SANDIA REPORT

SAND89—0485 • UC—32
Unlimited Release
Printed March 1989

GEN3D: A GENESIS Database 2D to 3D Transformation Program

Amy P. Gilkey, Gregory D. Sjaardema

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-76DP00789

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

NTIS price codes
Printed copy: A03
Microfiche copy: A01

SAND89-0485
Unlimited Release
Printed March 1989

Distribution
UC-32

GEN3D:
A GENESIS Database 2D to 3D Transformation Program

Amy P. Gilkey and Gregory D. Sjaardema
Engineering Analysis Department
Sandia National Laboratories
Albuquerque, New Mexico 87185

Abstract

GEN3D is a three-dimensional mesh generation program. The three-dimensional mesh is generated by mapping a two-dimensional mesh into three dimensions according to one of four types of transformations: translating, rotating, mapping onto a spherical surface, and mapping onto a cylindrical surface. The generated three-dimensional mesh can then be reoriented by offsetting, reflecting about an axis, and revolving about an axis. *GEN3D* can be used to mesh geometries that are axisymmetric or planar, but, due to three-dimensional loading or boundary conditions, require a three-dimensional finite element mesh and analysis. More importantly, it can be used to mesh complex three-dimensional geometries composed of several sections when the sections can be defined in terms of transformations of two-dimensional geometries. The code *GJOIN* is then used to join the separate sections into a single body. *GEN3D* reads and writes two-dimensional and three-dimensional mesh databases in the GENESIS database format; therefore, it is compatible with the preprocessing, postprocessing, and analysis codes used by the Engineering Analysis Department at Sandia National Laboratories, Albuquerque, NM.

Contents

Figures 6

1 Introduction 7

1.1 Terminology 8

1.2 Element Blocks 9

1.3 Front and Back Sets 10

1.4 Three-Dimensional Output 10

2 Command Input 19

2.1 Mesh Transformation 21

2.2 Mesh Orientation 24

2.3 Element Block Types 28

2.4 Front and Back Set Definition 30

2.5 Information and Processing 31

3 Informational and Error Messages 33

4 Executing GEN3D 35

4.1 Execution Files 35

4.2 Special Software 35

References 37

A The GENESIS Database Format 39

B Command Summary 43

C Site Supplements 45

C.1 VAX VMS 45

C.2 CRAY CTSS 45

Figures

1.1 Simple 2D mesh input to *GEN3D* 13

1.2 Translation 14

1.3 Rotation around line outside mesh 15

1.4 Rotation around mesh edge 16

1.5 Warp about the Y axis 17

2.1 Illustration of Projection Procedure for the WARP POINT Command 25

2.2 Illustration of Mapping Procedure for the WARP *axis* Command . . 26

1. Introduction

GEN3D is a mesh generation program that creates a three-dimensional mesh from a two-dimensional mesh by transforming the two-dimensional mesh. Four transformations are available: (1) translating normal to the plane of the two-dimensional mesh, (2) rotating the two-dimensional mesh about an axis, (3) mapping the two-dimensional mesh onto a spherical surface, and (4) mapping the two-dimensional mesh onto a cylindrical surface. The resulting geometry is commonly referred to as a $2^1/_2$-dimensional geometry since it can be completely defined in terms of a two-dimensional geometry and a single transformation. *GEN3D* is intended to be used with the companion code *GJOIN* [3] which joint two or more GENESIS databases into a single database.

GEN3D is compatible with and intended to complement the other mesh generation programs used in the Engineering Analysis Department at Sandia National Laboratories. The two programs *FASTQ* [4] and *PATRAN* [2] are used to generate the majority of the finite element meshes in the Engineering Analysis Department. *FASTQ* is a two-dimensional mesh generation program that is based on the earlier mesh generator *QMESH/RENUM* [11]. It has been totally rewritten and enhanced, and also employs higher-order primitives which have been developed to simplify meshing of commonly encountered shapes (for example, triangles) and conditions (for example, transitions). *PATRAN* is a general purpose, two-dimensional and three-dimensional commercial mesh generation program that can be used to generate, with varying degrees of difficulty, any desired geometry.

Although almost any desired geometry can be defined and meshed with the two programs *FASTQ* and *PATRAN*, additional mesh generation programs have been, or are being, developed to reduce the effort required to both define and discretize the geometry. Koteras [12,14], Blacker [6,5], and Chavez [7] are developing automatic, or semi-automatic, mesh generators for two-dimensional and three-dimensional geometries. These programs are intended to automatically discretize a user-defined geometry with varying amounts of involvement by the analyst. Research is also in progress to reduce the time and effort required to define the geometry. Sjaardema [17], Schutt [16], Chavez [7], and Koteras and Blacker [13] are investigating methods for linking the output from CAD programs and solid modelers to the input of mesh generation programs to eliminate or reduce the need to enter the geometry manually.

GEN3D is a simpler approach to three-dimensional mesh generation than the automatic and general-purpose programs cited above. The geometries of many of the bodies analyzed are axisymmetric or planar; however, due to three-dimensional load-

ing or boundary conditions, the body must be analyzed with a three-dimensional finite element mesh and analysis code. *GEN3D* can also be used to mesh complex three-dimensional geometries composed of several primitives or sections if the primitives can be defined in terms of transformations of two-dimensional geometries. Each of the primitives, or as many as possible, can be meshed using *GEN3D*, and then joined together using *GJOIN* [3]. The primitives that cannot be meshed with *GEN3D* can be meshed with *PATRAN* and then joined to the other primitives.

GEN3D reads and writes files in the GENESIS [19] database file format which is the geometry definition portion of the EXODUS [15] database file format used in the Engineering Analysis Department at Sandia National Laboratories, Albuquerque, NM. All of the mesh generation programs in the Engineering Analysis Department write files in the EXODUS format, and all of the postprocessing programs read EXODUS format files. Therefore, the two-dimensional mesh input to *GEN3D* can be generated by *FASTQ*, *PATRAN*, or any other two-dimensional mesh generator that writes the EXODUS file format. The output from *GEN3D* can be graphically displayed by *BLOT* [9] or *PATRAN*, and examined by *GROPE* [10] or *NUMBERS* [18]. The three-dimensional mesh generated by *GEN3D* can be joined to other three-dimensional meshes using *GJOIN*.

1.1 Terminology

GEN3D generates a three-dimensional mesh by transforming a two-dimensional mesh. The available transformations are:

translation For a translation transformation, the two-dimensional mesh is displaced a specified amount in the negative Z direction[1]. The three-dimensional mesh is then generated in the volume between the original mesh and the displaced mesh.

rotation The rotation transformation generates the three-dimensional mesh by rotating the two-dimensional mesh about an axis parallel to the Y coordinate axis. The three-dimensional mesh is then generated in the volume "swept" out by the rotating two-dimensional mesh.

warping The warping transformation maps the two-dimensional mesh onto a circular surface with a specified radius of curvature. Two surfaces can be specified: a spherical surface where the center of curvature is a point, and a cylindrical surface where the center of curvature is a line that lies in either of the planes defined by the equations $X = 0$ or $Y = 0$. The first warp type is called "point warp" and the second is called "axis warp" in this document. The warped two-dimensional mesh is then translated normal to the surface to form the three-dimensional mesh.

[1] the two-dimensional mesh is assumed to lie in the X–Y coordinate plane

The term "level" refers to a single translation or rotation. Three-dimensional nodes and elements that are created by doing a single translation or rotation of the two-dimensional nodes and elements are said to be "generated from" the two-dimensional nodes and elements. Normally, one three-dimensional element is generated per level from each two-dimensional element. One three-dimensional node is generated per level from each two-dimensional node and an additional node at the back side of the last level (unless the three-dimensional mesh is rotated 360 degrees). For example, Figure 1.1 shows a two-dimensional mesh with 28 elements and 40 nodes. Figure 1.2 shows the same mesh translated 10 times. The mesh has 10 levels; 10 elements are generated from each two-dimensional element resulting in 280 elements; 11 nodes are generated from each two-dimensional node resulting in 440 nodes.

Each element in the two-dimensional database must be a four-node quadrilateral. The three-dimensional elements generated from the two-dimensional four-node quadrilaterals are eight-node hexahedrons.

Figure 1.1 shows a two-dimensional rectangular mesh. Figures 1.2, 1.3, 1.4 and 1.5 display three-dimensional meshes generated from the input mesh. Figure 1.2 shows the result of a translation, Figure 1.3 shows a rotation around an axis outside the mesh, Figure 1.4 shows a rotation around an axis at the edge of the mesh, and Figure 1.5 shows a warp about the Y axis.

1.2 Element Blocks

The GENESIS database groups all elements in the mesh into element blocks. Each block represents a different element type and/or material.

If the mesh is translated, *GEN3D* allows the user to specify a "tunnel" block to implement a material change through the levels of an input element block. A tunnel block is transformed into several three-dimensional element blocks, each consisting of the elements in one or more levels. The user specifies the starting and ending levels and the number of levels in a three-dimensional block with the **TUNNEL** command. The tunnel block is commonly used to analyze the sequential mining of a drift in a geomechanics problem by "killing" the elements in the tunnel blocks as a function of time.

