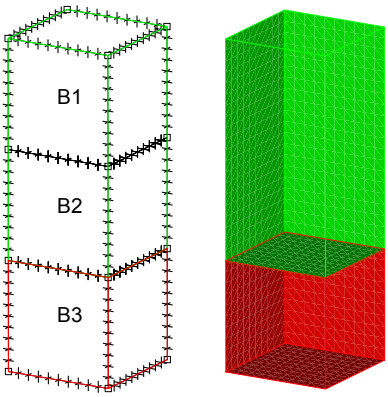


Figure 2.60: Illustration of the effects of `NCOINCIDE = GROUP`. Only meshes of the same element group are joined, as revealed by the culled rendering on the right.

2.6.2 *Joining and detaching meshes*

Command MESH-JOIN joins parts of the model by equivalencing coincident nodes. This is useful when a mesh imported via NASTRAN-ADINA or IMPORT-EXTERNAL is to be joined with another mesh. It can also be used if an error was made with NCOINCIDE during meshing, and parts that should form a congruent mesh remained separate.

The parameter SEARCH controls which nodes are included in the coincidence search. ADINA can either search over all nodes of the parts specified (SEARCH = ALL) or on the boundaries (SEARCH = BOUNDARY). The parameter ACTION controls whether the parts with coincident nodes are found and/or joined. ACTION = JOIN equivalences coincident nodes, and ACTION = FIND creates a node set of all the coincident nodes without joining the parts. ACTION = JOIN-SAVE equivalences coincident nodes *and* creates a node set of all the coincident nodes.

Command input 2.61 creates two adjacent bodies, subdivides them, and then meshes them with the setting GBODY NCOINCIDE = NO. The meshes remain separate (Figure 2.61, panel *a*).

Command input 2.62 demonstrates how to use MESH-JOIN to join separate meshes. The culled rendering in Figure 2.61 (panel *b*) shows the results.

Command Input 2.61: Creating and meshing two bodies. The setting NCOINCIDE = NO keeps the two meshes separate. Figure 2.61 (panel *a*) shows the two adjacent, but unjoined meshes.

```
feprogram program=adina
body block 1 dx1=1.0 dx2=1.0 dx3=1.0
body block 2 cx1=-1.0 dx1=1.0 dx2=1.0 dx3=1.0
*
subdivide body 1 mode=length size=0.1
subdivide body 2 mode=length size=0.1
*
egroup threedsolid 1
egroup threedsolid 2
*
gbody 1 nodes=8 group=1 ncoincide=no
gbody 2 nodes=8 group=2 ncoincide=no
```

Command Input 2.62: Continuation of command input 2.61 joining two meshed bodies. The culled rendering in Figure 2.61 (panel *b*) shows that the two meshes are now joined.

```
mesh-join gtype=body search=boundary
1
2
```

MESH-DETACH detaches parts of the model by splitting nodes shared by those parts satisfying parameter SEARCH. This is useful if an error was made with NCOINCIDE during meshing, and parts that should have remained separate were erroneously joined. Command input 2.63 detaches the joined meshes. The culled rendering in Figure 2.61 (panel *c*) shows the two meshed bodies, no longer joined.

Command Input 2.63: Continuation of command input 2.62 detaching two meshed bodies. Setting SEARCH = ALL searches over all nodes. Figure 2.61 (panel *c*) shows the resulting separated meshes.

```
mesh-detach gtype=body search=all
1
2
```

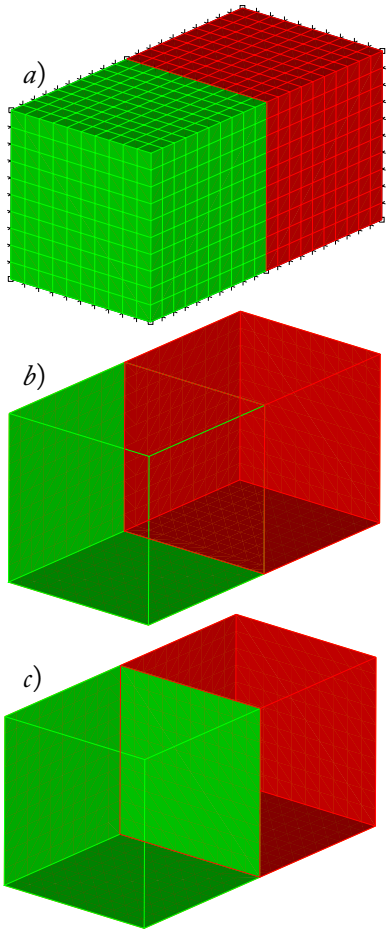


Figure 2.61: Illustration of the effects of MESH-JOIN and MESH-DETACH. Panel *a* shows two adjacent, meshed bodies. Each mesh belongs to a different element group (shown in green and red, for clarity). The two meshes are not joined. The command MESH-JOIN is then used to join the meshes (visible in the culled rendering in panel *b*). Finally (panel *c*), the command MESH-DETACH can be used to separate the two meshes again.

2.6.3 *Splitting meshes*

It may also be necessary to split a mesh at an interface after coincident nodes have been equivalenced. In these cases (as with, for example, fracture problems), the command MESH-SPLIT is useful.

Parameter BOUNDARY-SPLIT specifies how the split is treated at its boundaries (*i.e.*, the precise nodal location of the initial crack tip/front). The three options for BOUNDARY-SPLIT are illustrated, in two dimensions, in Figure 2.62.

Command inputs 2.64 and 2.65, respectively, illustrate two- and three-dimensional cases using MESH-SPLIT.

Command Input 2.64: Example showing MESH-SPLIT GTYPE = TWOD in use. Three separate surfaces are created and meshed while checking for nodal coincidence. Finally, the mesh is split along the internal horizontal line, as shown in Figure 2.63.

```
feprogram program=adina
coordinates point
1 0.0 0.0 0.0 0
2 0.0 1.0 0.0 0
3 0.0 1.0 1.0 0
4 0.0 0.0 1.0 0
5 0.0 0.5 0.5 0
6 0.0 1.0 0.5 0
7 0.0 0.5 1.0 0
8 0.0 0.5 0.0 0
*
surface vertex name=1 p1=1 p2=8 p3=7 p4=4
surface vertex name=2 p1=2 p2=6 p3=5 p4=8
surface vertex name=3 p1=6 p2=3 p3=7 p4=5
*
subdivide surface name=1 mode=length size=0.125
subdivide surface name=2 mode=length size=0.125
subdivide surface name=3 mode=length size=0.125
*
egroup twodsolid subtype=stress2
*
gsurface 1 nodes=4 ncoincide=boundary
gsurface 2 nodes=4 ncoincide=boundary
gsurface 3 nodes=4 ncoincide=boundary
*
mesh-split gtype=twod boundary-split=external
6
```

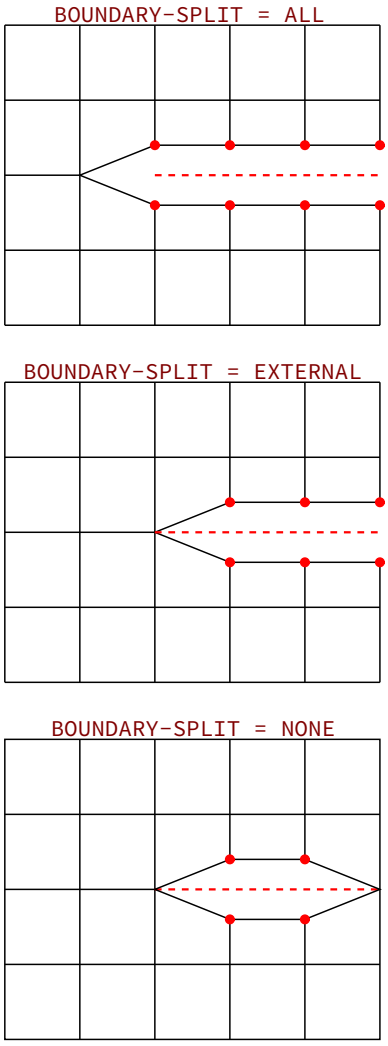


Figure 2.62: Effects of BOUNDARY-SPLIT for indicated values. The dashed red line indicates the position of the split interface. Split nodes are shown in red; gap shown for clarity.

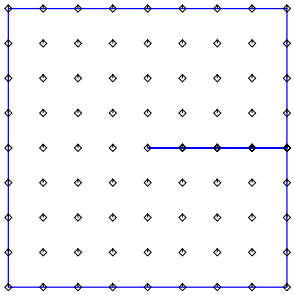


Figure 2.63: Model outline with hidden lines/edges following two dimensional MESH-SPLIT operation. Because BOUNDARY-SPLIT = EXTERNAL, the central, internal node was not split.

Command Input 2.65: An example showing MESH-SPLIT GTYPE = THREED in use. Four separate bodies are created and free-form meshed while checking for nodal coincidence. Finally, the mesh is split along the face separating bodies 3 and 4, as shown in Figure 2.64.

```
feprogram program=adina
body block name=1 option=centered position=vector,
     cx1=-0.25 cx2=0.0 cx3=-0.25 dx1=0.5 dx2=1.0 dx3=0.5
body block name=2 option=centered position=vector,
     cx1=-0.25 cx2=0.0 cx3=0.25 dx1=0.5 dx2=1.0 dx3=0.5
body block name=3 option=centered position=vector,
     cx1=0.25 cx2=0.0 cx3=-0.25 dx1=0.5 dx2=1.0 dx3=0.5
body block name=4 option=centered position=vector,
     cx1=0.25 cx2=0.0 cx3=0.25 dx1=0.5 dx2=1.0 dx3=0.5
*
subdivide body name=1 mode=length size=0.1
subdivide body name=2 mode=length size=0.1
subdivide body name=3 mode=length size=0.1
subdivide body name=4 mode=length size=0.1
*
egroup threedsolid
*
facelink option=all
*
gbody 1 nodes=4 meshing=free-form ncoincid=boundary
gbody 2 nodes=4 meshing=free-form ncoincid=boundary
gbody 3 nodes=4 meshing=free-form ncoincid=boundary
gbody 4 nodes=4 meshing=free-form ncoincid=boundary
*
mesh-split gtype=threed boundary-split=external
1 3
```

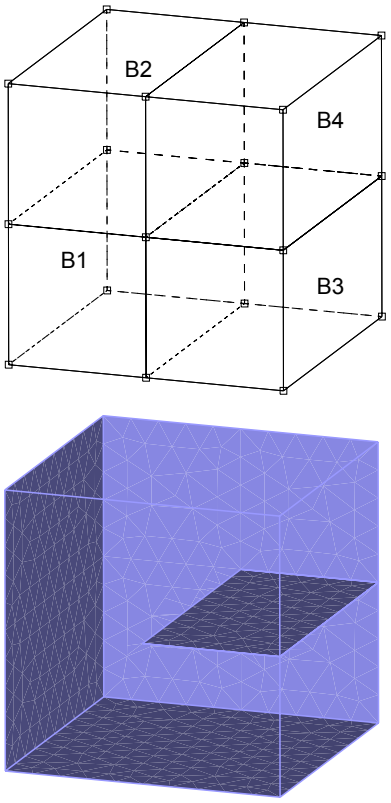


Figure 2.64: The four bodies shown (top) were subdivided and meshed using GBODY NCOINCIDE = ALL. This mesh was then split along the face separating bodies 3 and 4. The culled rendering on the bottom shows the result of the MESH-SPLIT operation.

2.6.4 *Checking for coincidence*

A simple way of visually inspecting a mesh to check for equivalence is to render the shaded mesh with front cases culled. See Face-linking on page 14 for a simple example.

ELEMENTSET OPTION = ATTACHED can be used to verify that parts of the model have not been inadvertently joined. Command input 2.66 demonstrates how to use this command to determine which parts of a mesh are joined. The command places all attached elements into a single element set, which can then be inspected.

Command Input 2.66: Meshing two bodies sharing a point and generating a set of elements attached to an element from one body (ELEMENTSET OPTION = ATTACHED). Figure 2.65 shows the element set, shaded in green.

```
feprogram program=adina
body block name=1 option=centered position=vector,
    cx1=0.0 cx2=0.0 cx3=0.0 dx1=1.0 dx2=1.0 dx3=1.0
body block name=2 option=centered position=vector,
    cx1=1.0 cx2=1.0 cx3=1.0 dx1=1.0 dx2=1.0 dx3=1.0
*
subdivide body name=1 mode=length size=0.1
subdivide body name=2 mode=length size=0.1
*
egroup threedsolid
*
gbody 1 nodes=8
gbody 2 nodes=8
*
elementset name=1 option=attached
1 1
```

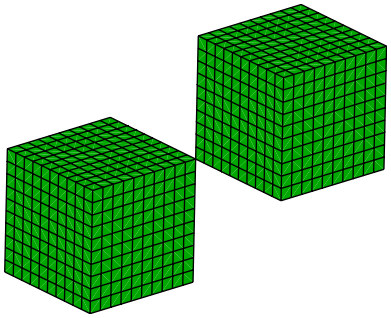


Figure 2.65: Shaded rendering of the set of all elements attached to element 1 of group 1 (in the body on the lower left). Since elements from both bodies are shown in green, it is clear that the two meshes are joined. In this case, the two meshes are connected by single equivalent node at the touching corners.

Figure 2.65 shows the set of all elements attached to element 1 of group 1 (in the body on the lower left). In this case, if the two meshed bodies were not intended to be joined, the command MESH-DETACH can be used to detach them.

Figure 2.66 illustrates how to use command ELEMENTSET OPTION = ADJACENT to determine if the mesh has been properly split. By creating a set of elements immediately adjacent to an element on the split interface, the user can check that only those elements on the same side of the selected element are included in that element set.

