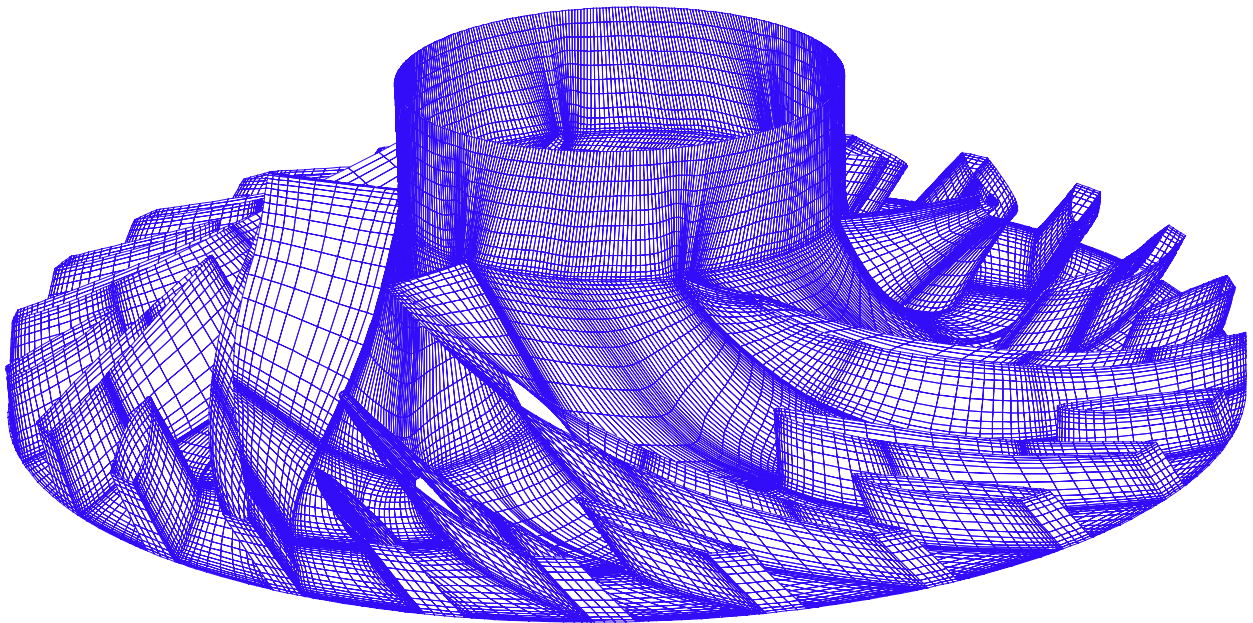


TrueGrid[®]

“A quality mesh in a
fraction of the time.”



Version 2.2

A Tutorial

Table of Contents

I. Introduction to TrueGrid®	3
II. The Phases of TrueGrid®	4
III. Text/Menu Window, Menus, Dialogue Boxes and Help	5
IV. Geometry	10
V. TrueGrid® Graphics	13
VI. Making a Single Block Part	19
VII. How TrueGrid® Built the Mesh	24
VIII. The History Table: Turning Commands Off and On	25
IX. Interactively Changing the Mesh Density	27
X. Clustering of Nodes (Zoning)	28
XI. Clustering of Nodes (Zoning) in Opposite Directions	29
XII. Saving and Rerunning the Session File	30
XIII. The Session and Input File Format	31
XIV. A Multiple-Block Part	33
XV. Putting a Square Peg into a Round Hole	39
XVI. Intersecting Pipes Example	42
XVII. Example of a Fuel Tank	52

I. Introduction to TrueGrid®

TrueGrid® is a general purpose tool for creating a multiple-block-structured mesh. Although the beginning of this Tutorial provides some introductory information, the new user is strongly advised to also read the first 2 chapters of the **TrueGrid®** User's Manual, A User's Guide and Reference Manual.

The processes of creating a mesh and creating geometry are separated within **TrueGrid®**. Surfaces and curves may be created internally, imported from a CAD/CAM system via an IGES file, or imported as polygon surfaces using the **vpsd** command. A block mesh is then created and molded to the shape of the geometry.

Making a mesh with **TrueGrid®** is analogous to sculpting: A block mesh is created using the **block** command. Pieces of the block mesh are removed so that its topology matches that of the object to be meshed. Portions of the mesh are moved, positioned along curves, projected to surfaces, etc. Interpolation, smoothing, and zoning requirements are used to fine-tune the mesh. Parts created separately are glued with the block boundary interfaces and common nodes merged together by specifying a tolerance.

TrueGrid® is based on a powerful technique known as the projection method. The projection method allows faces, edges, and nodes of the mesh to be directly placed on surfaces. Similarly, edges and nodes of the mesh can be placed along curves. In this sense, **TrueGrid®** is a constraint-based mesh generator.

Additionally, if **TrueGrid®** is required to place two faces of the mesh sharing a common edge onto two different surfaces, then it will automatically place the shared edge on the intersection of the two surfaces. There is no need to construct the intersection curve of two surfaces; **TrueGrid®** will find the intersection curve automatically. Likewise, a vertex required to be on three surfaces will be placed at the intersection point of all three surfaces.

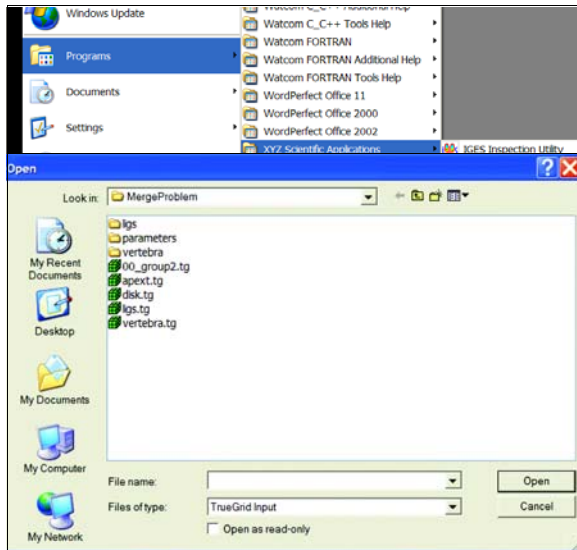
Other notable features of **TrueGrid®** include:

1. You can choose to run **TrueGrid®** in an interactive or batch mode or switch back and forth between these two modes.
2. Any number of commands can be deactivated and reactivated.
3. The most frequently used mesh generation commands can be performed using the mouse.
4. The mesh can be easily refined.
5. **TrueGrid®** produces a session files that can be easily edited and rerun in a batch mode.
6. Every feature of the mesh can be made into a parametric feature.

There are a few conventions used in this tutorial. All text to be typed or produced by a dialogue are placed in a *Courier* font. Keyword commands in a sentence are **Bold** face. The name of a button in the menus or the environment window are ***Bold and Italic***. It is also assumed that you are using a mouse with 3 buttons. If you only have 2 buttons, use the right mouse button with the control key down to simulate the third mouse button.

II. The Phases of TrueGrid®

To run TrueGrid® on a WINDOWS system, double click on the TG (TrueGrid®) icon or shortcut in the desktop or go to START , PROGRAMS, XYZ Scientific Applications and click on TG.



When TrueGrid® is run on a WINDOWS system, the first window to appear is

which can be ignored by clicking on cancel in the bottom right. This window lets you select an existing command file to be run in batch mode.

After clicking on cancel or when TrueGrid® is run on a UNIX or LINUX system with the tg command, a text and menu window appears on the screen in the upper left corner. The initial title of this window is *Control Phase*. The title of this particular window will always indicate the current phase of TrueGrid®. Because the current phase depends on the commands issued, the text and menu window title changes depending on the commands issued interactively and/or read from a batch file.

The Control Phase, the initial phase of the code, is where output options are chosen, material models defined, geometry created, and where parameters governing the final mesh output file are set. No graphical capabilities are available in the Control Phase.

The Part Phase is entered as soon as a block mesh is created using either the `block` or `cylinder` command. Three new windows appear in the Part Phase: the *Computational Window* (used for displaying logical blocks of the mesh), the *Physical Window* (used for displaying the actual mesh and any geometry), and the *Environment Window* (used to determine what is in a graphics window, how it is displayed, and how it can be manipulated using the mouse). It is in the Part Phase where the mesh is constructed by positioning, projecting, deleting, zoning, refining and smoothing parts of the mesh (as well as by performing other functions). Boundary conditions and loads for a part are also specified in the Part Phase.

The Merge Phase is where parts are assembled into one model by merging, i.e., gluing, nodes together that are within a specified tolerance of each other. Only the *Computational window* of the Part Phase is missing. Both the *Physical* and *Environment* windows are present in the Merge Phase. It is in the Merge Phase that the mesh output file is written. There are features available in the Merge Phase to view boundary conditions, loads, and properties, to diagnose the condition of the mesh, and to view the mesh and geometry.

III. Text/Menu Window, Menus, Dialogue Boxes and Help

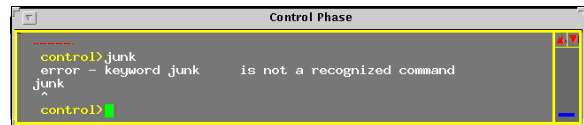
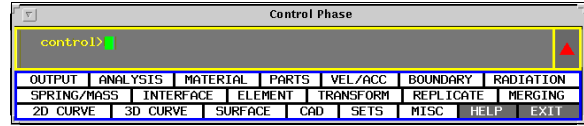
The Main Menu

The main menu of **TrueGrid**[®] appears in the upper left text/menu window. This text/menu window has a title indicating the current phase of the code (either *Control Phase*, *Part Phase*, or *Merge Phase*). Commands can be issued from this window, and any output such as warnings or error messages is directed to this window.

Because of the limited screen space, the text/menu window is not particularly large. This leaves room for other windows that appear in the other phases of the code. Consequently, when there is output from **TrueGrid**[®] to the user, the menu automatically disappears and the text portion of the text/menu window is enlarged to fill the window. For example, enter the illegal command

junk

on the command line. A message is issued indicating that junk is an illegal command. Notice that the menu is no longer visible.

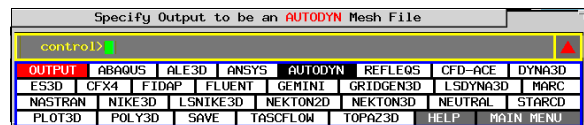


To bring the menu back, either press the Enter key or press the right mouse button (3-button mouse) in the text/menu window, or press the middle mouse button in the scroll bar area of the window.

To expand the text area to cover the window and remove the menu, press the upward pointing arrow appearing on the right of the text/menu window, or press the Page Up key or the Up Arrow key on the keyboard while the cursor is in the text window. Any of these actions indicate to **TrueGrid**[®] that you wish to view more of the previous text. In such cases, the text area expands to fill the window, thereby making more of the previous text visible. To scroll through the text, either use the Page Up, Page Down, Up Arrow or Down Arrow keyboard keys, or use the scroll bar. Pressing the left mouse button on the arrows above the scroll bar moves text by pages. Pressing the middle mouse button on the arrows moves the text by lines.

Getting Help For a Main Menu

In the main menu area of the text/menu window there are two grey buttons, the **HELP** button and the **EXIT** button.



Press the **EXIT** button with the left mouse button to quit **TrueGrid**[®].

Press the **HELP** button to activate the help system. Normally, pressing the left mouse button on an entry of the main menu causes a submenu to appear. However, when the **HELP** button is on, pressing the left mouse button on a main menu entry calls up a Help Window containing information about all the commands in the given category.

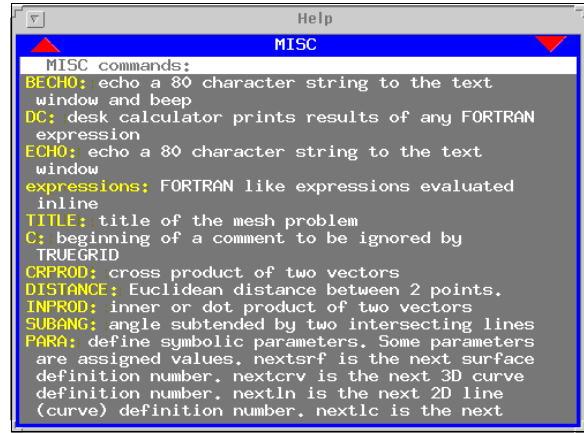
The Help Window contains a title bar with scroll arrows as required. Pressing the left mouse button scrolls by page. Pressing the middle mouse button scrolls by line. The Page Up, Page Up, Up Arrow and Down Arrow keys can also be used for scrolling.

Pressing the left mouse button on another category of the main menu replaces the contents of the current help window with information about the new category.

To kill the Help Window, either use window manager pop-ups or turn the **HELP** button off.

Sub-Menus

Submenus contain a list of all the commands in a main-menu category. One uses a submenu to access detailed help or a dialogue box for each command in the main-menu category. A submenu is viewed by pressing the left mouse button on a category of the main menu while the **HELP** button is off.

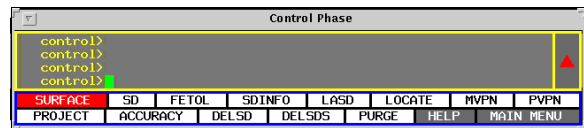


Each submenu contains two grey buttons: a **MAIN MENU** button and a **HELP** button. Press the **MAIN MENU** button to return to the main menu. The **HELP** button of a submenu works the same as that of the main menu. Each submenu also contains as its first entry a red button which indicates the name of the current submenu. The red button is not an active button.

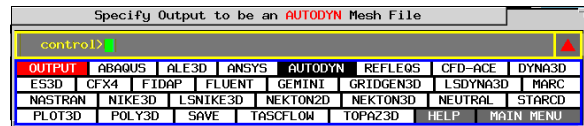
Getting Help For a Submenu Item

When the mouse is moved onto a command button in the submenu, a bar appears above the text/menu window with a short description of that command.

The help system works the same as for the main menu, but the Help Window for a command in a submenu is far more specific.

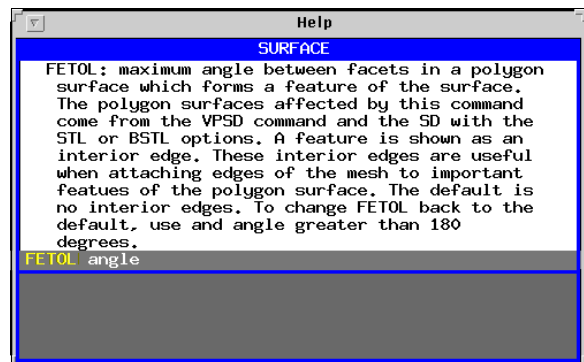


The first part of the help for a command is in white with black print and describes the function of the command. The second part of the help contains a complete syntax for the command. All literal options are in upper-case letters. Other strings (especially those containing underscores) are meant to describe the type of data to be enter. Semicolons should be regarded as literal.



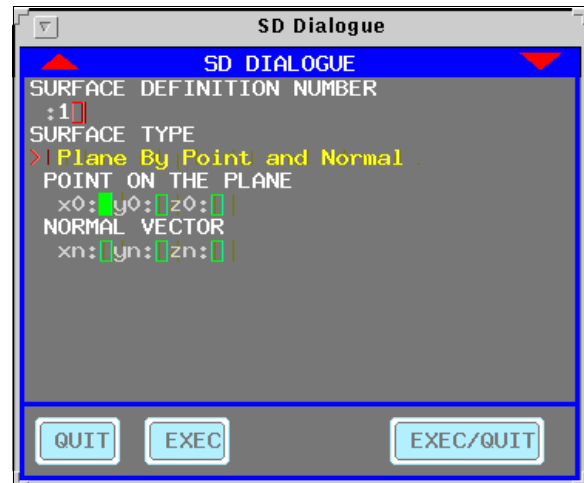
Dialogue Boxes

Dialogue boxes are available for every command. To obtain a dialogue box for a particular command, move to the submenu containing that command, and press the left mouse button on the command name. (Make sure the **HELP** button is off.)



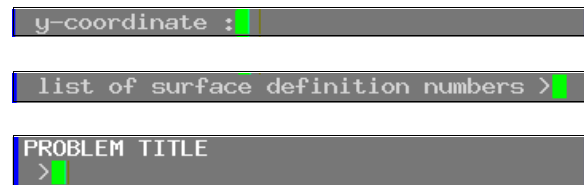
Some of the dialogue boxes contain complex decision trees, especially for commands used to set options for output to a particular simulation code. Dialogue boxes may contain input fields and option lists as well as descriptions and/or titles for such. An input field consists of an optional description (in white print), prompts (in light grey print), and cursor boxes (initially green).

There may be many input fields within a dialogue box. The current active field (the one to which text input is directed) is always displayed with a filled green cursor. The other active cursors are hollow green cursors.



There are three types of input fields:

1. Fields for entering a single number,
2. Fields for entering a list of numbers,
3. Fields for entering general strings.



In the first two cases, input is checked as it is entered. Characters that would create an invalid string are ignored. Furthermore, if just one number is required, then **TrueGrid**[®] will not allow a second number to be entered into that field.

Moving the Cursor in a Dialogue Box

The active cursor can be repositioned to another character of the current field, or to another field entirely using the left mouse button. Press the left mouse button on the prompt itself to position the cursor at the end of the input field. Alternatively, press the Enter key to move the cursor to the next available field.

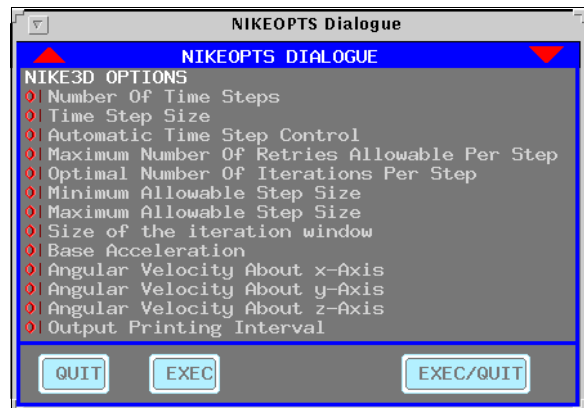
If the current field is empty, the cursor can always be moved from the field. However, if the current field is non-empty and is not complete, then **TrueGrid**[®] will not allow the user to move the cursor from the current field. For example, **1.0e-** can be entered, but is not complete. **TrueGrid**[®] informs the user of such problems by turning the active cursor blue. The user must correct the problem before the cursor can be moved to another field. The user can use the mouse to move the cursor to a previous character of the current field. The backspace can be used to delete a previous character. Or, CTRL+x can be used to delete the character beneath the cursor.

When the user presses the Enter key to end the current field and begin the next, the current field is marked "finished" with a red cursor. The cursor will not return to this field again without positioning the cursor using the mouse.

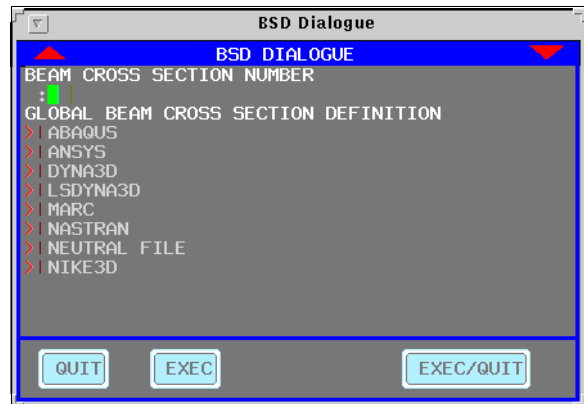
Lists in Dialogue Boxes

There are two types of lists that occur in dialogue boxes: exclusive (choose one) and non-exclusive (many items can be chosen). Exclusive lists have entries that begin with a red ">" character. Non-exclusive lists have entries that begin with a red "O". A title/description is in white above the list.

The following is an example of non-exclusive dialogue box list:

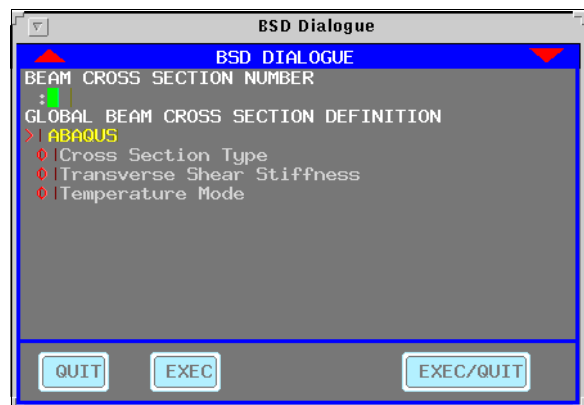


The following is an example of exclusive dialogue box list:



Choose an item of a list by pressing the left mouse button anywhere on the text of that entry. The text of the chosen entry will be highlighted in yellow. To deselect the entry, press the left mouse button on the entry again.

When an entry of an exclusive list is chosen, the other entries disappear from sight. The hidden entries reappear if the selected entry is deselected.



Executing a Dialogue Box

Press the *EXEC*ute button at the bottom of the dialogue box to issue the command. Any number of error conditions will prevent execution of the dialogue:



- (a) Not all input fields are filled in,

- (b) Not all input fields have been flagged as "complete". Pressing ENTER causes the field to be checked for completeness. Complete fields have red cursors at the end.
- (c) No entry has been chosen for a list that requires a choice to be made.
- (d) **TrueGrid**[®] is in a state where a command is in progress.

If (a) or (b) occur, then the input field will be moved just below the top of the dialogue, and the prompt for the first offending field is colored blue. If (c) occurs, then the first entry of the offending list appears near the top of the dialogue, and the marker (either ">" or "o" is colored blue). If (d) occurs, the **EXECute** button is colored red.

To **EXECute** and **QUIT** the dialogue box, use the **EXEC/QUIT** button. This button is usually the best choice because it requires the least number of mouse actions.

Pictures are not automatically redrawn when commands are issued in **TrueGrid**[®]. To insure that a new picture is drawn, press the **EXEC** button or the **EXEC/QUIT** button using the middle mouse button instead of the left mouse button. Doing so causes a draw command to be issued with the command.

IV. Geometry

Curves and surfaces play a unique role in **TrueGrid**[®] mesh generation because the mesh starts out with no relation to the geometry. You must take steps to associate parts of the mesh to the curves and surfaces. **TrueGrid**[®] has numerous methods to create the curves and surfaces. The geometry is also imported from CAD systems using the IGES data format.

Internal Geometry Library

TrueGrid[®] can create a wide variety of surface types using the `sd` command such as:

- sphere
- plane
- cylinder
- cone
- torus
- surface of revolution
- ruled surface
- swept surface (using skinning)
- extruded surface (general cylinders)
- interpolation between two surfaces
- tabulated surface (array of points)
- polygon surface
- pipe surface from a 3D curve
- surface given by equations
- cubic spline surface
- B-spline surface
- NURBS surface
- blended curves
- Stereo Lithography surfaces
- compositions of any of the above

An extensive set of options for building 2D curves using the `ld` command and 3D curves using the `curd` command is also provided within **TrueGrid**[®]. Surfaces and 3D curves are used to shape the mesh. 2D curves and 3D curves are used to create surfaces.