If the mesh is rotated around the edge of the mesh and the center elements are handled like other elements, the center elements become six-node wedge-shaped elements and new nodes are overlaid on the center nodes. To eliminate this problem, the element blocks which border the center of rotation must be identified (with the **CENTER** command) and specially handled. Only one three-dimensional element is generated from each two-dimensional element bordering the center for each 90-degree quadrant. As

the elements get farther from the center, the number of elements generated from each two-dimensional element increases. Eventually the elements merge with the normal rotated elements. Figure 1.4 shows a rotation around the edge of the mesh.

Elements in a center block (CENTER command) must conform to certain restrictions. An element borders the center if and only if its minimum X coordinate is equal to the minimum for the mesh. The elements adjacent to the centermost elements are found by searching the element connectivity. The adjacent elements must have the same connectivity structure. For example, the lower left node in each element must appear in the same slot in the connectivity sequence. The mesh must be rotated a complete quadrant (90, 180, 270, or 360 degrees) and the number of rotations per quadrant must be a multiple of 2. If *nrot/nquad* is the number of rotations per quadrant, there must be at least $nrot/(2nquad)$ elements along the X axis in the center block.

1.3 Front and Back Sets

Sets of nodes and sets of element sides may be defined in a GENESIS database. The two-dimensional sets are transformed into three-dimensional sets and repeated for each level. The user may create sets of only the front or back of the three-dimensional mesh. A front node set includes all of the two-dimensional nodes. Similarly, a front side set includes all of the four-node faces which make up the two-dimensional elements. A back node or side set includes the nodes or element faces on the back side of the last level. A back node or side set cannot be defined if the three-dimensional mesh is rotated 360 degrees since the generated shape has no "back"; however, front node or side sets may be defined.

1.4 Three-Dimensional Output

The three-dimensional nodes are numbered so that all the three-dimensional nodes generated from a two-dimensional node are numbered consecutively. The numbering of the three-dimensional elements depends on the element block type. For tunnel blocks, all elements in each level are grouped together with the elements in the next level. For non-tunnel blocks, all the three-dimensional elements generated from a two-dimensional element are grouped together.

In general, all three-dimensional items (nodes or elements or sets) generated from a two-dimensional item for each level are processed before proceeding to the next item. This is true for the nodal coordinates, the element order map, the element connectivity (for non-tunnel blocks), the node sets, and the side sets.

Translation Transformation: For translations, the X and Y coordinates are copied from each two-dimensional node to the three-dimensional nodes generated from it; the Z coordinate is zero for nodes on the first level and is decreased for the nodes on each successive level.

Rotation Transformation: For rotations, the X and Z coordinates are functions of the X coordinate of the two-dimensional node and the angle of each level. The rotation is counterclockwise if looking down the Y axis. The center of rotation offsets the X coordinate; the X coordinates of nodes on the first level are copied from the two-dimensional nodes. The Y coordinates from the two-dimensional nodes are copied to the generated three-dimensional nodes. The X and Z coordinates are:

$$\begin{aligned} Z &= -(x_0 - x_c)sin\theta_l \\ X &= (x_0 - x_c)cos\theta_l + x_c \end{aligned}$$

where θ_l is the rotation angle of level l, x_c is the center of rotation, and x_0 is the x coordinate of the two-dimensional node.

Warping Transformation: Two types of warping transformations are available: point-centered warp and axis-centered warp. In the warping transformations, the coordinates are functions of the type of warping and the edge type. The point-centered warp maps the two-dimensional mesh onto a spherical surface. The X and Y coordinates from the two-dimensional mesh are copied to the generated three-dimensional mesh and the Z coordinate is calculated from the radius of curvature. A cylindrical surface is generated by the axis-centered warp. The two-dimensional coordinates corresponding to the warping axis are copied to the three-dimensional mesh; the other two-dimensional coordinate and the Z coordinate are calculated from the radius of curvature such that the distance to the three-dimensional node measured along the generated surface is equal to the original distance in the two-dimensional mesh. The descriptions of the warp commands in the following chapter describe the warping procedure in more detail.

Gradient: A gradient can be specified for each of the transformation options. The gradient affects the spacing of the levels. The displacement or thickness of level i is t_i where:

$$\begin{aligned} t_1 &= \begin{cases} tottran \times \frac{(grad-1)}{(grad^{ntran}-1)} & \text{if } grad \neq 1; \\ tottran/ntran & \text{if } grad = 1. \end{cases} \\ t_i &= t_1 \times grad^{i-1} \end{aligned}$$

where *tottran* is the total translation, *grad* is the gradient, and *ntran* is the total number of translations. For rotations, *tottran* is the angle of rotation and t_i is the rotation of level i.

Orientation: After the three-dimensional mesh has been generated, the final position and orientation of the mesh are calculated by revolving, offsetting, reflecting, or zeroing. The specified revolutions are performed, followed by the coordinate offsets, and then it is reflected about the specified coordinate axes. All nodal coordinates less than the specified tolerance are then zeroed.

Element Attributes: The element attributes from the two-dimensional element are copied to the three-dimensional elements generated from it.

Side and Node Sets: Each two-dimensional node set is repeated on each level and the back side of the last level (if any). Each side of an two-dimensional side set is expanded to four nodes by adding the adjacent two nodes on the next level, and each set is repeated on each level.

Other Records: QA records and coordinate and element block names are only written to the output database if they existed on the input database. A QA record for the *GEN3D* program is added to the input QA record(s). The coordinate names are "X", "Y", and "Z". The element block names from the input database should all be "QUAD". If so, the names are changed to "HEX".