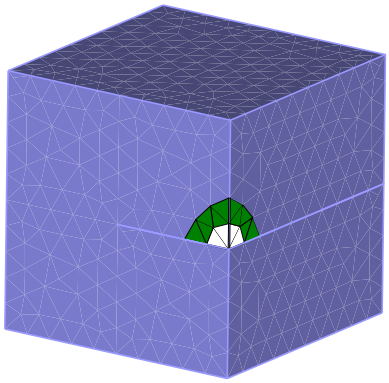


Figure 2.66: Set of elements (in green) adjacent to the elements (in white) just above the desired split interface (thick, internal horizontal line – see Figure 2.64), visible after using ELEMENTSET OPTION = ADJACENT. The elements on the other side of the interface are not detected as adjacent, which indicates that the mesh has been properly split.

2.7 Copying and Converting Meshes

It is often necessary or expedient to copy meshes in models involving repeating parts or domains. It may also be necessary to copy the triangulation from one surface to another (similar) surface, as when a cyclically symmetric part must be free-meshed and the nodes on the cylic boundaries must be coincident.

Or perhaps the user has decided that 8-node shells will not suffice and is faced with regenerating a 9-node shell meshes over several complex geometries. It would be far simpler and more efficient to convert the existing mesh, in place, rather than generating new meshes.

In all of the above cases, ADINA offers meshing tools to help the user achieve the desired result.

2.7.1 Copying meshes

The COPY-MESH-BODY command can be used to copy a mesh from one body to another (similar) body via a defined affine transformation. This transformation must be defined by a command of the TRANSFORMATION family, such as TRANSFORMATION TRANSLATION or TRANSFORMATION ROTATION.

Command input 2.67 demonstrates how COPY-MESH-BODY can be used to copy a mesh from one body to another. In this case, the bodies are adjacent and comprise a domain with repeating geometry. However, meshes may also be copied to bodies defined by more general transformations.

Command Input 2.67: COPY-MESH-BODY being used to copy a complex mesh from one body to another. The first body is meshed using GBODY options (see Figure 2.67, left panel). The mesh is then copied from the first to the second body (Figure 2.67, right panel). Note that GBODY is only used once and that COPY-MESH-BODY TRANSFORMATION = 1 references the appropriate transformation.

```
feprogram program=adina
body block dx1=1.0 dx2=1.0 dx3=0.075
body cylinder cx1=-0.15 cx2=-0.25 axis=zl,
     radius=0.125 length=.075
body cylinder cx1=0.10 cx2=0.125 axis=zl,
     radius=0.2 length=.075
*
body subtract name=1 keep-too=no keep-imp=no
2
3
*
subdivide body name=1 mode=length size=0.05
*
egroup shell name=1
egroup threedsolid name=2
*
gface name=1 nodes=9 prefshap=quad-direct body=1 group=1
*
body sweep name=3 face=1 dx=0.0 dy=0.0 dz=-.075,
     mesh=yes 3d-egroup=2 body=1
*
transformation translation name=1 dx=0.0 dy=-1.0 dz=0.0
*
body transformed name=2 parent=3 transformation=1
copy-mesh-body body1=3 body2=2 transformation=1
```

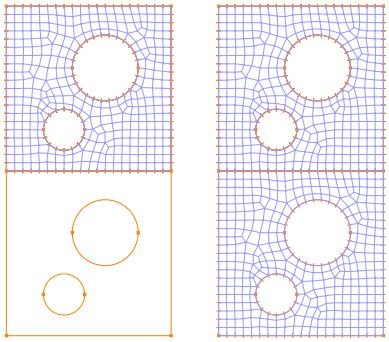


Figure 2.67: Using COPY-MESH-BODY to mesh two similar bodies using the same mesh. The same transformation is used to create the second body (left) and copy the mesh (right).

The command BODY TRANSFORMED can be used to copy a body and its mesh multiple times. This is particularly useful when a model contains multiple bodies which must be meshed and/or when a model consists of multiple repeating sections.[23]

For example, command input 2.68 illustrates the steps needed to copy a bolt's mesh multiple times. Figure 2.68 illustrates the results after defining an appropriate TRANSFORMATION and using BODY TRANSFORMED. Rather than using 8 relatively expensive GBODY operations, the original mesh is merely copied 7 times.

[23] Of course, COPY-MESH-BODY may also be used multiple times to achieve the same effect.

Command Input 2.68: BODY TRANSFORMED being used to copy a meshed bolt multiple times. The first body is meshed using GBODY; its mesh is shown in magenta. A transformation describing a 45° rotation about the x-axis is then defined. The body and its mesh are then copied 7 times using that transformation, resulting in the copied meshes (blue). Note that GBODY is only used once and that BODY TRANSFORMED TRANSFORMATION = 1 references the appropriate transformation.

```
feprogram program=adina
body cylinder cx1=0.0 cx2=0.0 cx3=2.0 radius=0.25 length=1.5
body cylinder cx1=0.75 cx2=0.0 cx3=2.0 radius=0.5 length=0.2
body cylinder cx1=-0.75 cx2=0.0 cx3=2.0 radius=0.5 length=0.2
*
body merge name=1 keep-too=no merge-im=yes
2
3
*
egroup threedsolid
*
subdivide body name=1 mode=length size=0.1
*
gbody name=1 nodes=27
*
transformation rotation name=1 mode=axis axis=xl angle=45.0
*
body transformed name=2 option=copy parent=1,
     transformation=1 ncopy=7 mesh=yes egroup=1
```

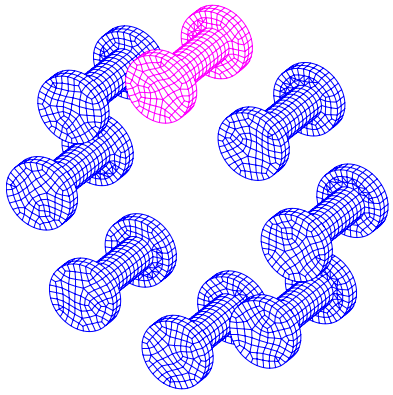


Figure 2.68: Illustration of BODY TRANSFORMED copying a bolt's mesh multiple times. In this case, the bolts are being arranged in a circular pattern (*e.g.*, a flanged joint). The original mesh is shown in magenta, and the copied meshes are shown in blue. The NCOPY = 7 option is used to specify that the body and its mesh are to be copied seven times.

The tradeoff when using BODY TRANSFORMED instead of COPY-MESH-BODY to copy meshes is that only transformations consisting of a single translation, a sequence of translations, or a single rotation may be used. COPY-MESH-BODY does not have this limitation; it can make use of all types of transformations and combinations.

2.7.2 *Copying triangulations*

The COPY-TRIANGULATION command copies face triangulations which can later be used by meshing commands like GFACE or GBODY. The enables the creation of identical free-form meshes on similar faces. This is very useful for substructuring or cyclic symmetry. Command input 2.69 demonstrates the use of COPY-TRIANGULATION.

Command Input 2.69: Copying a mesh triangulation from one cyclic boundary to another (as shown in Figure 2.69. The option TRANSFORMATION = 1 references the transformation necessary to map the triangulation using a 36° rotation (corresponding to the period of cyclic symmetry for this body) about the vertical axis.

```
subdivide body name=1 size=0.07
*
egroup shell
*
gface nodes=4 prefshape=quad-direct
12 1
*
transformation rotation axis=zl angle=36.0
*
copy-triangulation body1=1 face1=12 body2=1 face2=11,
    transformation=1
*
egroup threedsolid
*
gbody 1 nodes=8
```

Note that COPY-TRIANGULATION is not needed for Mapped Meshing; matching meshes will be generated on the cyclic boundaries, provided that identical subdivisions are used along those boundaries.

COPY-TRIANGULATION can also be used to generate a 'skin' of shell of 3D plane stress (membrane) elements from a 3D solid mesh – see Skin of elements on 3D-solid mesh.

In 3D analyses, if the faces (or surfaces) are touching, Face-linking can be used to ensure Nodal Coincidence.

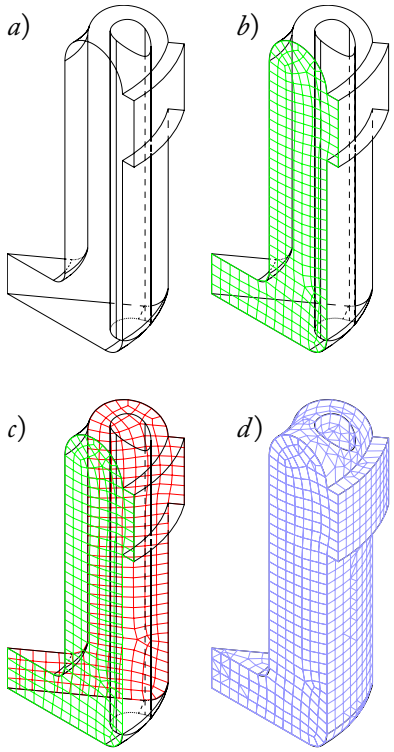


Figure 2.69: Using COPY-TRIANGULATION to copy a triangulation from one face to another. Panel *a* shows a cyclically-symmetric body which we would like to mesh; nodes on the cyclic boundaries must coincide. Once the first face is meshed (shown in green, panel *b*), COPY-TRIANGULATION is used to copy the triangulation to the appropriate face (shown in red, panel *c*). GBODY will use those triangulations as it generates the 3D solid mesh. Panel *d* shows the resulting 3D solid mesh.

2.7.3 *Converting meshes*

The MESH-CONVERT command can be used to add the center node(s) to higher-order serendipity meshes without the need to regenerate the mesh. Table 2.2 details which element types can be converted.

Note that if pyramid elements are present in a 3D solid mesh (see Mixed meshing), MESH-CONVERT converts 13-node pyramids to 14-node pyramids to match the 27-node brick elements.

Specific element groups and/or element types can be con-

Element Type	Original Element	Converted Element
2D Solid	8-node quadrilateral 6-node triangular	9-node quadrilateral 7-node triangular
Shell	8-node quadrilateral	9-node quadrilateral
3D Solid	20-node brick 10-node tetrahedral	27-node brick 11-node tetrahedral

Table 2.2: Supported element conversions.

verted using the ELEMENT-TYPE and GROUP parameters. Command input 2.70 demonstrates how an existing mixed mesh of 20- and 10-node elements can readily be converted to 27- and 11-node elements.

Command Input 2.70: Converting a higher-order mesh using MESH-CONVERT. Figure 2.70 shows the original and resulting meshes.

```
feprogram program=adina
body block name=1 option=centered position=vector,
    cx1=0.0 cx2=0.0 cx3=0.0 dx1=2.0 dx2=2.0 dx3=2.0
body cylinder name=2 option=centered position=vector,
    cx1=0.0 cx2=0.0 cx3=0.0 axis=xl radius=0.2 length=4.0
*
body subtract name=1 keep-too=no keep-imp=no
2
*
subdivide body 1 mode=length size=0.1
subdivide face 7 body=1 mode=length size=0.04
subdivide face 8 body=1 mode=length size=0.04
*
egroup threedsolid
*
gbody 1 nodes=20 meshing=free-form
*
boxzone b
-10 0 -10 10 -10 10
modeldepic geometry=no
frame/meshplot zone=b
*
mesh-convert element-type=threedsolid
```

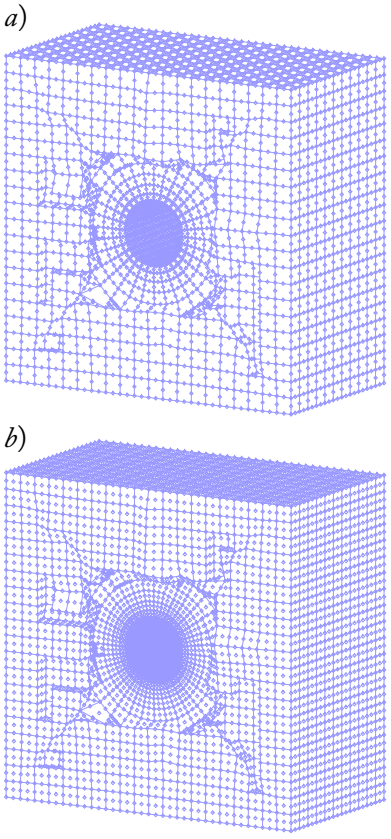


Figure 2.70: Converting a higher-order mesh using MESH-CONVERT. Panel *a* shows the mesh resulting from GBODY in command input 2.70. By using MESH-CONVERT, it is possible to add center nodes to the existing higher-order mesh without remeshing the body (panel *b*).

2.8 *Mesh Checking*

It is very useful to verify, before the start of an analysis, that a mesh is properly generated and that the elements within the mesh meet appropriate criteria for quality. ADINA helps users check meshes in a number of ways. This section describes tools for identifying meshing problems and for re-meshing problematic portions of a mesh.

2.8.1 *Fluid mesh compatibility*

Fluid meshes must be *compatible* – that is, the shared face between adjacent fluid elements must be geometrically and topologically identical. Furthermore, the nodes of the elements at the adjoining face must be coincident. Finally, fluid elements are only connected at a face; fluid elements cannot be connected only at an edge. For more information, see Face-linking on page 14, and Nodal Coincidence on page 77.

The command `CONTROL COMPATIBLE-MESH = YES` will check if a fluid mesh is compatible. If the fluid mesh is not compatible, a data (.dat) file is not generated, and an error message is output giving the element and element group labels which are not compatible.

A compatibility check will, of course, require additional time to generate a data file. To prevent long data file generation times, a *threshold number* can be input so that a compatible mesh check is run only if the number of elements in the model does not exceed this number. For example, setting `CONTROL COMPATIBLE-MESH = 1000000` (the default) will check for mesh compatibility if the model contains one million elements or fewer.