Importing IGES Entities

Most Computer Aided Design programs (CAD) export their geometry, such as surfaces, curves, and solids, to other programs with a file in a format known as IGES (Initial Graphics Exchange System). **TrueGrid**[®] can extract most surface and curve entities from an IGES file. The projection method in **TrueGrid**[®] uses trimmed surfaces, not solids. When exporting an IGES to **TrueGrid**[®], be sure to select trimmed surfaces as the type of export from your CAD system.

When import an IGES file into **TrueGrid**[®], be sure to be in either the part or merge phase where there is a graphics window to display the surfaces.

Some CAD systems automatically will export some extra planer surfaces used by some programs to define the extreme boundary of the entire model. These surfaces can be so large that they will dwarf the size of the surfaces and make them appear as a single point in the middle of the picture. If your CAD system has this characteristic, you must learn to remove these bounding planes from the display list and restore the picture so that the surfaces of interest are scaled to fill the display screen in front of you. You will learn to do some of these things in the next section.

Extracting All IGES Entities

All supported entities are extracted from an IGES file with the single command

```
iges IGES_file_name m n;
```

Issuing this command causes the specified IGES input file to be read and all curves and surfaces to be extracted. You can find this command in the CAD sub-menu. The surfaces are assigned **TrueGrid**[®] surface definition numbers beginning with *m*, and the curves are assigned **TrueGrid**[®] curve definition numbers beginning with *n* (where *m* and *n* are integers.) Note that curves used to construct composite curves or to trim a surface are not processed.

Example

```
iges airplane.igs 1 1;
```

Selectively Extracting IGES Entities

Entities can be selectively extracted from an IGES file using a two-step process. First, specify the IGES file using the command (also in the CAD sub-menu)

```
igesfile IGES_file_name
```

Second, use either the `igescd`, `igessd`, or `nurbsd` command to extract curves, surfaces, or NURBS surfaces, respectively. These commands are all found in the CAD sub-menu, as well.

Example

```
igesfile airplane.igs  
nurbsd 1 24 1;  
igessd 1 15 25;  
igescd 1 87 2;
```

will cause **TrueGrid**[®] to

- (a) read the IGES file 'airplane.igs';
- (b) to extract the first 24 NURBS surfaces, and to create **TrueGrid**[®] surface definitions 1 to 24;
- (c) to extract the first 15 surfaces (other than NURBS), and to create **TrueGrid**[®] surface definition numbers 25 to 40;
- (d) to extract the first 87 curves from the IGES file, and to create **TrueGrid**[®] curve definition numbers 2 to 88.

Saving Time

Large IGES files may take some time to process within **TrueGrid**[®]. It can be time-consuming to tile all the surfaces for graphics purposes. Every time **TrueGrid**[®] is rerun and, consequently, asked to read the same IGES file, all this work must be repeated. To eliminate the need for subsequent processings of an IGES file, use the command

```
saveiges binary_output_file
```

Example

Suppose that after issuing the commands from one of the previous examples, the command

```
saveiges airplane.bin
```

is issued. Then the next time the same set of surfaces and curves are to be used, issue the commands

```
useiges airplane.bin  
iges airplane.igs 1 1;
```

or

```
useiges airplane.bin  
igesfile airplane.igs  
nurbsd 1 24 1;  
igessd 1 15 25;  
igescd 1 87 2;
```

The `useiges` command is also found in the CAD sub-menu. **TrueGrid**[®] will process this set of commands many times faster than it was able to process the commands in the previous example.

IGES Levels (Layers) and Groups

The layer or level feature commonly found in CAD systems can be used to organize the geometry for **TrueGrid**[®] use. The associativity instance is used for grouping the entities into levels. **TrueGrid**[®] preserves this level structure.

V. TrueGrid[®] Graphics

TrueGrid[®] provides dynamic control for rotating, translating, and zooming. Also provided is readily accessible control over what curves, surfaces, regions, etc. are in the picture and whether or not they are labeled.

Example

Run **TrueGrid[®]**. If you are running on a PC with WINDOWS, start it as described in the introduction and click on the *Cancel* button in the file browser so that you are not running from an existing batch file.

If you are on a LINUX or UNIX system, use the command `tg` with no command-line options.

There is no graphics capability in the Control Phase of the code, so issue the command

```
merge
```

to enter the Merge Phase of the code. Next create a couple of surfaces using the `sd` **S**urface **D**efinition command:

```
sd 1 cyli    0 0 0  0 1 0  2
sd 2 plan   -1 0 0  1 0 0
```

You can define these surfaces using the *SD* dialogue box under the *SURFACE* menu.

The first surface is a right circular cylinder of radius 2 whose axis of rotation passes through (0,0,0) and is parallel to (0,1,0). The second surface is a plane passing through (-1,0,0) and normal to (1,0,0).

Even though the plane is actually perpendicular to the x-y plane, it appears somewhat slanted. This is because perspective is added to the picture, i.e., distant parts of the picture are smaller. The various circular cross sections of the cylinder differ from each other for the same reason. The perspective angle (the angle between the center of the picture and the edge of the screen) is controlled by the command

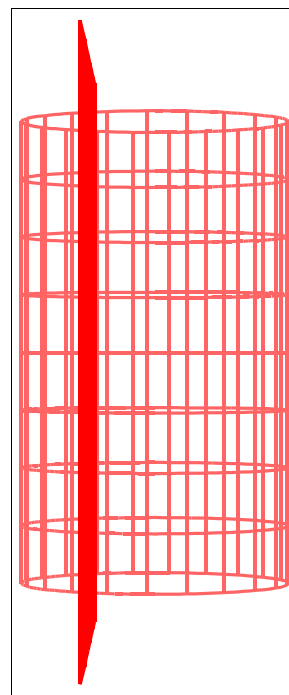
```
angle angle_of_perspective
```

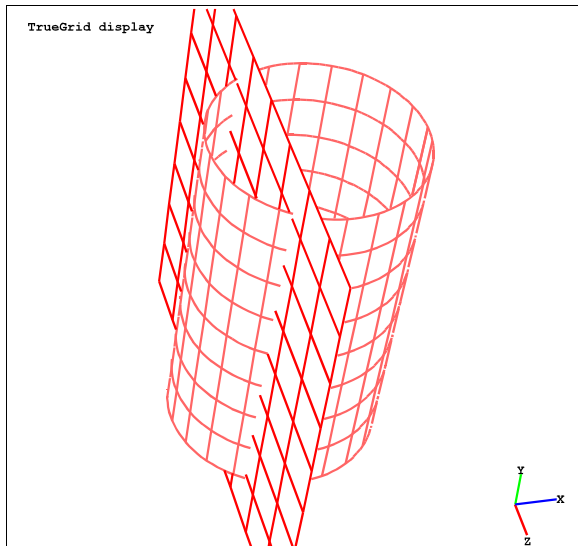
The default angle of perspective is 23 degrees. Setting the angle to 0 eliminates all effects of perspective.

Specifying the Type of Picture

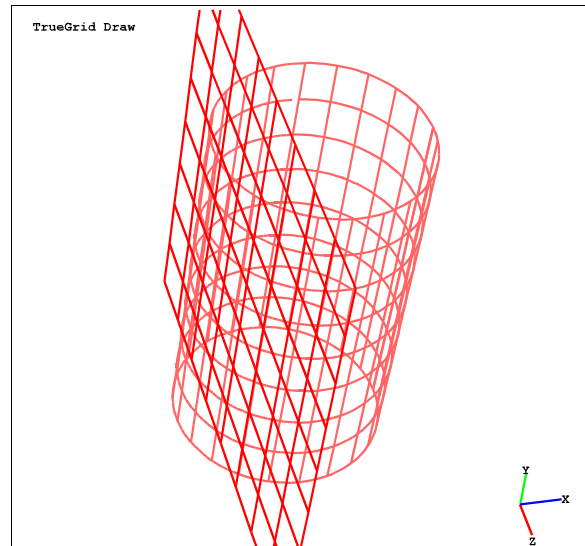


TrueGrid[®] can create several types of pictures. Which type of picture will be generated depends on the setting of the Picture button group in the Environment Window of **TrueGrid[®]** (shown above). The initial setting of the picture type is a *Hide* drawing (hidden lines are removed). The other types are described below. To choose a different picture type for all subsequently generated pictures, press the left mouse button on the desired picture type button (*Wire*, *Hide* or *Fill*). Changing the picture type does *not* cause a picture to be regenerated. A picture is redrawn by pressing either the *Draw*, *Center* or *Restore* button.





Picture Type: Hide



Picture Type: Wire

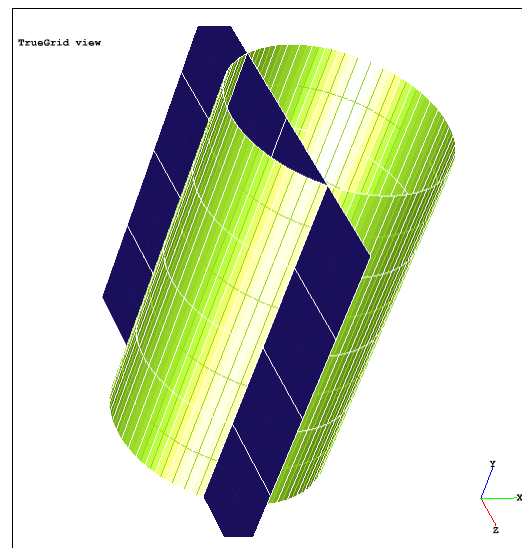
The simplest type of picture is a wire frame drawing of all objects in the picture (the preceding picture is an example). Objects that would be hidden by others (in the real world) are not hidden in a *Wire* picture. The *Hide* option is used for creating pictures whose hidden lines are removed. The

Fill option is used to create a picture where all polygons are filled and all hidden surfaces are removed. A two-source lighting model is used (along with dithering to create a very smoothly lighted picture when only 8 bit plane graphics is available.)

When the picture type is set to *Fill*, several options concerning the lighting model can be changed via a pop-up menu which is accessed with the right mouse button in the Physical Window.

Hardware Graphics

We currently support hardware graphics features only for SGI versions of **TrueGrid**[®]. To invoke the graphics hardware features (lighting, fogging, material models, etc.), press the *H.W.* button using the left mouse. When hardware is invoked, the *Wire* option is the same as before, except that lighting and fogging models now apply. The *Hide* and *Fill* options no longer function differently; both set the picture type to a *Fill* type. Many parameters of the hardware model can be interactively changed using the right mouse button to activate a pop-up menu in the Physical Window.



Picture Type: Fill

Moving Around Interactively

The Middle Mouse Button is used to manipulate 3D graphics windows in **TrueGrid**[®]. The effect of pressing the middle mouse button in either the Physical or Computational window depends on which of the four buttons, Rotate, Move, Zoom or Frame is pressed. The buttons in this Motion button group of



the Environment Window (pictured above) are reset by pressing the desired button using the left mouse button.

The Rotate, Move and Zoom options are all similar in nature. A dynamic operation is performed on the picture by moving the mouse to either the Physical or Computational Window, pressing the middle mouse button, and dragging the mouse to a new position while the middle mouse button remains pressed. As the mouse is moved, a minimal wire-frame representation of the picture is redrawn as quickly as possible. When the left mouse button is finally released, a new picture is redrawn. The type of the final picture is determined by the setting of the *Wire, Hide, Fill* button group.

Rotate

Horizontal movement of the mouse causes a rotation about a vertical line parallel to the screen plane. Vertical movement of the mouse causes a rotation about a horizontal line parallel to the screen plane. When the picture is magnified, rotations are reduced appropriately. This effect is eliminated by holding the “Shift” key down while rotating.

Move

This is also known as pan. The picture follows the mouse.

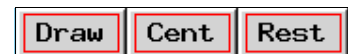
Zoom

Only vertical mouse movement causes a change. Move the mouse upward to enlarge the picture, and downward to shrink the picture. Moving the mouse half a window causes the picture to change by a factor of 10.

Frame

The *Frame* option is the fastest way to isolate some visible portion of the current picture. This is done by positioning a rubber-banded box around the part of the picture to be isolated; the framed part of the picture is then expanded to fill the entire graphics window. To frame part of the picture, first make sure the *Frame* button is pressed. Move the mouse into the graphics window and press the middle mouse button where you wish to place one corner of the framing box. Keep the middle mouse button pressed and drag the mouse to a new position. A rubber-banded square box will appear. Release the middle mouse button to view a wire frame, a hidden-line, or a polygon fill picture of the contents of the framing box (depending on the setting of the *Wire, Hide, Fill* buttons). To abort a frame operation in progress, drag the mouse outside of the graphics window and release the middle mouse button.

Drawing a New Picture



It is sometimes necessary to tell **TrueGrid**[®] to redraw the picture. This is especially true in the Part Phase, where commands are buffered – all buffered commands are executed whenever a new picture is generated. Also, if the user changes the type of picture to be drawn (*Wire, Hide, Fill* option), a new picture is not automatically redrawn; the user needs to tell **TrueGrid**[®] to redraw the picture.

There are three basic ways to regenerate the picture. The first uses the *Draw* option. Invoking the *Draw* option (by pressing the *Draw* button with the left mouse button) causes the code to execute all buffered commands, and to generate a picture of the specified type (*Wire, Hide* or *Fill*) in the same position and orientation as the previous picture. The *Center* option is used to adjust the picture

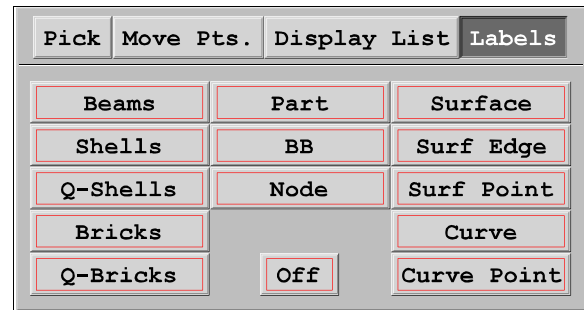
so that all objects are fully visible within the window. **Center** leaves the rotational orientation of the picture the same as for the previous picture. The **Restore** option also makes sure all objects are visible. However, the **Restore** command puts the picture back to the home (default) rotational orientation.

The **Center** and **Rest** commands are especially useful when objects are added or removed from the picture, or if the user gets lost in the picture.

Locating Objects Using Labels

Labels play an important role in interactive mesh generation. **TrueGrid**[®] allows the user to reference an object by pointing to its label. **TrueGrid**[®] never draws overlapping labels and, within this restriction, tries to center the label on the object. Furthermore, **TrueGrid**[®] highlights the object corresponding to a particular label when the user points to that label. (Assuming that the user has selected the **Label** button under the **Pick** panel.)

These features make it easy to find a particular object by its label, even when there are many objects in the picture.



Only one class of objects is labeled at any given time. The user selects the class of objects to be labeled by first selecting the **Labels** panel of the Environment Window by pressing the **Labels** button with the left mouse button. Then the user presses the button corresponding to the class of objects to be labeled. Pressing the **Off** button removes all labels.

Example Continued

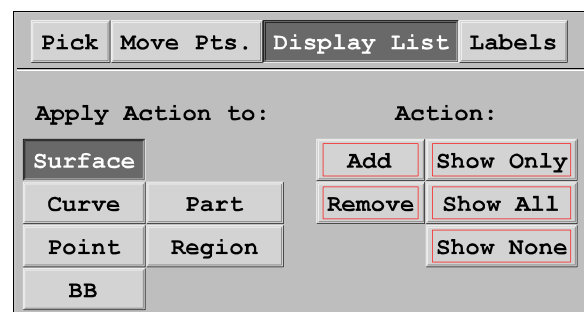
Press the **Surface** button to display labels for the plane and cylinder in the picture. Move the mouse into the Physical Window and press the left mouse button on one of the two labels in the picture. Pointing in this fashion to label 1 highlights surface 1, which is the cylinder. Similarly, pointing to label 2 highlights the sphere. Pressing the left mouse button anywhere else in the picture clears the highlighting.

Next press the **Surf Point** button. Points on the plane (used mainly for graphical purposes) are now labeled. **TrueGrid**[®] allows the user to attach points of the mesh directly to such surface points. Zooming in on part of the picture will reveal more labels and points that can be used while building the mesh.

The **Surf Edge** button will label the edges of all defined surfaces in the picture. Edges of the mesh can be directly placed on such surface edges. When surfaces are infinite, as in the example, they have no useable edges; their edges can be displayed and highlighted but cannot be used in interactive meshing. This is because edges of infinite surfaces change, depending on the size of the bounding box for the picture.

Selecting Objects For the Picture

The **Display List** panel of the Environment Window provides an interactive way to add and remove objects from the picture. Pressing the **Display List** button of the Environment Window activates this panel.



The group of buttons on the left is used to select the class of objects. The group of buttons on the right determine if objects are added or removed from the picture.

Removing an object from the picture is done by highlighting the object (using the label highlighting feature described above), and then by pressing the **Remove** button with the left mouse button.

To remove all objects except one, highlight the object to remain (using the label highlighting feature described above), and then press the **Show Only** button with the left mouse button.

Remove all objects of the specified class by pressing the **Show None** button with the left mouse button. Removing all surfaces, for example, makes the mesh easier to view.

Add all objects of the specified class to the picture by pressing the **Show All** button with the left mouse button.

The **Add** feature also works on selected objects. If **TrueGrid**[®] where in the part phase, then a click on the **Region** button and the **Show None** button would remove all components of the part from the picture. Using the Computational Highlighting System for the mesh in the Part Phase, it is possible to select a region of the mesh that is not in the picture. Such a region is then added to the picture by pressing the **Add** button with the left mouse button. Notice that the **Region** button is grayed out in the Merge Phase because regions of the mesh cannot be selected in the merge phase.

Controlling the Picture From the Command Line

All of the interactive functions presented so far (except for the frame feature) correspond to commands that can be issued from the text and menu window of **TrueGrid**[®]. On the other hand, there are graphics commands that have no interactive counterparts. For example, there is no interactive function to add a surface, curve, etc. to the picture.

Presented below is a list of **TrueGrid**[®] commands that are analogous to the interactive functions presented so far.

Generate a New Picture

Generate a line drawing `draw`
Generate a hidden-line drawing `disp`
Generate a polygon fill drawing `tvv`
Center the picture (do not change the rotational orientation) `center`
Center the picture in the default orientation (with no rotations) `rest` or `restore`

Rotate, Translate, Zoom

Rotate the picture about a horizontal line parallel to the screen plane `rx angle_in_degrees`
Rotate the picture about a vertical line parallel to the screen plane `ry angle_in_degrees`
Rotate the picture about a line perpendicular to the screen plane `rz angle_in_degrees`
Zoom Forward by a factor (that can be less than 1) `zf zoom_factor`
Zoom Backward by a factor (that can be greater than 1) `zb inverse_zoom_factor`

Label Objects in the Picture

Label surfaces `labels sd`
Label curves `labels crv`
Label surface edges `labels sledge`

Label points on surfaces `labels sdpt`
Label points on curves `labels crvpt`
Remove all labels `labels off`

Specify the Surfaces Shown

Display All Surface Definitions `dasd`
Remove All Surface Definitions `rasd`
Display a single Surface Definition `dsd surface_definition_number`
Display a list of Surface Definitions `dsds list_of_surface_definitions ;`
Add a Surface Definition `asd surface_definition_number`
Remove a Surface Definition `rsd surface_definition_number`

Specify the Curves Shown

Display All Curve Definitions `dacd`
Remove All Curve Definitions `racd`
Display a single Curve Definition `dcd curve_definition_number`
Display a list of Curve Definitions `dcds list_of_curve_definition_numbers ;`
Add a Curve Definition `acd curve_definition_number`
Remove a Curve Definition `rcd curve_definition_number`

Specify the Parts Shown

Display All Parts `dap`
Display one particular Part `p part_number`
Add a Part to the picture `ap part_number`
Remove a Part from the picture `rp part_number`

Specify the Materials Shown

Display All Materials `dam`
Display a specific Material `dm material_number`
Add a Material to the picture `am material_number`
Remove a Material from the picture `rm material_number`

Specify the Regions Shown

These commands apply only in the Part Phase. The distinction between a Region specification and an Index Progression specification are explained later.

Display All Regions `darg`
Display a specific Region `rg region`
Display a Region specified by Index Progression `rgi progression`
Add a Region to the picture `arg region`
Add a Region specified by Index Progression `argi progression`
Remove a Region from the picture `rrg region`
Remove a Region specified by Index Progression `rrgi region`

VI. Making a Single Block Part

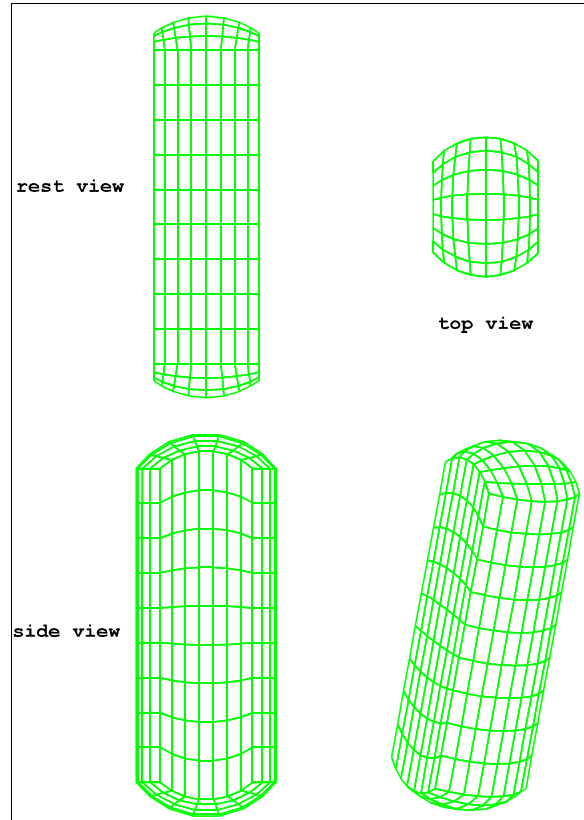
A single block part is created. Each of the six faces is projected to a different surface to form a rounded-end rod. The edges of the block are automatically placed along the intersections of the surfaces. The corners of the block are automatically placed at the intersections of three surfaces. The default interpolation takes care of the interior nodes.

Getting Started

Run **TrueGrid**[®] with no input file. Enter the Merge Phase by typing the command

```
merge
```

followed by a return. This allows you to view the geometry for this problem as it is being created.



