# Appendix B. Mesh File Format

The content and format of Fluent mesh files is described in the following sections:



**Note**

A *mesh* file is a subset of a case file; it contains only those sections of the case file that pertain to the mesh. The currently defined sections relevant for the mesh file are explained in the following sections. For information about sections in the case file, refer to Grid Sections (p. 3856) and Other (Non-Grid) Case Sections (p. 3867) in the Fluent User's Guide (p. 1).

## B.1. Guidelines

The mesh files are broken into several sections according to the following guidelines:

- Each section is enclosed in parentheses and begins with a decimal integer indicating its type. This integer is different for formatted and binary files (see Formatting Conventions in Binary Files and Formatted Files (p. 493)).
- All groups of items are enclosed in parentheses. This makes skipping to ends of (sub)sections and parsing them very easy. It also allows for easy and compatible addition of new items in future releases.
- Header information for lists of items is enclosed in separate sets of parentheses, preceding the items, which are in their own parentheses.

## B.2. Formatting Conventions in Binary Files and Formatted Files

For formatted files, examples of file sections are given in Grid Sections (p. 494) and Non-Grid Sections (p. 502). For binary files, the header indices described in subsequent sections (for example, `10` for the node section) are preceded by `20` for single-precision binary data, or by `30` for double-precision binary data (that is, `2010` or `3010` instead of 10). The end of the binary data is indicated by `End of Binary Section 2010` or `End of Binary Section 3010` before the closing parameters of the section.

An example with the binary data represented by periods is as follows:

```
(3010 (2 1 2aad 2 3)(
.
.
.
)
End of Binary Section 3010)
```

## B.3. Grid Sections

Grid sections are stored in the case file. A *mesh* file is a subset of a case file, containing only those sections pertaining to the mesh. The currently defined grid sections are explained in the following sections.



The section ID numbers are indicated in both symbolic and numeric forms. The symbolic representations are available as symbols in a Scheme source file (`xfile.scm`), which is available from ANSYS, Inc., or as macros in a C header file (`xfile.h`), which is located in your installation area.

### B.3.1. Comment

| | |
|---|---|
| Index: | 0 |
| Scheme symbol: | `xf-comment` |
| C macro: | `XF_COMMENT` |
| Status: | optional |

Comment sections can appear anywhere in the file (except within other sections) as:

```
(0 "comment text")
```

You should precede each long section or group of related sections, by a comment section explaining what is to follow.

For example,

```
(0 "Variables:")
 (60 (
 (max-skew-limit 1.)
 (max-cell-skew 0.85)
 (skewness-method 0)
 ))
```

### B.3.2. Header

| | |
|---|---|
| Index: | 1 |
| Scheme symbol: | `xf-header` |
| C macro: | `XF_HEADER` |
| Status: | optional |

Header sections can appear anywhere in the file (except within other sections). The following is an example:

```
(1 "ANSYS(R) TGrid(TM) 3D, serial 15.0.0")
```

The purpose of this section is to identify the program that wrote the file. Although this section can appear anywhere, it is typically one of the first sections in the file. Additional header sections indicate other programs that may have been used in generating the file. This provides a history mechanism showing where the file came from and how it was processed.

## B.3.3. Dimensions

| | |
|---|---|
| Index: | 2 |
| Scheme symbol: | `xf-dimension` |
| C macro: | `XF_DIMENSION` |
| Status: | optional |

The dimensions of the grid appear as:

```
(2 ND)
```

where *ND* is 2 or 3. This section is supported as a check that the grid has the appropriate dimensions.

## B.3.4. Nodes

| | |
|---|---|
| Index: | 10 |
| Scheme symbol: | `xf-node` |
| C macro: | `XF_NODE` |
| Status: | required |

The nodes section appears as:

```
(10 (zone-id first-index last-index type ND)(
 x1 y1 z1
 x2 y2 z2
 .
 .
 .
 ))
```

- If `zone-id` is zero, this provides the total number of nodes in the mesh. In this case, `first-index` will be one, `last-index` will be the total number of nodes *in hexadecimal*, `type` is zero, *ND* is omitted, and there are no coordinates following (the parentheses for the coordinates are omitted as well).