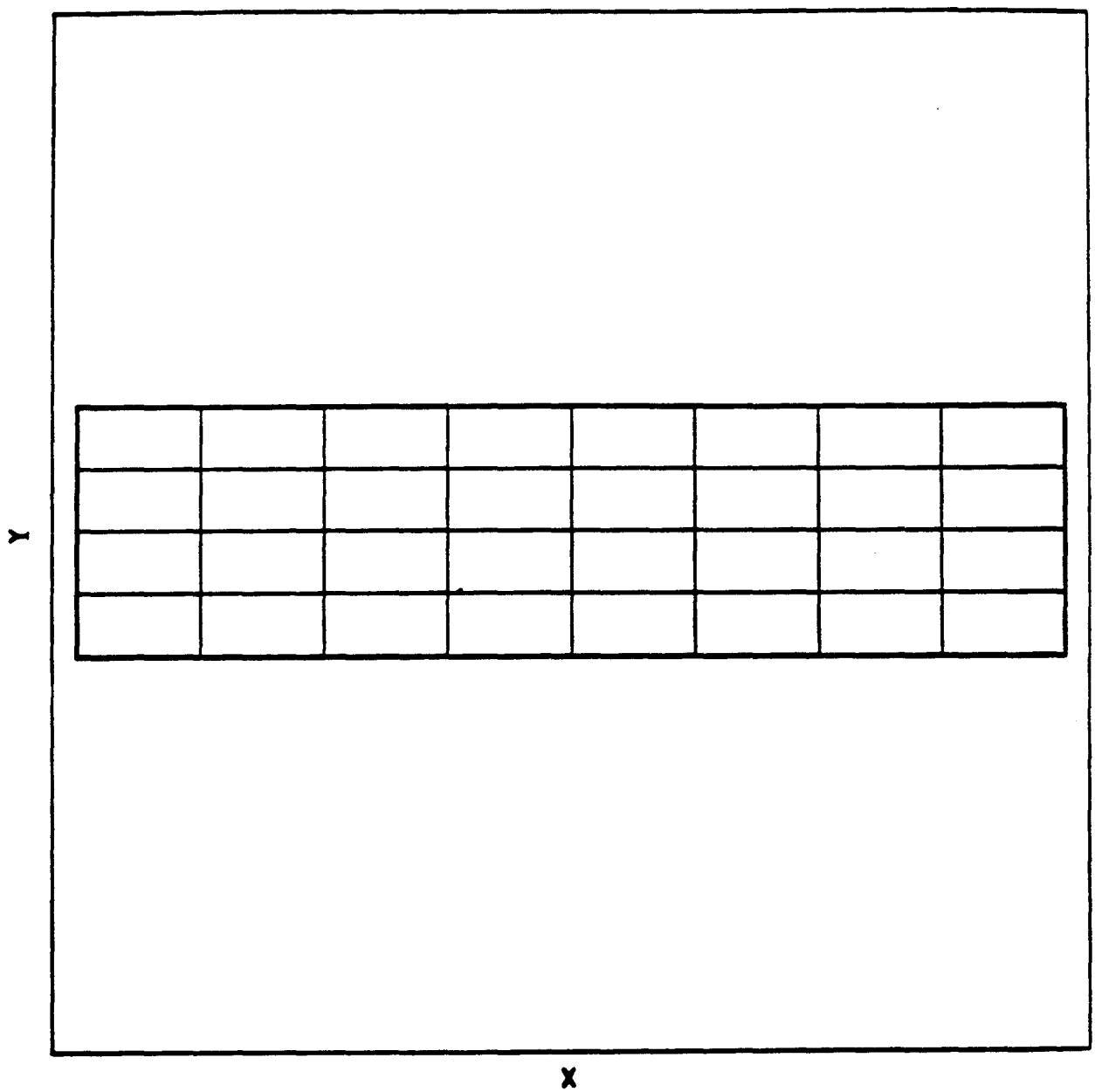

Figure 1.1. Simple 2D mesh input to *GEN3D*

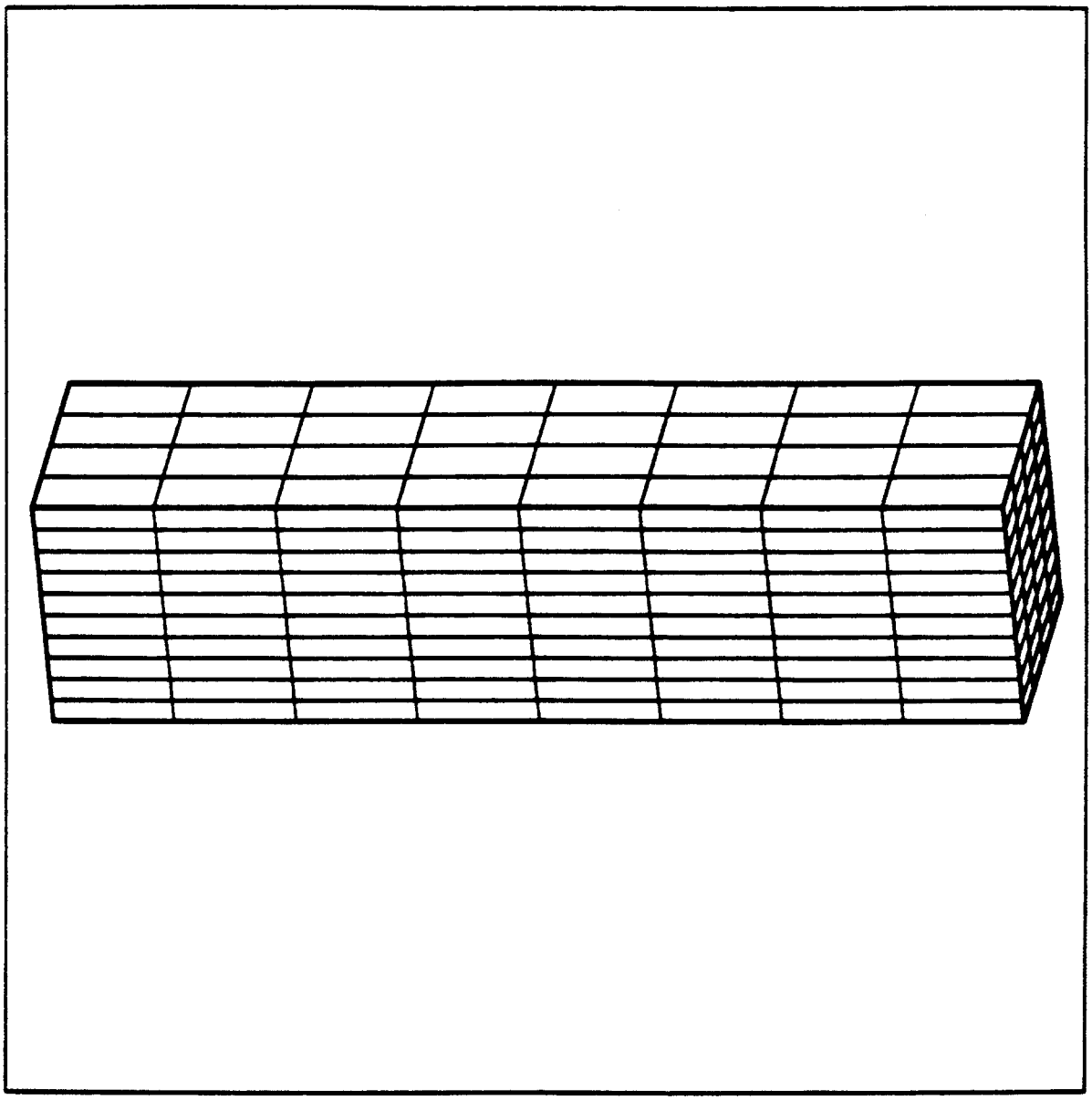

Figure 1.2. Translation

Figure 1.2 shows the mesh in Figure 1.1 translated 10 levels. The mesh was created with the following commands:

```
TRANSLATE 10, 0.1
END
```

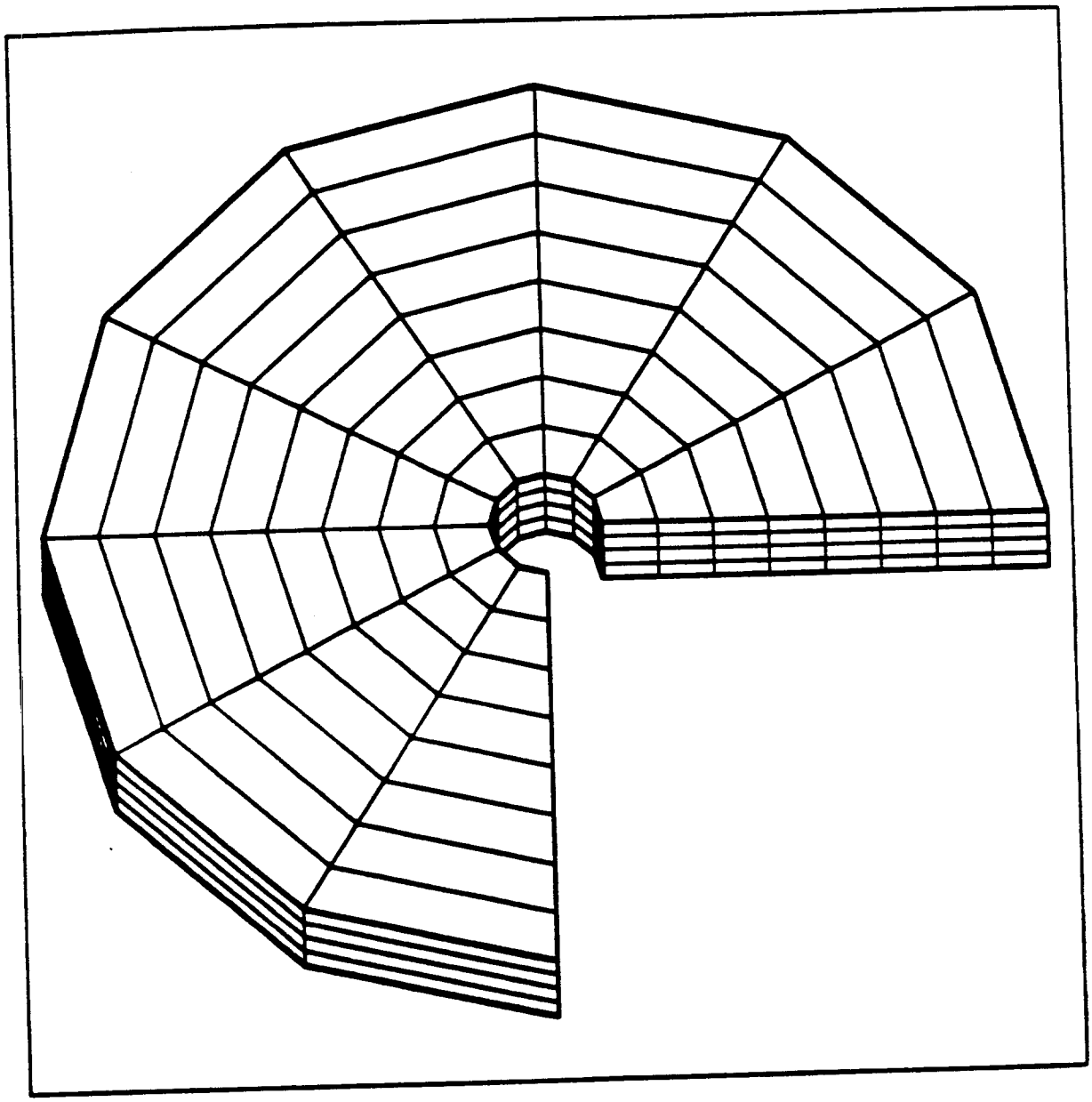

Figure 1.3. Rotation around line outside mesh

Figure 1.3 shows the mesh in Figure 1.1 rotated 9 levels for a total of 270 degrees. The center of rotation is slightly outside the mesh edge. The mesh was created with the following commands:

```
ROTATE 9, 270, 1, -0.05
END
```