The Geometry

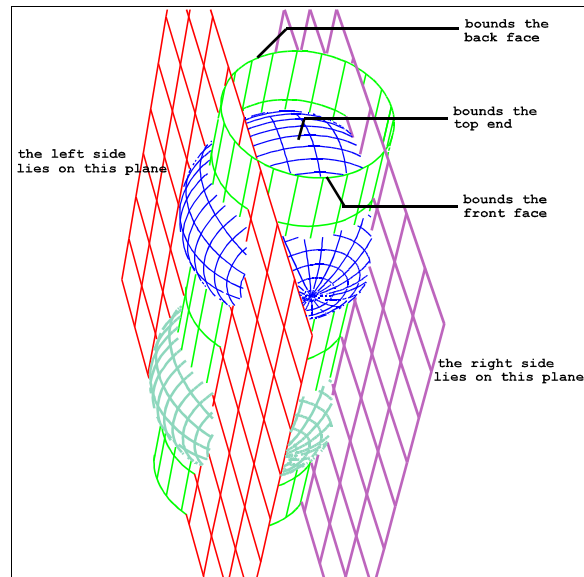
The mesh is determined by just a few surfaces. The commands to create the geometry are listed below.

```
sd 1 cyli 0 0 0 0 1 0 2
sd 2 plan -1.5 0 0 1 0 0
sd 3 plan 1.5 0 0 1 0 0
sd 4 sphe 0 -3 0 2.5
sd 5 sphe 0 3 0 2.5
```

As an experiment, enter only part of a given command before pressing Enter key. **TrueGrid**[®] will prompt you for the next required parameter of the command. You may enter as much of the command after the prompt as you wish (even part of the next command).

Pressing the ESCape key in the middle of a command aborts that command. Another way to correct any mistake is to simply redefine the surface.

Surface definition 1 is a right circular cylinder of radius 2 whose axis of rotation passes through (0,0,0) and is parallel to (0,1,0). Both the front and back faces of the rod will lie on this cylinder



Surface definition 2 is a plane passing through the point $(-1.5,0,0)$ and perpendicular to $(1,0,0)$. The left face of the rod will lie on this plane.

Surface definition 3 is a plane passing through the point $(1.5,0,0)$ and perpendicular to $(1,0,0)$. The right side of the rod will lie on this plane.

Surface definition 4 is a sphere centered at $(0,-3,0)$ and of radius 2.5. The bottom end of the rod will lie on this sphere.

Surface definition 5 is a sphere centered at $(0,3,0)$ and of radius 2.5. The top end of the rod will lie on this sphere.

The Block Command

The actual mesh is now created using the `block` command. At the `merge>` prompt enter

```
block 1 9; 1 10; 1 8; -2 2; -6 6; -2 2;
```

(The command can be entered on one line or split across lines as shown.)

The Part Phase

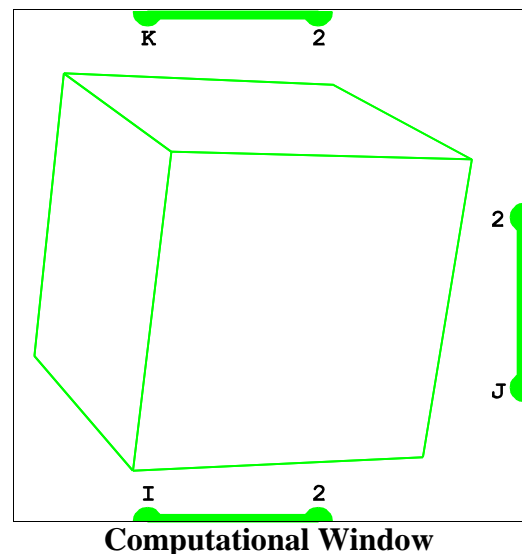
This block command creates a single block mesh with

9 nodes in the i-direction,
10 nodes in the j-direction,
8 nodes in the k-direction,
x-coordinates evenly spaced between -2 and 2,
y-coordinates evenly spaced between -6 and 6, and
z-coordinates evenly spaced between -2 and 2.

The `block` command switches you to the Part Phase. A new window with a title `Computational` appears in the upper right corner of the screen. The `Computational` and `Physical` pictures will not be drawn until you issue a `draw` command.

The `Computational Window` contains a block representation of the mesh. Along the lower, upper, and right borders of the window are index bars. These index bars provide a means by which the user can select any corner, edge or face of the block, as well as the solid block. Such a selection can then be used directly in any command requiring a description of part of the mesh as one of its arguments.

The usefulness of the computational window becomes apparent when creating complex three-dimensional arrays of blocks using the `block` command. In fact, all but the simplest meshes will tend to be multiple-block meshes.



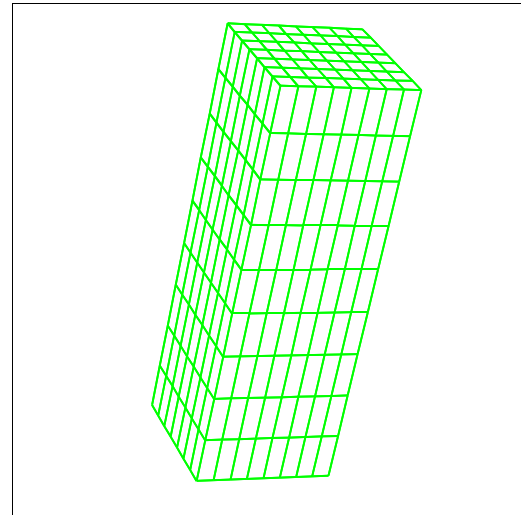
The Physical Window contains a picture of the actual mesh. To distinguish between geometry and mesh, mesh lines are green, surfaces are red, and curves are



Setting the Window

yellow. Of course, highlighting changes the color of the highlighted object. Any time the physical window is drawn, all buffered commands are executed and the actual mesh is displayed.

Because there are two graphics windows in the Part Phase, you must choose which windows are to be acted upon. This is done by pressing the left mouse button on either the *Physical*, *Computational* or *Both* button.



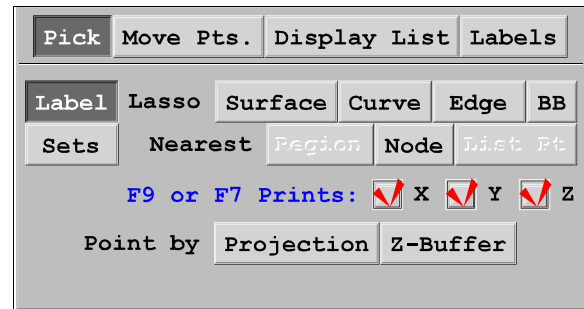
Physical Window

When either the *Draw*, *Center* or *Restore* buttons is pressed, the appropriate window is redrawn. The same is true of manually issued commands such as **rx**, **ry**, **rz**, etc. The user should also be aware that commands such as those to reset the angle of perspective are also directed to the window indicated by the setting of *Phys*, *Both* and *Comp* button group.

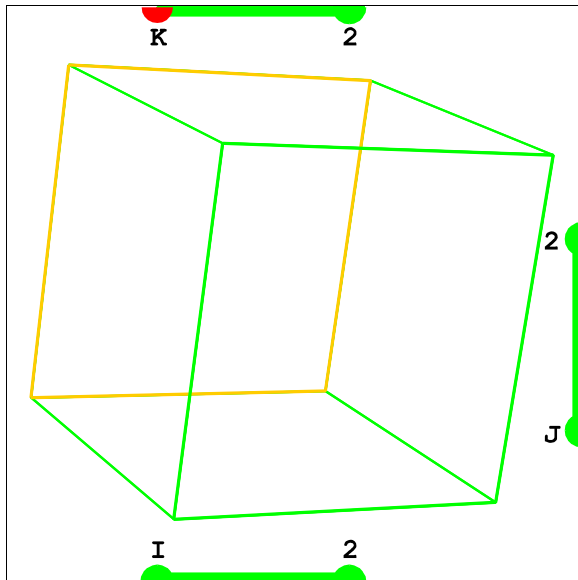
Selecting a Surface

Choose the *Label* button under the *Pick* panel to select surfaces by their labels.

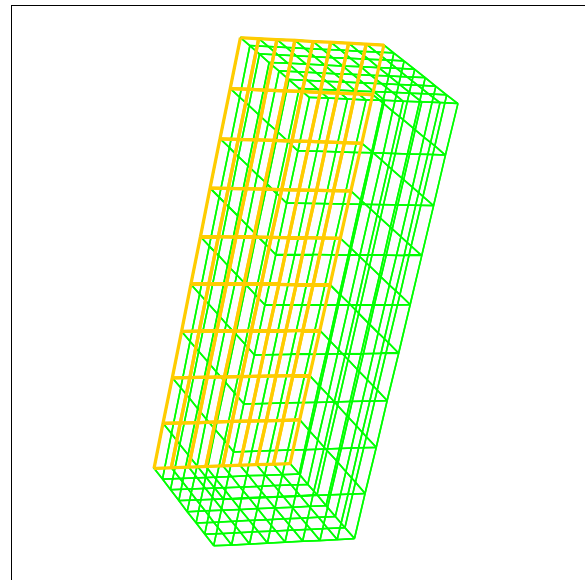
There are three different functions for the left mouse in the Part Phase Physical Window. The left mouse button can be used to choose objects by their labels, to choose the nearest mesh vertex, and to interactively move portions of the mesh. By clicking on the *Label* button, you have chosen the function for the left mouse button.



To finish the example will require a few interactive projections. An interactive projection requires the user to highlight a face of the mesh, and to select a surface. Select the cylinder by clicking on the appropriate label (this is surface definition 1).



The Minimum k Slicing Plane



Corresponding Physical Mesh

Highlight Faces of the Mesh and Project

The front and back faces of the mesh are to be projected to the cylinder that was just highlighted. These faces will be selected simultaneously.

The slicing plane feature is a fast way to identify a face in the Physical and Computational windows. It is activated whenever the mouse is moved into the Computational Window near one of the dots of the index bars. The dot near the mouse turns white, and a corresponding block face in the Computational and Physical Window are highlighted in white.

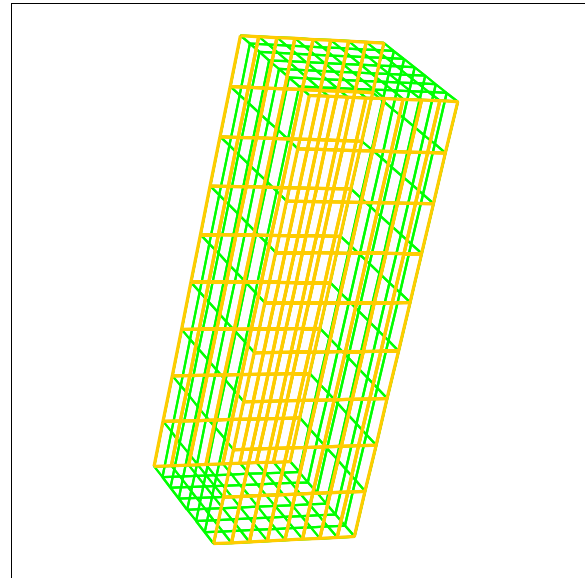
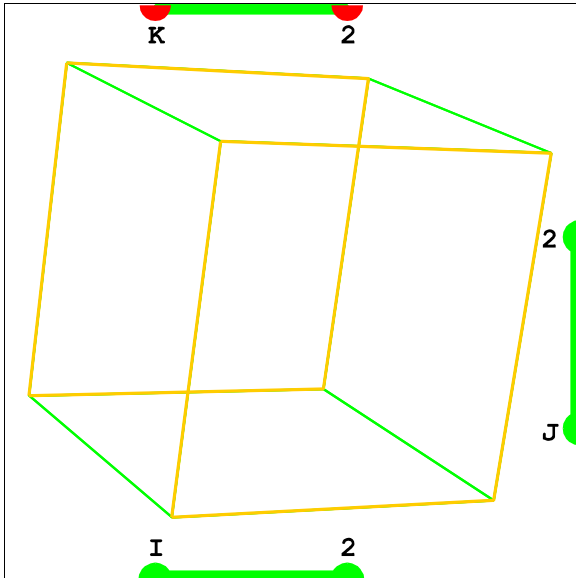
Dots on the lower index bar correspond to faces where the *i*-index is held constant; dots on the right index bar correspond to constant *j*-index faces; dots on the top index bar correspond to constant *k*-index faces. By convention, the actual values of fixed indices increase from left to right and bottom to top.

Now select the back face of the mesh by first highlighting that face (as shown above) and then by clicking the left mouse button while the minimum constant *k* slicing plane is highlighted. You must move the mouse to the top index bar, and near to the left most dot on the left end of the index bar. When you click on this dot, the back face of the computational block and the back face of the mesh will both turn yellow. This is the minimum *k* slicing plane.

If you make a mistake while making the selection, press the **F2** key to clear the selection.

Next move the mouse until the maximum *k* slicing plane is active (equivalently, when the right dot on the top index bar is white). The highlighted face of the mesh is the front face. Add this face to the current selection by pressing the left mouse button while the maximum *k* slicing plane is highlighted.

At this point, both the front and back faces of the mesh will be highlighted in yellow. The two dots of the upper index bar will be red. The cylindrical surface you highlighted earlier should still be



Front and Back Mesh Faces

highlighted.

Now press the *Project* button.



In case of a mistake, deactivate the projection command by pressing the *Undo* button in the Edit button group of the Environment Window. Pressing the *Undo* button will undo the last active meshing command. So pressing the *Undo* button many times will undo many meshing commands. (The Undo feature does not apply to graphics commands.) There are more sophisticated ways to selectively deactivate meshing commands. Such techniques will be discussed later.



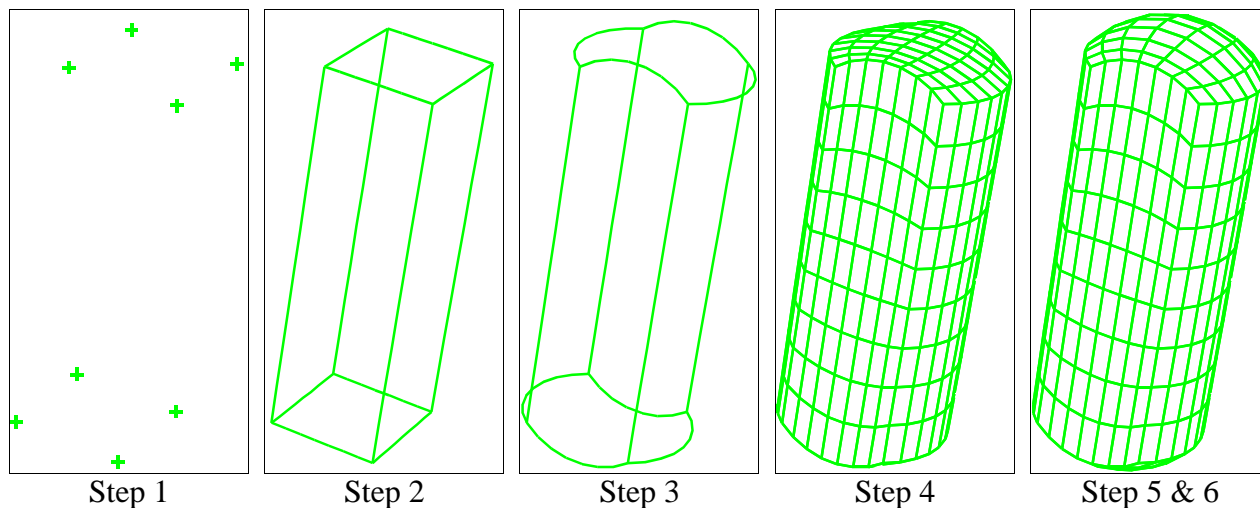
It should not be difficult to finish this problem. There are four other projection commands required.

1. The minimum i-face projected to the plane on the left (surface definition 2).
2. The maximum i-face projected to the plane on the right (surface definition 3).
3. The minimum j-face (bottom) projected to the bottom sphere (surface definition 4).
4. The maximum j-face (top) projected to the top sphere (surface definition 5).

Keep in mind that the **F2** key clears the computational selection, and that *Undo* will reverse the effects of any meshing command such as a projection.

VII. How TrueGrid[®] Built the Mesh

The previous example provides a wonderful illustration of the projection method. Here we describe the method. Snapshots of the mesh in various unfinished states are used for illustration. You never actually see these intermediate states of the mesh.



Step 1: Each corner of the mesh is placed at the intersection of the 3 surfaces. The picture for this step only shows the locations for these 8 corners (referred to as the vertices) of the block mesh. At this stage, none of the other nodes on the block have valid coordinates, so they cannot be shown.

Step 2: The nodes that form the edges between the corners (or vertices) are interpolated along a line. The location of the nodes cannot be determined from the picture for this step because the interior nodes of an edge are equally spaced along the line between the two end vertices.

Step 3: The edges of the blocks are now projected to the surfaces on which they must lie. In this example, all edges are required to be on two different surfaces. The nodes are equally spaced along the curve of intersection of the two appropriate surfaces. It is difficult to detect the location of the interior nodes at the ends of the line segments which connect the nodes.

Step 4: Using the edges that form the boundary of a face, all the interior nodes of each face of the mesh are interpolated. Many interior face nodes end up in their final positions after this interpolation step, without having to be projected, because the linear interpolation is based upon the shape of the bounding edges of the face. For example, the faces that must lie on planes have edges that already lie on planes and, consequently, the linear interpolation put all of the interior nodes of the faces onto the planes. However, the top and bottom ends of the rod are not correctly positioned. The top and bottom ends are required to be on spheres.

Step 5: The interior nodes of faces of the mesh are projected onto the surfaces on which they are required to lie.

Step 6: As a final step, the interior nodes of the block are interpolated using the positions of the nodes on the 6 faces. The mesh is complete. The picture is the same as step 5 because the interior nodes of the block are not visible.

VIII. The History Table: Turning Commands Off and On

A list of mesh commands for the part being generated can be found in what is referred to as the history table. All the commands in the history table can be viewed by pressing the *History* button in the **Edit** button group of the Environment Window.



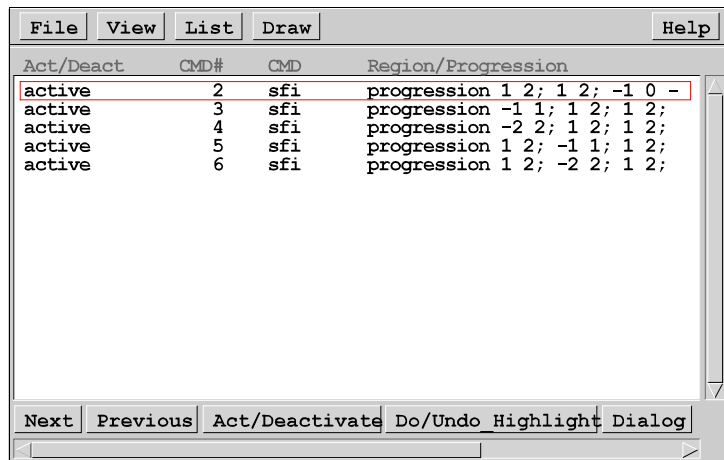
The history is also summoned by entering the `history` command at the text window prompt.

Highlighting the Part of the Mesh to which a Command Applies

Click on one of the lines in the history window with the Middle Mouse Button just below the header

Region/Progression

You have selected a region or progression of the mesh associated with a command in the history table. Notice that the mesh in both the computational and physical windows are highlighted. This highlighted region can be used in subsequent commands.

A screenshot of a software window titled 'History Table'. It has a menu bar with 'File', 'View', 'List', 'Draw', and 'Help'. The main area contains a table with columns 'Act/Deact', 'CMD#', 'CMD', and 'Region/Progression'. The first row is highlighted with a red border. Below the table is a toolbar with buttons: 'Next', 'Previous', 'Act/Deactivate', 'Do/Undo', 'Highlight', and 'Dialog'.

Act/Deact	CMD#	CMD	Region/Progression
active	2	sfi	progression 1 2; 1 2; -1 0 -
active	3	sfi	progression -1 1; 1 2; 1 2;
active	4	sfi	progression -2 2; 1 2; 1 2;
active	5	sfi	progression 1 2; -1 1; 1 2;
active	6	sfi	progression 1 2; -2 2; 1 2;

The history table can be used to debug a mesh. There are many features that you will learn as your understanding of the projection method improves. This feature to highlight the portion of the mesh that a command applies to can be used to locate a command in the table by the region it is applied to.

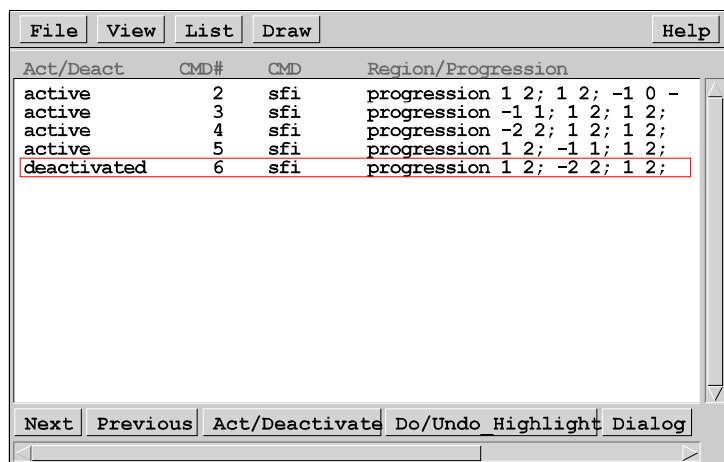
Deactivating and Reactivating

Once you have located a command in the table, you can deactivate that command. This has the same effect as the Undo button, except you can select any command to deactivate, not just the last one. To deactivate a command, click the middle mouse button on the word `active` associated with the command you wish to deactivate. The word `active` will be changed to `deactivated`.

Example (cont.)

Use the history highlighting feature to isolate the command requiring the top end of the rod to be projected. The progression line of that command appears as

```
progression 1 2;-2 2;1 2;
```

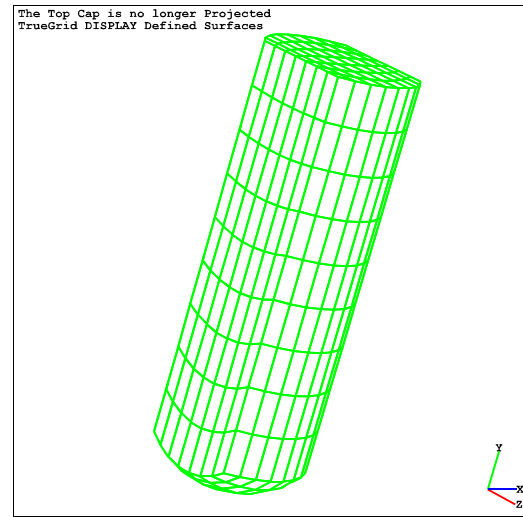
A screenshot of the 'History Table' window, similar to the previous one, but the sixth row is highlighted with a red border and the word 'active' in the first column has been replaced by 'deactivated'.