  For example,

  ```
  (10 (0 1 2d5 0))
  ```

- If `zone-id` is greater than zero, it indicates the zone to which the nodes belong. In this case, `first-index` and `last-index` are the indices of the nodes in the zone *in hexadecimal*. The values of `last-index` in each zone must be less than or equal to the value in the declaration section. `type` indicates the type of nodes in the zone. The following values are used to indicate the node type:

  – Zero for "virtual" nodes.

  – One for no (any) type.

– Two for boundary nodes.

Nodes of type zero are ignored but types one and two are read and written.

*ND* is an optional argument that indicates the dimensionality of the node data, where *ND* is 2 or 3. If the number of dimensions in the grid is two, as specified in the Dimensions (p. 495) or in the node header, then only x and y coordinates are present on each line.

The following is an example of a two-dimensional grid:

```
(10 (1 1 2d5 1 2)(
 1.500000e-01 2.500000e-02
 1.625000e-01 1.250000e-02
  .
  .
  .
 1.750000e-01 0.000000e+00
 2.000000e-01 2.500000e-02
 1.875000e-01 1.250000e-02
 ))
```

As the grid connectivity is composed of integers representing pointers (see Cells (p. 497) and Faces (p. 498)), using hexadecimal conserves space in the file and provides for faster file input and output. The header indices are also in hexadecimal so that they match the indices in the bodies of the grid connectivity sections. The `zone-id` and `type` are also in hexadecimal for consistency.

## B.3.5. Periodic Shadow Faces

| | |
|---|---|
| Index: | 18 |
| Scheme symbol: | `xf-periodic-face` |
| C macro: | `XF_PERIODIC_FACE` |
| Status: | required only for grids with periodic boundaries |

This section indicates the pairings of periodic faces on periodic boundaries. Grids without periodic boundaries do not have sections of this type.

The format of the section is as follows:

```
(18 (first-index last-index periodic-zone shadow-zone)(
 f00 f01
 f10 f11
 f20 f21
 .
 .
 .
 ))
```

where:

- `first-index` is the index of the first periodic face pair in the list.
- `last-index` is the index of the last periodic face pair in the list.
- `periodic-zone` is the zone ID of the periodic face zone.
- `shadow-zone` is the zone ID of the corresponding shadow face zone.

These are in hexadecimal format.

The indices in the section body (`f*`) refer to the faces on each of the periodic boundaries (in hexadecimal), the indices being offsets into the list of faces for the grid.

> **Note**
>
> The `first-index` and `last-index` do *not* refer to face indices; they refer to indices in the list of periodic pairs.

An example of such a section is as follows:

```
(18 (1 2b a c) (
 12 1f
 13 21
 ad 1c2
 .
 .
 .
 ))
```

## B.3.6. Cells

| | |
|---|---|
| Index: | 12 |
| Scheme symbol: | `xf-cell` |
| C macro: | `XF_CELL` |
| Status: | required |

The declaration section for cells is similar to that for nodes.

```
(12 (zone-id first-index last-index type element-type))
```

When `zone-id` is zero, it indicates that it is a declaration of the total number of cells. If `last-index` is zero, then there are no cells in the grid. This is useful when the file contains only a surface mesh as it serves to alert Fluent that it cannot be used in the solver.

In a declaration section, the `type` has a value of zero, while the `element-type` is not present.

For example,

```
(12 (0 1 3e3 0))
```

states that there are 3e3 (hexadecimal) = 995 cells in the grid. This declaration section is required and must precede the regular cell sections.

The `element-type` in a regular cell section header indicates the type of cells in the section, as follows:

| element-type | description | nodes/cell | faces/cell |
|---|---|---|---|
| 0 | mixed | | |
| 1 | triangular | 3 | 3 |
| 2 | tetrahedral | 4 | 4 |
| 3 | quadrilateral | 4 | 4 |
| 4 | hexahedral | 8 | 6 |
| 5 | pyramid | 5 | 5 |
| 6 | wedge | 6 | 5 |

| element-type | description | nodes/cell | faces/cell |
|---|---|---|---|
| 7 | polyhedral | *NN* | *NF* |

where *NN* and *NF* will vary, depending on the specific polyhedral cell.

Regular cell sections have no body, but they have a header of the same format where `first-index` and `last-index` indicate the range for the particular zone, `type` indicates whether the cell zone is fluid (`type` = 1) or solid (`type` = 17).

A `type` of zero indicates a dead zone and will be skipped when the file is read in solution mode in Fluent. If a zone is of mixed type (`element-type`=0), it will have a body that lists the `element-type` of each cell.

