gmsh scripts

Release 0.3.3

Sep 07, 2023

CONTENTS:

1 Tutorials **3**
- 1.1 Simple Children . . . 3
  - 1.1.1 Concept . . . 3
  - 1.1.2 Children . . . 3
  - 1.1.3 Items Children . . . 8
  - 1.1.4 All together . . . 13
  - 1.1.5 Mesh . . . 14
- 1.2 Simple Layer . . . 18
  - 1.2.1 Concept . . . 18
  - 1.2.2 Number of layers . . . 23
  - 1.2.3 Zones . . . 24
- 1.3 Cross section 2 . . . 25
  - 1.3.1 Decomposition . . . 28
  - 1.3.2 Geometry . . . 28
  - 1.3.3 Mesh . . . 31
    - 1.3.3.1 Unstructured Tetrahedral . . . 31
    - 1.3.3.2 Unstructured Hexahedral . . . 38
    - 1.3.3.3 Structured Tetrahedral . . . 40
    - 1.3.3.4 Structured Hexahedral . . . 43
  - 1.3.4 Zones . . . 48
  - 1.3.5 Result . . . 49

2 API Reference **53**
- 2.1 `gmsh_scripts` . . . 53
  - 2.1.1 Subpackages . . . 53
    - 2.1.1.1 `gmsh_scripts.block` . . . 53
    - 2.1.1.2 `gmsh_scripts.boolean` . . . 66
    - 2.1.1.3 `gmsh_scripts.coordinate_system` . . . 67
    - 2.1.1.4 `gmsh_scripts.entity` . . . 71
    - 2.1.1.5 `gmsh_scripts.optimize` . . . 77
    - 2.1.1.6 `gmsh_scripts.parse` . . . 79
    - 2.1.1.7 `gmsh_scripts.quadrate` . . . 84
    - 2.1.1.8 `gmsh_scripts.refine` . . . 85
    - 2.1.1.9 `gmsh_scripts.size` . . . 86
    - 2.1.1.10 `gmsh_scripts.smooth` . . . 87
    - 2.1.1.11 `gmsh_scripts.strategy` . . . 88
    - 2.1.1.12 `gmsh_scripts.structure` . . . 89
    - 2.1.1.13 `gmsh_scripts.support` . . . 90
    - 2.1.1.14 `gmsh_scripts.transform` . . . 96
    - 2.1.1.15 `gmsh_scripts.utils` . . . 100

2.1.1.16 gmsh_scripts.zone 101
2.1.2 Submodules 103
2.1.2.1 gmsh_scripts.factory 103
2.1.2.2 gmsh_scripts.load 103
2.1.2.3 gmsh_scripts.plot 104
2.1.2.4 gmsh_scripts.registry 104
2.1.2.5 gmsh_scripts.run 110

3 Indices and tables **111**

Python Module Index **113**

Index **115**

3D structured/unstructured tetrahedral/hexahedral multi-block mesh generator with boolean operations based on gmsh

CHAPTER

ONE

TUTORIALS

1.1 Simple Children

1.1.1 Concept

Each successor of class *Block* can have children blocks. Each class with `items` (currently *Matrix* and *Layer*) can have children for each `item`.

Every child is a gmsh_scripts file in JSON or YAML format.

Children are specified in `data.children` field. One could set `transforms` for children at `data.children_transforms` field.

Items children are specified in `data.items_children` One should set `data.items_children_transforms` and `data.items_children_transforms_map` to apply transformations to them.

1.1.2 Children

Let's create two children and one parent. Children are instances of class *Layer*, i.e. they are cylinders with different radius. Parent is an instance of class *Block* and has 3 `items` arranged along X-axis.

We create them structured and quadrated for convenience.

- Child 1

```yaml
metadata:
  run:
    factory: geo
    strategy:
      class: strategy.NoBoolean
data:
  class: block.Layer
  layer: [ [ 0.05;;4, 0.15;;4 ],
           [ 0.1;;2, 0.3;;2 ] ]
  layer_curves: [ [ line, circle_arc ],
                  [ line, line ] ]
  items_zone: [ CHILD_1 ]
  items_do_structure_map: 1
  items_do_quadrate_map: 1
```

```
python -m gmsh_scripts child_1.yaml
```