Act/Deact	CMD#	CMD	Region/Progression
active	2	sfi	progression 1 2; 1 2; -1 0 -
active	3	sfi	progression -1 1; 1 2; 1 2;
active	4	sfi	progression -2 2; 1 2; 1 2;
active	5	sfi	progression 1 2; -1 1; 1 2;
deactivated	6	sfi	progression 1 2; -2 2; 1 2;

On the same line click the middle mouse button on the word "active". A picture is not automatically regenerated, so press the Draw button in the environment window.

Notice that the top end of the bar is no longer projected to the sphere and that the history table reports the status of the command you selected to be "deactivated".

Reactivate the command by clicking the middle mouse button on the word "deactivated". Again, draw the picture to see that the top end of the rod is back on the sphere.



Command Deactivated

IX. Interactively Changing the Mesh Density

The `mseq` command is used to change the mesh density. For example, the block command

```
block 1 9; 1 10; 1 8; -2 2;-6 6;-2 2;
```

creates the same mesh as these commands

```
block 1 2; 1 2; 1 2; -2 2;-6 6;-2 2;  
mseq i 7 mseq j 8 mseq k 6
```

The first `mseq` command adds 7 nodes in the *i*-direction. The next `mseq` command adds 8 nodes in the *j*-direction. The last `mseq` command adds 6 nodes in the *k*-direction. The `mseq` command can also be used to remove nodes by using a negative number.

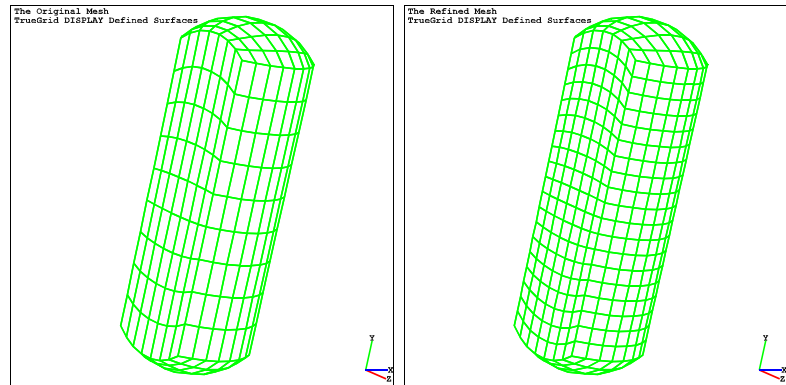
Refining the Example Mesh

To illustrate how easy it is to refine the mesh, try the command

```
mseq j 10
```

To view the change, click on the ***draw*** button. To undo this, try the command

```
mseq j -10
```



The Original Mesh

After the MSEQ Command

X. Clustering of Nodes (Zoning)

The **Relative Spacing Command (res)** is the simplest command used to cluster nodes along an edge of the mesh.

SYNTAX: `res i_min j_min k_min i_max j_max k_max direction ratio`
where *direction* can be either *i*, *j*, or *k*, and *ratio* is ration of element widths.

DESCRIPTION: The nodes of all block edges of a specified region of the mesh are clustered in the specified direction. The amount of clustering depends on the ratio. The nodes are spaced so that the ratio of distances between adjacent nodes in the specified *direction* equals the given *ratio*.

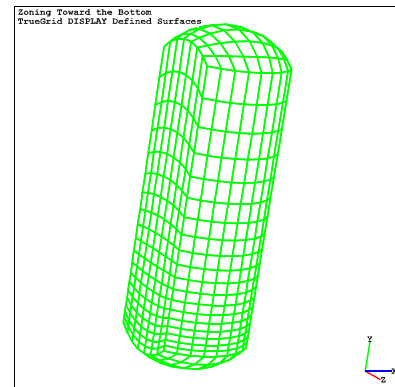
If spacing is specified in the *i* direction with a ratio of 1.2, then the nodes with smaller *i*-index values are closer together than those with larger *i*-index value. In particular, for a fixed *j* and *k* index, the distance between the nodes with *i*-index 2 and 3 will be 1.2 times as large as the distance between nodes with *i*-index 1 and 2. So in this case, nodes "cluster" in the negative *i*-direction.

Example (cont.)

To cluster the nodes toward the bottom end of the bar, issue the command

```
res 1 1 1 2 2 2 j 1.1
```

To avoid typing the region description, first clear the computational highlighting by pressing the **F2** key. Then type `res` on the command line, and press the **F1** key. A description of the highlighted region is put onto the command-line following `res`. Because a null selection makes no sense in a command, **TrueGrid**[®] converts the null selection to a description of the *entire mesh*. Finish the command by entering `j 1.1`.



Zoning in the j-direction

XI. Clustering of Nodes (Zoning) in Opposite Directions

You can cluster nodes at both ends of an edge by using the **Double Relative Spacing Command (drs)**.

SYNTAX: `res i_min j_min k_min i_max j_max k_max direction ratio1 ratio2`
 where *direction* can be either *i*, *j*, or *k*
 ratio1 is the ratio of element widths at the start
 ratio2 is the ratio of element widths at the end

DESCRIPTION: The nodes of the specified region are clustered in a given direction. Zoning near the minimum index of the given direction is controlled by the first ratio. Zoning near the maximum index of the given direction is controlled by the second ratio. If the second ratio is the reciprocal of the first, then `drs` works the same way as `res`.

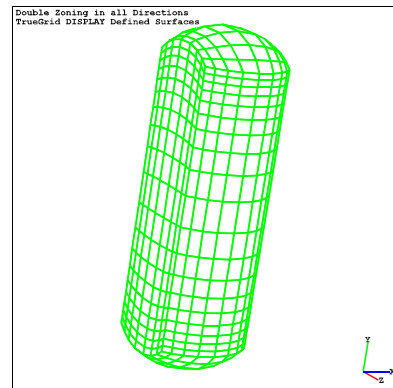
Example (cont.)

The `drs` command is particularly useful for fluids problems where nodes need to be clustered toward the walls of a mesh in order to resolve the boundary layer. To achieve such clustering for the current example, issue the three commands

```
drs 1 1 1 2 2 2 i 1.2 1.2
drs 1 1 1 2 2 2 j 1.2 1.2
drs 1 1 1 2 2 2 k 1.2 1.2
```

To avoid typing the region each time, use the same procedure described in the example of the `res` command.

NOTE: If you have already issued the `res` command of the previous example, there is no need to deactivate that command. Zoning commands for a region override previously issued zoning commands for that region.



Double Zoning

XII. Saving and Rerunning the Session File

A **TrueGrid**[®] session file is written every time that **TrueGrid**[®] is run. If no session file name is specified on the command line (using the `s=` option), then the session file is named **tsave** and is placed in the current working directory. If you are running **TrueGrid**[®] under WINDOWS (other than from a command prompt window), you do not have the option to name the session file: it will always be named **tsave**. If the session file is named **tsave**, then make sure to rename the session file when finished, or it will be over-written after running **TrueGrid**[®] three more times.

Rerunning the Session File

A session file can be rerun as an input file. On a UNIX or LINUX system or on WINDOWS running from a command prompt window, using the `i=filename`. If you are running **TrueGrid**[®] in WINDOWS without the aid of a command prompt window, you can browse to select the command file. Running the session file as an input file is a good way to pick up where you left off the last time **TrueGrid**[®] was run. To run the session file as an input file, first remove the final command used to exit **TrueGrid**[®] (which can be `end`, `exit`, `adios`, `quit`). Otherwise **TrueGrid**[®] will exit without pausing for an interactive session. (There are times when you may wish to rerun the session file without any interactive session. In this case, leaving an exit command in the file is appropriate.)

When **TrueGrid**[®] is finished processing the input file, the user interface becomes active. A prompt appears in the text/menu window to indicate this fact. **TrueGrid**[®] will not automatically draw a picture unless a draw command has been placed in the input file.

The new session file will contain the lines from the original input file as well as any commands issued interactively after processing the input file.

Mixing Batch and Interactive Processing

You can interrupt an input file at any point by placing **interrupt** command in the input file. When **TrueGrid**[®] reaches the **interrupt** command, the user interface becomes active. A prompt appears in the text/menu window (upper left corner) to signal this fact. To process all the input file commands up to the next interrupt (or to the end of input file, whichever comes first), issue the command **resume** command. Alternatively, if **TrueGrid**[®] is stopped in either the Part or Merge Phase, then the **Resume** button in the Environment Window can be used (the Environment Window does not exist in the Control Phase).



The new session file produced will contain all the interactive and input file commands, in the order in which they were issued but the **interrupt** and **resume** commands are removed.

XIII. The Session and Input File Format

The session file is an ASCII file that is easily edited. Only the first 256 characters of a line are considered by **TrueGrid**[®]. Within a session file it is permissible to

1. Use upper or lower case letters,
2. Insert a comment anywhere,
3. Use any format for numbers,
4. Insert extra spaces anywhere,
5. Break a command across lines,
6. Insert graphics commands, or
7. Insert interrupt commands.

Comments are placed in the input file by placing a single **c** (or **C**) either at the beginning of the line or sandwiched between at least one space. All characters after the **c** and on the same line are treated as part of the comment. A comment may be inserted in the middle of a command. For example, **TrueGrid**[®] has no problem processing

```
c Create the Block Part
  block
  c Determine the Node
  c Distribution
    1  9;  c i-list
    1 10;  c j-list
    1  8;  c k-list
  c Specify the Coordinates
    -2  2;  c x-coordinates
    -6 -6;  c y-coordinates
    -2  2;  c z-coordinates
```

TrueGrid[®] uses a flexible format for numbers as well. For example, if **TrueGrid**[®] is looking for a floating point number, then any of the following will work (and will be treated equivalently):

```
1.0
.10E+01
.10e1
10.0E-01
1
```

If **TrueGrid**[®] is looking for an integer, any of the above will be interpreted as 1.

TrueGrid[®] also understands FORTRAN like expressions, so long as they are enclosed by square brackets. All FORTRAN intrinsic functions are supported, including trigonometric functions and their inverses. All angles are assumed to be in degrees. For example, these are valid floating point numbers:

```
[tan(atan2(2,1))*3]
[(-2)*(-3)*4*5/4+3]
[sqrt(2)*sqrt(2)]
[2.3**2.5]
```

Parameters are defined using the **para** command and are referenced by preceding the parameter name with a % sign. The syntax for the **para** command calls for pairs of items (parameter name followed by a value) followed by a terminating ;. For example, after the command

```
para x1 10 x2 12 x3 24;
```

then

```
[sqrt(%x1*%x2+%x3)]
```

is a valid expression whose value is 12. Note that parameters can be used as soon as they are defined. In particular,

```
para x1 10 x2 [2*%x1];
```

results in a value of 20 for the parameter named x2. A single parameter can be referenced without brackets. For example, **TrueGrid**[®] understand %x1 by itself, but it does not understand -%x1. [-%x1] is, however, understood by **TrueGrid**[®].

XIV. A Multiple-Block Part

A multiple-block mesh is constructed of the space inside a box and outside a sphere contained within the box. This mesh is created with just four commands that:

1. Create the multiple-block part,
2. Define the Sphere,
3. Make a hole in the mesh where the spherical cavity will be,
4. Project the faces of the hole onto the sphere.

Creating the Part

Run **TrueGrid**[®] with no input file. Instead of entering the Merge Phase as before, this time enter the block command straightaway.

```
block 1 5 9 13; 1 5 9 13; 1
5 9 13;
-3 -1 1 3; -3 -1 1 3; -3
-1 1 3;
```

This command creates a 27-block part. The block boundaries are shown above. The syntax of the block command is

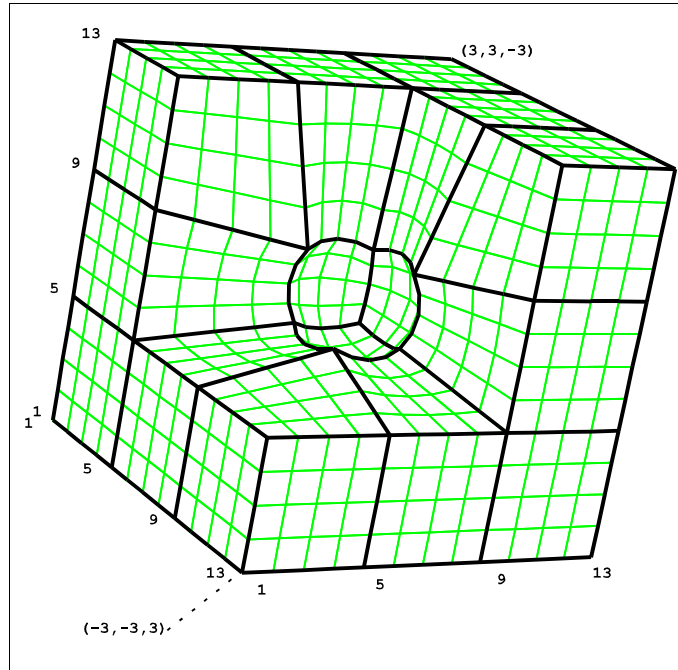
block *i-list* ; *j-list* ; *k-list* ; *x-list* ; *y-list* ; *z-list* ;

The *i-list*, *j-list* and *k-list* must be terminated by a semicolon (;). In this case, *i-list* consists of the numbers 1, 5, 9, 13, and *x-list* consists of the numbers -3, -1, 1, 3. If intermediate numbers are used in the *i-list* (5 and 9 are intermediate in this case), then **TrueGrid**[®] will subdivide the mesh in the *i*-direction at each place where the *i*-index is equal to one of these intermediate numbers. The corresponding intermediate values in the *x-list* are used to set the *x*-coordinates of the nodes at the subdivisions. An analogous statement applies to the *j*- and *k*-directions.

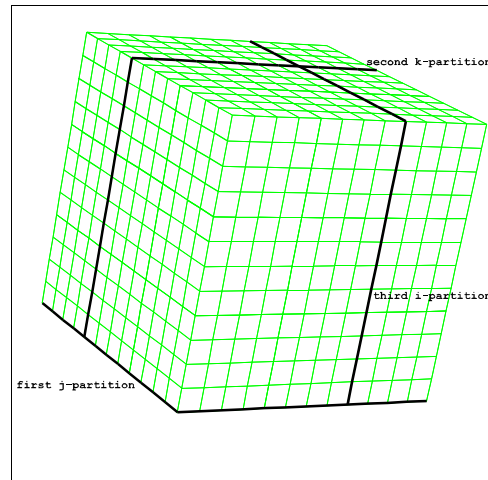
Terminology

If *n* is a number in the *i-list*, then the face where the *i*-index is held constant at *n* is called an ***i*-partition**. Similarly, a ***j*-partition** is a face where the *j*-index is held constant and equal to a number in the *j-list* of the block command. And, a ***k*-partition** is a face where the *k*-index is held constant and equal to a number in the *k-list*.

The mesh partitions shown here can be highlighted using the slicing plane feature.



Cut-A-Way View of the Final 27-Block Mesh



Sample Mesh Partitions

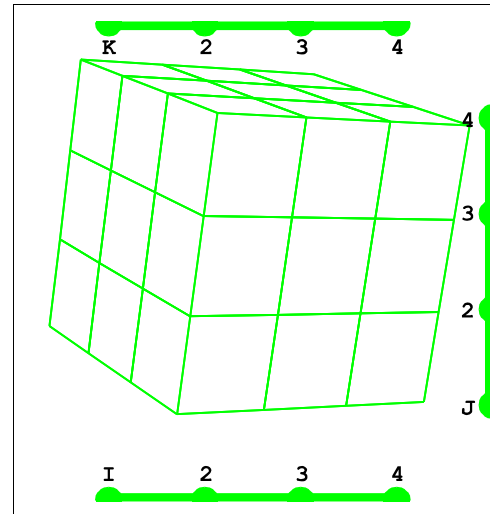
The Computational Window always contains just the partitions created using the `block` command. The computational partitions are always shown a fixed distance from each other and, therefore, each block in the computational picture is always a cube. For the current example, the computational picture contains 27 cubes (the user must issue a draw command to see the initial computational and/or physical pictures).

For the previous single-block example, no intermediate numbers were specified in either the *i-list*, *j-list* or *k-list* of the `block` command. Therefore, there were just two partitions in each direction, leading to a computational representation which was a single cube.

Each of the 27 blocks of the current mesh are treated as though they were single block meshes.

In particular, it is not possible to delete only part of a block. It is not possible to project only part of a block face onto a surface. It is not possible to zone just part of a block edge.

Furthermore, the same interpolation used to position the nodes within a single-block mesh is used here: The coordinates of corner nodes of blocks are used to interpolate edge node coordinates. Edge node positions are then used to interpolate face nodes. Face node positions are used to interpolate interior nodes. So, even though only corner node coordinates of the 27 blocks were specified using the `block` command, all nodes of the mesh have coordinates assigned to them.



Computational Representation

Because of the way that individual blocks are treated, any unaltered mesh created using the `block` command will always consist of rectangular boxes whose nodes are equally spaced in the *i*-direction, equally spaced in the *j*-direction, and equally spaced in the *k*-direction. However, globally this need not be true because adjacent blocks do not necessarily contain the same number of nodes. All that can be said in general is that all blocks between two adjacent *i*-partitions have the same number of nodes in the *i*-direction, etc.

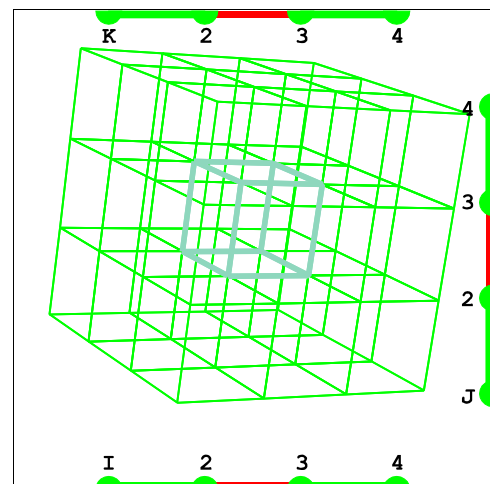
Shaping the Mesh

First create a sphere of radius 1 centered at the origin using the `Surface Definition (sd)` command.

```
sd 1 sphe 0 0 0 1
```

This sphere will not be visible using the *Hide* option because the sphere is fully contained within the physical mesh.

Second, delete the center block of the 27-block part. This is easy to do using the Computational Highlighting System. To do this, first highlight the solid center block as follows: Move the mouse into the computational window close to the second dot on the *i*-index bar (bottom window border). Press the left mouse button and drag the mouse to the third dot on the *i*-index bar. As you do this, the middle segment



Highlight The Center Block

of the i-index bar turns red, and all blocks between the second and third i-partitions are highlighted in cyan. Release the left mouse button. Do the same in the j-direction using the j-index bar (right border of the window). This time only the blocks between the second and third i-partitions and those between the second and third j-partitions are highlighted in cyan. Do the same in the k-direction. Now only the middle block is highlighted (use the **Wire** option to see inside). Note: Pressing the F2 key clears all highlighting.

Next, press the **Delete** button in the Edit button group of the Environment Window.



Recall that meshing commands can be deactivated using the **History** feature. Alternatively, use the **Undo** button to deactivate the last active command.

Notice that after the new pictures are drawn, the highlighting of the center block has changed from cyan to magenta. This is a feature of the computational highlighting system: When part of a highlighted solid region is deleted, then faces of the deleted region contained within the solid are colored magenta. This always happens when you select a region that has been, at least, partially deleted. It is a warning that you may have selected the wrong region.

Another unique feature of the computational highlighting system is that any combination of the 6 faces of a highlighted block can be selected in one step. As soon as one dot on a highlighted segment is activated, then the highlighting switches from solid to faces. Faces are highlighted in yellow as opposed to cyan for solids.

Notice the two activated dots on the i-index bar in the above picture. Activating these dots highlights the two corresponding i-faces of the center block. Similarly, any or all of the remaining 4 faces of the center block can be added to the current selection.

We want to project all six faces of the center block onto the sphere. So highlight all six faces by activating all six endpoints of the activated segments.

Now everything is set up to project. Instead of using a label to reference the sphere, try the command-line approach instead. First enter

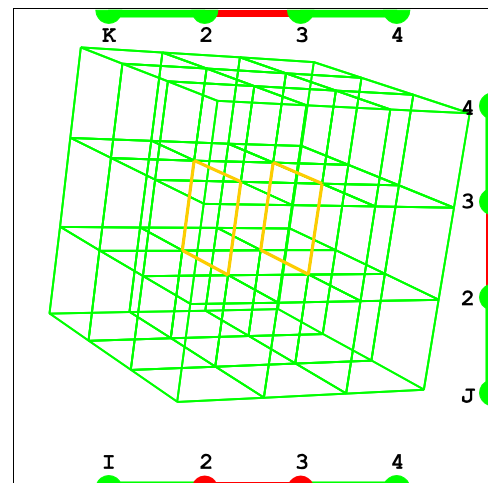
```
sfi
```

on the command-line (in the text/menu window). Then press the **F1** key to translate the highlighted selection into a numerical description of the selection. The numerical description will be appended to the current command line to give

```
sfi -2 -3;-2 -3;-2 -3;
```

The 2's and 3's indicate the partition numbers (second and third partitions in each direction). The negative signs indicate that the dots at 2 and 3 in all directions were turned on. If the segments connecting the dots had not been activated, then a 0 would have been inserted between the -2's and -3's.

Finish the command by adding `sd 1` to the command-line and by pressing the **Enter** key. The final command is



Two Faces of The Center Block

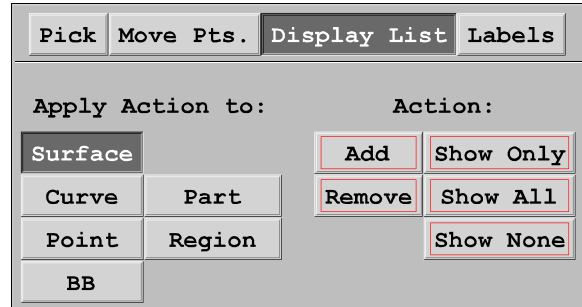
```
sfi -2 -3;-2 -3;-2 -3; sd 1
```

You must issue a draw command to view the results. Meshing commands issued from the command-line, or using dialogue boxes, do not cause the mesh to be automatically displayed. This way the experienced user can issue many commands before viewing the next picture. Furthermore, **TrueGrid**[®] does not have to recompute the mesh until a draw command is issued.