In the following example, there are 3d (hexadecimal) = 61 cells in cell zone 9, of which the first 3 are triangles, the next 2 are quadrilaterals, and so on.

```
(12 (9 1 3d 0 0)(
 1 1 1 3 3 1 1 3 1
 .
 .
 .
 ))
```

**Note**

The cell section is not required in meshing mode when the file contains only a surface mesh.

## B.3.7. Faces

| Index: | 13 |
|---|---|
| Scheme symbol: | `xf-face` |
| C macro: | `XF_FACE` |
| Status: | required |

The face section has a header with the same format as that for cells, but with a section index of 13. The format is as follows:

```
(13 (zone-id first-index last-index bc-type face-type))
```

where:

- `zone-id` = zone ID of the face section
- `first-index` = index of the first face in the list
- `last-index` = index of the last face in the list
- `bc-type` = ID of the boundary condition represented by the face section
- `face-type` = ID of the type(s) of face(s) in the section

A `zone-id` of zero indicates a declaration section, which provides a count of the total number of faces in the mesh. Such a section omits the `bc-type` and is not followed by a body with further information.

A non-zero `zone-id` indicates a regular face section and will be followed by a body containing information about the grid connectivity. Each line describes one face and appears as follows:

```
n0 n1 n2 c0 c1
```

where n* are the defining nodes (vertices) of the face, and c* are the adjacent cells.

This is an example of the format for a 3D grid with a triangular face format. The actual number of nodes depends on the face-type. The order of the cell indices is important and is determined by the right-hand rule: if you curl the fingers of your right hand in the order of the nodes, your thumb will point toward c0.

If the face zone is of mixed type (face-type= 0) or of polygonal type (face-type= 5), each line of the section body will begin with a reference to the number of nodes that make up that particular face, and has the following format:

```
x n0 n1 ... nf c0 c1
```

where x = number of nodes (vertices) of the face and nf is the final node of the face.

All cells, faces, and nodes have positive indices. If a face has a cell only on one side, then either c0 or c1 is zero. For files containing only a boundary mesh, both these values are zero.

bc-type indicates the ID of the boundary condition represented by the face section. The current valid boundary condition types are defined in the following table:

| bc-type | Description |
|---|---|
| 2 | interior |
| 3 | wall |
| 4 | pressure-inlet, inlet-vent, intake-fan |
| 5 | pressure-outlet, exhaust-fan, outlet-vent |
| 7 | symmetry |
| 8 | periodic-shadow |
| 9 | pressure-far-field |
| 10 | velocity-inlet |
| 12 | periodic |
| 14 | fan, porous-jump, radiator |
| 20 | mass-flow-inlet |
| 24 | interface |
| 31 | parent (hanging node) |
| 36 | outflow |
| 37 | axis |

The faces resulting from the intersection of the non-conformal grids are placed in a separate face zone. A factor of 1000 is added to the bc-type of these sections; for example, 1003 is a wall zone.

face-type indicates the type of faces in the zone as defined in the following table:

| face-type | description | nodes/face |
|---|---|---|
| 0 | mixed | |
| 2 | linear | 2 |
| 3 | triangular | 3 |
| 4 | quadrilateral | 4 |

| face-type | description | nodes/face |
|---|---|---|
| 5 | polygonal | *NN* |

where *NN* will vary, depending on the specific polygonal face.

## B.3.8. Edges

| Index: | 11 |
|---|---|
| Scheme symbol: | `xf-edge` |
| C macro: | `XF_EDGE` |
| Status: | optional |

The edge section has a header of the following format:

```
(11 (zone-id first-index last-index type element-type))
```

where:

- `zone-id` is the zone ID for the edge section
- `first-index` and `last-index` are the index of the first and last edge in the list, respectively
- `type` indicates the edge type
- The `element-type` is ignored completely.

A `zone-id` of zero indicates a declaration section with no body, which provides a count of the total number of edges in the mesh. A non-zero `zone-id` indicates a regular edge section, and will be followed by a body containing information about the grid connectivity. Each line describes one edge and appears as follows:

```
v0 v1
```

where `v0`, `v1` are the vertices defining the edge.

`type` denotes the edge type as defined in the following table:

| Description | type |
|---|---|
| boundary edge | 5 |
| interior edge | 6 |

**Note**

In case files written for Fluent, the Edges section will be omitted.