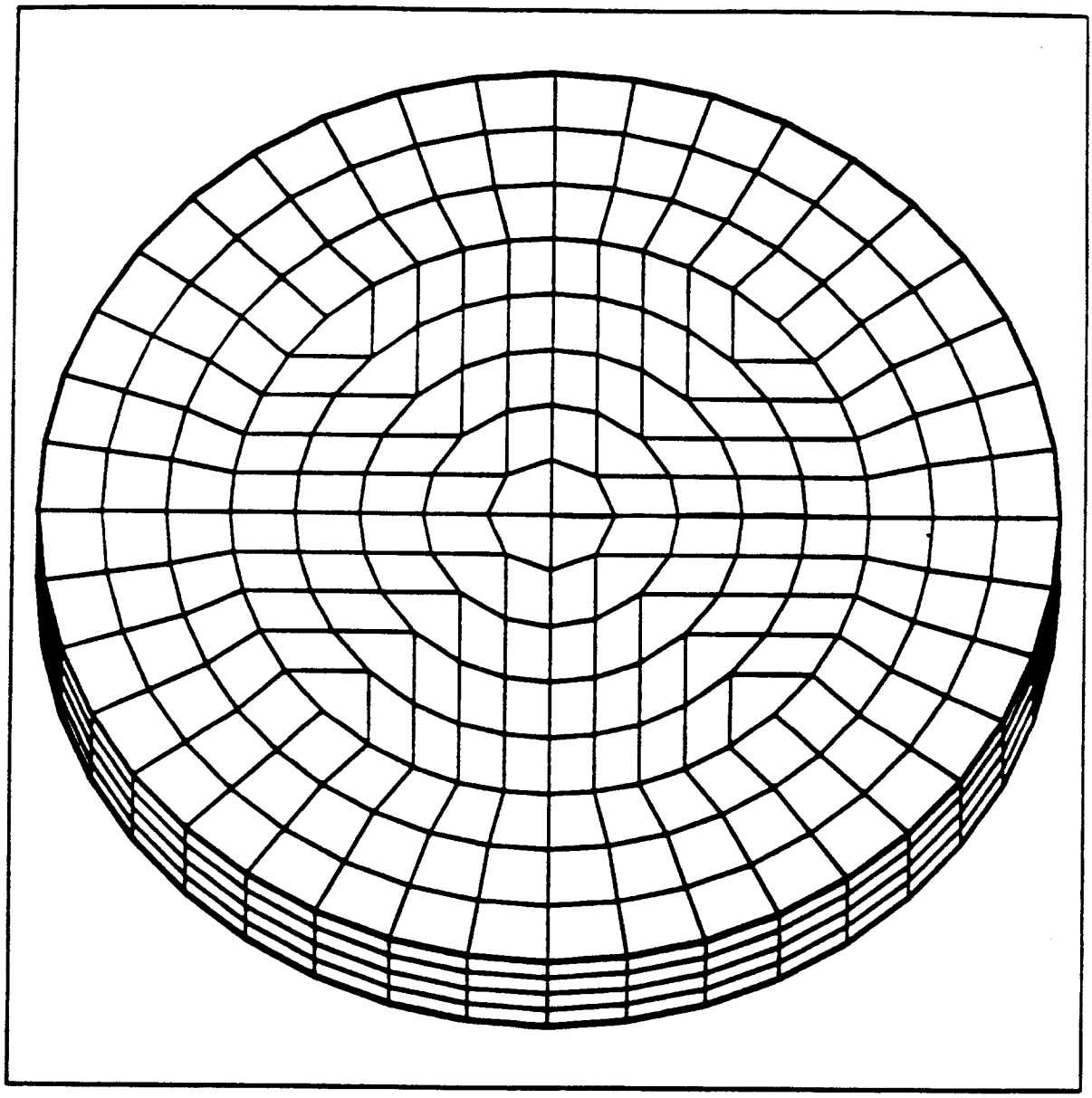

Figure 1.4. Rotation around mesh edge

Figure 1.4 shows the mesh in Figure 1.1 rotated 40 levels for a total of 360 degrees. The center of rotation is at the edge of the mesh. The mesh was created with the following commands:

```
ROTATE 40, 360
CENTER 1
END
```

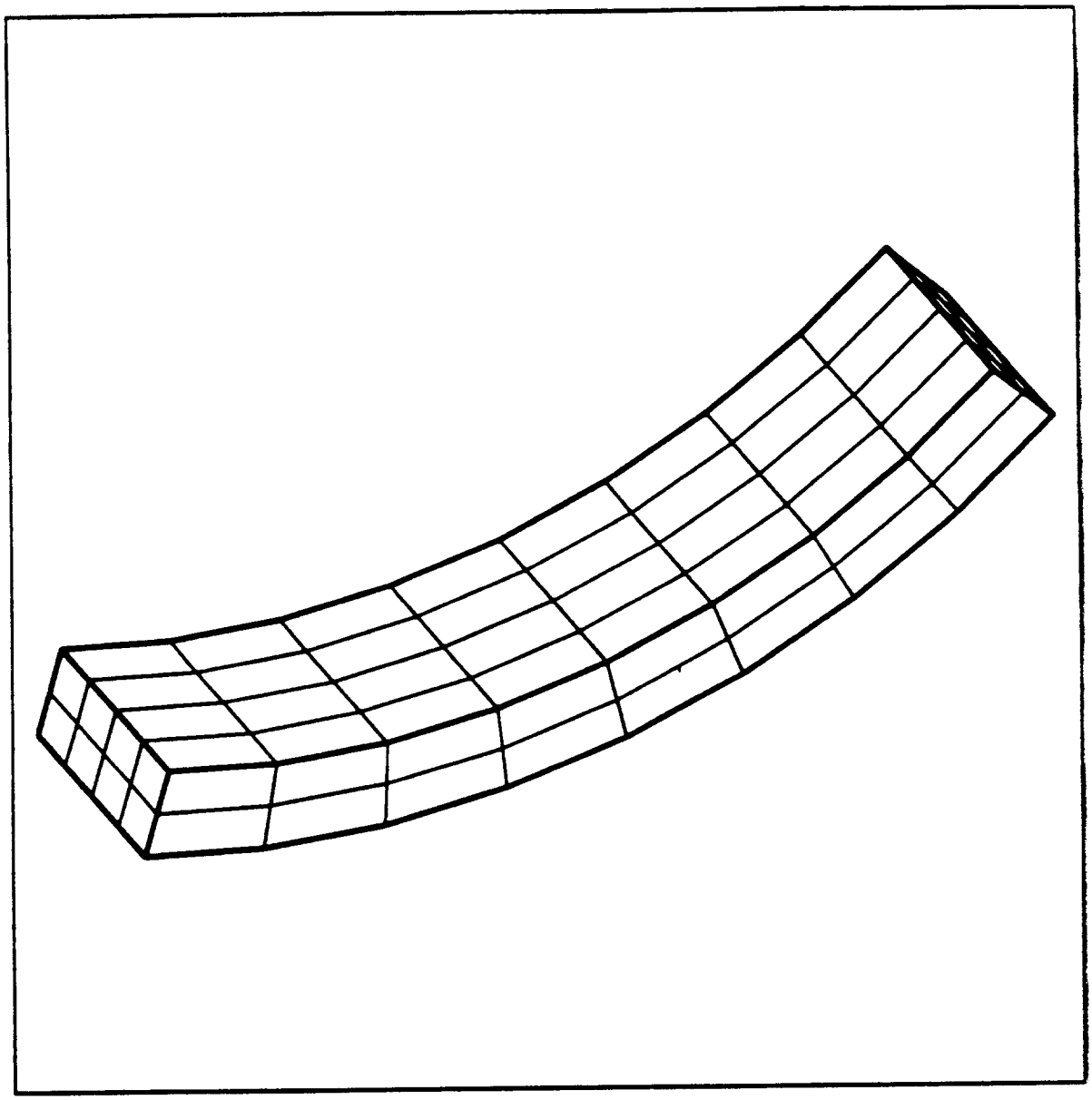

Figure 1.5. Warp about the Y axis

Figure 1.5 shows the mesh in Figure 1.1 warped about the Y axis for 2 levels for a total of 0.04 with a radius of curvature of 0.4. The mesh was created with the following commands:

```
WARP Y 2 .04 1 0.4
END
```

2. Command Input

The user directs the processing by entering commands to set processing parameters. The commands are in free-format and must adhere to the following syntax rules.

- Valid delimiters are a comma or one or more blanks.
- Either lowercase or uppercase letters are acceptable, but lowercase letters are converted to uppercase.
- A "$" character in any command line starts a comment. The "$" and any characters following it on the line are ignored.
- A command may be continued over several lines with an ">" character. The ">" and any characters following it on the current line are ignored and the next line is appended to the current line.

Each command has an action keyword or "verb" followed by a variable number of parameters.

A command verb or keyword is a character string matching one of the valid commands. It may be abbreviated as long as enough characters are used to distinguish it from other commands.

The meaning and type of the parameters depends on the command verb. Most parameters are optional. If an optional parameter field is blank, a command-dependent default value is supplied. Valid entries for parameters are:

- A numeric parameter may be a real number or an integer. A real number may be in any legal FORTRAN numeric format (e.g., 1, 0.2, -1E-2). An integer parameter may be in any legal integer format.
- A string parameter is a literal character string. Most string parameters may be abbreviated.

The notation conventions used in the command descriptions are:

- The command verb is in **bold** type.
- A literal string is in all uppercase SANSERIF type and should be entered as shown (or abbreviated).
- The value of a parameter is represented by the parameter name in *italics.*

- The default value of a parameter is in angle brackets ("< >"). The initial value of a parameter set by a command is usually the default parameter value. If not, the initial setting is given with the default or in the command description.