Peel Away Outer Layers

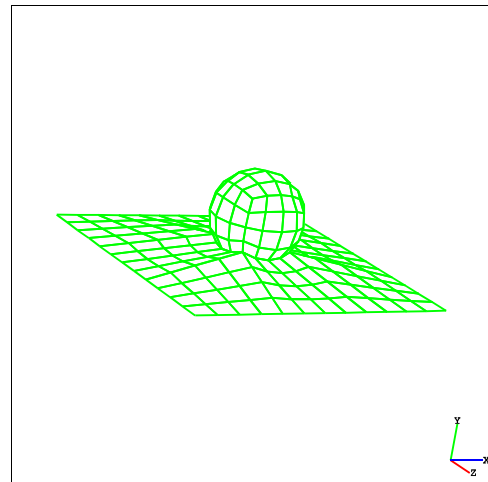
Before clearing the highlighting, try out the graphics peel commands. Select the Display List dialogue within the Environment Window. This is the same dialogue used to add and remove surfaces.

Press the Region button under the "Apply Action to:" category. Then press the **Show Only** button under the "Action:" category. **TrueGrid**[®] now shows only the highlighted selection. (Nothing has been deleted, only removed from the picture.) Because the current selection consists of the faces of the hole projected to the sphere, only these faces are shown.



Now clear the highlighting by pressing the **F2** key. Activate the second dot on the j-index bar (corresponding to the second j-partition). While the **Region** button is still pressed, press the **Add** button in the "Action:" category.

By activating only the second dot on the j-index bar, you selected the entire second j-partition. Then by pressing the **Add** button, you added this entire partition to the picture. Notice how easy it is to point to a region that is not in the picture and add it to the picture. With no other object is this possible. For in order to point to another type of object, that object must already be in the picture.



Only Part of the Mesh

About the Mesh

Each of the 27 blocks is built the same way a single-block mesh is built.

First the corners of the block are positioned. Some corners are required to be on the sphere and others are not. Those required to be on the sphere are projected to the closest point on the sphere relative to the corners initial position. After all corners are positioned, then block edges are interpolated. This amounts to evenly distributing the edge nodes along a line connecting the corners. Edges required to be on the sphere are projected to the sphere. Next, edge nodes are used to interpolate face nodes. Faces required to be on the sphere are then projected to the sphere. Finally, interior nodes of each block are interpolated from the face nodes.

Notice that several blocks have some edges that are straight and others that are curved. The faces interpolated from these edges are not simple surfaces (i.e., non-planar).

The TrueGrid® Challenge

This mesh was built with just FOUR commands. Try building a similar mesh using another mesh generator. Compare the time it takes and the mesh quality. It could be argued that this problem is ideally suited to demonstrate TrueGrid®. The truth is somewhere in the middle because a more complex model has many small problems, each of which can be solved using these techniques just demonstrated.

Mesh Diagnostics

The mesh diagnostics can be used in the Part or Merge Phase. For this exercise, change to the merge phase to measure the quality of the mesh. The commands

```
endpart merge
```

will end the part and switch TrueGrid® to the Merge Phase.

The computational window no longer exists. Also, nodes may be eliminated (merged) in this phase, thereby destroyed the correspondence between the computational and physical domains. No functions requiring the computational highlighting system are available, including the peel commands. Consequently, the entire mesh is displayed after a draw command.

The MEASURE Command

The *MEASURE* command is in the DIAGNOSTICS submenu. Move to that submenu and locate the *MEASURE* command. Press the left mouse button on the *MEASURE* button and a dialogue box appears.

In the measure dialogue, choose the Orthogonal Test option by pressing the left mouse button anywhere on the string. Execute the dialogue by pressing the *EXECute* button.

A new window whose title is *2D Curves* appears. In that window is a plot of the deviation angle (from 90 degrees) vs. the number of segments. The tallest peak is near 0, meaning that a substantial number of elements have nearly perfect 90 degree angles.

The absolute range of the test is printed in the text window:

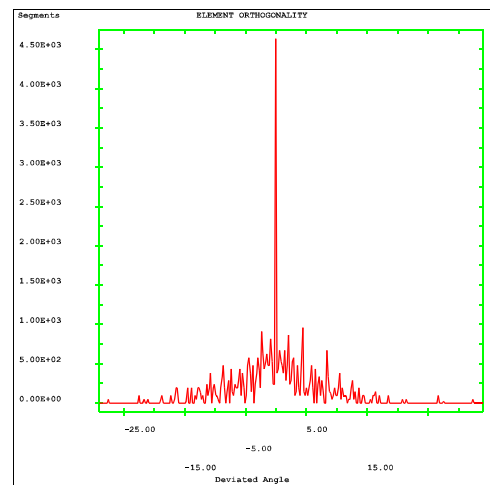
```
measure ranges from  
-2.71598E+01 to 3.249846E+01
```

indicating that the element angles are roughly between 62.5 and 122.5 degrees.

The ELM Command

The *elm* command is in the same DIAGNOSTICS submenu as the measure command. The *elm* command allows you to isolate elements whose measure is within a given range. Either use a dialogue box to specify a range between -28 -25, or directly issue the command

```
elm -28 -25
```



Orthogonality Test

All the elements with angles between 62 and 65 degrees will be highlighted magenta. The part can be removed from the picture using the command

rp 1

Then only the selected elements remain in the picture. Before trying another diagnostic command, add the part back to the picture

ap 1

This is necessary because the measure command only measures those elements that the user has specified to be in the picture. That is not to say only elements within the frame of the window are measured by this command, but all elements that are currently active and selected by the user.

To kill the 2D Curve Window, use the window manager "kill" or "quit" option for the window.

XV. Putting a Square Peg into a Round Hole

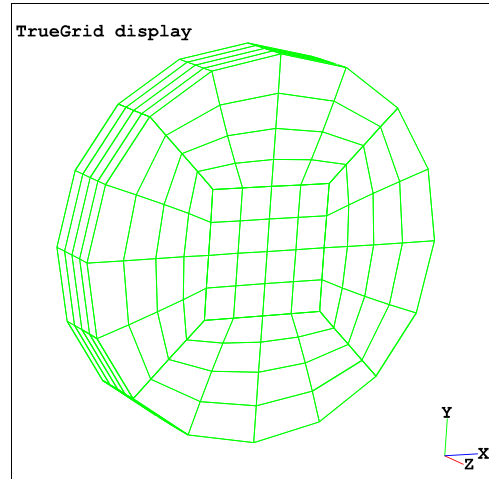
A technique is introduced for creating a good mesh of a rounded object using a block mesh in Cartesian coordinates. The mesh is created using four commands.

Creating the Part

Run **TrueGrid**[®] with no input file. Create a 9-block part using the block command:

```
block 1 5 9 13; 1 5 9 13; 1 5;  
      -1 -1 1 1; -1 -1 1 1; 0 1;
```

The resulting mesh contains several collapsed blocks. The first and second i-partitions have the same x-coordinates. Likewise, the second and third i-partitions have the same x-coordinates. An analogous statement is true for the j-direction and y-coordinate. All in all, only the center block of the mesh is not collapsed. A good way to see that some regions are collapsed is to first draw both pictures and then explore the mesh using the Slicing Plane feature.



The Butterfly Mesh

The Method

Notice that in the final mesh shown above there are only five blocks. The center block is easy to spot. Four other blocks surround this center block.

This mesh was created by first deleting the four corner blocks of the mesh. Then the outside faces of the four outside blocks were projected (in one command).

The Geometry

The geometry for this problem consists of a single right circular cylinder created with the following Surface Definition command:

```
sd 1 cyli 0 0 0 0 0 1 3
```

(Extra spaces were added for the sake of clarity – you may omit these when entering the command.) This creates a cylinder of radius 3 whose axis passes through (0,0,0) and is parallel to (0,0,1).

Cutting Corners

All four corner blocks can be highlighted and deleted at once. Do this by activating the first and last segments of both the i- and j-index bars (these are on the lower and right window borders, respectively). Then press the **Delete** button in the environment window.

Projecting to the Cylinder

Clear the current computational selection by pressing either the **F2** key or CTRL+d. Now activate all segments on both the i- and j-index bars. Activate all four endpoint dots of these index bars. This selection consists of four block faces.

Use the *Labels* dialogue of the Environment Window to label the surface. Use the *Pick* panel of the Environment Window to configure **TrueGrid**[®] to choose objects by label. Highlight the cylindrical surface by pressing the left mouse button on the label for this surface.

Finally, press the *Project* button in the Environment Window to project the four highlighted faces to the cylinder.

The mesh is finished, except for removing the redundant nodes on the block faces. To do this, end the part and change to the Merge Phase by issuing the commands

```
endpart merge
```

The redundant nodes are merged using the exterior. **Surface Part Tolerance** command

```
stp 0.01
```

View the merged nodes within the part using the graphics command

```
labels TOL 1 1.
```

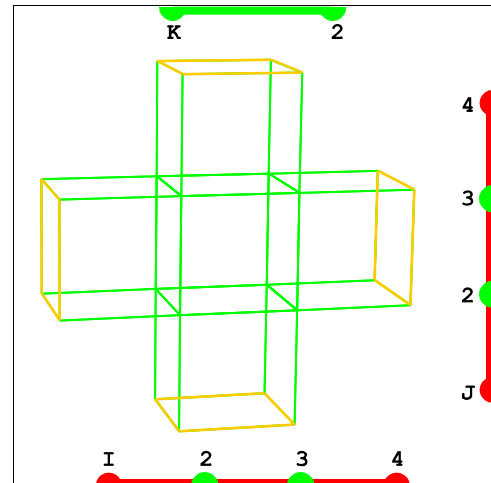
The Reason for the Butterfly Mesh

The alternative to the butterfly mesh involves just projecting the blocks to the cylinder. Actually, the mesh could be a single-block mesh. Such a mesh is shown on this page.

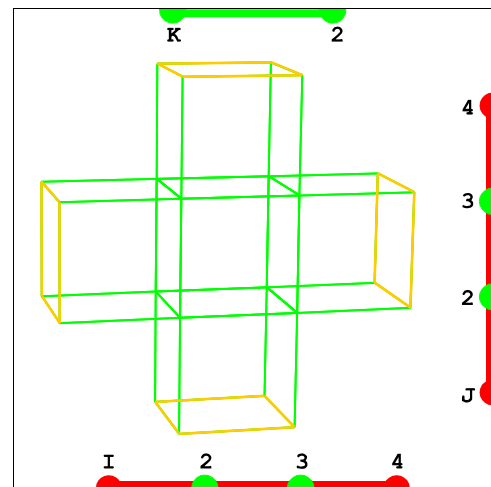
Pay particular attention to the region near the corners of the block. The angles of elements in this region are not good.

One of the corners of the block has been darkened to illustrate what goes wrong with a mesh of this type.

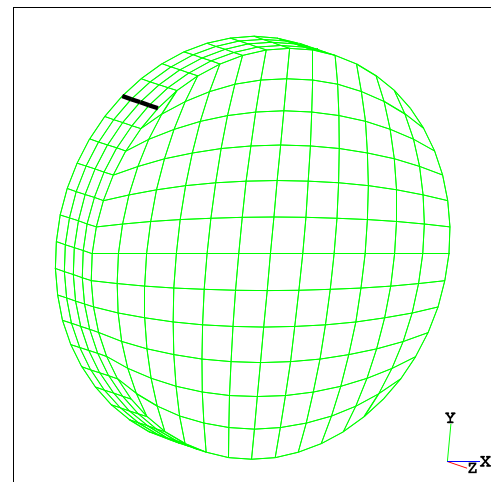
The outer two faces of the block at a corner of the mesh must be folded back to conform to the cylinder. The result is that the elements near this corner have angles that are nearly 180 degrees.



Delete the Four Corner Blocks



Faces Projected to the Cylinder



A Square Peg in a Round Hole

If the mesh is refined, then the problem just gets worse. This is because two points on a circle that are very close are connected by a segment whose slope is nearly the same as a tangent to the circle. Therefore, as the mesh is refined, the elements at the corner end up with two faces that are very nearly tangent to the cylinder. In other words, the two faces of the corner elements align themselves with a tangent plane to the cylinder (and, thus, with each other).

In general, the fact that the mesh quality actually goes down as the mesh is refined is a classic symptom of a bad topology. In these cases, the user should consider a different topology for the mesh.

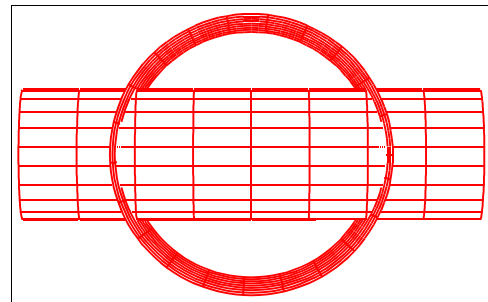
XVI. Intersecting Pipes Example

In this example, the fluid inside two intersecting pipes will be meshed with two parts. This is an example with detailed explanations in the use of the Graphical User Interface. It is also a good example to demonstrate the use of the projection method and block boundary interfaces.

Run **TrueGrid**[®]. You will have a single window showing. This is the Control Phase window. You will want to see the surfaces as you create them, so click on the **MERGE** button in the **MERGING** menu. If you click on this button with the left mouse button, you will get a dialogue box for the merge command. Then click on the **EXEC/QUIT** button to execute the command. You can reduce the number of steps you take when executing a command, such as merge, which has no arguments, by clicking on the **MERGE** button with the middle mouse button. In this case, no dialogue box will appear and the command will be executed immediately. You could also simply type the **merge** command in the text window to accomplish the same thing.

Creating the Geometry

Two cylinder surfaces are needed for this example. The dialogue box for surfaces is found in the **SURFACES** menu. Click on the **SD** button. In the dialogue box, click on the cylinder option. Enter the parameters to define surface 1 with an axis through the point (0,0,0). For the axis direction vector, use (0,0,1) and use 2 for the radius. This forms a cylinder with an axis of symmetry that passes through the origin in the z-direction. Be sure to type **Enter** to close each line of text. When each argument is terminated with a red box, click on the **EXEC/QUIT** button in the dialogue box. This creates the first surface and prints the command in both the text window and the session file.



2 Pipes Surfaces

```
sd 1 cy 0 0 0 0 0 1 2
```

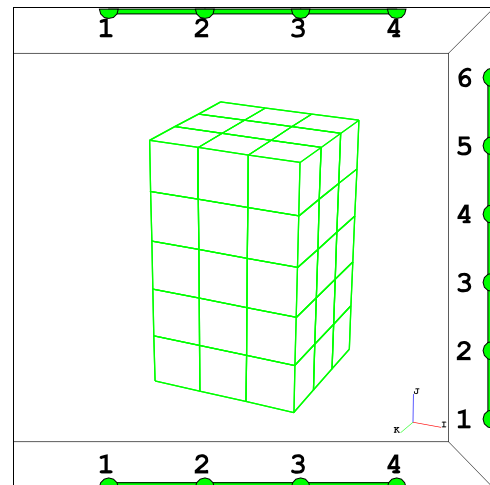
The second surface is also a cylinder. Repeat the above instructions for surface number 2 using the vector (1,0,0) for the axial direction and a radius of 1. When you click on the **EXEC/QUIT** button in the dialogue box, the following command will be printed:

```
sd 2 cy 0 0 0 1 0 0 1
```

You can view these surfaces by clicking on the **Show All** button in the **Display List** Panel of the environment window.

Creating the 1st Part

This part is a standard butterfly topology for the 1st cylinder pipe. It has two additional partitions in the j-direction and two partitions in the mid section in the k-direction for the second smaller pipe. Coordinates were carefully chosen to position these partitions to minimize the number of steps needed to perform the projection to the two surfaces. In particular, the first two partitions in both the i- and j-directions have the same coordinates. Also, the last two partitions in the i- and j-directions are assigned the same



Part in the Computational Space

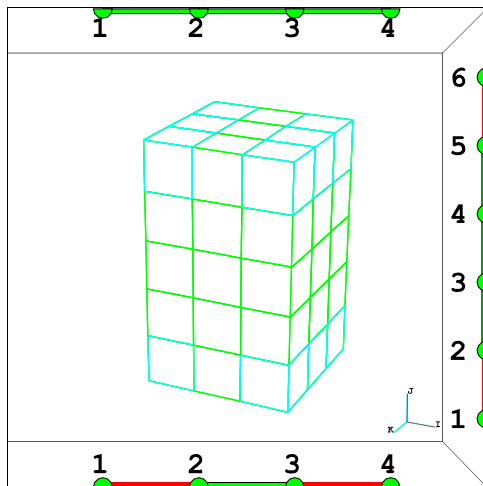
coordinates. This does not make much sense physically, but there is no need to make sense of this as a physical object when you are in the middle of the process of forming the mesh. The best description of this is that the outer regions have zero volume. The outer partitions have collapsed onto the inner partitions like a deflated balloon. Since the physical window can be confusing at this stage, it is advisable to use the computational window to track the progress of the mesh.

Create the part by clicking on the **BLOCK** button in the **PARTS** menu. There are six lists of number below to be transcribed into the **block** dialogue box. Be sure to type **Enter** at the end of each list in the dialogue box. Then click on the **EXEC/QUIT** button.

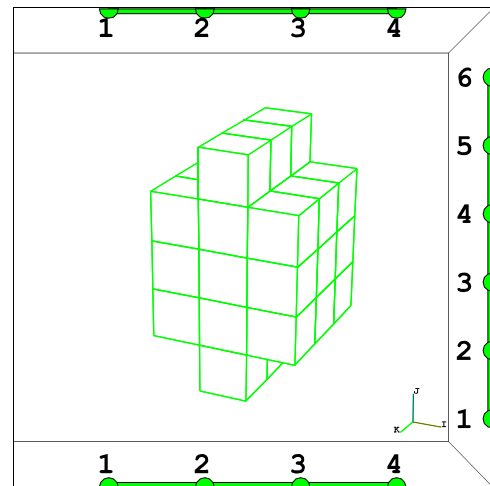
```
block 1 5 15 19;
      1 5 7 13 15 19;
      1 5 11 15;
      -1 -1 1 1;
      -1 -1 -.6 .6 1 1;
      -1.6 -.6 .6 1.6;
```

To form the butterfly topology, the four corners must be deleted. In the computational window, click and drag the mouse from the 1st to the 2nd dots on the i-index bar (bottom bar). The bar will turn red. Do the same for the interval between the 3rd and 4th dots. Also select the intervals between 1st and 2nd dots and between the 5th and 6th dots in the j-index bar (right bar). This selects the four corners of the part to be deleted. Click on the **Delete** button. Then type the **F2** function key to deselect.

```
dei 1 2 0 3 4; 1 2 0 5 6;;
```

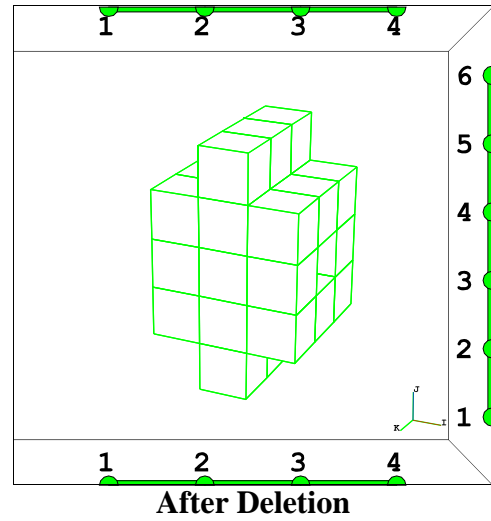
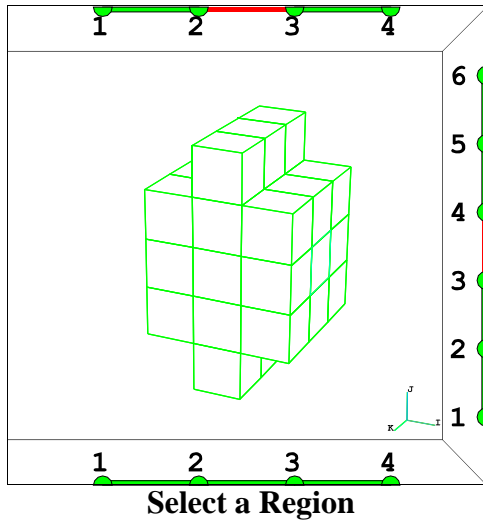


Select Regions to Delete



After Deletion

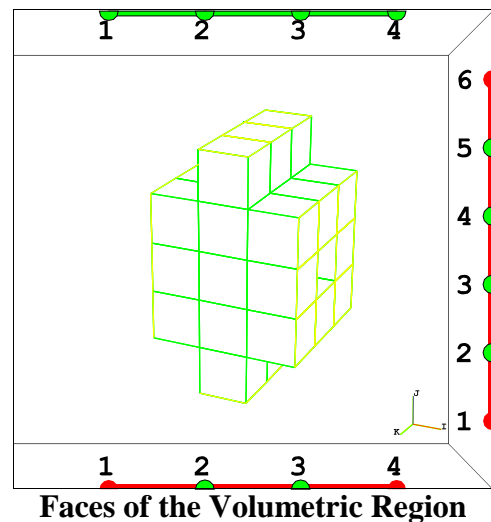
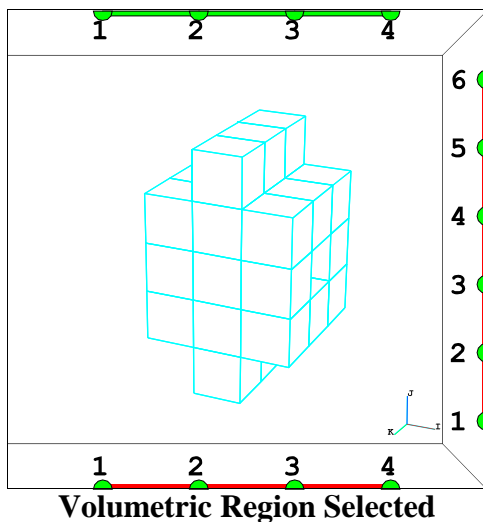
Another region must be deleted because the mesh for the smaller pipe must pass entirely through the mesh for the larger pipe. To select this region, click and drag the mouse along the j-index bar from dot 3 to 4. Then click and drag the mouse from dot 2 to dot 3 in the k-index bar (top bar). There is no need to make a selection in the i-direction since the entire bar is needed, and by default, the entire bar is selected. Click on the **Delete** button. Then type the **F2** key to deselect.



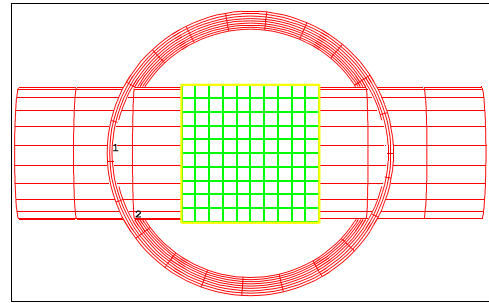
dei ; 3 4; 2 3;

Projecting to the Pipe Surfaces

The next step requires selecting faces, selecting a surface, and projecting. The minimum and maximum i- and j- faces are to be projected to the larger pipe surface and are selected in two steps. First, select the entire volume in both the i- and j-directions with a click and drag from the starting dot to the ending dot of each index bar. Then click on the end dots of both the i- and j-index bars. Clicking on the dots selects the corresponding face of the selected volume. This will get you all four faces to be projected to surface 1.