2.8.2 *Duplicate elements*

ADINA can also check for *duplicate elements* during data file generation. Duplicate elements are defined as elements of the same type sharing the same corner nodes. [24]

It is possible to have elements from different elements groups meet the criteria as duplicates. The command `CONTROL DUPLICATE = MODEL` will check for duplicates

[24] In ADINA Structures, elements are only considered duplicates if they *1*) have the same material definition, *2*) cross-section definition (if applicable), and *3*) have the same birth and death times.

across different element groups within the model. However, the user may wish to check for duplicate elements within the same element group only by using `CONTROL DUPLICATE = EGROUP`. For example, by using `CONTROL DUPLICATE = EGROUP`, elements from different element groups (of the same element type) which share the same corner nodes will not be considered duplicates.

2.8.3 *Unique element labels*

When saving results in OP2 format, it is necessary to create unique element labels using the `CONTROL ELEMENT-LABEL = UNIQUE` command.[25] Note that `CONTROL ELEMENT-LABEL` cannot be changed from `REPEAT` to `UNIQUE` after the model has already been meshed; if results are to be saved in OP2 format, `CONTROL ELEMENT-LABEL` should be set to `UNIQUE` before the model is meshed.

[25] OP2 format does not support element group labels.

2.8.4 *Mesh quality checks and re-meshing*

ADINA supports mesh quality checks via the `MESH-QUALITY` command. This command can be used to identify poor quality elements and generate mesh quality reports displaying element quality distribution plots. Mesh quality checks can be performed to identify elements which fail to meet mesh quality criteria, or *metrics*.

By default, `MESH-QUALITY` will automatically use appropriate metrics for the current model type.[26] For example, if `FEPROGRAM PROGRAM = ADINA`, the program will search for elements with negative Jacobian ratios. If `FEPROGRAM PROGRAM = ADINA-F`, the program will search for elements with minimum face angles of less than thirty degrees.[27]

By default, `MESH-QUALITY` will check the quality of all elements attached to the entity specified by `MESH-QUALITY TYPE`. Some possibilities are

- `TYPE = BODY`: mesh quality is checked for elements attached to the specified bodies.
- `TYPE = EGROUP`: mesh quality is checked for elements in the specified element groups.
- `TYPE = ELSET`: mesh quality is checked for elements in

[26] By default, the mesh quality criteria will identify elements that will fail in the solver. However, *even if a mesh passes the default quality criteria*, it is not a guarantee that all elements are of good quality. Sliver elements, for example, may become overdistorted in large deformation anaylsis as the solution advances.

[27] However, the user may select any combination of element quality criteria by using the `MESH-QUALITY METRIC = MANUAL`.

the specified element sets.

- `TYPE = MODEL`: mesh quality is checked for all elements in the model.

Users may specify their own mesh quality criteria, which should depend upon the specific model – that is, the element type and the analysis type. For example, a rubber model expected to undergo large deformations may need to satisfy stricter criteria than a model which undergoes only small deformations.

The Jacobian ratio for constant strain 3-node triangular and 4-node tetrahedral elements is always equal to 1. This is because the derivatives of strain (the terms in the Jacobian matrix) are constant, regardless of how distorted the element becomes. As a result, the Jacobian ratio criterion is not useful for tri-3 and tet-4 elements. The minimum and maximum corner angle are better mesh quality criteria for tri-3 and tet-4 elements. The minimum and maximum corner angle criteria is also useful for identifying sliver elements.

As an example of how the maximum and minimum corner angle criteria can be more useful in practical problems than the Jacobian ratio criterion, we can consider Primer Problem 61: *Analysis of a gasketed assemblage*. After importing the `.nas` file containing the mesh shown in Figure 2.71, we can run the mesh quality check with its default criterion for structural problems, as shown in command input 2.71.

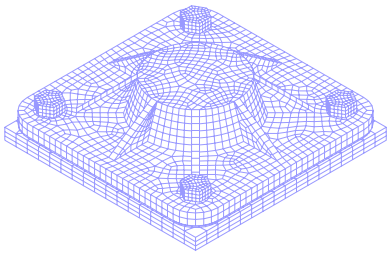


Figure 2.71: Original mesh from Primer Problem 61: *Analysis of a gasketed assemblage*.

Command Input 2.71: `MESH-QUALITY` being used, with default structures settings, to check the mesh from Primer Problem 61.

```
feprogram adina
*
nastran-adin filename='prob61.nas'
*
mesh-quality
```

All the elements pass the Jacobian ratio criterion. Now, we check the mesh using `CANGMAX = YES` and `CANGMIN = YES`. Command input 2.72 demonstrates how to use the maximum and minimum corner angle criteria to search for poor elements.

Command Input 2.72: Continuation of command input 2.71 using manually defined criteria `CANGMAX = YES` and `CANGMIN = YES`. The output can be seen in Figure 2.72.

```
mesh-quality metric=manual,
             cangmax=yes cangmax-thld=179.9,
             cangmin=yes cangmin-thld=1
```

Figure 2.72: Poor elements detected when using the maximum and minimum corner angle criteria. Note the flattened pyramid elements and sliver tetrahedral elements.

Figure 2.72 shows that the program detects several poor pyramid elements and sliver tegrahedral elements. These elements were not detected using the default Jacobian ratio criterion and would likely cause the program convergence difficulties in large strain analysis.

The MESH-QUALITY SAVETO parameter can be used to save all elements which fail the mesh quality metrics into a zone or an element set. The user may then easily view the failed elements and then proceed to re-mesh the elements using the GELEM command.

Command input 2.73 demonstrates how to use MESH-QUALITY to identify and display poor quality elements within a mesh.

Command Input 2.73: Checking the mesh quality of a free-formed mixed solid mesh using MESH-QUALITY. Figure 2.73 shows the elements which fail the quality check, which are stored in the "failed-elems" zone.

```
feprogram program=adina
*
body block name=1 dx1=2.0 dx2=2.0 dx3=0.7
body sphere name=2 radius=1.0
*
body merge name=2 keep-too=no merge-im=yes
1
*
egroup threedsolid
*
subdivide body name=2 mode=length size=0.2
*
gbody nodes=27
2 0
*
mesh-quality saveto=zone zone=failed-elems
*
nodedepiction name=1 symbolplot=yes
frame/meshplot zone=failed-elems nodedepiction=1
```

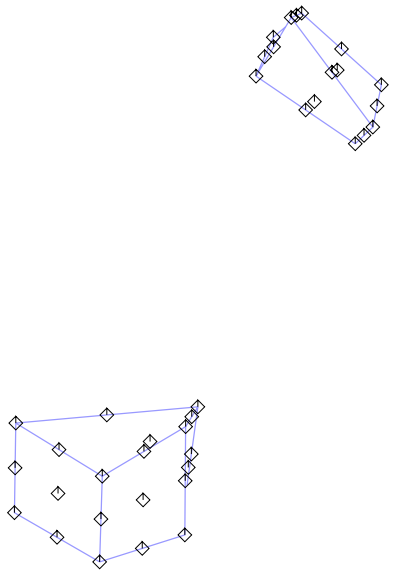


Figure 2.73: Elements which have failed a mesh quality check. In this case, two elements have failed the Jacobian metric check.

The GELEM command can be used to re-mesh elements of poor quality, as identified by MESH-QUALITY. GELEM can also be used to locally refine or coarsen an existing mesh. The original 3D elements are removed and replaced with tetrahedral and transitional pyramids, if GELEM PYRAMIDS = YES.

The GELEM TYPE parameter is used to define the entity to be re-meshed.

- TYPE = BODY: re-mesh all elements attached to the specified bodies.
- TYPE = ELEM: re-mesh the specified elements.
- TYPE = EGROUP: re-mesh all elements in the specified element groups.

- TYPE = ELSET: re-mesh all elements in the specified element sets.
- TYPE = ZONE: re-mesh all elements in the specified zones.

Command input 2.74 demonstrates how to use GELEM to remesh the poor elements identified by MESH-QUALITY which reside in the "failed-elem" zone.

Command Input 2.74: Continuation of command input 2.73 using GELEM to re-mesh the poor elements shown in Figure 2.73 and stored in the "failed-elem" zone. A subsequent mesh quality check confirms that all elements in the new mesh pass the criterion.

```
gelem type=zone zone=failed-elems
*
mesh-quality saveto=zone zone=failed-elems
```

3 Moving Mesh in ADINA CFD/FSI

ADINA offers state-of-the-art moving mesh capabilities for a range of CFD and Fluid-Structure Interaction (FSI) simulations. This chapter offers a best-practice approach and guide to setting up a class of simulations requiring a moving mesh.

3.1 Overview

In ADINA CFD and FSI applications, the fluid domain may deform at moving boundaries, including moving walls with prescribed displacements or fluid-structure interfaces. Maintaining a valid moving mesh with good quality elements is important for convergence and solution accuracy.

For models that have a moving mesh, the fluid governing equations are solved using the Arbitrary Lagrangian-Eulerian (ALE) formulation, which includes the mesh velocity. The moving mesh equations are solved by a Laplacian procedure in which the location of element vertices are determined by their original position and displacement. The displacement of these element nodes are decided by the physical condition at the fluid boundary and is solved for using the Laplace equation. This chapter will discuss all aspects pertinent to simulations that require a moving mesh, including background mesh options, solver selection, domain subdivisions, and ALE mesh constraints and conditions.

3.2 Basic Procedures

In ADINA-CFD, the moving mesh equations are solved in the order of point, to line, to surface, to volume. The steps for solving the moving mesh using the Laplace equation are as follows:

1. Determine the nodal displacements on the moving mesh boundary from physical conditions (*i.e.*, a prescribed moving wall or a fluid-structure interface). Nodes on these boundaries may also be constrained by Leader-Follower Constraints (see page 103), in which the nodes can slide along but cannot leave the specified boundary, thus maintaining the boundary shape.
2. Determine the local displacement of corner nodal points of the moving mesh domains. These corner points are those that are intercepts of two lines or edges, or joint points of three surfaces or faces. These points are often constrained by Leader-Follower conditions.
3. Solve the Laplace equation on all lines of the moving mesh domain to determine the nodal displacements on these lines. The end points on these lines are the corner points which are solved for in the previous step and are used as the prescribed boundary condition in this step. These lines can be the boundaries of the moving mesh domain, or, if the fluid domain is divided into several sub-domains, these lines can also be the boundaries of the sub-domains, which comprise the internal lines of the computational domain.
4. Lastly, for three-dimensional cases, solve the Laplace equation on all volumes. With the displacements solved for on all boundaries in the previous step, this step is to determine the displacements for all internal nodes.

3.3 Defining ALE Domain Geometry

The moving mesh domain is sometimes referred to as the ALE domain. The ALE domain will be automatically created in the AUI, and, depending on the source of model geometry, can be defined in the AUI in the following two ways:

1. The model geometry is defined using AUI Native Geometry, namely points, lines, surfaces, and volumes, or can be imported as a parasolid body using ADINA-M. In this case, the points, lines (or edges), surfaces (or faces), and volumes (bodies) will be used as the ALE domain geometry automatically.
2. There is no model geometry defined in the AUI, and the model consists only of element data, imported into the AUI. In this case, the AUI will use the element data to create the ALE domain. In two-dimensional cases, surfaces are defined by element groups, lines are defined by element-edge sets, and points are defined by the intersection of two edge sets. In three-dimensional cases, volumes are defined by element groups, surfaces are defined by element-face sets, lines are defined by the intersection of two face sets, and points are defined by the intersection of three face sets, see Figure 3.1.

Please note the following:

- It is not permissible to define overlapping element-face sets when the ALE moving mesh formulation is used. The AUI will issue an error message during data file generation when this occurs.
- Edges and corners are only preserved as the mesh moves when the ALE edges and corners have been previously defined. See Figure 3.2.
- The AUI automatically creates an element-face set during data file generation containing all remaining element faces not previously defined in other element-face sets.
- Shell element groups can be defined in the NASTRAN database, and when imported into the AUI, element-face sets can be created based on these shell element groups using the `ELFACESET` parameter in the `NASTRAN-ADINA` command.

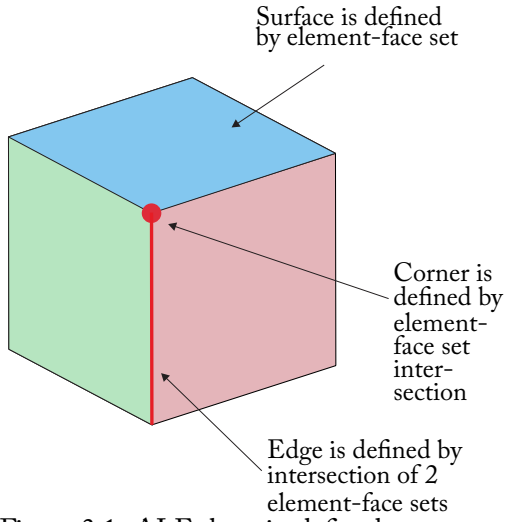


Figure 3.1: ALE domain defined by element-face sets.

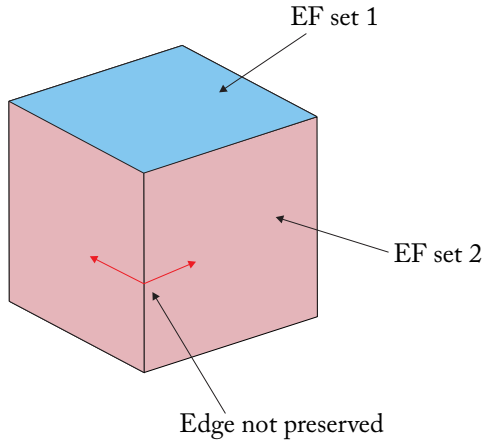


Figure 3.2: Defined element-face sets do not preserve edge. Because a corner is defined as the intersection between three face sets, the highlighted edge is not preserved. Nodes can move anywhere along element-face set 2.