2.1 Mesh Transformation

TRANSLATE *ntran* <1>, *tottran* <1.0>, *grad* <1.0>, ...

TRANSLATE causes the 2D mesh to be translated to create the 3D mesh. The number of levels is *ntran*, which is also the number of 3D elements derived from each input 2D element. The total range of the Z coordinate is *tottran* with a gradient of *grad*. The translation is always in the negative Z direction. This command supersedes previous transformation commands.

The gradient affects the spacing of the levels. The displacement or thickness of level i is z_i where:

$$z_1 = \begin{cases} tottran \times \frac{(grad-1)}{(grad^{ntran}-1)} & \text{if } grad \neq 1; \\ tottran/ntran & \text{if } grad = 1. \end{cases}$$

$$z_i = z_1 \times grad^{i-1}$$

Multiple translation increments can be specified with a single translate command by repeating the *ntran*, *tottran*, *grad* parameters on the command line. For example, the following command specifies two translation increments of thickness 1.0 for a total translation of 2.0:

TRANSLATE 5 1.0 0.5, 5 1.0 2.0

All increments must be specified with a single TRANSLATE command.

ROTATE *nrot* <1>, *totdeg* <360.0>, *grad* <1.0>, *cenrot* <0.0>

ROTATE causes the 2D mesh to be rotated to create the 3D mesh. The number of rotation levels is *nrot*, which is also the number of 3D elements derived from each input 2D element (with the exception of those affected by the CENTER command). The mesh is rotated a total of *nrot* rotations through a total arc of *totdeg* degrees. The angle of each rotation is equal to *grad* times the previous rotation. The center of rotation *cenrot* and the gradient *grad* are only meaningful if no center element blocks are defined (see the CENTER command).

The gradient affects the rotation of the levels. The angular rotation of level i is θ_i where:

$$\theta_1 = \begin{cases} totdeg \times \frac{(grad-1)}{(grad^{nrot}-1)} & \text{if } grad \neq 1; \\ totdeg/nrot & \text{if } grad = 1. \end{cases}$$

$$\theta_i = \theta_1 \times grad^{i-1}$$

Rotation is always counterclockwise. This command supersedes previous transformation commands.

WARP POINT *ntran* <1>, *tottran* <1.0>, *grad* <1.0>, *radius* <no default>, *edge_type* <RADIAL>

WARP POINT causes the 2D mesh to be mapped onto a spherical surface to create the 3D mesh. The spherical surface has a radius of curvature equal to *radius*. The center of curvature is located on the Z-axis, and it is a distance of *radius* above the X-Y plane. The number of levels is *ntran*, which is also the number of 3D elements derived from each input 2D element. The total thickness (measured radially) is *tottran* with a gradient of *grad*. Note that *radius* must be greater than the maximum distance from the Z-axis to the boundary of the 2D mesh. This command supersedes previous transformation commands.

The *edge_type*, which can be either VERTICAL or RADIAL, determines how the created nodes are generated. If VERTICAL is selected, the X and Y coordinates of the generated nodes are equal to the X and Y coordinates of the original 2D node. If RADIAL is selected, the X and Y coordinates of the generated nodes are calculated to lie on a line from the center of curvature (0.0, 0.0, *radius*) to the coordinates of the warped node (x_w, y_w, z_w) where x_w and y_w are the coordinates of the original 2D node, and z_w is determined such that the distance from the warped node to the center of curvature is equal to *radius*. Figure 2.1 illustrates the VERTICAL edge type, and Figure 2.2 illustrates the RADIAL edge type.

The mesh transformation is performed in two parts. First, the warped nodal positions (x_w, y_w, z_w) are calculated by mapping the original 2D mesh onto a spherical surface with a radius of curvature equal to *radius*. The original X and Y coordinates of the 2D mesh remain at the same values; the Z coordinate is calculated such that the distance to the center of curvature is equal to *radius*.

$$\begin{aligned} x_w &= x_0 \\ y_w &= y_0 \\ z_w &= radius - \sqrt{radius^2 - x_0^2 - y_0^2} \end{aligned}$$

The warped nodal positions are projections parallel to the Z-axis onto a spherical surface of radius *radius*; Figure 2.1 illustrates this process. Then, the generated nodal positions are determined by translating either vertically or radially from the warped nodal position. A total of *ntran* translations are performed through a distance of *tottran* with a gradient of *grad*. Note that the thickness is measured radially for either *edge_type*.

The gradient affects the spacing of the levels. The thickness or length of level i

is z_i where:

$$z_1 = \begin{cases} tottran \times \frac{(grad-1)}{(grad^{ntran}-1)} & \text{if } grad \neq 1: \\ tottran/ntran & \text{if } grad = 1. \end{cases}$$

$$z_i = z_1 \times grad^{i-1}$$

WARP *axis* <no default>, *ntran* <1>, *tottran* <1.0>, *grad* <1.0>, *radius* <no default>, *edge_type* <RADIAL>

This second form of the WARP command maps the 2D mesh to a cylindrical surface centered on the *axis*-axis to create the 3D mesh. The *axis* parameter must be either X or Y. The cylindrical surface has a radius of curvature equal to *radius*. The center of curvature is located a distance of *radius* above the X-Y plane. The number of levels is *ntran*, which is also the number of 3D elements derived from each input 2D element. The total thickness (measured radially) is *tottran* with a gradient of *grad*. This command supersedes previous transformation commands.

The *edge_type*, which can be either VERTICAL or RADIAL, determines how the created nodes are generated. If VERTICAL is selected, the X and Y coordinates of the generated nodes are equal to the X and Y coordinates of the projected 2D node. If RADIAL is selected, the X and Y coordinates of the generated nodes are calculated to lie on a line from the center of curvature to the coordinates of the warped node (x_w, y_w, z_w) where x_w, y_w, and z_w are the coordinates of the mapped 2D node.

The mesh transformation is performed in two parts. First, the warped nodal positions (x_w, y_w, z_w) are calculated by mapping the original 2D mesh onto a cylinder about the *axis*-axis with a radius of curvature equal to *radius*. If *axis* is X, then the original X-coordinate remains at the same value. The generated Y and Z coordinates are calculated such that the distance from the generated node to the X-Z plane measured along the cylindrical surface is equal to the X coordinate of the node in the 2D mesh. This is illustrated in Figure 2.2. If *axis* is Y, the X's and Y's are switched in the above discussion. Then, the generated nodal positions are determined by translating either vertically or radially from the warped nodal position. A total of *ntran* translations are performed through a distance of *tottran* with a gradient of *grad*. Note that the distance is measured radially for either *edge_type*.

The gradient affects the spacing of the levels. The thickness or length of level i

is z_i where:

$$z_1 = \begin{cases} tottran \times \frac{(grad-1)}{(grad^{ntran}-1)} & \text{if } grad \neq 1; \\ tottran/ntran & \text{if } grad = 1. \end{cases}$$

$$z_i = z_1 \times grad^{i-1}$$

The resulting 3D mesh will have an cylindrical angle of $x_{\mathbf{max}}/radius$ radians if warped about the Y axis, or $y_{\mathbf{max}}/radius$ radians if warped about the X axis, where $x_{\mathbf{max}}$ and $y_{\mathbf{max}}$ are the maximum x and y coordinates in the 2D mesh.

2.2 Mesh Orientation

REVOLVE $axis_1$, $ndeg_1$, $axis_2$, $ndeg_2$, ... <last selection>

REVOLVE RESET <initial condition>

REVOLVE causes the transformed 3D mesh to be rotated. Each (*axis*, *ndeg*) parameter pair specifies an axis (X or Y or Z) and the number of degrees to rotate. The axis refers to the "viewing" axis, not to the object axis. The rotations are according to right-hand rule. The center of the rotation is specified by the REVCEN command.

Revolutions are cumulative; however, only one center of revolution may be specified. The REVOLVE RESET command resets to no rotation.

REVCEN *xcen* <2D minimum X coordinate>, *ycen* <2D minimum Y coordinate>, *zcen* <0.0>

REVCEN sets the center of revolution for the REVOLVE command to the point (*xcen*,*ycen*,*zcen*).

OFFSET *xoff* <0.0>, *yoff* <0.0>, *zoff* <0.0>

OFFSET specifies offsets to be added to the transformed 3D coordinates. If a REVOLVE command has been issued, the 3D mesh is rotated before it is offset.

MIRROR $axis_1$, $axis_2$, ... <no default>

MIRROR RESET <no reflections>

MIRROR causes the transformed 3D mesh to be reflected about a coordinate plane. Each *axis* parameter specifies an axis (X or Y or Z) which is the normal to the reflection plane. Reflections are performed after the mesh has been repositioned by the REVOLVE and OFFSET commands.