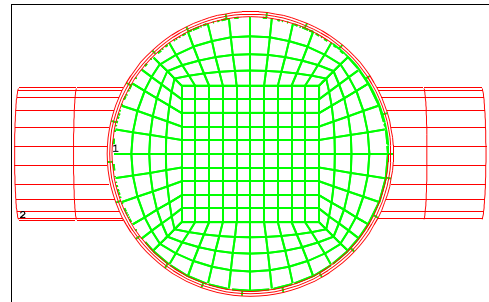
You can now shift to the physical window to see the results of the projection. To select surface 1, first display the surfaces (in the physical window). The easiest way is to click on the *Show All* button in the *Display List* Panel. Label the surfaces by clicking on the *Surface* button in the *Labels* Panel. If your picture is different because you have moved or rotated the view, simple click on the *Rest* button. Then click on *Label* in the *Pick* Panel so that you can pick a surface by clicking on the surface's label in the physical picture. Click on the label for the larger pipe surface. It will change colors when you have done this successfully.



Before Projection

The last step in the process is to click on the *Project* button. This produces the following command that is printed in the text window and to the session file. Type the **F2** key to deselect.

```
sfi -1 -4; -1 -6;;sd 1
```



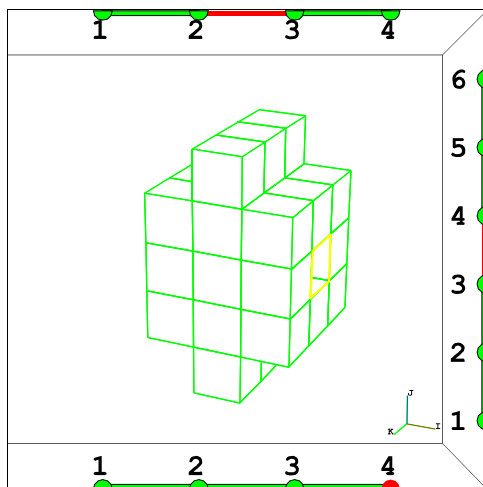
After Projection

Four edges of a deleted face must be projected to the smaller pipe surface. Surface number 1 is no longer needed in the picture, so click on the label for this surface. Then click on *Remove* in the *Display List* Panel. Type the following in the text window and type **Enter**

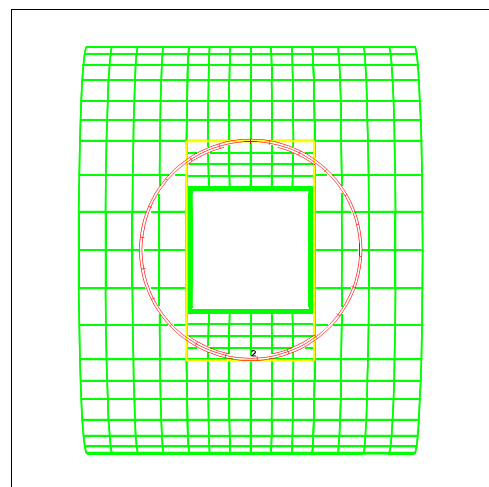
```
ry -90
```

to rotate the picture a negative 90 degrees about the y-axis. You can do this using the mouse, but it will be less accurate.

The process is the same when doing any projection. Select the desired region of the mesh, select the



Edges of a Deleted Face



View of Physical Mesh

surface, and click on the **Project** button. In this case, select the smaller surface, which should be the only surface in the picture. To select the region, click on the last dot in the i-index bar and select the middle interval in both the j- and k-index bars.

```
sfi -4; 3 4; 2 3;sd 2
```

Saving Block Boundary Interfaces

As a final step for this part, use the **BB** command in the **INTERFACE** menu to create 4 block boundary interfaces formed from the four faces of the hole for the smaller pipe. The interfaces must go all of the way through this part. Their identification numbers must be unique. These block boundaries will be key to building the second pipe and fitting it into this first pipe. It is left up to the reader to work out the details. Only a short description of each region is given.

All four interfaces use the entire interval in the i-direction. It is easiest to make no selections in the i-direction.

Choose the 4th dot in the j-index bar and the interval between dots 2 and 3 in the k-direction.

```
bb 1 4 2 4 4 3 1;
```

Choose the interval between dots 3 and 4 in the j-direction and the 2nd dot in the k-index bar.

```
bb 1 3 2 4 4 2 2;
```

Choose the 3rd dot in the j-index bar and the interval between dots 2 and 3 in the k-direction.

```
bb 1 3 2 4 3 3 3;
```

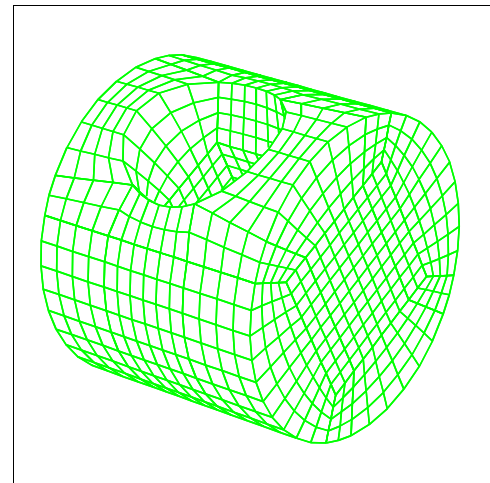
Choose the interval between dots 3 and 4 in the j-index bar and the 3rd dot in the k-direction.

```
bb 1 3 3 4 4 3 4;
endpart
```

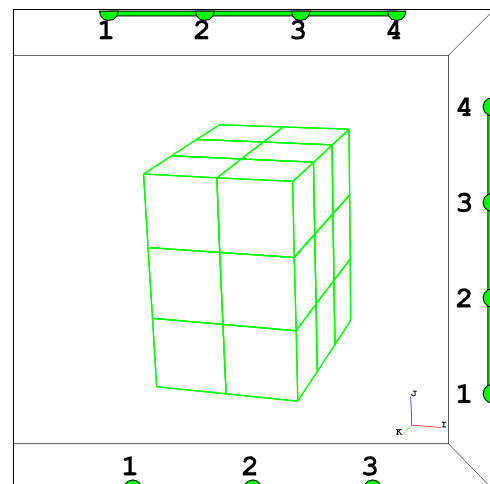
Creating Part 2

This part also has a butterfly topology. The cross section of the pipe is formed by sections of the mesh with a constant i-index (constant x-coordinate). There are two regions in the i-direction - one for the portion that extrudes from the first pipe and another embedded within the pipe. Similar techniques are used for this part with the initial coordinates carefully chosen to minimize the number of steps to completion. The outer regions have no volume. The projection to the pipe surface will expand the outer regions to give them positive volume.

```
block 1 19 23;
      1 4 10 13;
```



Part 1 Complete



Initial Topology Part 2

```

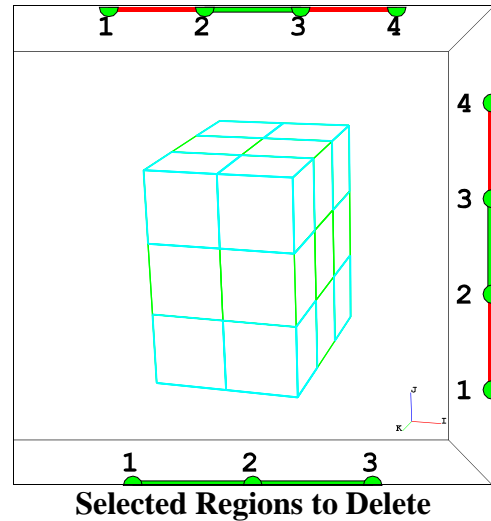
1 4 10 13;
-2 2 3
-.6 -.6 .6 .6
-.6 -.6 .6 .6

```

Deleting the Four Corners

The four corners are deleted to allow for the typical butterfly miter.

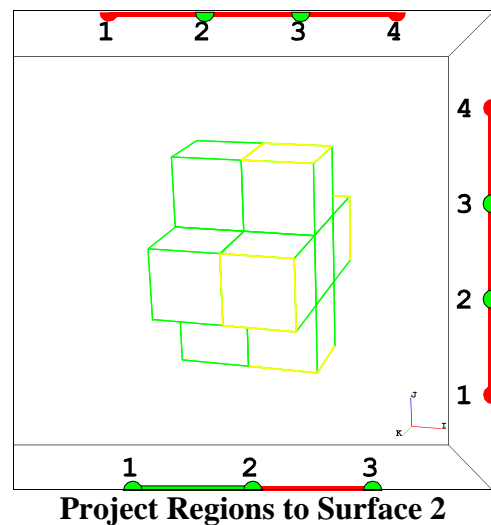
```
dei ; 1 2 0 3 4; 1 2 0 3 4;
```



Projecting to the Surfaces

The outer faces of the top portion of the mesh are projected to the smaller pipe surface.

```
sfi 2 3; -1 -4; -1 -4;sd 2
```

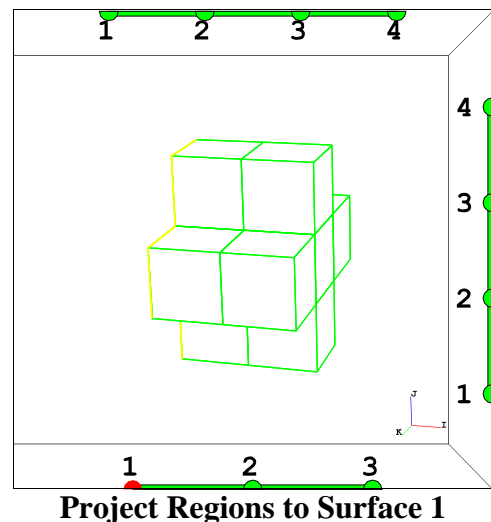


The bottom faces of this pipe must be projected to the larger pipe surface.

```
sfi -1;;;sd 1
```

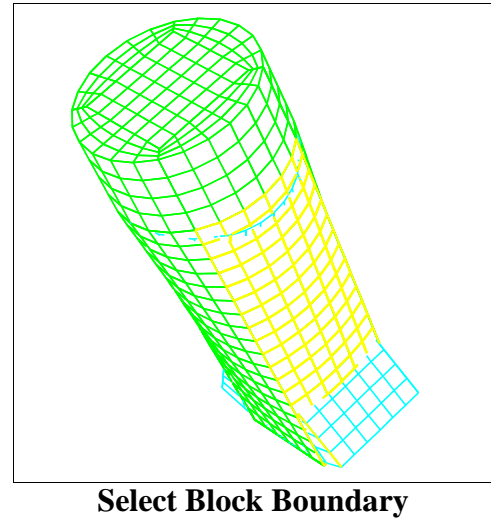
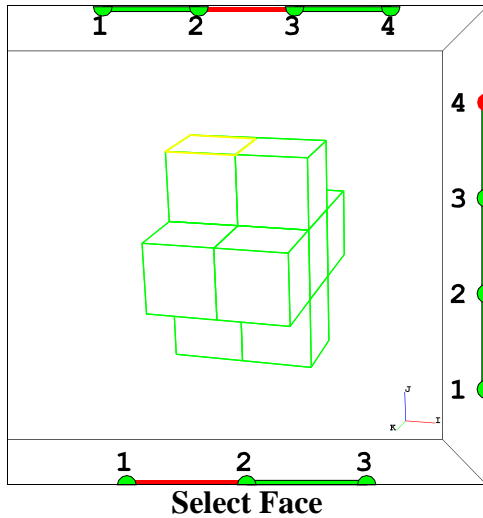
Attaching to Block Boundaries

The lower portion does not need to be projected. It is formed by attaching these faces to the block boundary interfaces formed in the first part. The Block Boundaries need to be in the picture. Click on **BB** in the **Display List** panel. Then click on **Show All**. Then click on the **BB** in the **Pick** panel. Select the face to be attached. Then select the appropriate Block Boundary using the left mouse button in the physical window. This done with a click and drag to form a small box. The Block Boundaries are colored cyan and there are 4 of them in the picture. Be careful to select only the Block Boundary that is on the same side as the face to be attached. When you select the correct one, the number 1 will appear in the **Show** window



of the *Pick* panel and the Block Boundary next to the selected face will change colors. Click on the *Attach* button.

```
bb 1 4 2 2 4 3 1;
```



Repeat this procedure for the remaining 3 Block Boundaries.

```
bb 1 2 1 2 3 1 2;
```

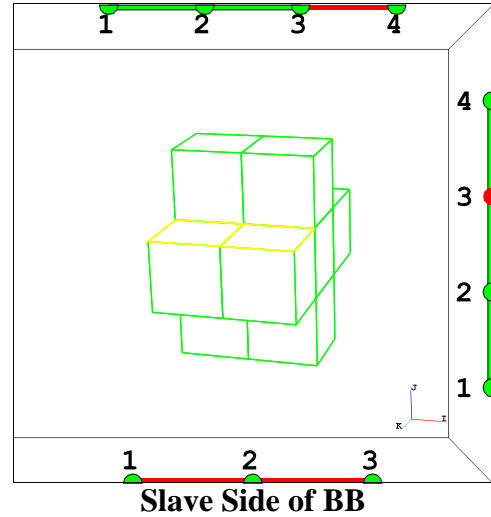
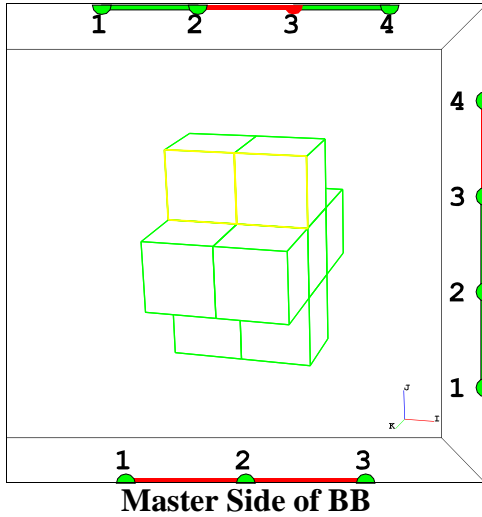
```
bb 1 1 2 2 1 3 3;
```

```
bb 1 2 4 2 3 4 4;
```

Gluing the Miters

In order to smooth the entire part using the most modern method, the coincident faces at the butterfly miters must be flagged so that the smoothing can include the nodes along the miter. This is done with the Intra-Part Block Boundary interface. It is called Intra-Part because both the master and slave side of the interface are found in the same part. Note that in the previous use of the Block Boundary interface, the master side was created in part 1 and the slave sides that were glued to the master side, were found in part 2. There are 4 miters to the butterfly topology requiring 4 pairs of the bb command. It does not matter which of each pair is the first (master side) and which of each pair is second (slave side).

Make the first selection in the computational window like in the first picture below. Then click on the **BB** button in the **INTERFACE** menu. Type the **F1** key to enter your selection into the dialogue box. Also type a 5 for the interface number and click on the **EXEC/QUIT** button. This creates the master side of the intra-Block Boundary interface number 5. To create the slave side, choose the region shown in the second picture below. Click on the **BB** button in the **INTERFACE** menu. Type **F1** and enter the interface number 5. Click on **EXEC/QUIT**. The two selected faces are now glued together.



```
bb 1 3 3 3 4 3 5;
bb 1 3 3 3 3 4 5;
```

Repeat this process for each of the butterfly miters. Your commands will be printer in the text window and should look like the following.

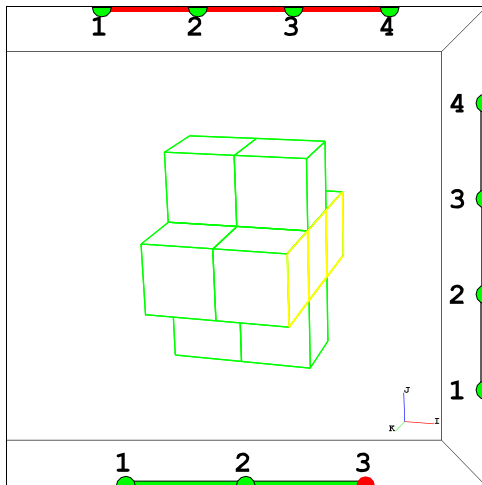
```
bb 1 2 3 3 2 4 6;
bb 1 1 3 3 2 3 6;
bb 1 1 2 3 2 2 7;
bb 1 2 1 3 2 2 7;
bb 1 3 1 3 3 2 8;
bb 1 3 2 3 4 2 8;
```

Smoothing the Mesh

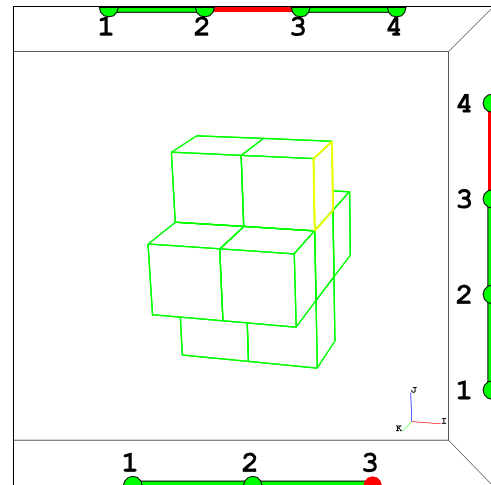
The Uniform smoothing method (**unifm**) allows for multiple regions. The Relaxation method (**relax**) for smoothing is the only other command with this option for multiple regions. Each region is separated by an ampersand. Of course, if you use the dialogue box for this command, you do not need to know this.

You should know that smoothing the interior of a volumetric region does not change the mesh on the outer faces. If you wish to smooth the outer faces, you must issue addition commands to do just that.

Select the face shown in the first picture. Click on the **UNIFM** button in the **MESH** menu. Then type the **F1** key. This enters the first region. In the dialogue box, click on the **Another Region** button. Select the face shown in the second picture and type **F1** To enter your choice into the second region in the **unifm** dialogue box. Select the region shown in the third picture. Click on the **Another Region** button and type **F1**. Choose 20 iterations in the dialogue box. Also choose 0 for the tolerance and a step size of 1. This gives you the maximum effect without becoming unstable.

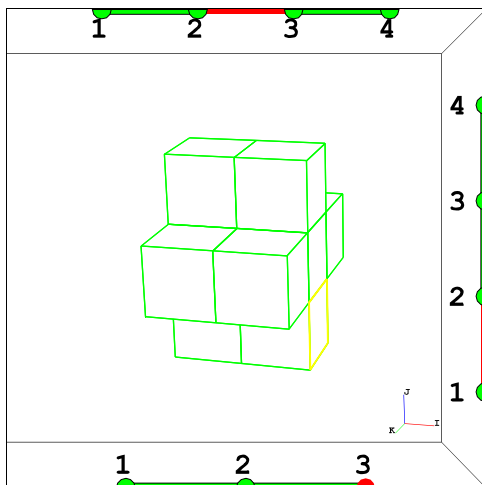


First Region to Smooth

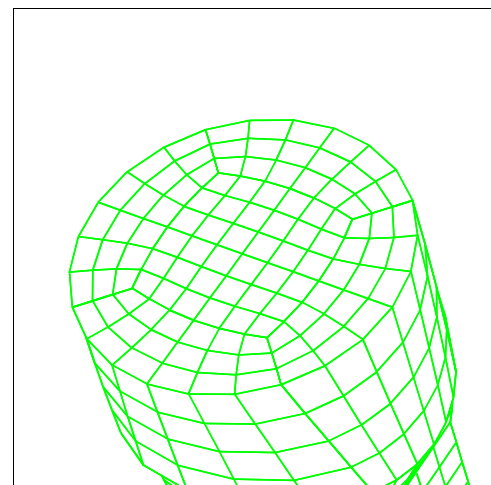


Second Region to Smooth

```
unifm 3 2 1 3 3 4 & 3 1 2 3 2 3 & 3 3 2 3 4 3 20 0 1 ;
```



Third Region to Smooth



Smoothed End Plate

Repeat this process for the face indicated in the command below.

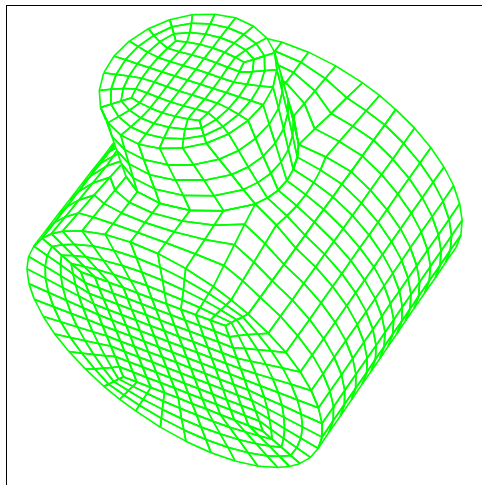
```
unifm 1 2 1 1 3 4 & 1 3 2 1 4 3 & 1 1 2 1 2 3 20 0 1 ;
```

The final step in this part is to smooth the interior. In this problem, smoothing is not just a luxury but necessary. If you rotate this part, you will notice that some of the interior partitions are extruding through the outer faces. This is easily fixed by choosing better coordinates for the face of the mesh in the middle i-partition. But instead, the **unifm** command will be used to pull those extruding elements back into the interior and give them ideal coordinates.

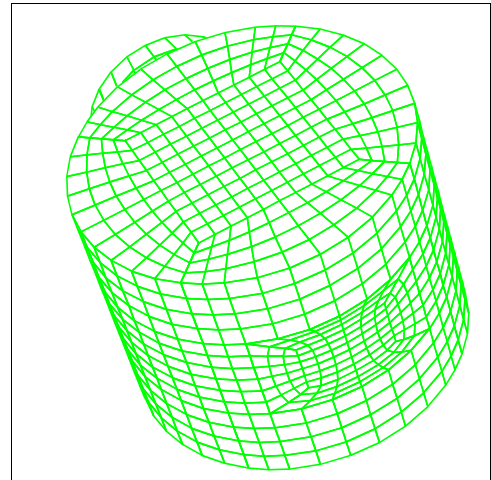
```
unifm 1 2 1 3 3 4 & 1 3 2 3 4 3 & 1 1 2 3 2 3 50 0 1 ;  
endpart
```

This ends the example. You can go into the merge phase and the two parts will be assembled. More will be said about the assembly process in the next example.

```
merge
```



Final Mesh



Final Mesh

XVII. Example of a Fuel Tank

In this section it is assumed that you can navigate through the menus and dialogues. It is also assumed that you have a basic understanding of the strategy used to create a model. There are many new commands introduced in this section. No further effort will be made to aid you in issuing these commands. There are five basic ways to issue these commands.