3.4 *Solving the Moving Mesh*

To solve the Laplace equation for the moving mesh, there a number of options that can be chosen in ADINA.

3.4.1 *Mesh Solver*

The fluid mesh solver obtains the displacements of each element node by solving the Laplace equation. For FCBI elements, the moving mesh solver is the sparse solver. For FCBI-C elements, there are two solver options, which are selected using the `MS-SOLVER` parameter of the `OUTER-ITERATION` command:

- `MS-SOLVER = SPARSE`: Use the sparse solver. This is the default option.
- `MS-SOLVER = FD-SOLVER`: Use the same solver as the one selected for the fluid flow solution (AMG Type 1 iterative solver).

When the sparse solver is used, the mesh displacements are directly computed at the nodes. When the iterative solver (`MS-SOLVER = FD-SOLVER`) is used, the mesh displacements are considered as an additional degree of freedom at the element center, and these displacements are then interpolated to the corner nodes.

Convergence of the moving mesh equations is difficult when the iterative solver is used because the iterative solution of the (diffusive) Laplace equation is highly sensitive to any errors in the displacement solution, such as those that occur when the displacements are interpolated to the corner nodes.[1]

The drawback of using the spare solver is that it requires much more memory than the iterative solver, because the direct solver needs to store the moving mesh matrix equations. Hence, for very large problems, the AMG iterative solver with the current mesh must be used.

To prevent errors in the fluid mesh displacement solution from accumulating when the iterative solver is used, care must be taken to ensure that the mesh displacement solution has tightly converged. If this solution fails to converge when the iterative solver is used, the inner iteration equation and variable residual relaxation factors for displacement should be tightened. See command input 3.1.

[1] For most moving mesh applications, it is recommended to use the sparse solver, provided sufficient memory is available.

Command Input 3.1: Inner iteration equation and variable residual relaxation factors for displacement in moving mesh problems with the iterative solver. Note the tolerances are tightened from the default `1.0E-1`.

```
ivar-control red-eqn displacement=1.0e-2
iver control red-var displacement=1.0e-2
```

Also, the maximum number of outer iterations in the fluid variable loop should be increased to between 2 and 10 iterations, as shown in command input 3.2.

Command Input 3.2: Maximum number of iterations in fluid variable loop for moving mesh problems using the iterative solver. Note the number of iterations are increased from the default single iteration.

```
outer-teration fluid-maxit=5
```

3.4.2 *Background Mesh*

When solving the Laplace equation, the calculations must be based on a background mesh. The `MESHUPDATE` parameter of the `MASTER` command specifies the background mesh to be used. There are two options:

- `MESHUPDATE = CURRENT`: The current mesh, *i.e.,* the mesh obtained in the last converged solution step is used as the background mesh for solving the increment in displacements.
- `MESHUPDATE = ORIGINAL`: The original mesh, *i.e.,* the undeformed mesh at the start of the solution is used as the background mesh for solving the increment in displacements.

By default, `MESHUPDATE = ORIGINAL`.

The benefits of using the original mesh are:

- Moving mesh equations are only factorized once at the beginning of the analysis when they are solved using the sparse solver. If the current mesh is used, the equations have to be factorized at each time step, which is expensive, especially for large CFD meshes.
- The mesh returns back to its original state for cyclic problems (not guarenteed when the current mesh is

used).

- Conditioning of the coefficient matrix tends to be better when the original mesh is used for the background mesh.

The benefit of using the current mesh is that it is more robust for certain problems with non-convex fluid domains. This option should only be used if the original mesh option is unsuccessful.

3.4.3 *The Solving Domain*

If the solving domain is subdivided into smaller domains by lines (in 2D) or surfaces (in 3D), each of the subdomains are solved separately by the Laplacian procedure, and the moving nodes will not cross the subdomain boundaries. It is important to note that the Laplacian solution cannot guarantee a valid fluid mesh, especially when the solving domain is non-convex and the mesh undergoes compression. In Euclidean space, an object is convex if a line joining any two interior points remains in the domain. Examples of a convex and non-convex domain are given in Figure 3.3. Figure 3.4 (panel *a*) shows a case where the element inverts under compression in a non-convex domain.

From a mathematical point of view, the closer the domain is to convex, the greater chance the mesh will converge under compressive mesh movement. Hence, the best strategy for mesh control is to divide the computational domain into multiple convex-like subdomains, and then solve the Laplace equation on these subdomains. The deformed element would then be acceptable, as shown in Figure 3.4 (panel *b*).

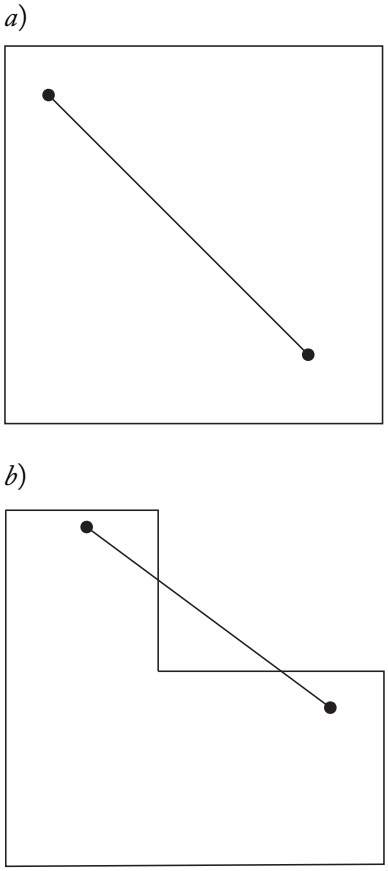


Figure 3.3: Convex (panel *a*), and non-convex (panel *b*) domains. Note in *b*), a line-segment exists outside the non-convex set.

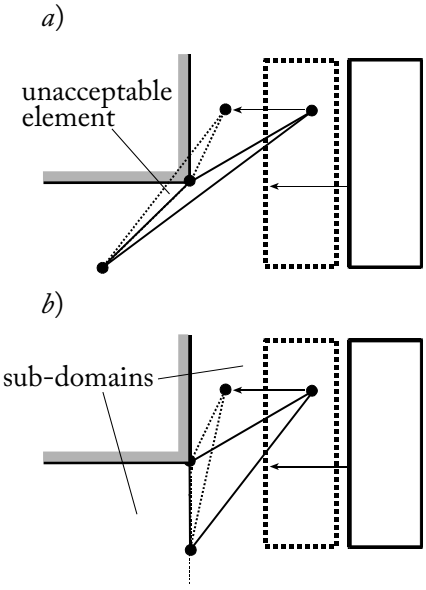


Figure 3.4: Element deformation in non-convex (panel *a*) and convex (panel *b*) fluid domains.

3.4.4 *Choice of Background Mesh and Subdomains*

An illustrative example showing different moving mesh results under a combination of the options discussed previously is given in the following section. Consider the non-convex domain in Figure 3.5; it is compressed by a right-side moving boundary, which may be a moving wall or an FSI interface. The figure shows the undeformed mesh, and the whole fluid domain (non-convex) is the ALE solving domain.

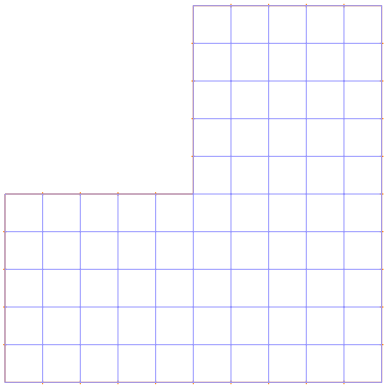


Figure 3.5: Undeformed, non-convex mesh. The right side wall is a moving boundary, prescribed to translate in the left direction by 90% of the top side length.

The right-side wall is a moving boundary, prescribed to translate in the left direction by 90% of the top side length. The background mesh is `MESHUPDATE = ORIGINAL`.

Figure 3.6 (panel *a*) shows element overlap when the boundary moves 80% of its displacement, demonstrating that the Laplacian equation of a non-convex domain may not result in a valid mesh.

A valid mesh results when the background mesh option is switched to `MESHUPDATE = CURRENT`, as shown in Figure 3.6 (panel *b*).

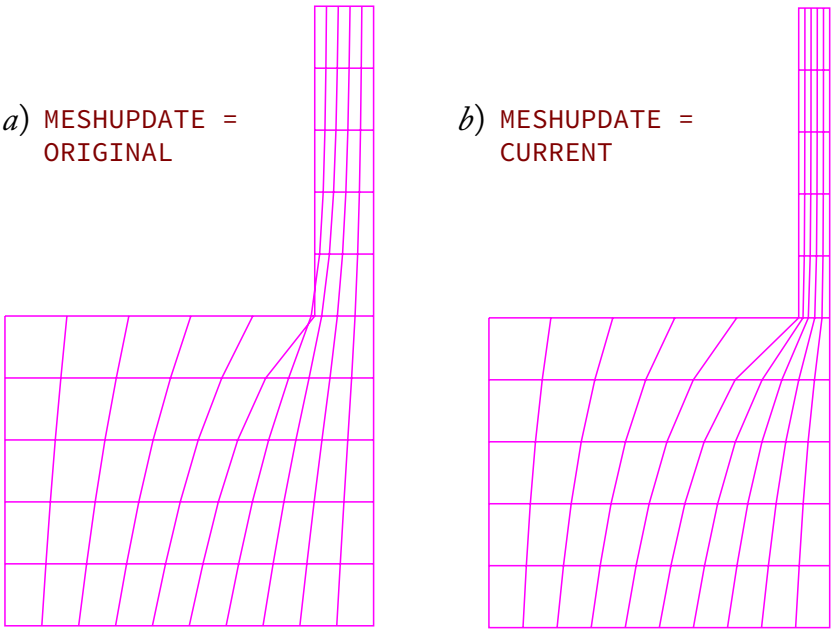


Figure 3.6: Moving mesh results using original (panel *a*) and current (panel *b*) background meshes. Using the current mesh with this non-convex domain results in a valid mesh.

Using `MESHUPDATE = CURRENT` when dealing with non-convex domains may solve certain problems that `MESHUPDATE = ORIGINAL` cannot. However, if the motion is periodic,

using a current mesh cannot guarantee a cyclic mesh. (A cyclic mesh is one that returns to its original topology under a periodic motion). As a second example, the same model is given a periodic boundary motion, and the right wall displacement is set to be 80% of the top side length, allowing a valid mesh for both background mesh options. An overlay of mesh results are plotted in Figure 3.7.

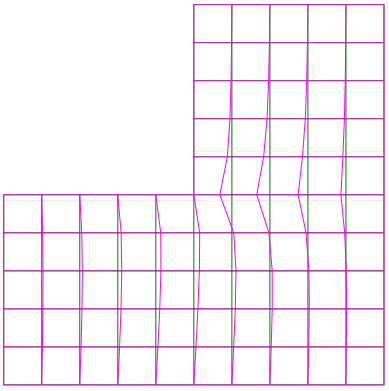


Figure 3.7: Overlay of cyclic Original (gray) and Current (magenta) meshes. Using the current mesh does not guarantee a cyclic mesh when a periodic mesh motion is imposed.

In a third example, the fluid domain is divided into two convex domains, as shown in Figure 3.8 (panel *a*). Here, the Laplacian equation is solved within each subdomain based on their respective boundaries. Allowing the moving distance of the wall to be more than 90% of the top side length, and by using the `MESHUPDATE = ORIGINAL`, the mesh is perfectly cyclic after several periods of motion. The results are given in Figure 3.8 (panel *b*).

In summary, the first choice for a background mesh should be the (default) original mesh. The fluid domain should also be divided into multiple convex-like subdomiains for which the Laplace equation is more likely to yield valid meshes. The `MESHUPDATE = CURRENT` option may be used to solve difficult non-convex domain problems for which the original mesh is unsuccessful.

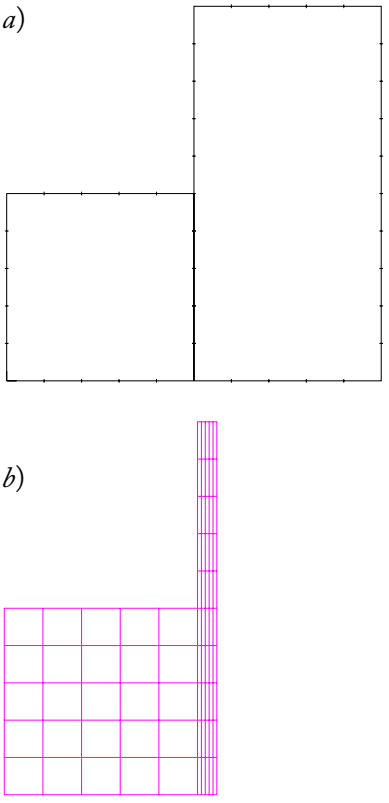


Figure 3.8: Moving mesh results using convex sub-domains and original background mesh. The geometry is first subdivided into convex subdomains shown in panel *a*). Using the original mesh as the background mesh recovers the mesh shown in panel *b*).

3.5 *ALE Conditions*

The Laplace procedure for arbitrary moving meshes typically performs well for small dispalcement problems. However, the Laplace procedure alone may not adequately control mesh quality in large displacement problems. ADINA provides additional ALE conditions to enhance moving mesh quality for problems involving large displacements.