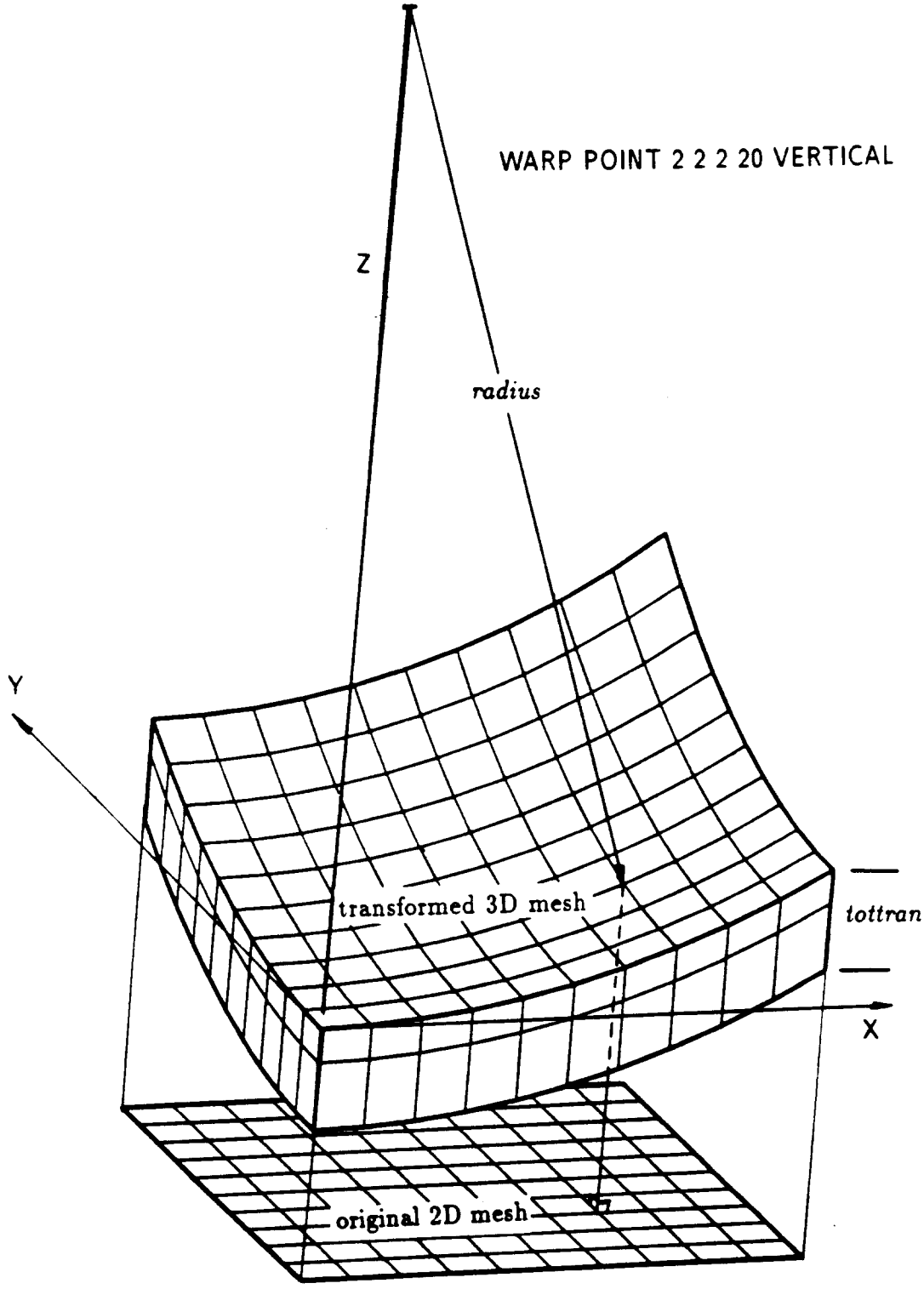

Figure 2.1. Illustration of Projection Procedure for the WARP POINT Command

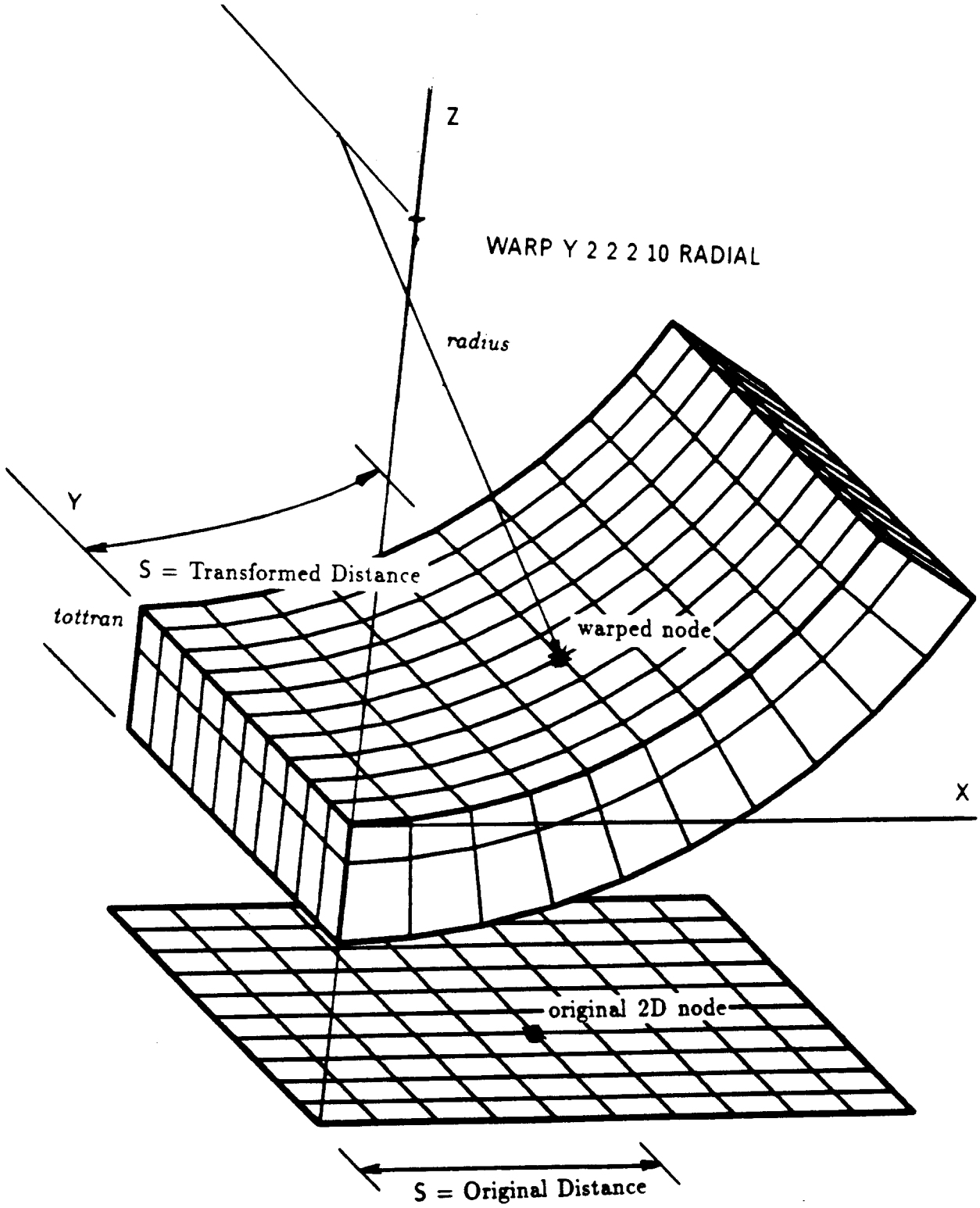

Figure 2.2. Illustration of Mapping Procedure for the **WARP** *axis* Command

The MIRROR RESET command resets to no reflection.

Reflections are not cumulative, that is, if MIRROR X Y X is entered, only one reflection about the X axis will be performed. If an odd number of reflections are performed, the element connectivity and the sideset face numberings will be correctly reordered.

ZERO $axis_1$, min_1, $axis_2$, min_2, ...

ZERO RESET <no automatic zeroing>

ZERO sets all $axis_i$ coordinates with an absolute value less than min_i equal to zero. The ZERO RESET command resets to no automatic zeroing. This command is used to zero nodal coordinates that should be equal to zero, but due to roundoff errors they have slightly nonzero values.

2.3 Element Block Types

Each element block is assigned one of the following block types:

- A normal block requires no special handling.
- A tunnel block (translation only) changes materials as it is translated.
- A center block (rotation around mesh edge only) has some elements which border the mesh edge that is the center of rotation.

Initially, all blocks are normal blocks.

The *block_id* parameter below refers to the element block identifier. The identifiers are displayed by the SHOW BLOCKS command.

BLOCK *block_id*$_1$, *block_id*$_2$, ... <all element blocks>

BLOCK defines the specified element blocks as normal blocks. This command supersedes any previous TUNNEL or CENTER commands.

TUNNEL *block_id* <no default>, *start* <1>, *end* <number of levels>, *inc* <1>

TUNNEL defines the specified element block as a tunnel block. A TRANSLATE command must be in effect before this command is issued. If a ROTATE command is issued, all tunnel blocks are changed to normal blocks.