1. In many cases, the command can be executed using buttons in the environment window. The **dei**, for deleting regions of the part, are executed with the **Delete** button. Projection to surfaces is done using the **Project** button. This produces the **sfi** command. The **Attach** button performs the **pb**, **curs**, **edge**, or **bb** command, depending on the context. The Move Pts. panel will issue the **tri**, **pb**, or **mbi** command depending on the context. All of these keyword commands can be found under the **MESH** menu except the **bb** command which is found in the **INTERFACE** menu.
2. Click on the menu that contains the desired command. Then click on the desired command to activate the dialogue box for that command. Fill in the required arguments and execute and quit that dialogue box. The **F1** key will enter the selected regions of the mesh into the dialogue box.
3. Type the command in the text window. The **F1** key will enter the selected regions of the mesh into a command typed in the text window.
4. Copy and paste commands with the mouse into the text window.
5. Modify the session file and run **TrueGrid**[®] with a command file in batch mode. Be sure to enter the **interrupt** command into the session file so that **TrueGrid**[®] will become interactive at the desired places in this process.

Selecting Output Options

This example will be designed for Ls-dyna, although it can easily be modified for other simulation codes. Select the **LSDYNA** command under the **OUTPUT** menu. Choose the key word option.

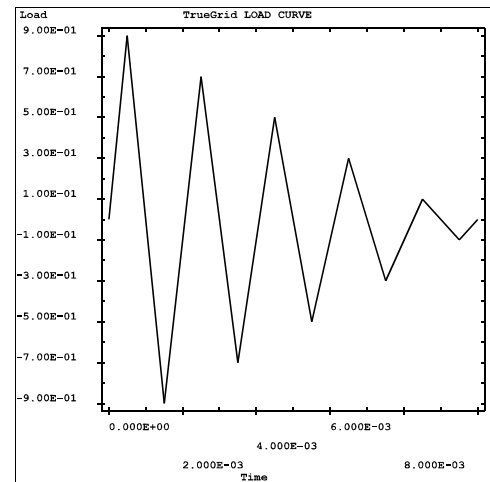
A material model is defined using the **LSDYMATS** command under the **MATERIAL** menu.

```
lsdymats 1 1 struct shell elfor bt c shell element formulation
shth .05 c shell thickness
rho .00073 c
density
e 3.0e+07 c
Young's Modulus
pr .3 ; c
Poison's Ratio
```

Several analysis options must be made using the **LSDYOPTS** under the **ANALYSIS** menu.

```
lsdyopts endtim .01 c termination
time
d3plot dtcycl .001 ; ; c data dump
interval
```

A load curve will be needed to vary the acceleration in



Load Curve

time. A load curve is defined using the *LCD* command found under the *2D CURVES* menu. Use the *LCV* command to draw the load curve.

```
lcd 1 0 0 .0005
     .9 .0015 -.9 .0025
     .7 .0035 -.7 .0045
     .5 .0055 -.5 .0065
     .3 .0075 -.3 .0085
     .1 .0095 -.1
     .01 0;
```

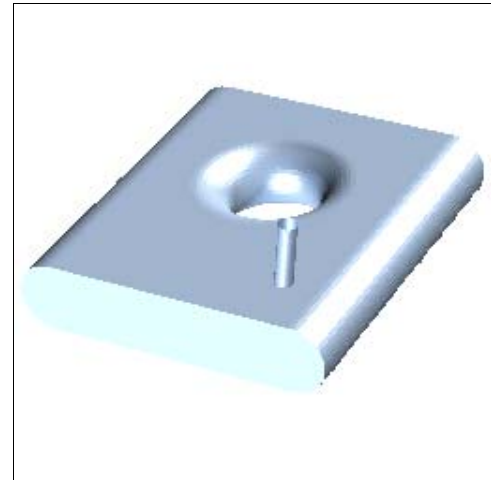
Creating the Geometry

In this example, you will create a shell structure of a tank.

The rounded walls are formed from half cylinders. Define two full cylinders of infinite length using the *sd* command in the *SURFACES* menu. The first three numbers are the coordinates of a point along the axis of symmetry. The second set of three numbers form the orientation vector. The last number is the radius.

```
sd 1 cy 3.5 0 0 0 0 1 1
sd 2 cy -3.5 0 0 0 0 1 1
```

You will only use a portion of these cylinders and there will be no additional effort to use these infinite surfaces. You should use this type of cylinder when possible because they are much easier to define.



Fuel Tank

Define the four (infinite) planes by using the *sd* option for a plane with a point and a normal. The gap between planes 3 and 4 is the diameter of the two cylinders.

```
sd 3 plan 0 1 0 0 1 0
sd 4 plan 0 -1 0 0 1 0
sd 5 plan 0 0 0 0 0 1
sd 6 plan 0 0 10 0 0 1
```

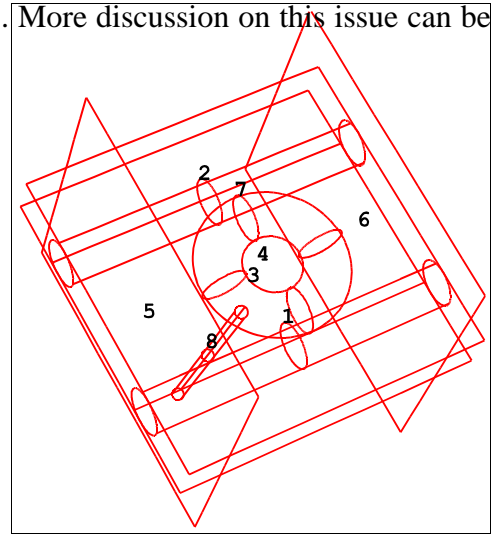
The next surface is a torus. You will use the inner half of this torus for the hole in the middle of the tank. The torus has the same minor radius as the two cylinders. The torus is also tangent to planes numbered 3 and 4 and the intersection of the torus with each of these planes is a circle. The intersection of the cylinders with these two planes are also along tangents. Tangent surfaces can be a problem because the Newton method used to project to the intersection of surfaces tries to intersect the tangent planes of the surfaces. In this case, the tangent planes of the two intersecting surfaces will both be the same tangent plane. **TrueGrid**[®] usually does a good job when the surfaces are accurate (unlike geometry that originates from a CAD system), but it may run out of iterations before it

converges. There are techniques to avoid tangent problems. More discussion on this issue can be found below.

```
sd 7 ts 0 0 6 0 1 0 2.25 0 1 ;
```

The inlet pipe is also an infinite cylinder. The orientation is defined using expressions. This pipe will be made in the cylindrical coordinate system. The angles used to orient the pipe will come in handy when you have to orient the cylindrical coordinate system. The pipe has a radius of .25.

```
sd 8 cy 2.5 -1 1.75
    [cos(-80)*cos(-25)] [sin(-80)]
    [cos(-80)*sin(-25)] .25
```

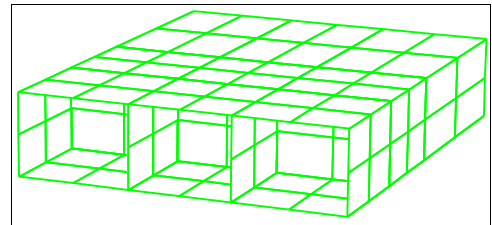


Tank Surfaces

Initializing Part 1

The first part forms the main body of the mesh. The front and back face plates and the inlet pipe will be done a separate parts. Initial coordinates used in the block part are carefully chosen to minimize the need to move regions to accommodate the projection to the surfaces.

```
block -1 -3 -5 -7;
      -1 -3;
      1 -3 -5 7;
      -4 -1.25 1.25 4
      -1 1
      0 4.75 7.25 10;
```

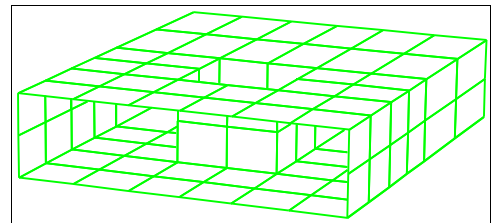


Initial Block Part

Deleting Unneeded Shells

All of the i-partitions are shells because the i-index has all negative indices. The first and last partitions will form the rounded side walls. The two inner partitions are used to form the torus. This is also the reason that the two middle k-partitions, corresponding to the two negative indices in the k-index list. The two shell j-partitions form the top and bottom faces of the tank. There are numerous shell faces that are not needed and must be deleted.

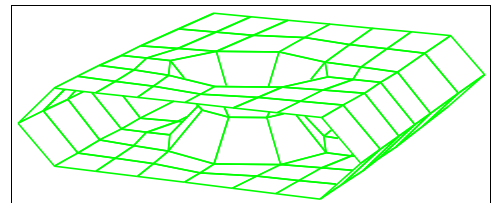
```
dei -2 0 -3; 1 2; 1 2 0 3 4;
dei 1 2 0 3 4; 1 2; -2 0 -3;
dei 2 3; -1 0 -2; 2 3;
```



Unneeded Regions Removed

Shaping the Mesh Using Projections

Faces of the mesh are now selected, corresponding surfaces are selected, and the **Project** button is clicked. This will produce a **sfi** command in the text window. Do this



Faces Projected to Surfaces

systematically so that each face is projected to a surface. If you are careful in the selection of regions you can minimize the number of projection commands to the following:

```
sfi -4; 1 2; 1 4;sd 1
sfi -1; 1 2; 1 4;sd 2
sfi 1 4; -1; 1 4;sd 4
sfi 1 4; -2; 1 4;sd 3
sfi -2 -3; 1 2; -2 -3;sd 7
sfi ;; -1;sd 5
sfi ;; -4;sd 6
```

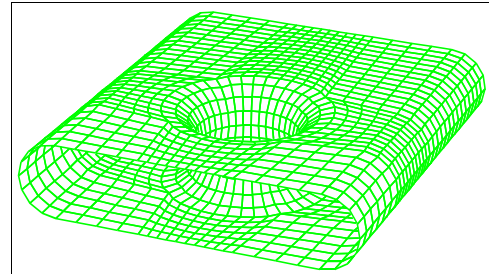
Adding Elements

This crude mesh can be refined using the **mseq** command. this command requires the number of elements to be added (positive integer), 0 (for no change), or the number of elements to remove (negative integer) for each region of the mesh.

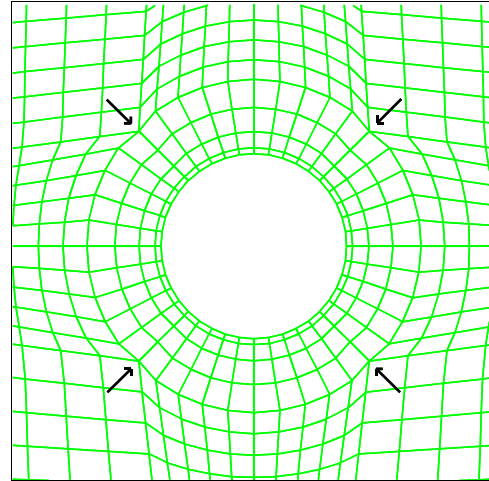
```
mseq i 2 8 2
mseq j 6
mseq k 12 8 4
```

Intersecting Tangent Surfaces

As was mentioned above, there may be a problem in the way the intersection of the torus and the plane are intersected. This is most noticeable at the vertices.



Increased Mesh Density

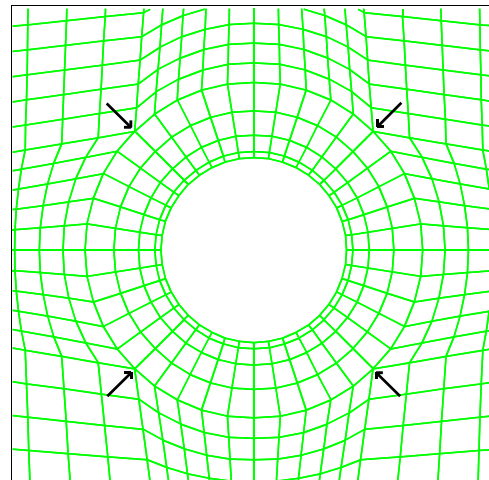


Inaccuracies at Tangents

One way to improve this is to issue the accuracy command (found in the **SURFACE** menu):

```
accuracy 50
```

This command will use 50 times as many iterations in the Newton method. It will also mean that the tolerance to measure convergence has been reduced by a factor of 50. It will be used the next time the mesh is recalculated. This happens when ever you issue another command that affects the mesh. This will slow the calculation down. Intersections to tangent surfaces are done with a method that is much slower to converge and that is why this situation should be avoided.

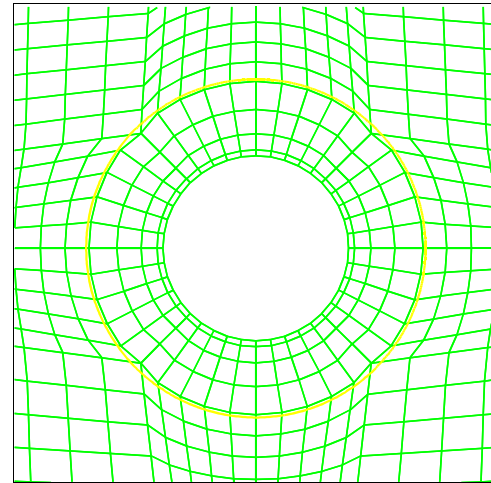


Improved Accuracy

Attaching Edges to 3D Curves

An alternative method can be used to avoid the intersection, if you have trimmed surfaces from a CAD system (**TrueGrid**[®] does not create trimmed surfaces but is able to use trimmed surfaces from a CAD system). Use the **sd** command with the **sds** option to form the composition of surfaces. Then project both faces to this new (composite) surface.

In this case, however, we do not have trimmed surfaces. The problem is solved using 3D curves. Create a 3D curve which is exactly the intersection of these two tangent surfaces.

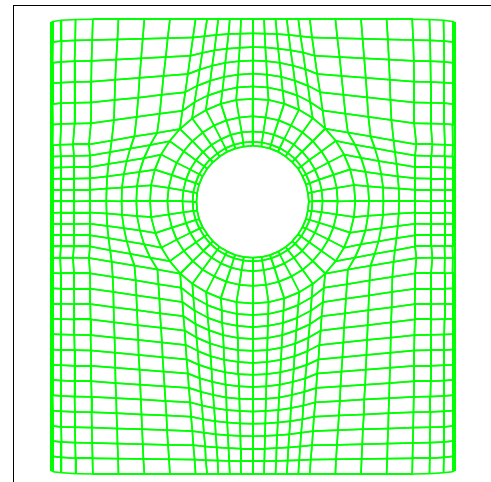


3D Curve at Intersection

```
curd 1 arc3 whole rt 0 -1 8.25 rt 2.25  
-1 6 rt 0 -1 3.75 ;  
curd 2 arc3 whole rt 0 1 8.25 rt 2.25 1 6 rt 0 1 3.75 ;
```

Then attach the appropriate edges of the mesh to these curves. This can be done by selecting an edge, selecting the curve, and clicking on the **Attach** button. The edges will be moved to the appropriate curves and then projected to the intersection of the tangent surfaces.

```
curs 2 1 2 2 1 3 1  
curs 3 1 2 3 1 3 1  
curs 2 1 3 3 1 3 1  
curs 2 1 2 3 1 2 1  
curs 2 2 2 2 2 3 2  
curs 3 2 2 3 2 3 2  
curs 2 2 3 3 2 3 2  
curs 2 2 2 3 2 2 2
```



Edges Attached to Curves

The projection algorithm will detect that the edge nodes are already at the intersection and skip the intersection calculations.

You might argue that this example emphasizes the intersection of tangent surfaces to an extreme when it should focus on the main stream use of **TrueGrid**[®]. But in response, you can argue that in order to understand the problem of intersecting tangent surfaces, you must understand the projection method and the limitations of its iterative Newton method. With this understanding, you can take full advantage of the projection method.

Distributing Nodes Equally

Notice that nodes along edges are not equally spaced. This is due to the initial position of some of the partitions and the number of elements chosen for each region. This can be changed by applying the **res** interpolation constraint which forces all of the nodes be equally spaced.


```

res 1 1 4 4 2 4 i 1
res 1 1 1 4 2 1 i 1
res 4 1 1 4 2 4 k 1
res 1 1 1 1 2 4 k 1

```

The entire minimum and maximum edges in both the i- and k-directions are equally spaced. These constraints include 2 interior partitions. By default, the two interior vertices along each of these edges has the freedom to be moved. By including them in the interior of a **res** command, they no longer have that freedom, but are interpolated as any other interior node along the edge.

Inserting Partitions for the Inlet

A hole is needed in the mesh for the inlet pipe. No partitions were formed in the block command to anticipate this hole. Partitions will be added using the **insprt** command. This method of forming partitions in the part has two advantages. First, you do not have to plan for the location of the required partitions. It is possible that you will discover that some existing partitions can be used for the unplanned feature, reducing the complexity of the part. In fact, you will use the existing 3rd i-partition for the inlet pipe hole. Secondly, when a partition is inserted, it is automatically placed along an existing mesh line so that you not have to determine the initial coordinates for the partition as you would if the partition were specified in the block command.

The inlet pipe part will have a flange that connects to this part, so the hole in this part must be made larger than the inlet pipe. The cylinder used for this hole is similar to surface 8, but with a larger radius of .5.

```

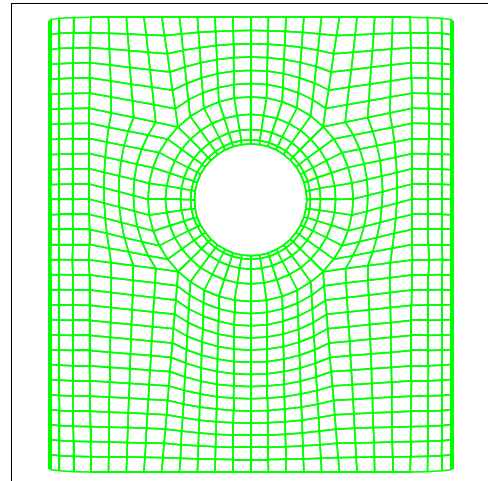
sd 9 cy 2.5 -1 1.75
   [cos(-80)*cos(-25)]
   [sin(-80)]
   [cos(-80)*sin(-25)] .5

```

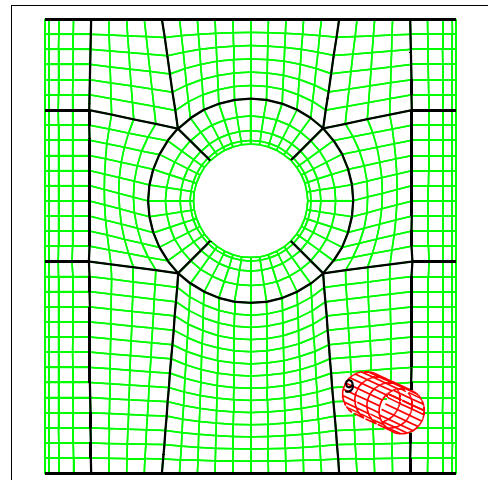
Three partitions must be inserted. The **insprt** command has 4 arguments. The first number is almost always 1 indicating a standard partition. When a partition is inserted, it is always relative to an existing reference partition. The new partition will be either before or after this reference partition. The second argument can be one of the following values:

- 1 - before the reference i-partition
- 2 - after the reference i-partition
- 3 - before the reference j-partition
- 4 - after the reference j-partition
- 5 - before the reference k-partition
- 6 - after the reference k-partition

This number is followed by the reference partition number and the number of elements away from that reference partition.



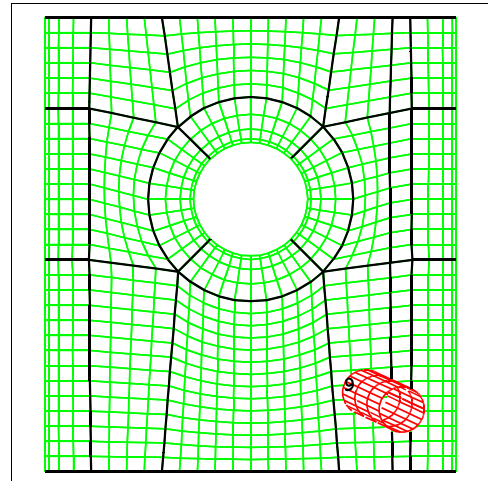
Equal Spacing on the boundaries



Existing Partitions

The first partition to be added will be an i-partition 3 elements after the existing 3rd i-partition.

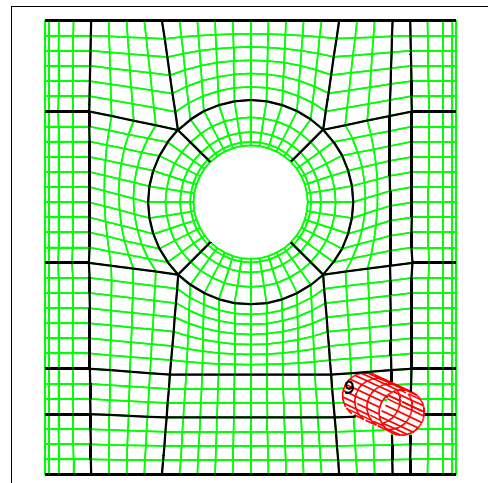
```
insprt 1 2 3 3
```



Inserting an I-Partition

Two more partitions are inserted to form two new k-partitions from existing mesh lines.

```
insprt 1 6 1 4  
insprt 1 6 2 3
```

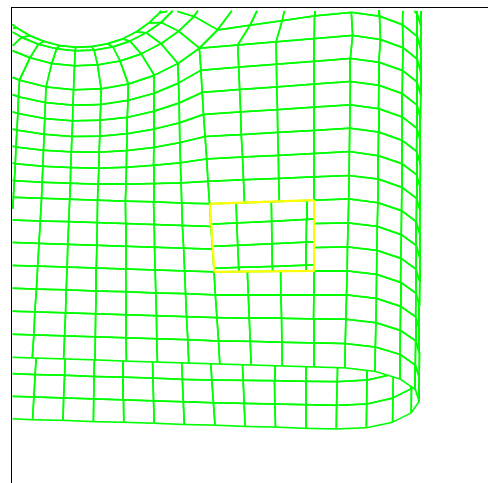


2 K-Partitions Added

Creating the Hole for the Inlet Pipe

A hole is created by selecting the region just formed by these three new partitions. Then the *Delete* button is clicked.