## B.3.9. Face Tree

| Index: | 59 |
|---|---|
| Scheme symbol: | `xf-face-tree` |
| C macro: | `XF_FACE_TREE` |
| Status: | only for grids with hanging-node adaption |

This section indicates the face hierarchy of the grid containing hanging nodes. The format of the section is as follows:

```
(59 (face-id0 face-id1 parent-zone-id child-zone-id)
 (
  number-of-kids kid-id-0 kid-id-1 ... kid-id-n
  .
  .
  .
 ))
```

where

`face-id0` is the index of the first parent face in the section.

`face-id1` is the index of the last parent face in the section.

`parent-zone-id` is the ID of the zone containing the parent faces

`child-zone-id` is the ID of the zone containing the children faces.

`number-of-kids` is the number of children of the parent face.

`kid-id-n` are the face IDs of the children.

These are in hexadecimal format. You can read files that contain this section in the meshing mode in Fluent.

## B.3.10. Cell Tree

| | |
|---|---|
| Index: | 58 |
| Scheme symbol: | `xf-cell-tree` |
| C macro: | `XF_CELL_TREE` |
| Status: | only for grids with hanging-node adaption |

This section indicates the cell hierarchy of the grid containing hanging nodes.

The format of the section is as follows:

```
(58 (cell-id0 cell-id1 parent-zone-id child-zone-id)
 (
 number-of-kids kid-id-0 kid-id-1 ... kid-id-n
 .
 .
 .
 ))
```

where:

- `cell-id0` is the index of the first parent cell in the section.
- `cell-id1` is the index of the last parent cell in the section.
- `parent-zone-id` is the ID of the zone containing the parent cells.
- `child-zone-id` is the ID of the zone containing the children cells.
- `number-of-kids` is the number of children of the parent cell.
- `kid-id-n` are the cell IDs of the children.

These are in hexadecimal format. You cannot read files that contain this section in the meshing mode in Fluent.

### B.3.11. Interface Face Parents

| | |
|---|---|
| Index: | 61 |
| Scheme symbol: | `xf-face-parents` |
| C macro: | `XF_FACE_PARENTS` |
| Status: | only for grids with non-conformal interfaces |

This section indicates the relationship between the intersection faces and original faces. The intersection faces (children) are produced from intersecting two non-conformal surfaces (parents) and are some fraction of the original face. Each child will refer to at least one parent.

The format of the section is as follows:

```
(61 (face-id0 face-id1)
 (
  parent-id-0 parent-id-1
  .
  .
  .
 ))
```

where:

- `face-id0` is the index of the first child face in the section.
- `face-id1` is the index of the last child face in the section.
- `parent-id-*` is the index of the parent faces.

These are in hexadecimal format.

If you set up and save a non-conformal mesh in the solution mode and then read it into the meshing mode of Fluent, this section will be skipped. Hence, all the information necessary to preserve the non-conformal interface will not be maintained. When you switch to the solution mode or read the mesh back into the solution mode, you will need to recreate the interface.

## B.4. Non-Grid Sections

The non-grid sections contain the boundary conditions, material properties, and solver control settings.
B.4.1. Zone

### B.4.1. Zone

| | |
|---|---|
| Index: | 39 or 45 |
| Scheme symbol: | `xf-rp-tv` |
| C macro: | `XF_RP_TV` |
| Status: | required |

Typically, there is one zone section for each zone referenced by the grid. Although some grid zones may not have corresponding sections, there cannot be more than one zone section for each zone.

The zone section has the following form:

```
(39 (zone-id zone-type zone-name domain-id)(
(condition1 . value1)
(condition2 . value2)
(condition3 . value3)
 .
 .
 .
  ))
```

Grid generators and preprocessors need only provide the section header and leave the list of conditions empty, as in:

```
(39 (zone-id zone-type zone-name domain-id)())
```

The empty parentheses at the end are required. The solver adds conditions as appropriate, depending on the zone type.

When only `zone-id`, `zone-type`, `zone-name`, and `domain-id` are specified, the index 45 may be used for a zone section. However, the index 39 must be used if boundary conditions are present, because any and all remaining information in a section of index 45 after `zone-id`, `zone-type`, `zone-name`, and `domain-id` will be ignored.