For each tunnel block, a separate 3D element block is created starting at level *start*, with each block having *inc* levels. Any levels after level *end* are put in a single block.

For example, the commands

```
TRANSLATE 15, 15.0
TUNNEL 999, 5, 9, 2
```

create five blocks consisting of the following 3D elements (derived from the 2D elements of element block 999):

1) the elements in levels 1, 2, 3, and 4,
2) the elements in levels 5 and 6,
3) the elements in levels 7 and 8,
4) the elements in level 9,
5) the elements in levels 10, 11, 12, 13, 14, and 15.

The block identifier of the first block is always *block_id*. The new blocks are assigned consecutive identifiers greater than the maximum existing (and new) identifier.

CENTER *block_id*$_1$, *block_id*$_2$, ... <all element blocks>

CENTER defines the specified element blocks as center blocks. A ROTATE command must be in effect before this command is issued. The mesh must be rotated a complete quadrant (90, 180, 270 or 360 degrees) and the number of rotation levels must be a multiple of 2 for each 90 degrees of rotation. If *nrot* is the number of rotations, there must be at least *nrot*/2 elements along the X axis in the center block.

If a TRANSLATE command is issued, all center blocks are changed to normal blocks.

If center blocks are defined, the center of rotation defined by the ROTATE command is ignored. The center of rotation is the minimum coordinate of all elements in the center blocks.

2.4 Front and Back Set Definition

NSETS FRONT or BACK <no default>, set_id_1, set_id_2, ... <no default>

NSETS defines front or back node sets with the given identifiers. The identifiers must be unique from existing node set identifiers and previously defined front and back node set identifiers.

Back sets cannot be defined on a 360-degree rotation.

SSETS FRONT or BACK <no default>, set_id_1, set_id_2, ... <no default>

SSETS is equivalent to the NSETS command except that it defines side sets.

2.5 Information and Processing

SHOW *command* <no parameter>

SHOW displays the settings of parameters relevant to the *command.* For example. the command SHOW BLOCK displays information about all the element blocks.

LIST VARS

LIST VARS displays a summary of the input database. The summary includes the database title; the number of nodes, elements, and element blocks; and the number of node sets and side sets.

HELP *command* <no parameter>

HELP displays information about the program command given as the parameter. If no parameter is given, all the command verbs are displayed. This command is system-dependent and may not be available on some systems.

END

END ends the command input and starts the database transformation.

QUIT

QUIT ends the command input and exits the program immediately without writing an output database.

3. Informational and Error Messages

GEN3D first reads the input database, which must be a valid 2D GENESIS database. If a database format error is discovered, the program prints an error of the following format:

DATABASE ERROR - READING *database item*

and aborts.

After the database is read, command input is requested from the user. An error or warning message may appear in response to a command. If an error message appears, the command is usually ignored. If only a warning is printed, the command is usually performed. If the message is not sufficiently informative, the appropriate command description may be helpful. The display after the command shows the effect of the command.

When the command input is complete, GEN3D transforms the 2D mesh and writes the 3D database. The transformation is explained in Section 1.4.

The program allocates memory dynamically as it is needed. If the system runs out of memory, the following message is printed:

FATAL ERROR - TOO MUCH DYNAMIC MEMORY REQUESTED

and the program aborts. The user should first try to obtain more memory on the system. Another solution is to run the program in a less memory-intensive fashion. For example, reducing the number of transformations requires less memory.

GEN3D has certain programmer-defined limitations. The limits are not specified in this manual since they may change. In most cases the limits are chosen to be more than adequate. If the user exceeds a limit, a message is printed. If the user feels the limit is too restrictive, the code sponsor should be notified so the limit may be raised in future releases of GEN3D.

4. Executing GEN3D

The details of executing GEN3D are dependent on the system being used. The system manager of any system that runs GEN3D should provide a supplement to this manual that explains how to run the program on that particular system. Site supplements for all currently supported systems are in Appendix C.

4.1 Execution Files

The table below summarizes GEN3D's file usage.

Description	Unit Number	Type	File Format
User input	standard input	input	Section 2
User output	standard output	output	ASCII
GENESIS database	9	input	Appendix A
GENESIS database	10	output	Appendix A

All files must be connected to the appropriate unit before GEN3D is run. Each file (except standard input and output) is opened with the name retrieved by the EXNAME routine of the SUPES library [8].

4.2 Special Software

GEN3D is written in ANSI FORTRAN-77 [1] with the exception of the following system-dependent features:

- the VAX VMS help facility and
- the OPEN options for the files.

GEN3D uses the following software package:

- the SUPES package [8] which includes dynamic memory allocation, a free-field reader, and FORTRAN extensions.

References

[1] *American National Standard Programming Language FORTRAN.* Tech. Rep. ANSI X3.9-1978. American National Standards Institute, New York, 1978.

[2] *PDA/PATRAN-G Version 2.1 User's Guide.* PDA Engineering Software Products Division, Santa Ana, CA, 1986.

[3] Biffle, J. H., 1523, "Three-Dimensional Mesh Generation with GJOIN," (Memo to 1520 Staff), Sandia National Laboratories, Albuquerque, NM, October 7, 1988.

[4] Blacker, T. D., "FASTQ Users Manual, Version 2.1," SAND88-1326, Sandia National Laboratories, Albuquerque, NM, July 1988.

[5] Blacker, T. D., Mitchiner, J. L., Phillips, L. R., and Lin, Y. T., "Knowledge System Approach to Automated Two-Dimensional Quadrilateral Mesh Generation," in *Proceedings of ASME Computers in Engineering Conference*, pp. 153-162, 1988.

[6] Blacker, T. D., Stephenson, M. B., Mitchiner, J. L., Phillips, L. R., and Lin, Y. T., "Automated Quadrilateral Mesh Generation: A Knowledge System Approach," in *Proceedings of ASME Winter Annual Meeting*, 1988.

[7] Chavez, P. F., Henderson, M., and Razdan, A., "Automatic Three-Dimensional Finite Element Modeling Using Solid Model Data and Artificial Intelligence Techniques," in *Proceedings of 1988 ASME International Computers in Engineering Conference and Exhibition*, August 1988.

[8] Flanagan, D. P., Mills-Curran, W. C., and Taylor, L. M., "SUPES A Software Utilities Package for the Engineering Sciences," SAND86-0911, Sandia National Laboratories, Albuquerque, NM, September 1986.

[9] Gilkey, A. P., "BLOT—A Mesh and Curve Plot Program for the Output of a Finite Element Analysis," SAND88-1432, Sandia National Laboratories, Albuquerque, NM. In preparation.

[10] Gilkey, A. P., "GROPE - A GENESIS/EXODUS Database Examination Program," (RS1523/88/02), Sandia National Laboratories, Albuquerque, NM. In preparation.

[11] Jones, R. E., "Users Manual for QMESH, A Self-Organizing Mesh Generation Program," SLA-74-0239, Sandia National Laboratories, Albuquerque, NM, 1974.

[12] Koteras, J. R., "Lamination: A Method for Restructuring Quadrilateral and Hexahedral Meshes," SAND88-3411, Sandia National Laboratories, Albuquerque, NM, February 1989.

[13] Koteras, J. R. and Blacker, T. D. Communication to G. D. Sjaardema.

[14] Koteras, J. R., Selleck, C. B., and Jones, R. E., "An Algorithm for Automated Quadrilateral Mesh Generation for Planar Regions," SAND86-2059, Sandia National Laboratories, Albuquerque, NM, January 1989.

[15] Mills-Curran, W. C., Gilkey, A. P., and Flanagan, D. P., "EXODUS: A Finite Element File Format for Pre- and Post-processing," SAND87-2977, Sandia National Laboratories, Albuquerque, NM, September 1988.

[16] Schutt, J. A., 1513, "Comments on Transferring Model Geometry Definitions for Thermal Analysis," (Memo to D. L. Moore, 2857), Sandia National Laboratories, Albuquerque, NM, April 6, 1989.

[17] Sjaardema, G. D., "Finite Element Analysis on Microcomputers: Code Evaluation Report," SAND87-2885, Sandia National Laboratories, Albuquerque, NM, March 1988.

[18] Sjaardema, G. D., "NUMBERS, A Mass Properties Calculation Program for Finite Element Models," SAND88-0737, Sandia National Laboratories, Albuquerque, NM, In preparation.

[19] Taylor, L. M., Flanagan, D. P., and Mills-Curran, W. C., "The GENESIS Finite Element Mesh File Format," SAND86-0910, Sandia National Laboratories, Albuquerque, NM, May 1986.