3.5.1 *Leader-Follower Constraints*

Consider the example given in Figure 3.9. Here, the center block is moving circularly. The ALE domain is subdivided into eight convex subdomains. Running first with `MESHUPDATE = ORIGINAL`, the mesh fails (Figure 3.10, panel *a*). Switching to `MESHUPDATE = CURRENT`, the simulation successfully completes the first cycle, but the mesh crashes in its second cycle. As shown in Figure 3.10 (panel *b*), the mesh quality becomes progressively poor. This example shows that, even with all subdomains being convex, the mesh may still fail if the displacement is large. Recall that in the Laplacian procedure, the interior mesh is solved after the boundaries are determined. These boundaries can be controlled by Leader-Follower constraints to improve the mesh outcome.

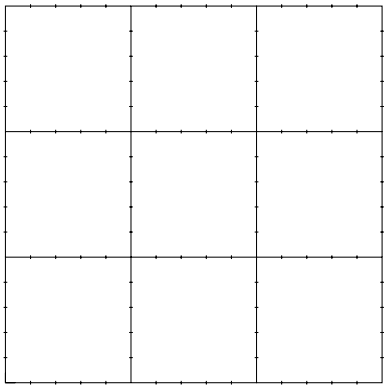


Figure 3.9: Circularly orbiting block with convex subdomains.

a) `MESHUPDATE = ORIGINAL` *b)* `MESHUPDATE = CURRENT`

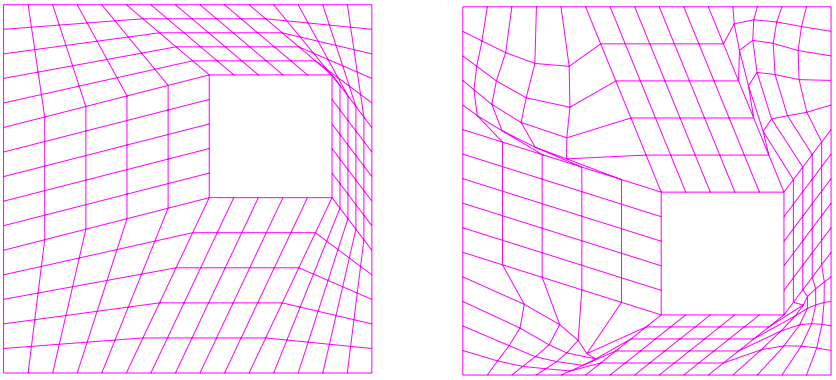


Figure 3.10: Original (panel *a*) and Current (panel *b*) moving mesh results. Both background meshes eventually fail, despite having the ALE domain divided into convex subdomains.

Consider a moving ball as shown in Figure 3.11 (panel *a*). Because there are no moving conditions on the outer boundary of the domain, the element becomes highly distorted as the ball translates to the right. Here, point 2 is fixed, and the line 1-2 becomes largely skewed. If we force

point 2 to follow point 1, then the line 1-2 will not skew, and the element will retain its quality, as shown in panel *b*). In this case, point 1 is the deemed the leader, and point 2 the follower.

In the previous circularly moving block example, we can set Leader-Follower pairs to maintain the mesh quality as the block displaces. Figure 3.12 (panel *a*) shows the eight Leader-Follower pairs. Note that each leader point, as denoted by "L", is situated on a moving wall. Note also that there are two follower points per leader. It is permitted to have more than one follower per leader, but not more than one leader per follower. Additionally, a point assigned as a follower cannot also be assigned as a leader. In this example, the original mesh is selected as the background mesh. Figure 3.12 (panel *b*) shows the resulting mesh.

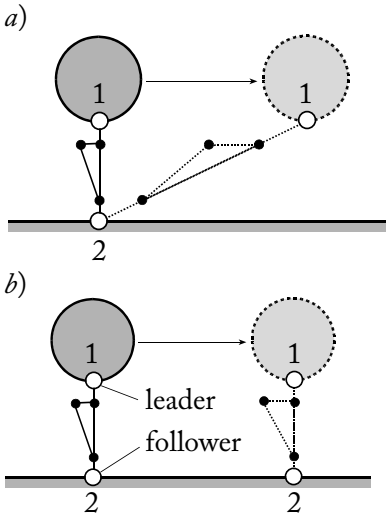


Figure 3.11: Translating ball with Leader-Follower constraints.

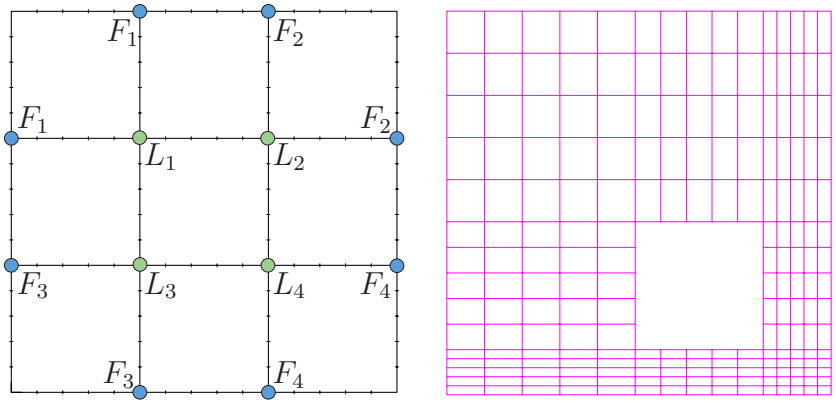


Figure 3.12: Circularly moving block with assigned Leader-Follower pairs. Note there are two followers assigned for each leader, as shown in panel *a*). The original mesh is selected as the background mesh, with the results presented in panel *b*).

3.5.2 *Types of Leader-Followers*

The Leader-Follower pair is assigned to the two end points a moving mesh boundary line. The leader must be located on a physically moving boundary (*i.e.*, a moving wall, a fluid-structure interface, or a free surface). The follower can be on the boundary of the fluid domain, or it can be an internal point. The displacement of the moving boundary determines the displacement of the leader. The displacement of the leader, in turn, determines the displacement of the follower. There are three options when deciding the type of Leader-Follower pair, and these are application-dependent. Each type is an option in the `LEADER-FOLLOWER` command.

- `TYPE = PARALLEL`: The follower first displaces with the leader, and then the follower is projected back onto the wall along the wall normal, as illustrated in Figure 3.13. If the follower is not on a wall boundary, FSI boundary, or slipping boundary, its motion is completely determined by its leader, and the Leader-Follower type can only be `TYPE = PARALLEL`. This is the default setting.
- `TYPE = CLOSEST`: The follower will move to the locally closest location to the leader on the wall or slipping boundary, as a illustrated in Figure 3.14. If there are multiple closest points, the program picks the first one as the follower's location point.
- `TYPE = CONE`: The follower is located on a wall or slipping boundary at the locally closest location to the leader within a conical search domain. The leader defines the apex of the conical domain, and the cone axis is normal to the leader boundary. The cone angle is a user-specified variable. If a zero cone angle is specified, the Leader-Follower vector is normal to the leader boundary.

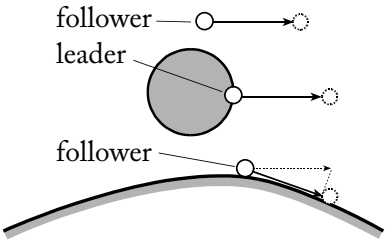


Figure 3.13: Parallel type Leader-Follower constraint. The follower first displaces with the leader and then is projected back onto the wall.

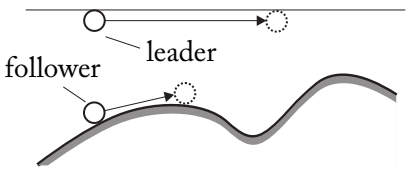


Figure 3.14: Closest type Leader-Follower constraint.

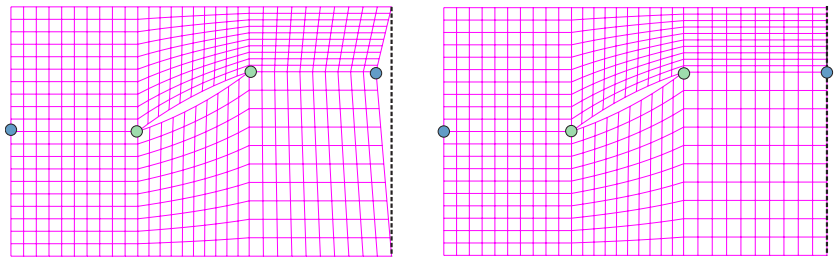
3.5.3 *Slipping Boundary*

If the follower point is not on a wall or FSI boundary, its motion is unconstrained, and only `TYPE = PARALLEL` can be used. A slipping boundary can be used to constrain the motion of the follower so that it may only move along that boundary. Consider the example given in Figure 3.15. An oscillating airfoil inside a channel is studied.

To preserve the mesh quality, leader points are assigned at the left and right tips of the airfoil, and the respective followers are assigned at the left boundary (inlet) and right boundary (outlet). Because there is no wall condition at the outlet of the domain, the follower will move according to its leader, as shown in Figure 3.16 (panel *a*). Note how the outlet boundary curves inward.

To maintain the outlet boundary shape, a slipping boundary can be defined along this line. In this case, nodes can only move along its original shape (a straight line), thus preserving the geometry. Figure 3.16 (panel *b*) shows the results with a defined slipping boundary on the outlet.

a) *b*)



For 2D cases, slipping boundaries are lines/edges or element edge sets. In 3D, slipping boundaries are surfaces/-faces or element face sets.

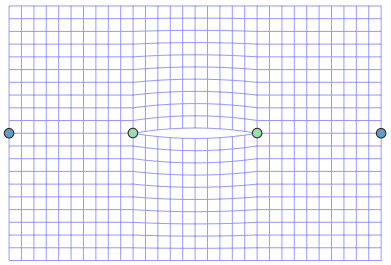


Figure 3.15: Airfoil with rule-based, undeformed mesh. The leader points are shown in green, and their respective followers are shown in blue.

Figure 3.16: Moving mesh results of airfoil without (panel *a*) and with (panel *b*) a slipping boundary. With a slipping boundary defined, nodes along the outlet are constrained to move only along the outlet boundary, thus preserving the shape of that edge.

3.5.4 *Extended Wall*

In some cases, the flow domain will expand due to a moving boundary. If this expansion has to follow a certain geometric shape, the Extended Wall may be needed. Figure 3.17 gives a sample problem. Here, the FSI boundary (highlighted in red) translates, following the extended wall. The subdivided wall is shown in orange.

When defining an extended wall, it is important to remember to discretize the line by subdividing it. Otherwise, the curved extended wall will act as a straight line between the starting and ending points. Figure 3.18 shows the moving mesh after defining the Extended Wall and setting the appropriate subdivisions.

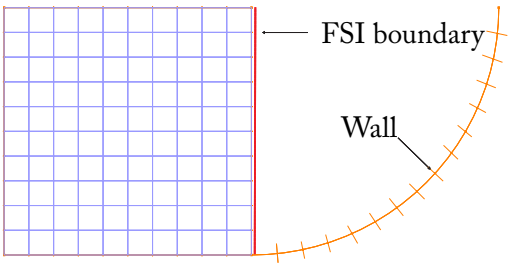


Figure 3.17: FSI extended wall problem. Here, the FSI boundary (shown in red) translates following the extended wall (shown in orange).

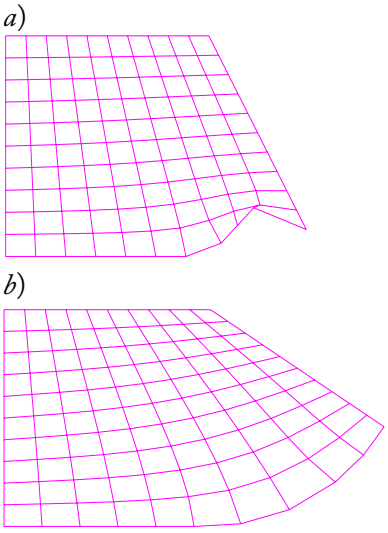


Figure 3.18: Moving mesh results for extended wall FSI problem. An invalid mesh results without line subdivision on the extended wall, as shown in panel *a*). The corner node of the undeformed mesh follows the subdivided line, as shown in panel *b*).

4 Fast Graphics Mode

Both Linux and Windows versions of the AUI support *Fast Graphics Mode* (FGM) visualizations.

FGM takes advantage of powerful graphics cards, resulting in high performance graphical rendering and manipulation. FGM also offers a wide variety of visualization enhancements. In addition, user interactions within FGM are more intuitive and natural.

Of course, the *standard mode* visualization style is also available and remains unchanged.

Vector graphics snapshots are not available in FGM.

4.1 Hardware Requirements

Fast Graphics Mode requires a medium range graphic card. The minimum requirements are:

- OpenGL 3.3 compatible graphic card
- 1 GB on board video memory
- 64-bit Linux or Windows (Vista or later) operating system

The following graphics cards have been tested:

- NVIDIA Quadro NVS 295
- NVIDIA Quadro 410
- NVIDIA Quadro 600
- NVIDIA Quadro K2000
- AMD FirePro V4900

Intel graphics cards are not compatible with FGM.

For best performance, upgrading to the latest vendor-supplied graphics card drivers is highly recommended.

4.2 *Activating FGM*

To run FGM, the Graphics System must be set to OpenGL (use Edit → Graphics System to select OpenGL).

When FGM is available, the Fast Graphics icon is ungrayed. Clicking this icon enables the FGM. To disable FGM and return to standard mode, click the Fast Graphics icon again.

4.3 *General FGM Settings*

As with standard mode graphics, FGM allows viewport customization. Users may adjust these settings to suit their preferences.