In meshing mode, the zone name and type can be extracted from the boundary condition section `39` (refer to the Fluent User's Guide for details) or `45`, but only section `39` can be written.

The `zone-id` is in decimal format. This is in contrast to the use of hexadecimal in the grid sections.

The `zone-type` is one of the following:

```
degassing
exhaust-fan
fan
fluid
geometry
inlet-vent
intake-fan
interface
interior
internal
mass-flow-inlet
outflow
outlet-vent
parent-face
porous-jump
pressure-far-field
pressure-inlet
pressure-outlet
radiator
solid
symmetry
velocity-inlet
wall
wrapper
```

The `interior`, `fan`, `porous-jump`, and `radiator` types can be assigned only to zones of faces inside the domain. The `interior` type is used for the faces within a cell zone; the others are for interior faces that form infinitely thin surfaces within the domain. Fluent allows the `wall` type to be assigned to face zones both on the inside and on the boundaries of the domain.

Some zone types are valid only for certain types of grid components (for example, cell zones can be assigned only either `fluid` or `solid` type). All other types listed can be used for only boundary (face) zones.

The `zone-name` is a label for the zone. It must be a valid Scheme symbol and is written without quotes. The rules for a valid `zone-name` are as follows:

- The first character must be a lowercase letter or a special-initial.
- Each subsequent character must be a lowercase letter, a special-initial, a digit, or a special-subsequent.

  A special-initial character is one of the following:

  ```
  ! $ % & * / : < = > ? ~ _ ^
  ```

  and a special subsequent is one of the following:

  ```
  . + -
  ```

Some examples of zone sections produced in the meshing mode are as follows:

```
(39 (1 fluid fuel 1)())
(39 (8 pressure-inlet pressure-inlet-8 2)())
(39 (2 wall wing-skin 3)())
(39 (3 symmetry mid-plane 1)())
```

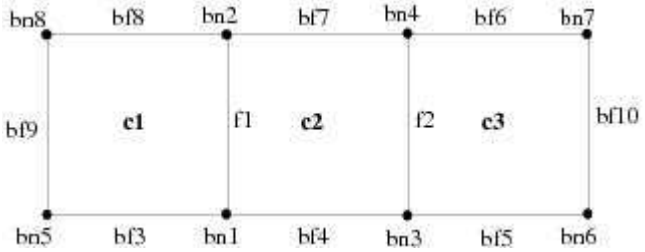
## B.5. Example Files

*The examples show 2D quadrilateral meshes for easier illustration. The same concepts are applied to 3D meshes.*

### Example 1

Figure 1: Quadrilateral Mesh (p. 504) illustrates a simple quadrilateral mesh with no periodic boundaries or hanging nodes.

**Figure 1: Quadrilateral Mesh**




The following describes this mesh:

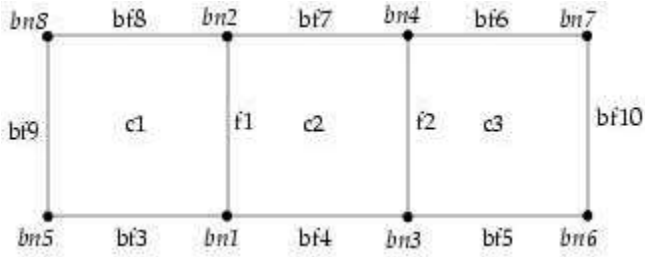
```
(0 "Grid:")

(0 "Dimensions:")
(2 2)

(12 (0 1 3 0))
(13 (0 1 a 0))
(10 (0 1 8 0 2))

(12 (7 1 3 1 3))

(13 (2 1 2 2 2)(
1 2 1 2
3 4 2 3))

(13 (3 3 5 3 2)(
5 1 1 0
```

```
1 3 2 0
3 6 3 0))

(13 (4 6 8 3 2)(
7 4 3 0
4 2 2 0
2 8 1 0))

(13 (5 9 9 a 2)(
8 5 1 0))

(13 (6 a a 24 2)(
6 7 3 0))

(10 (1 1 8 1 2)
(
1.00000000e+00 0.00000000e+00
1.00000000e+00 1.00000000e+00
2.00000000e+00 0.00000000e+00
2.00000000e+00 1.00000000e+00
0.00000000e+00 0.00000000e+00
3.00000000e+00 0.00000000e+00
3.00000000e+00 1.00000000e+00
0.00000000e+00 1.00000000e+00))
```

### Example 2

Figure 2: Quadrilateral Mesh with Periodic Boundaries (p. 505) illustrates a simple quadrilateral mesh with periodic boundaries but no hanging nodes. In this example, `bf9` and `bf10` are faces on the periodic zones.