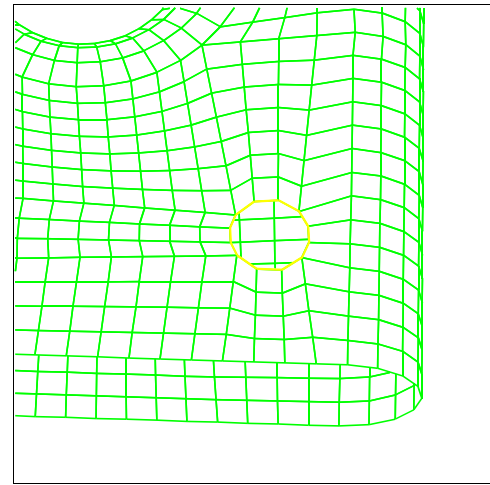
```
dei 3 4; -1; 2 3;
```



Deleted Region

The edges are then projected to surface 9. Using the same settings to select the region to be deleted, select surface 9 and click on the **Project** button.

```
sfi 3 4; -1; 2 3;sd 9
```



Projected Edges

Saving the Interfaces

To insure that the inlet pipe part matches the hole in the part, block boundaries will be defined. Each of the four edges of the hole will form a different block boundary. This is done by selecting one edge at a time. Click on the **BB** command in the **INTERFACE** menu. Type the **F1** key to enter your selection. Then type an identification number (usually starting at 1) into the dialogue box. Do this for all four edges.

```
bb 3 1 3 4 1 3 1;
bb 4 1 2 4 1 3 2;
bb 3 1 2 4 1 2 3;
bb 3 1 2 3 1 3 4;
```

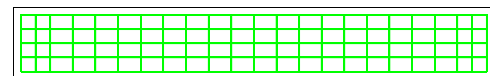
Setting the Loads

Four corner nodes are assigned accelerations in the y-direction which will vary in time. The command **ACCI** in the **DIS/VEL/ACC** menu is used and it refers to the load curve that was defined at the beginning of this example.

```
acc1 -1 0 -5;-2;-1 0 -6;1 1 0 1 0 ;
endpart
```

This completes the first part.

Initializing Part 2



Initial Part

The second part of this example forms the two end plates. Only one end plate will be made and then copied to form the other. This block command initializes some of the partitions with the same coordinates. The initial effect is that the outer regions are collapsed. This is only temporary because the outer edges of this part are going to be projected outward to the cylinders and planes, giving these regions a positive area. The coordinates were carefully chosen so that most of the interior regions are initialized to their final position. This part includes a butterfly topology at all four corners.

```
block 1 3 5 23 25 27;
      1 3 7 9;
      -1;
      -4 -4 -3.5 3.5 4 4;
```

```

-.5 -.5 .5 .5
0

```

Deleting the Four Corner Regions

Using the index bars in the computational window, you can select all four corner regions. Then click on the **Delete** button. These are deleted so that you can form a butterfly topology at these corners.

```
dei 1 2 0 5 6; 1 2 0 3 4; -1;
```

Project Boundary Edges

Now project the outer edges to the appropriate surfaces. Since the edges to be projected are coincident with interior edges of the part, you must select the edges in the computational window. Then select the appropriate surface and click on the **Project** button. The boundaries of this part must match the boundaries of the first part, so it is critical that you choose the same surfaces used in the first part.

```

sfi -6; 2 3; -1; sd 1
sfi 4 5; -1 0 -4; -1; sd 1
sfi -1; 2 3; -1; sd 2
sfi 2 3; -1 0 -4; -1; sd 2
sfi ;; -1; sd 5

```

Optimize the Mesh

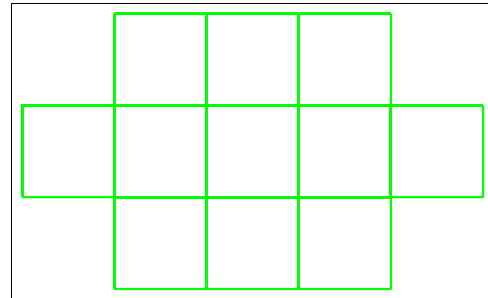
The interior vertex of the three blocks that form each butterfly can be moved to a better position. The **TRICENT** function in the **MISC** menu will calculate the ideal position for these nodes. In the **tricent** dialogue, click on the “Based on the edge midpoints” option. This calculation is based on the midpoints of the bounding topological triangle, indicated by the marks in the picture. A topological triangle is formed from 3 regions enclosing a common vertex and with edges that may not be line segments. Click on the **Node** button in the **Pick** panel. Then click on one of the marked vertices with the left mouse. Then type **F7** to enter the coordinates into the dialogue box. Repeat this for the other two marked vertices on the boundary of the topological triangle..

```

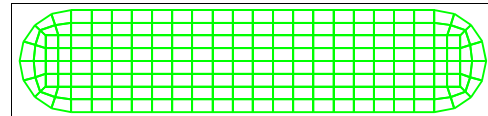
tricent 2 4.2071066 .70710677 0 4 0 0
3.5 .5 0

```

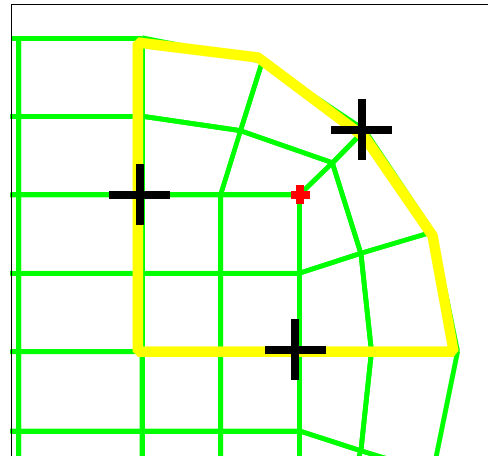
When you execute this function, you will receive the coordinates for the center vertex. Issue a **pb** command for



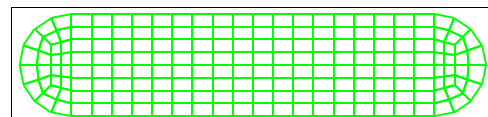
Computational Mesh



After Projecting



Topological Triangle

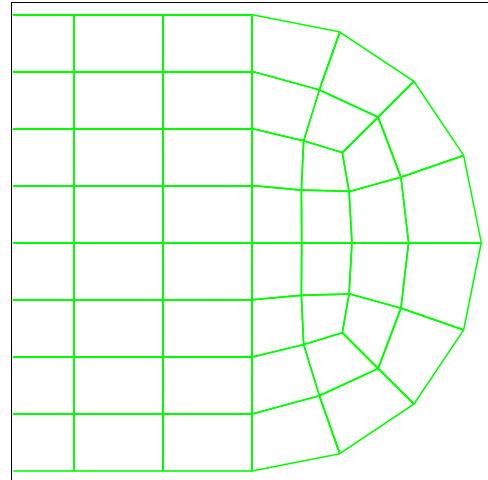


Interior Vertices Moved

each of the four center vertices forming the four corner butterflies. You only need to issue the **tricent** command once and use cut and paste in the text window to form the coordinates for each **pb** command. Take care to put the minus sign in the proper places.

```
pb 5 3 1 5 3 1 xy 3.894338 3.943377e-01
pb 5 2 1 5 2 1 xy 3.894338 -.3943377
pb 2 3 1 2 3 1 xy -3.894338 .3943377
pb 2 2 1 2 2 1 xy -3.894338e -.3943377
```

This mesh can be improved a little by using one of the smoothing functions. The one that is easiest to use in the **RELAX** function in the **MESH** menu because you can have holes in the selected region. Select the end regions of the part to be smoothed. Use 10 iterations for a small improvement in the quality. You can experiment with this and you will probably find that a large number of iterations will not give you a better mesh. In some cases, a larger number of iterations will make a difference. Use a tolerance of 0 in this case since we are not doing enough iterations to be concerned with convergence. You almost always want to use a step size of 1. Never go larger than 1 or the smoothing may become unstable. A positive number smaller than 1 will dampen the effect.

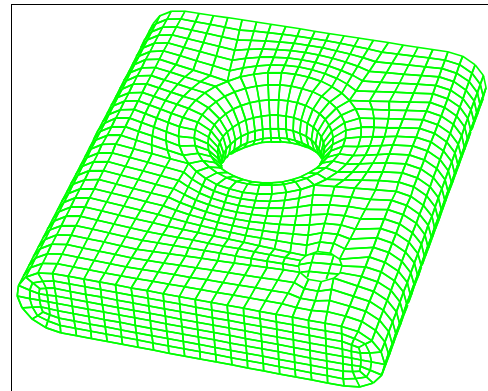


Smoothed End Caps

```
relaxi 1 3 0 4 6;1 4;;10 0 1
```

Replicate the Part

You need to replicate this part for the opposite end. The opposite end is 10 units in the z-direction. A local coordinate system transformation is defined with the **LCT** command in the **REPLICATE** menu.



Two Parts Merged

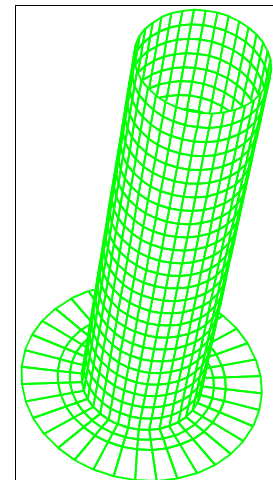
This command creates a table of transformations. In this case, you will create a table of transformations containing just one transformation. This transformation moves an object 10 units in the z-direction using the **mz** option. This transformation is then applied with the **LREP** command also found in the **REPLICATE** menu. The **lrep** command creates a copy of the part for each transformation identified in the list that follows it. The 0 transformation is the identity transformation. The identity transformation must be included if you want to keep the original part. You will not see the results of this replication until you enter the merge phase.

```
lct 1 mz 10;lrep 0 1;
endpart
```

This completes the second part.

Initializing Part 3

This part is ideal for the cylinder part. The cylinder part is similar to the



Inlet w/ Flange

block part with the exceptions that the x coordinate is replaced by the radius and the y-coordinate is replaced by the angular coordinate. Also, all interpolations between vertices and across faces are done in the cylindrical coordinate system. The i-index list contains a negative 1 which means there will be a shell partition with a constant radius. The first index in the k-direction is also negative indicating that there is a shell flange at the minimum k partition.

```
cylinder -1 3 4;
         1 10 19 28 37;
         -1 26;
         .25 .35 .5;
         0 90 180 270 360;
         0 2;
```

The default position aligns the local z-axis of the cylindrical part with the z-axis of the global coordinate system. This must be modified so that the part is generated in position. The **CYCORSY** command in the **PARTS** menu transforms the local coordinate system. Using the coordinates and the normal vector used to form surface 8, you can derive the transformation that defines this new frame of reference.

```
cycorsy rx 90 rz 10 ry 25
        v 2.5 -1 1.75;
```

Projecting The Pipe & Flange

The intermediate i-partition will be used to set aside enough space in the transition region. Another cylinder is needed with an intermediate radius of .35 for this partition.

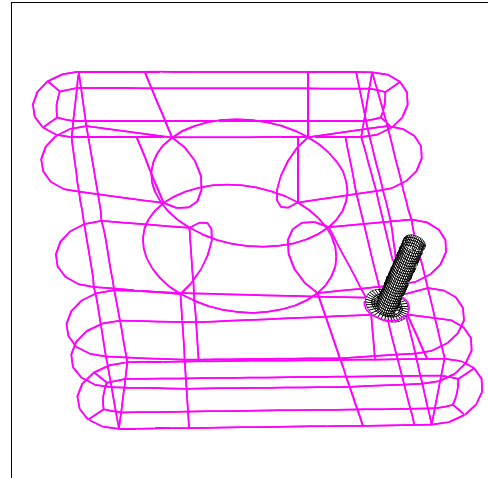
```
sd 10 cy 2.5 -1 1.75
    [cos(-80)*cos(-25)]
    [sin(-80)]
    [cos(-80)*sin(-25)] .35
```

The outer radius does not need to be projected because those edges will be attached to the block boundaries defined in the first part. The intermediate i-partition is projected to this new cylinder. The inner i-partition is projected to the smallest cylinder. The flange is projected to the bottom plane.

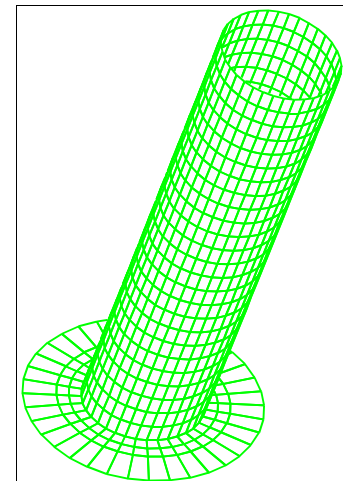
```
sfi ;; -1;sd 4
sfi -1;;;sd 8
sfi -2;;;sd 10
```

Gluing to Part 1

The four outer edges are attached to the four corresponding block boundaries. To do this, display all of the block boundaries by typing the command **dabb** or use the **Display List** panel. Go to the **Labels** panel and label the block boundaries so that you can **Pick** by **Label**. Select one of the outer edges, click on the **TRBB** command in the **INTERFACE** menu. Type



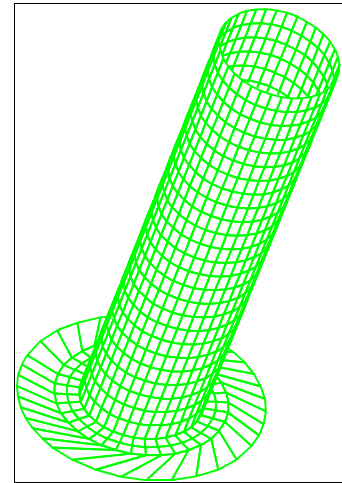
Transformed Cylinder Part



Projections

F1 to enter you selection. You must determine the appropriate block boundary from the picture. Then type the appropriate block boundary into the dialogue box and execute. The edge will be shifted so that it aligns with part 1. The **trbb** command creates a transition from one mesh density to another, but these transition elements are not created until you end the part and go to the merge phase.

```
trbb 3 1 1 3 2 1 2;
trbb 3 2 1 3 3 1 1;
trbb 3 3 1 3 4 1 4;
trbb 3 4 1 3 5 1 3;
```

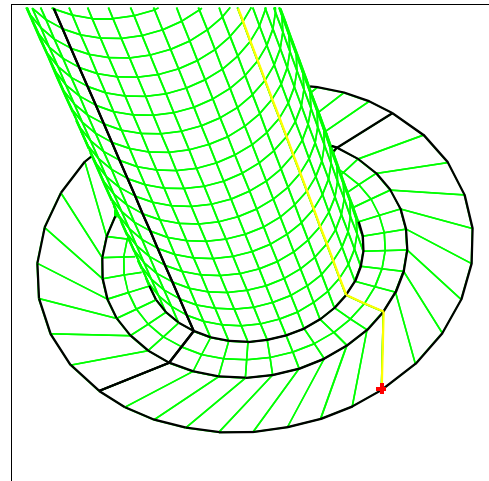


Glued Edges

Repositioning the J-Partitions

The j-partitions do not align with the block boundaries. This causes a twist or shear in the elements in the boundary elements. The angular coordinate assigned to each partition must be corrected. The angular coordinate from the boundary can be assigned to j-partition. Follow these steps:

1. select the 2nd j-partition in the computational window.
2. click on *Node* in the *Pick* panel
3. uncheck the x- and z-coordinates in the *Pick* panel
4. click on the boundary node (marked in red) with the desired coordinate
5. click on the *Attach* button

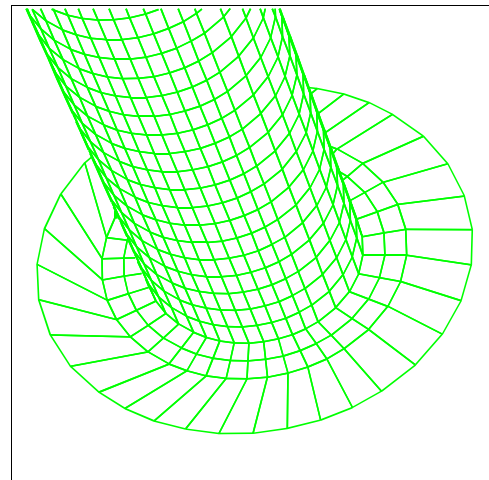


Rotate J-Partition

The entire partition will then be aligned with the node on the boundary. This should be repeated for all of the j-partitions. The 1st and last j-partitions will look the same, but the last partition will have an angular coordinate that is 360 degrees greater than the 1st j-partition.

```
pb 1 1 1 3 1 2 y -1.107621e+01
pb 1 2 1 3 2 2 y 6.817803e+01
pb 1 3 1 3 3 2 y 1.732756e+02
pb 1 4 1 3 4 2 y 2.417690e+02
pb 1 5 1 3 5 2 y 3.489238e+02
endpart
```

This completes the third part.



2nd J-Partition in Place

Assembling the Model

The next step is to enter the merge phase to assemble the entire model. The **MERGE** command is found in the **MERGING** menu.

```
merge
```

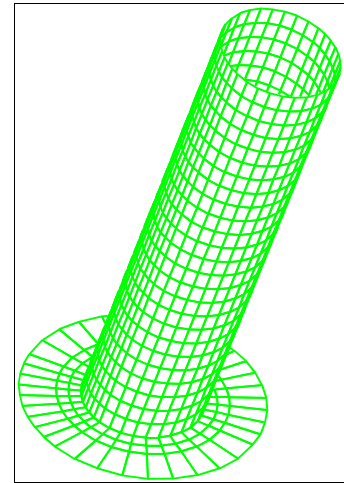
The nodes must be merged and the output file written. Use the **STP** command found in the **MERGE** menu.

```
stp .01
```

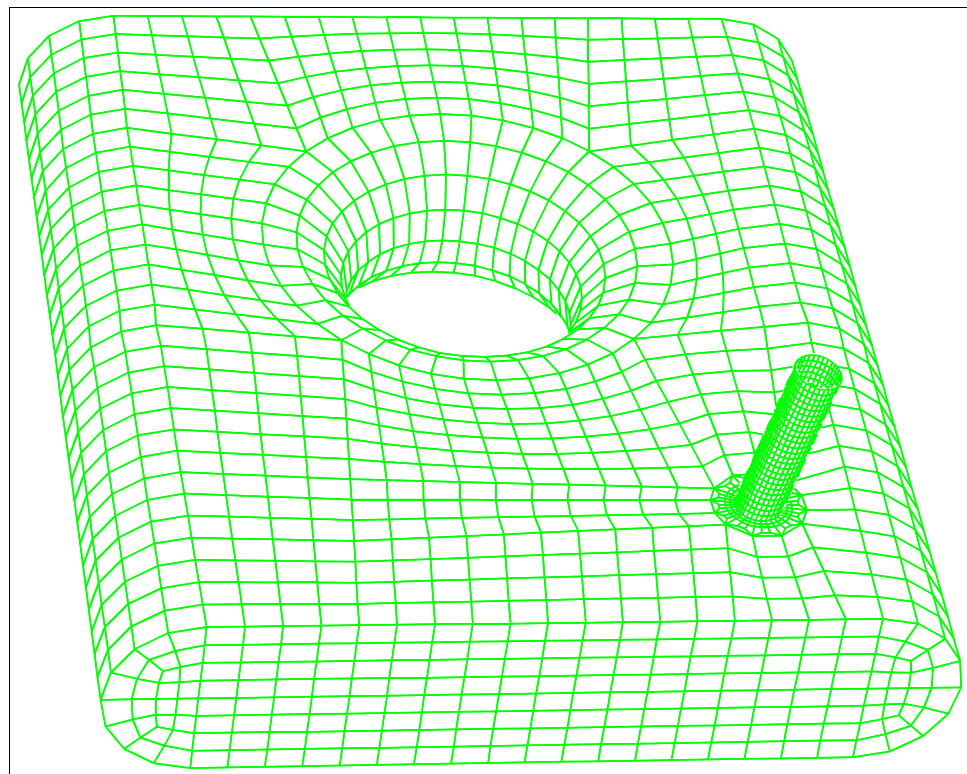
The tolerance will vary depending on the problem. In this problem, part 2 was not carefully generated and the nodes at the interface between parts 1 and 2 do not match. For this reason, the tolerance needed is relatively large. Since part 3 was shaped using block boundaries, the interface nodes have identical coordinates so a very small tolerance would be sufficient.

The last step is to write an output file with the **WRITE** command found in the **OUTPUT** menu.

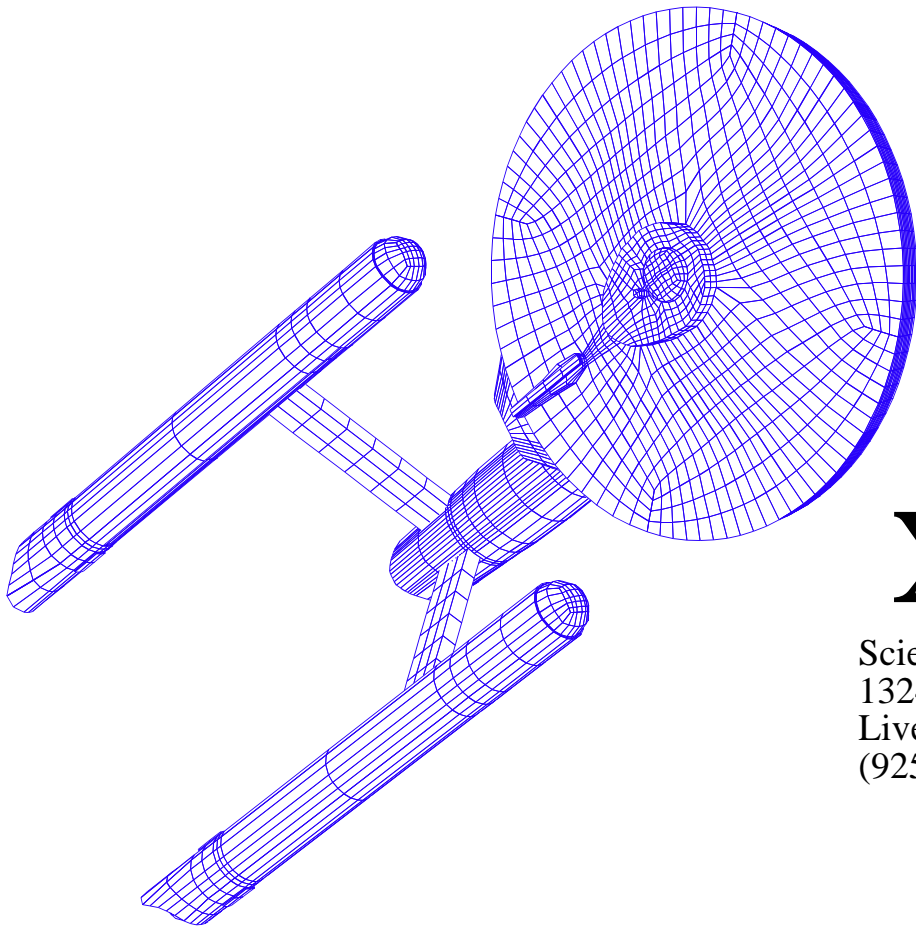
```
write
```



Completed Part 3



Final Model



XYZ

Scientific Applications, Inc.
1324 Concannon Blvd.
Livermore, CA 94550
(925) 373-0628