4.3.1 *Projections*

FGM can display either an orthographic (parallel) or a perspective projection. To select a projection, right-click in the graphics window, then choose Fast Graphics → View → Parallel or View → Perspective, as desired.

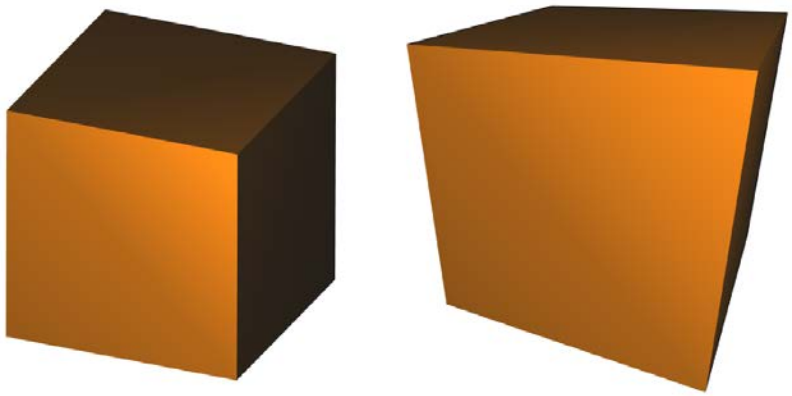


Figure 4.1: Parallel (left) and Perspective projection (right) of a body.

In *Parallel* mode, the model is displayed with all points projected along lines parallel to their positions on the

screen. In Parallel mode, three-dimensional models appear to lack depth.

In *Perspective* mode, the model is displayed in a more natural fashion. Objects further from the camera appear smaller than those nearer.

4.3.2 *Coordinate axes*

FGM's coordinate axes have been enhanced. The new axes are located at the bottom left corner of the graphics window.

To toggle the visualization of the axes:

1. Right-click in the graphics window and choose Fast Graphics → Configuration.
2. Select the Viewport tab.
3. In the Viewport box, check/uncheck the Show Axes option.

4.3.3 *Scene bounding box*

The scene bounding box option draws a box around the entire model. This box is useful to show the limits of the model. The scene bounding box is used by a number of features to let the user activate or define a working plane.

To toggle the scene bounding box:

1. Right-click in the graphics window and choose Fast Graphics → Configuration.
2. Select the Viewport tab.
3. In the Viewport box, check/uncheck the Scene BBox option.

4.3.4 *Background*

To customize the background:

1. Right-click in the graphics window and choose Fast Graphics → Background.

2. Select a background type (see below)

Solid: Works in the same way as the background in the standard mode. Select the background color using the Top field.

Ramp: Creates a vertical gradation between the Top color and the Bottom color.

File: Allows an existing image file to be used. Use the File field to choose the image file. The image file must be in the .tga file format.

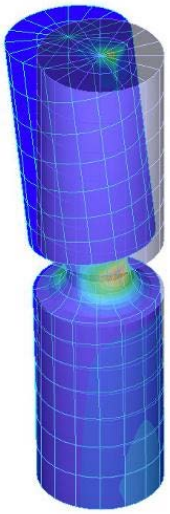


Figure 4.2: Original and deformed mesh renderings.

4.4 Scene Rendering

Fast Graphics Mode offers enhanced flexibility over standard mode when rendering scenes. Thus, the user can more easily explore the model during pre- and post-processing.

4.4.1 Original and deformed meshes

When plotted together in standard mode, the deformed mesh obscures the original mesh.

In FGM, the Translucency feature (see page 133) can be used to control the level of transparency of the original mesh. Note that although both meshes are displayed, selections can only be made in the deformed mesh.

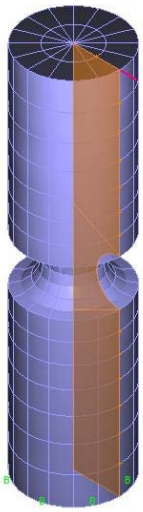


Figure 4.3: Geometry and original mesh.

4.4.2 Geometry and original meshes

When plotted together in standard mode, the original mesh always obscures the geometry.

In FGM, the Translucency feature can be used to control the level of transparency of the geometry, see above. However, selections can only be made on the mesh, not on the geometry.

4.4.3 Pattern lines

FGM introduces a new type of representation named Pattern Lines. These lines use a texture when drawing thick

lines.

The types of geometries represented with Pattern Lines are:

- Contact surfaces.
- Fluid structure boundaries.
- Gluemesh surfaces.

To change the current pattern:

1. Right-click in the graphics window and choose Fast Graphics → Configuration.
2. Select the Pattern Lines tab.
3. Select the desired pattern.
4. Check the Enabled field for each channel for which the color should be customized. Other channels use the base color.
5. Select a color to apply to each channel using the corresponding button.
6. Select the transparency level by changing its Alpha value. A value of 0 disables the channel.

The colors in the patterns can be changed by changing the colors associated with the channels. By default, channel 0 is red, channel 1 is green and channel 2 is blue.

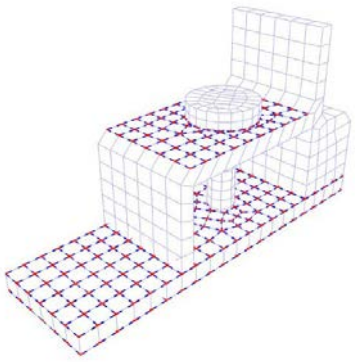


Figure 4.4: Representation of contact surfaces using pattern lines.

4.4.4 *Labels*

Entity Labels are objects represented as a text that maintain an orientation and size on the screen. Examples of Entity Labels are point labels, node labels, etc.

In standard mode, a large number of objects is both computationally expensive and can obscure the plot. FGM solves both problems by using adaptive occlusion. Labels in FGM are grouped in *node boxes*.[1] Each node box is a bounding box that holds a set of labels that are within the node box boundary. The size of each node box is calculated based on the distance between labels and number of labels per node box.

[1] *Node boxes* are not to be confused with finite element *nodes*.

To change the current Label visualization:

1. Right-click in the graphics window and choose Fast Graphics → Configuration.
2. Select the Labels tab.
3. change the desired values (see below).

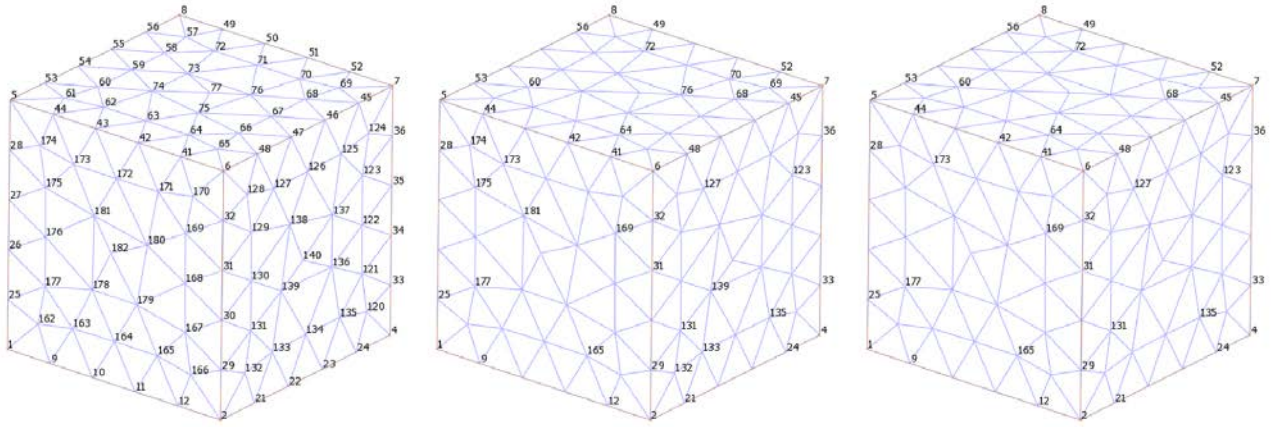


Figure 4.5: From left to right: absolute threshold values 0, 5, and 10.

Threshold: Controls the maximum number of labels that can be inside a node box. A high number of labels per node box will increment the node box size and reduce the total number of labels displayed. Threshold values can be absolute or percentage.

Distance: Controls if a node box is active or not based on the distance to the camera. Node boxes out of range are represented by a single label.

Density: When a node box is active (within distance

range), Density controls how many labels are displayed.

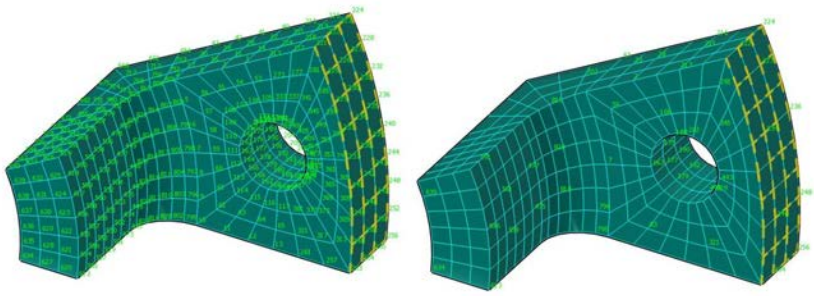


Figure 4.6: Left: Density 0, Distance 0; Right: Density 1, Distance 1.

Setting Distance and Density to zero will display all labels. Selected Labels are always visualized.

4.5 Navigation Tools

Fast Graphics Mode offers its own navigation interface and associated tools for visually exploring the model. Users can select to use FGM's navigation interface and tools by:

1. Right-clicking in the graphics window and choose Fast Graphics → Configuration.
2. Selecting the Shortcut Keys tab.
3. Unchecking the Standard box.

4.5.1 FGM navigation interface

FGM's navigation interface helps users interact with the scene more intuitively. It has been optimized for fast response. As a result, the display quality will be affected during navigation. All navigation operations applied inside the FGM view will not modify the current view configuration in standard mode. Because of this, navigation operations do not support the standard Undo/Redo command.

4.5.2 Hot navigation tool

FGM's navigation interface uses hot navigation keys buttons to activate interface tools. The advantage of using these keys is that the view can be manipulated at any time without leaving the current tool/mode. To deactivate a tool, simply release the key.

Table 4.1: Hot navigation tool keys

F2	Zoom View Tool
F3	Pan View Tool
F4	Orbit View Tool
I	Zoom Region Tool
J	Unzoom All Oneshot
Q	Zoom Selection Oneshot

4.5.3 Orbit view tool

The orbit view tool rotates the current view about a fixed pivot point. To activate the orbit view tool, click the Dynamic Rotate (XY) icon or press and hold F4.

After the orbit view tool is activated, the Navigation Trackball is displayed on the screen. The mouse cursor will change to show the function of the action available for the trackball area where the cursor is. The trackball interface is divided into four action zones:

a) Inside the circle, any drag action over this zone will act like the standard Dynamic Rotate (XY) icon.

b) Outside the circle, the drag actions over this zone will act like the standard Dynamic Rotate (Z) icon.

c) On vertical lines, the rotation will be limited to the Y axis of the screen only.

d) On horizontal lines, the rotation will be limited to the X axis of the screen only.

To rotate the view:

1. Select the desired rotation type by positioning the cursor over one of the four action zones.
2. Press the left mouse button and while it is pressed, drag the cursor in the desired direction of rotation.
3. Release the left mouse button.

The orbit tool uses the Navigation pivot point (see page 119) as the center of rotation.

It is possible to rotate the model about the model's X, Y, or Z axes by holding down left Ctrl while dragging the trackball.

a) Click on a vertical line to rotate about the model's X axis

b) Click on the circle to rotate about the model's Y axis.

c) Click on a horizontal line to rotate about the model's Z axis

To cancel the current rotation action, right-click without releasing the left mouse button.

The rotation increment can be adjusted using the Angle snap mode (see page 127).

To exit the Orbit tool when using the Dynamic Rotation (XY) icon, press Esc, or single click on the screen, or click an icon such as the Pick, Pan, Dynamic Rotate (Z) or Dynamic Resize.

The Orbit Tool supports Double click and go (see page 119).

4.5.4 *Camera spin mode*

In Camera Spin Mode, the camera rotates around the model.

To activate Camera Spin Mode, release the left mouse button while moving the mouse. The faster the mouse is moving, the faster the spin. To exit Camera Spin Mode, press the left mouse button or exit the Orbit View Tool.

4.5.5 *Pan view tool*

To pan the view in the XY plane of the screen:

1. Click the Dynamic Pan icon or press and hold F3.
2. Press the left mouse button and while it is pressed, drag the cursor in the desired direction.
3. Release the left mouse button.

It is possible to cancel the current Pan action by right-clicking without release the left mouse button.

Moving the view using the Pan View tool changes the Navigation pivot point.

The displacement factor is proportional to the distance of the camera from the navigation pivot point. The Pan View Tool might not work correctly when the camera is very close to the navigation pivot point. In this case, click the Unzoom All icon to restore the view.

To exit the Pan View Tool while using the Dynamic Pan icon, press Esc, or single click on the screen, or click an icon such as the Pick, Dynamic Rotate (XY), Dynamic Rotate (Z), or Dynamic Resize.

4.5.6 *Zoom view tool*

The Zoom View tool zooms into and out of the view. Its behavior is similar to the standard Dynamic Resize.

To zoom the view:

1. Click the Dynamic Resize icon, or press and hold F2.
2. Press the left mouse button and while it is pressed, drag

the cursor up and down until the desired zoom level is reached.

3. Release the left mouse button.

It is also possible to zoom by using the mouse wheel.

The zoom factor is proportional to the distance of the camera from the Navigation pivot point. If the camera is very close to the navigation pivot point, the Zoom View Tool might not work correctly. In that case, click the Unzoom All icon to restore the view.