**Figure 2: Quadrilateral Mesh with Periodic Boundaries**




The following describes this mesh:

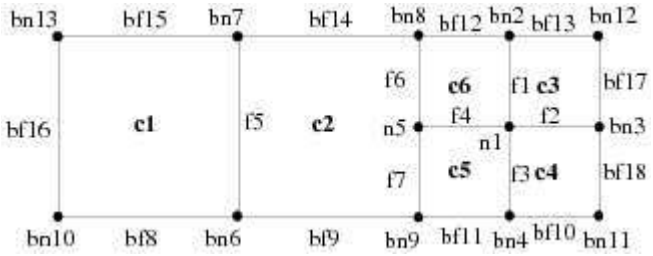
```
(0 "Dimensions:")
(2 2)

(0 "Grid:")

(12 (0 1 3 0))
(13 (0 1 a 0))
(10 (0 1 8 0 2))

(12 (7 1 3 1 3))

(13 (2 1 2 2 2)(
1 2 1 2
3 4 2 3))

(13 (3 3 5 3 2)(
5 1 1 0
1 3 2 0
3 6 3 0))

(13 (4 6 8 3 2)(
7 4 3 0
4 2 2 0
```

```
2 8 1 0))

(13 (5 9 9 c 2)(
8 5 1 0))

(13 (1 a a 8 2)(
6 7 3 0))

(18 (1 1 5 1)(
9 a))

(10 (1 1 8 1 2)(
1.00000000e+00 0.00000000e+00
1.00000000e+00 1.00000000e+00
2.00000000e+00 0.00000000e+00
2.00000000e+00 1.00000000e+00
0.00000000e+00 0.00000000e+00
3.00000000e+00 0.00000000e+00
3.00000000e+00 1.00000000e+00
0.00000000e+00 1.00000000e+00))
```

### Example 3

Figure 3: Quadrilateral Mesh with Hanging Nodes (p. 506) illustrates a simple quadrilateral mesh with hanging nodes.

**Figure 3: Quadrilateral Mesh with Hanging Nodes**




The following describes this mesh:

```
(0 "Grid:")

(0 "Dimensions:")
(2 2)

(12 (0 1 7 0))
(13 (0 1 16 0))
(10 (0 1 d 0 2))

(12 (7 1 6 1 3))
(12 (1 7 7 20 3))

(58 (7 7 1 7)(
4 6 5 4 3))

(13 (2 1 7 2 2)(
1 2 6 3
1 3 3 4
1 4 4 5
1 5 5 6
6 7 1 2
5 8 2 6
9 5 2 5))

(13 (3 8 b 3 2)(
a 6 1 0
6 9 2 0
4 b 4 0
9 4 5 0))
```

```
(13 (4 c f 3 2)(
2 8 6 0
c 2 3 0
8 7 2 0
7 d 1 0))

(13 (5 10 10 a 2)(
d a 1 0))

(13 (6 11 12 24 2)(
3 c 3 0
b 3 4 0))

(13 (b 13 13 1f 2)(
c 8 7 0))

(13 (a 14 14 1f 2)(
b c 7 0))

(13 (9 15 15 1f 2)(
9 b 7 0))

(13 (8 16 16 1f 2)(
9 8 2 7))

(59 (13 13 b 4)(
2 d c))

(59 (14 14 a 6)(
2 12 11))

(59 (15 15 9 3)(
2 b a))

(59 (16 16 8 2)(
2 7 6))

(10 (1 1 d 1 2)
(
2.50000000e+00 5.00000000e-01
2.50000000e+00 1.00000000e+00
3.00000000e+00 5.00000000e-01
2.50000000e+00 0.00000000e+00
2.00000000e+00 5.00000000e-01
1.00000000e+00 0.00000000e+00
1.00000000e+00 1.00000000e+00
2.00000000e+00 1.00000000e+00
2.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00
3.00000000e+00 0.00000000e+00
3.00000000e+00 1.00000000e+00
0.00000000e+00 1.00000000e+00))
```