A. The GENESIS Database Format

The following code segment reads a GENESIS database.

```
C   --Open the GENESIS database file

      NDB = 9
      OPEN (UNIT=NDB, ..., STATUS='OLD', FORM='UNFORMATTED')

C   --Read the title

      READ (NDB) TITLE
C       --TITLE - the title of the database (CHARACTER*80)

C   --Read the database sizing parameters

      READ (NDB) NUMNP, NDIM, NUMEL, NELBLK,
     &    NUMNPS, LNPSNL, NUMESS, LESSEL, LESSNL
C       --NUMNP - the number of nodes
C       --NDIM - the number of coordinates per node
C       --NUMEL - the number of elements
C       --NELBLK - the number of element blocks
C       --NUMNPS - the number of node sets
C       --LNPSNL - the length of the node sets node list
C       --NUMESS - the number of side sets
C       --LESSEL - the length of the side sets element list
C       --LESSNL - the length of the side sets node list

C   --Read the nodal coordinates

      READ (NDB) ((CORD(INP,I), INP=1,NUMNP), I=1,NDIM)

C   --Read the element order map (each element must be listed once)

      READ (NDB) (MAPEL(IEL), IEL=1,NUMEL)
```

```
C   --Read the element blocks

      DO 100 IEB = 1, NELBLK

C      --Read the sizing parameters for this element block

         READ (NDB) IDELB, NUMELB, NUMLNK, NATRIB
C         --IDELB - the element block identification (must be unique)
C         --NUMELB - the number of elements in this block
C         --   (the sum of NUMELB for all blocks must equal NUMEL)
C         --NUMLNK - the number of nodes defining the connectivity
C         --   for an element in this block
C         --NATRIB - the number of element attributes for an element
C         --   in this block

C      --Read the connectivity for all elements in this block

         READ (NDB) ((LINK(J,I), J=1,NUMLNK, I=1,NUMELB)

C      --Read the attributes for all elements in this block

         READ (NDB) ((ATRIB(J,I), J=1,NATRIB, I=1,NUMELB)

  100 CONTINUE
```

```
C   --Read the node sets

      READ (NDB) (IDNPS(I), I=1,NUMNPS)
C      --IDNPS - the ID of each node set
      READ (NDB) (NNNPS(I), I=1,NUMNPS)
C      --NNNPS - the number of nodes in each node set
      READ (NDB) (IXNNPS(I), I=1,NUMNPS)
C      --IXNNPS - the index of the first node in each node set
C      --   (in LTNNPS and FACNPS)

      READ (NDB) (LTNNPS(I), I=1,LNPSNL)
C      --LTNNPS - the nodes in all the node sets
      READ (NDB) (FACNPS(I), I=1,LNPSNL)
C      --FACNPS - the factor for each node in LTNNPS

C   --Read the side sets

      READ (NDB) (IDESS(I), I=1,NUMESS)
C      --IDESS - the ID of each side set
      READ (NDB) (NEESS(I), I=1,NUMESS)
C      --NEESS - the number of elements in each side set
      READ (NDB) (NNESS(I), I=1,NUMESS)
C      --NNESS - the number of nodes in each side set
      READ (NDB) (IXEESS(I), I=1,NUMESS)
C      --IXEESS - the index of the first element in each side set
C      --   (in LTEESS)
      READ (NDB) (IXNESS(I), I=1,NUMESS)
C      --IXNESS - the index of the first node in each side set
C      --   (in LTNESS and FACESS)

      READ (NDB) (LTEESS(I), I=1,LESSEL)
C      --LTEESS - the elements in all the side sets
      READ (NDB) (LTNESS(I), I=1,LESSNL)
C      --LTNESS - the nodes in all the side sets
      READ (NDB) (FACESS(I), I=1,LESSNL)
C      --FACESS - the factor for each node in LTNESS
```

A valid GENESIS database may end at this point or after any point described below.

```
C   --Read the QA header information

      READ (NDB, END=...) NQAREC
C      --NQAREC - the number of QA records (must be at least 1)

      DO 110 IQA = 1, MAX(1,NQAREC)
         READ (NDB) (QATITL(I,IQA), I=1,4)
C         --QATITL - the QA title records; each record contains:
C         --   1) analysis code name (CHARACTER*8)
C         --   2) analysis code qa descriptor (CHARACTER*8)
C         --   3) analysis date (CHARACTER*8)
C         --   4) analysis time (CHARACTER*8)
  110 CONTINUE

C   --Read the optional header text

      READ (NDB, END=...) NINFO
C      --NINFO - the number of information records

      DO 120 I = 1, NINFO
         READ (NDB) INFO(I)
C         --INFO - extra information records (optional) that contain
C         --   any supportive documentation that the analysis code
C         --   developer wishes (CHARACTER*80)
  120 CONTINUE

C   --Read the coordinate names

      READ (NDB, END=...) (NAMECO(I), I=1,NDIM)
C      --NAMECO - the coordinate names (CHARACTER*8)

C   --Read the element type names

      READ (NDB, END=...) (NAMELB(I), I=1,NELBLK)
C      --NAMELB - the element type names (CHARACTER*8)
```

B. Command Summary

Mesh Transformation (page 21)

TRANSLATE *ntran*, *tottran*, *grad*, ...

causes the 2D mesh to be translated to create the 3D mesh.

ROTATE *nrot*, *totdeg*, *grad*, *cenrot*

causes the 2D mesh to be rotated to create the 3D mesh.

WARP POINT *ntran*, *tottran*, *grad*, *radius*, *edge_type*

maps the 2D mesh onto a sphere of radius *radius* and then translates to create the 3D mesh.

WARP *axis*, *ntran*, *tottran*, *grad*, *radius*, *edge_type*

maps the 2D mesh onto a cylinder of radius *radius* about the *axis*-axis and then translates to create the 3D mesh.

Mesh Orientation (page 24)

REVOLVE $axis_1$, $ndeg_1$, $axis_2$, $ndeg_2$, ...
REVOLVE RESET

causes the transformed 3D mesh to be rotated.

REVCEN *xcen*, *ycen*, *zcen*

sets the center of rotation for the REVOLVE command.

OFFSET *xoff*, *yoff*, *zoff*

specifies the coordinate offsets for the transformed 3D mesh.

MIRROR $axis_1$, $axis_2$, ...
MIRROR RESET

causes the transformed 3D mesh to be reflected about the specified axes.

ZERO $axis_1$, min_1, $axis_2$, min_2, ...
ZERO RESET

sets all $axis_i$ coordinates with an absolute value less than min_i equal to zero.

Element Block Types (page 28)

BLOCK *block_id*$_1$, *block_id*$_2$, ...

defines element blocks as normal blocks (no special handling).

TUNNEL *block_id*, *start*, *end*, *inc*

defines an element block as a tunnel block, with tunnels starting at level *start* to level *end*, incrementing by *inc*.

CENTER *block_id*$_1$, *block_id*$_2$, ...

defines element blocks as center blocks (bordering the mesh edge that is the center of rotation).

Front and Back Set Definition (page 30)

NSETS FRONT/BACK, *set_id*$_1$, *set_id*$_2$, ...

defines front or back node sets.

SSETS FRONT/BACK, *set_id*$_1$, *set_id*$_2$, ...

defines front or back side sets.

Information and Processing (page 31)

SHOW *command*

displays the processing parameters set by a command.

LIST VARS

displays summary information about the input database.

HELP *command*

displays information about a GEN3D command.

END

ends command input and starts processing.

QUIT

quits command input and aborts processing.

C. Site Supplements

C.1 VAX VMS

The command to execute GEN3D on VMS is:

GEN3D *2D_database 3D_database user_input*

2D_database is the filename of the input GENESIS database. A prompt appears if *2D_database* is omitted. The default is TAPE9.GEN.

3D_database is the filename of the output GENESIS database. A prompt appears if *3D_database* is omitted. The default is the base filename of *2D_database* with the extension .GEN3D.

If *user_input* is given, the user input is read from this file. Otherwise it is read from SYS$INPUT (the terminal keyboard). User output is directed to SYS$OUTPUT (the terminal).

GEN3D operates in either interactive or batch modes.

C.2 CRAY CTSS

To execute GEN3D, the user must have selected the acclib library and be running ccl.

The command to execute GEN3D on CTSS is:

gen3d *2D_database 3D_database* i=*input* o=*output*

2D_database is the filename of the input GENESIS database. The default is tape9.

3D_database is the filename of the output GENESIS database. The default is tape10.

User input is read from *input*, which defaults to tty (the terminal). User output is directed to *output*, which defaults to tty (the terminal).

Distribution:

1510 J. W. Nunziato
1511 D. K. Gartling
1520 L. W. Davison
1521 R. D. Krieg and Staff (12)
1521 G. D. Sjaardema (30)
1522 R. C. Reuter, Jr. and Staff (15)
1523 J. H. Biffle and Staff (12)
1524 A. K. Miller and Staff (12)
1530 D. B. Hayes
1531 S. L. Thompson
1533 S. T. Montgomery
1533 A. C. Robinson
1550 C. W. Peterson, Jr.
1556 W. L. Oberkampf
2814 P. F. Chavez
3141 S. A. Landenberger (5)
3151 W. I. Klein (3)
3154-1 for DOE/OSTI (8)
6258 D. S. Preece
6314 L. S. Costin
6322 R. E. Glass
6334 H. J. Iuzzolino
6334 R. D. McCurley
6334 J. S. Rath
6334 R. P. Rechard
6334 E. Shepherd
8240 C. W. Robinson
8241 G. A. Benedetti
8242 M. R. Birnbaum
8243 M. L. Callabresi
8244 C. M. Hartwig
8245 R. J. Kee
8524 J. A. Wackerly