It is possible to cancel the current zoom action by right-clicking without releasing the left mouse button (however not when zooming using the mouse wheel).

To exit the Zoom View Tool when using the Dynamic Resize icon, press Esc, or single click on the screen, or click an icon such as the Pick, Dynamic Rotate (XY), Dynamic Rotate (Z), or Pan.

4.5.7 *Zoom region tool*

The Zoom Region tool zooms into a region of the screen. The behavior is similar to the standard Zoom icon.

To zoom into a region of the screen:

1. Click the Zoom icon or press and release I.
2. Left-click and drag the Zoom region rectangle as desired.
3. Release the left mouse button.

To exit the Zoom View tool, press Esc, or click an icon such as the Pick, Dynamic Rotate (XY), Dynamic Rotate (Z) or Pan icons.

The Zoom Region Tool will change the Navigation pivot point. Because the Zoom Region Tool changes the navigation pivot point, perspective projections might become imprecise. In that case, use the Pan View tool and the Zoom View tool instead.

4.5.8 *Unzoom all oneshot*

This option centers all visible objects in the view by calculating the best zoom factor.

To Unzoom All, click the Unzoom All icon or press and release 9. This action will move the navigation pivot point to the center of the scene.

4.5.9 *Zoom selection oneshot*

Apply a Zoom All action to the selected area of the model (see Selection Tools on page 123 for instructions about selecting regions of the model). This tool is useful for navigating to a selected area within the model.

To Zoom Select, press and release 0. This action will change the Navigation pivot point.

4.5.10 *Double click and go*

Double-clicking in the graphics window moves the clicked point to the center of the screen. This also changes the Navigation pivot point. If the double click was made on a three-dimensional model or Scene bounding box, the three coordinates of the navigation pivot point will be updated – otherwise only the two coordinates corresponding to the screen's XY system will be modified.

4.5.11 *Navigation pivot point*

Usually, the navigation pivot point is positioned at the center of the screen and does not change after a navigation command. The Orbit view tool uses this point as center of rotation. The Pan view tool and Zoom view tool use the navigation pivot point to calculate the factor of transformation to apply to the action. This factor varies depending on the distance between the Camera and the navigation pivot point. Small distances cause small offset values. It is possible to modify the navigation pivot point as discussed above, or by using the Navigation pivot point tool (see page 128).

4.6 *Visualization Tools*

Fast Graphics Mode lets users display portions of the model using any combination of Hide and Unhide commands. Note that only meshplots can be hidden. Also, hidden geometry cannot be selected directly with selection tools (Region Selection, etc).

4.6.1 *Hide selection*

Hides the current selection. This tool is useful when you want to exclude from the rendering certain parts of a model without altering the current zone. To hide a selection, press and release H, or right-click in the graphics window and choose Fast Graphics → Display → Hide Selection. Hidden geometry cannot be selected.

4.6.2 *Unhide all*

Makes visible all previously hidden geometry. To unhide all the hidden geometry, press and release Ctrl +H, or right-click in the graphics window and choose Fast Graphics → Display → Unhide All.

4.6.3 *Hide invert*

Hides everything that is visible and unhides everything that is hidden. To unhide all the hidden geometry, right-click in the graphics window and choose Fast Graphics → Display → Invert.

4.6.4 *Hide unselected*

Hides everything that is not selected. To hide all the unselected geometry, right-click in the graphics window and choose Fast Graphics → Display → Hide Unselected.

4.7 *Visualization Objects*

Fast Graphics Mode allows the user to view internal parts of the model through the familiar cutting plane and cutting volumes. The orientation of the cutting plane or volume can be defined interactively, using the mouse.

4.7.1 *Cutting plane*

Cutting planes temporarily slice away a portion of the model. Sliced objects will be not affected by selection tools.

To create a cutting plane:

1. Activate the cutting plane tool by pressing and releasing T or clicking on the Cutting Plane icon  .
2. Move the mouse over the model or Scene bounding box to select the initial plane position and orientation.
3. Left click to create the plane.
4. Manipulate the plane using the Manipulators (see page 126).
5. To add a new cutting plane, press +, then repeat steps 2 to 4. Up to four planes can be defined.
6. Press Esc to finish using the tool.

The created cutting planes will remain active until they are disabled. The sliced surface will be represented in one of the planes colors (yellow, red, green, blue). It is possible to not render the sliced surface, by disabling the Cap sections option (see page 122)

To remove the cutting planes, click on the Cutting Plane icon  , or press and release T.

The direction of a cutting plane can be reversed by right clicking on the plane.

All cutting planes are removed automatically upon returning to standard mode.

4.7.2 *Cutting volume*

The Cutting Volume tool temporarily excludes from the scene a portion of a model that lies inside or outside of a volume shape. Objects affected by the volume will be not selected by selection tools.

The available cutting volumes are:

- Box 
- Sphere 
- Cylinder 

To create a cutting volume:

1. Click on the Cutting Box, Cutting Sphere or Cutting Cylinder icon.
2. Manipulate the volume using the Manipulators (see page 126).
3. Press Esc to finish using the tool.

The created cutting volume is active until it is disabled. The sliced surface is represented in yellow. It is possible to not render the sliced surfaces, by disabling Cap sections.

To remove a cutting volume, click on the active cutting volume icon. The cutting volume will be removed automatically upon returning to standard mode.

The direction of the volume section can be reversed by right clicking while defining the volume. In reverse mode, the volume will be painted in red and all the elements outside the volume will be affected.

4.7.3 *Cap sections*

The Cap Sections option controls the representation of the sliced surface resulting from Cutting Planes or Volumes.

By default, Cap Sections are enabled. To toggle the Cap Sections option:

1. Right-click in the graphics window and choose Fast Graphics → Configuration.
2. Select the Viewport Tab.

3. In the Viewport group, check/uncheck the Cap Section box.

4.7.4 *Known issues*

Shell elements drawn in midsurface depiction are not drawn correctly. Open or flipped surfaces are not displayed correctly when rendering the cap surface.

4.8 *Selection Tools*

Fast Graphics Mode allows users to select model entities in a variety of ways.

4.8.1 *Locator action*

Query: The Query icon lets you select objects within the mesh plots, such as elements and nodes. Information is displayed about the selection in the Message Window.

Pick: In standard mode, the Pick icon lets you select mesh plots and annotations such as band tables.

Dialog box selection: Pressing the P key, or double-clicking in a cyan column of a dialog box.

4.8.2 *Region selection*

The region selection tools allow you to select one or more objects by defining an outline or area. The type of created region depends on the current region type.

- Rectangular Region.
- Circular Region.
- Lasso Region.

Use Tab to switch between region types. The default region type is a rectangular region. The Query icon and Pick icon only work with a rectangular region.

To select using a rectangle:

1. Activate a region selection tool.
2. If the rectangular region is not active, use Tab to switch between region types until the rectangular region is active. The cursor changes to +.
3. Drag in the view, then release the mouse. The first location you click is one corner of the rectangle, and where you release the mouse defines the opposite corner.

To cancel the selection, right click before you release the mouse. Press Esc to finish using the Region Tool.

To select using a circle:

1. Activate a region selection tool.
2. If the circular region is not active, use Tab to switch between region types until the circular region is active. The cursor changes to ◯.
3. Drag in the view, then release the mouse. The first location you click is the center of the circle, where you release the mouse defines the circle radius.

To cancel the selection, right click before you release the mouse. Press Esc to finish using the Region Tool.

To select using a lasso:

1. Activatate a region selection tool.
2. If the lasso region is not active, use Tab to switch between region types until the lasso region is active. The cursor changes to .
3. Left-click to define the first segment point.
4. Move the mouse and left-click to define the second segment point. Repeat steps 3 to 4 as needed.
5. Complete the region by left-clicking without moving the mouse.

To cancel the selection, right click before you release the mouse. Press Esc to finish using the Region Tool.

If you hold down Shift while selecting a region, the selected objects are added to the current selection. If you hold down Ctrl, the selected objects are removed from the current selection.

Click Esc to finish using the Region Selection tool.

4.8.3 *Deep selection*

Deep selection allows the user to select objects obscured by other objects.

By default, deep selection is disabled. To enable/disable deep selection, right-click in the graphics window and choose Fast Graphics → Selection → Deep Selection.

4.9 *Selection Representation*

In standard mode, the selected items are 'highlighted.' FGM represents selected items in different ways depending on the item type. Selected items are represented in the defined selection color and are not obscured by other items.

4.9.1 *Selection silhouette*

Shows a black silhouette around the current selection.

To toggle the visualization of the Selection Silhouette:

1. Right-click in the graphics window and choose Fast Graphics → Configuration.
2. Check/Uncheck the Selection Silhouette option.

4.9.2 *Selection box*

Shows a box bounding the current selection.

To toggle the visualization of the Selection Box:

1. Right-click in the graphics window and choose Fast Graphics → Configuration.
2. Check/Uncheck the Selection Box option.

4.10 *Manipulators*

Manipulators transform one or more objects when the mouse is dragged. Manipulators appear when one or more Visualization Objects are created (see page 121) and a transformation tool is active.

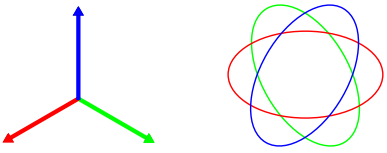


Figure 4.7: Move (left) and rotate (right) manipulators.

4.10.1 *Move manipulator*

The Move Manipulator moves one or more objects when the mouse is dragged. To use the Move Manipulator:

1. Choose an axis by positioning the mouse over one of the three axes of the Axes icon. The axis turns yellow to indicate that it is active.
2. Drag the mouse to move the selection along that axis

Choose an axis by first positioning the mouse over any axis of the icon, then dragging the mouse to move the selection along that axis. When you position the mouse over any axis, it turns yellow to indicate that it is active.

4.10.2 *Rotate manipulator*

The Rotate Manipulator lets you choose a plane of rotation when rotating a selection with the mouse. Choose a plane of rotation by first positioning the mouse over any plane of the icon, then dragging the mouse to rotate the selection in that plane. When you position the mouse over a plane of rotation, it turns yellow to indicate that it is active. The Rotate Manipulator uses Angle snap mode to set the rotation increment.

4.11 *Snap Mode*

Snap Mode sets rotation and scaling increments to a given value.

4.11.1 *Angle snap mode*

Use this mode to set the incremental rotation to be applied to a number of features. To toggle Angle Snap, click the Angle Snap icon .

To change the incremental angle of rotation:

1. Click the down arrow to the right of the Angle Snap Icon.
2. Enter the desired snap increment into the dialog box, and click OK.

The default rotation snap angle is 10 degrees.

4.11.2 *Scaling snap mode*

The Scaling Snap Mode controls the percent of increment to be applied to a number of features. For example, the Scale Manipulator uses Scaling Snap.

To toggle Scaling Snap, click the Scaling Snap icon .

To change the scaling increment:

1. Click the down arrow to the right of the Scaling Snap icon.

2. Enter the desired snap increment into the dialog box, and click OK.

4.12 *Transform Tools*

As with the standard graphics mode, all objects displayed in the FGM viewport can be arbitrarily transformed (*e.g.*, scaled and/or rotated).

4.12.1 *Navigation pivot point tool*

FGM uses the Navigation pivot point (see page 119) as a reference position to calculate a number of features related with the view. Users can move the navigation pivot point via the Navigation Pivot Point Tool.

To move the navigation pivot point using this tool:

1. Activate the tool by pressing Insert.
2. Use the Move manipulator to move the navigation pivot point.
3. Press Enter to apply the change to the view.

It is possible to cancel the current translation by clicking on the right mouse button without releasing the left mouse button.

To deactivate the tool, press Esc, or activate a different tool.[2]

[2] If you deactivate the tool without applying the changes by pressing Enter the navigation pivot point will not be modified.

4.12.2 *2D transform*

Any annotation in the graphics window can be moved or scaled.

To transform a selected annotation:

1. Select the annotation to be transformed using the Pick icon.
2. Right-click in the graphics window and choose Fast Graphics →Transform →Move or Scale.
3. Click and Drag the mouse.

4. Repeat steps 2 and 3, as necessary.
5. Exit the tool pressing Esc or changing the tool.

Alternatively it is possiblee to activate 2D Transform Move or 2D Transform Scale using the assigned key shortcut (see Fast Graphics Options-Shortcuts). It is possible to cancel the current transform by clicking on the right mouse button while holding the left mouse button.

The scale increment can be set by activating the Scaling snap mode (see page 127). The 2D Transform tool is only available when the FGM navigation interface option (see page 115) is disabled.

4.13 *Visual Appearance and Effects*

Fast Graphics Mode offers several features for controlling the appearance of shapes and surfaces. These capabilities can be used to generate more realistic visual representations of complex models.

4.13.1 *Surface smooth*

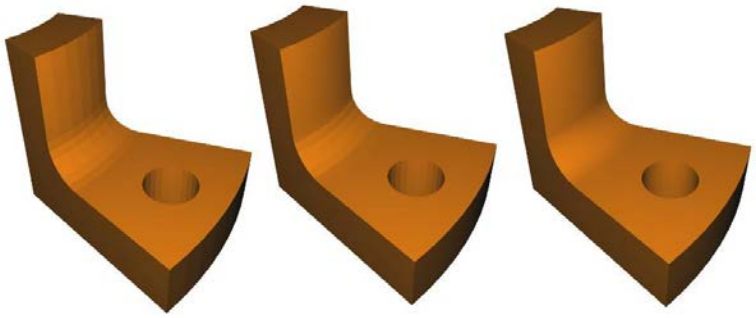


Figure 4.8: Smoothing applied with values of 0, 15, and 30, from left to right.

Smoothing blends the shading at the edges between adjacent faces based on the angle to produce the appearance of a smooth surface. The advantage of this technique is that smooth surfaces are generated without increasing the surface density (using, for example, the Surface Depiction dialog box).

To change the smooth angle:

1. Right-click in the graphics window and choose Fast Graphics → Configuration.
2. Select the Advanced Tab.
3. Change the Surface Smooth value.

By default, the smooth angle is 15 degrees.

4.13.2 *Shading*

FGM offers several shaders. By default, FGM uses Gouraud shading. To select a shader:

1. Right-click in the graphics window and choose Fast Graphics → Shader.
2. Choose the desired shader.

Gouraud: The Gouraud shader calculates the color for each vertex (per-vertex shading) and associates the color with that vertex. The colors are then interpolated across the face of the polygon to produce a smooth effect. The final quality of the represented surface depends on the face density.

The shader properties used for the lighting are Ambient, Diffuse, Specular and Emissive colors.

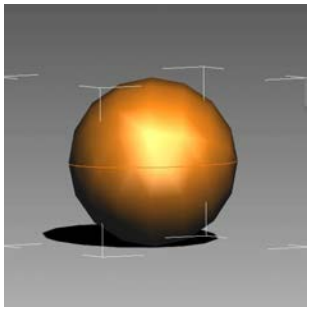


Figure 4.9: Gouraud shading.

Gouraud-Lambert: Like Gouraud, the Gouraud-Lambert shader calculates the lighting calculations per-vertex, but the Gourand-Lambert shader does not apply the specular component. The lighting calculation for this shader is similar to that used in the standard UI.

Gouraud-Lambert is the simplest and fastest shader. Its lighting properties are Ambient, Diffuse, and Emissive colors.

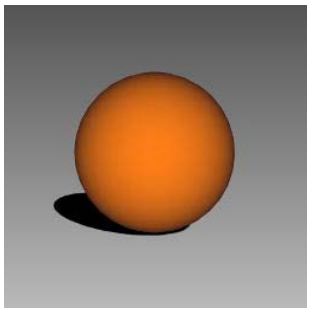


Figure 4.10: Gouraud-Lambert shading.

Lambert: The Lambert shader represents matte surfaces with no specular highlights. The lighting calculation is performed per pixel.

The shader properties used for the lighting are Ambient, Diffuse and Emissive colors.

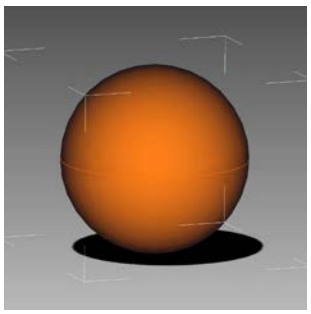


Figure 4.11: Lambert shading.

Flat: The Flat shader calculates the lighting on each face independently of its neighbors, so that there is no variation of color across the face, causing each polygon to have a flat appearance. This can help visualize element faces in a complex mesh.

The properties used for the lighting are Ambient, Diffuse, Specular and Emissive colors.

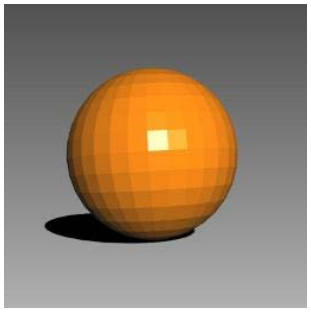


Figure 4.12: Flat shading.

Blinn: This shader is particularly effective at simulating metallic surfaces (*e.g.*, brass or copper) which typically have soft specular characteristics.

Phong: The Phong shader mimics glossy surfaces with a hard specular highlight.

Image Based Lighting (IBL): IBL uses an environment texture to illuminate the scene.

Environment: This shader effective simulates environmental reflections on polished metallic surfaces.

Wireframe: This shader simply represents surface facets as white with red edges.

Gooch: Gooch (cool to warm) shading is a type of non-realistic lighting.

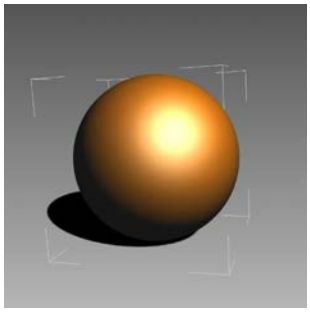


Figure 4.13: Blinn shading.

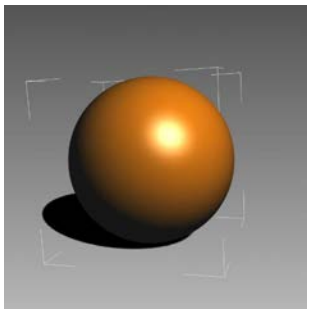


Figure 4.14: Phong shading.

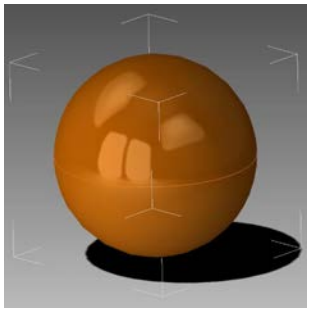


Figure 4.15: Image based lighting shading.

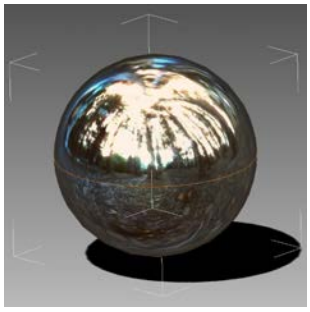


Figure 4.16: Environment shading.

4.13.3 *Translucency*

FGM introduces translucency for visualizing the interior of a solid. To toggle translucency, click the Cull Front Faces icon .

To configure the translucency:

1. Right-click in the graphics window and choose Fast Graphics → Configuration.
2. Select the Advanced tab.
3. Choose a Method in the Translucency Method field (see below for choices)

Average: This is the default translucency mode; it offers good performance and moderate quality.

Fast: This mode offers the best performance but lowest quality. It is useful for complex scenes where performance is required.

XRay: Similar to Average in performance, this method imitates an X-ray film of the object.

Dual Peeling: Dual peeling is the slowest method of representation and requires more graphical processing unit (GPU) memory.

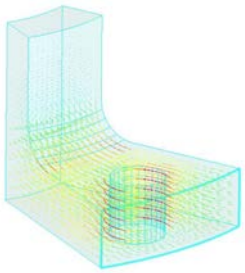


Figure 4.17: Translucent rendering.

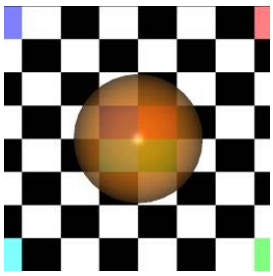


Figure 4.18: Average translucency method.

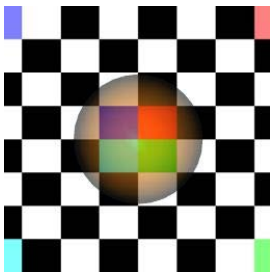


Figure 4.19: Fast translucency method.

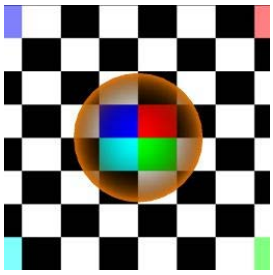


Figure 4.20: XRay translucency method.

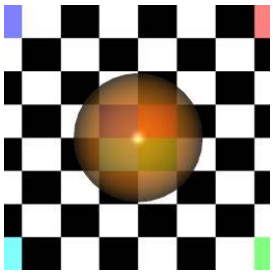


Figure 4.21: Dual Peeling translucency method.

4.14 *Advanced Configuration*

VSync instructs the graphics card how to synchronize its actions with the monitor. That means the graphics card can swap its frame buffer and send a new frame to the monitor only when the monitor is ready to redraw a new screen. This affects the *framerate*, or the number of frames per second.

To change the Vertical Sync:

1. Right-click in the graphics window and choose Fast Graphics → Configuration.
2. Select the Advanced tab.
3. In the Visualization box, make a selection (see below).

On: When VSync is enabled, the graphics card is instructed to wait for the monitor to signal when it's ready for

a new frame before supplying a single whole frame. The first noticeable impact is that framerate is limited to the monitor's current refresh rate. For example, if the monitor's refresh rate is 60Hz, the frame rate cannot exceed 60 frames per second.

Off: When VSync is disabled, the framerate can be greater but the 'Tearing' effect might appear.

Auto: When Auto is chosen, VSync will only be enabled whenever the framerate exceeds the refresh rate. If the framerate falls below the refresh rate, VSync is instantly disabled.

4.15 Animation

ADINA FGM implements a new method of representing pre-generated animations. Unlike the standard mode, FGM allows the manipulation of the camera in real time. It is possible to save and load the animation independently from the database. Depending on the type of solution and the number of steps, the required memory will be different.

4.15.1 Visualization

FGM can reproduce only animations generated while in FGM, any animation generated and stored in the database using the standard mode won't be compatible.

To Playback an animation:

1. Load a pre saved animation (see below) or generate a new animation using the AUI interface.
2. The Time Slider (horizontal scroll bar) will be visible.
3. Press Play ▷ icon in the FGM toolbar to see the animation.
4. Click Stop ■ icon to end the animation playback.
5. Use the Time Slider (Horizontal scroll bar) to go to different steps.

In case of a solution with multiple mesh plots, only the animation of the active will be played, it is possible to change the active mesh plot while reproducing the animation by picking a different mesh plot. During the reproduction of an animation it is possible to change the location and orientation of the camera by using any Navigation Tools, note that Particle Selection and Query will be disabled.

To customize:

1. With a loaded animation Open the Reproduction Config by pressing the arrow icon next to the Stop ■ icon.
2. Select the desired reproduction mode by checking One-Way, Two-Way or none.
3. Adjust the reproduction speed clicking on the Speed combo-box and selection one of the options.

4. Click the OK button to apply the changes and close the dialog.

Note: It is possible to change the speed of reproduction of the animation by pressing Page Up and Page Down.

One-Way Reproduction: The loaded animation will be played in a loop going in one direction from beginning to the end.

Two-Way Reproduction: The animation will be played in a loop from begin to end and reverse.

4.15.2 Save & Load

Animation files generated while in FGM mode, are independent of the database and can be visualized without the need to have a database loaded. To save an animation:

1. While in FGM, generate a animation using one of the standard method.
2. Select File → Fast Graphics → Save Animation (ani) from the main menu.
3. The FGM Save Dialog will be open, select the folder to save the animation.
4. Enter the name of the animation file name.
5. Press the Save button.

An informative message will be shown in the message dialog.

To load an animation:

1. Select File → Fast Graphics → Load Animation (ani).
2. The FGM Load Dialog will be open, select the animation file to be loaded.
3. Press the Open button.

An informative message will be shown in the message dialog. The loaded animation can now be played.

5 Differences in default solver settings for ADINA and SOL 601/701

Below is a complete list of the differences in default solver settings for ADINA and SOL 601/701.

5.1 Implicit Time Integration Method

The implicit time integration method for implicit dynamic analysis, low-speed dynamics, and the stabilized total-load-application (TLA-S) procedure is different for ADINA and SOL 601/701.

- **ADINA Solver:** Uses the Bathe implicit time integration method by default.
- **SOL 601/701 Solver:** Uses the Newmark implicit time integration method by default.

The below Tech Brief shows examples where the Bathe method leads to stable and accurate solutions and the Newmark method fails.

Implicit Time Integration — What Can Go Wrong

5.2 Incompatible Modes Formulation

The use of the incompatible modes formulation for 4-node 2D-solid elements, 8-node 3D-solid elements, and 4-node shell elements is different for ADINA and SOL 601/701.

- **ADINA Solver:** Does not use the incompatible modes formulation by default.

- **SOL 601/701 Solver:** Uses the incompatible modes formulation by default.

The below Tech Brief shows a surprising unphysical buckling mode that can occur when the incompatible modes formulation is used in geometrically nonlinear analysis. For this reason, the incompatible modes formulation must be used with great care in nonlinear analysis.

Surprises in Analyses with Incompatible Modes Elements

5.3 Mixed (u/p) Formulation

The use of the mixed (u/p) formulation for material models is different for ADINA and SOL 601/701.

- **ADINA Solver:** Uses the mixed (u/p) formulation by default for all rubber material models and for ALL elastic-plastic material models.
- **SOL 601/701 Solver:** Uses the mixed (u/p) formulation by default for all rubber material models and for the plastic-bilinear and plastic-multilinear material models.

5.4 Through-Thickness Integration Order for Shell Elements

The through-thickness ($t-$) direction integration order for shell elements is different for ADINA and SOL 601/701.

- **ADINA Solver:** Uses 2-point Gauss integration by default for single-layer shell elements, and 2-point Gauss integration for each layer for multi-layer shell elements.
- **SOL 601/701 Solver:** Uses, for the below material models, 5-point Newton-Cotes integration by default for single-layer shell elements, and 3-point Newton-Cotes integration for each layer for multi-layer shell elements.

1. Plastic-bilinear.
2. Plastic-multilinear.
3. Nonlinear elastic.
4. SMA
5. Thermo-plastic

6. Plastic creep multilinear.

5.5 *Strain Increments in Viscoelastic Materials*

The number of subdivisions of strain increments used in the integration of stresses for viscoelastic materials is different for ADINA and SOL 601/701.

- **ADINA Solver:** Uses 10 subdivisions of strain increments by default in the integration of stresses for viscoelastic materials (`MATERIAL VISCOELASTIC NSUBD=10`).
- **SOL 601/701 Solver:** Uses 1 strain increment by default in the integration of stresses for viscoelastic materials (`MATERIAL VISCOELASTIC NSUBD=1`).

Topic Index

Command Index