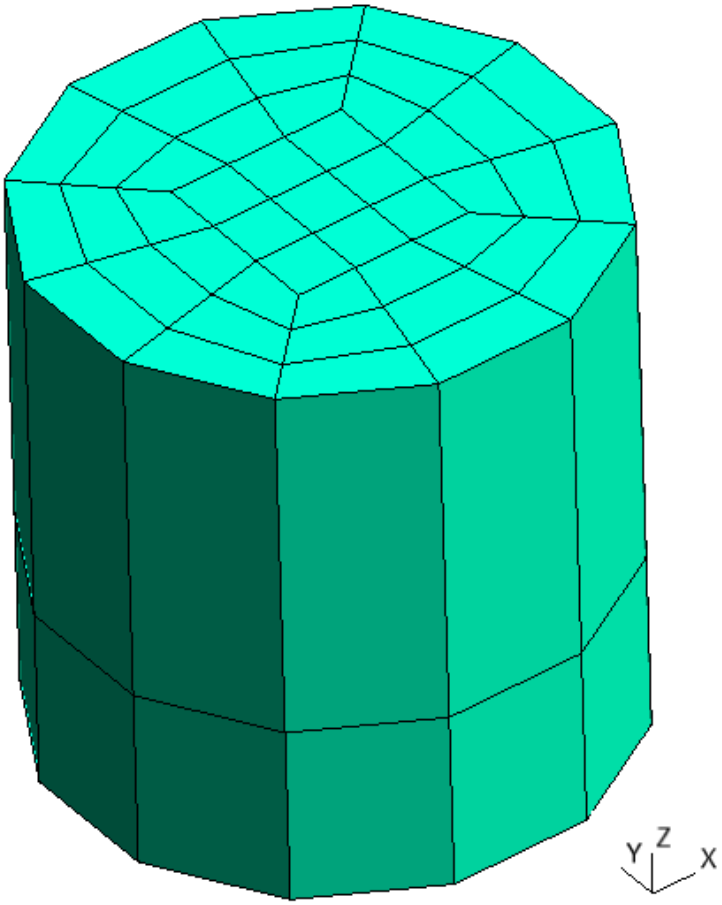

Fig. 1: Child 1

- Child 2

```yaml
metadata:
  run:
    factory: geo
    strategy:
      class: strategy.NoBoolean
data:
  class: block.Layer
  layer: [ [ 0.1;;4, 0.3;;4 ],
           [ 0.1;;2, 0.3;;2 ] ]
  layer_curves: [ [ line, circle_arc ],
                  [ line, line ] ]
  items_zone: [ CHILD_2 ]
  items_do_structure_map: 1
  items_do_quadrate_map: 1
```

```
python -m gmsh_scripts child_2.yaml
```

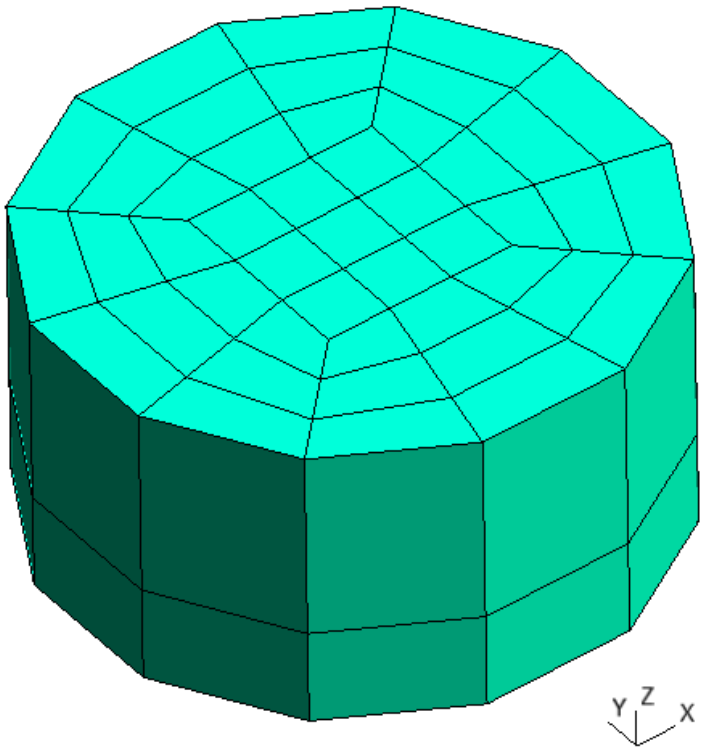

Fig. 2: Child 2

- Parent

```yaml
metadata:
  run:
    factory: geo
```

(continues on next page)

(continued from previous page)

```
    strategy:
      class: strategy.NoBoolean
data:
  class: block.Matrix
  matrix: [ [ 0;;2, 1;;2, 2;;2, 3;;2 ],
            [ 0;;2, 1;;2 ],
            [ 0;;2, 1;;2 ] ]
  items_zone: [ PARENT ]
  items_do_structure_map: 1
  items_do_quadrate_map: 1
```

```
python -m gmsh_scripts parent.yaml
```

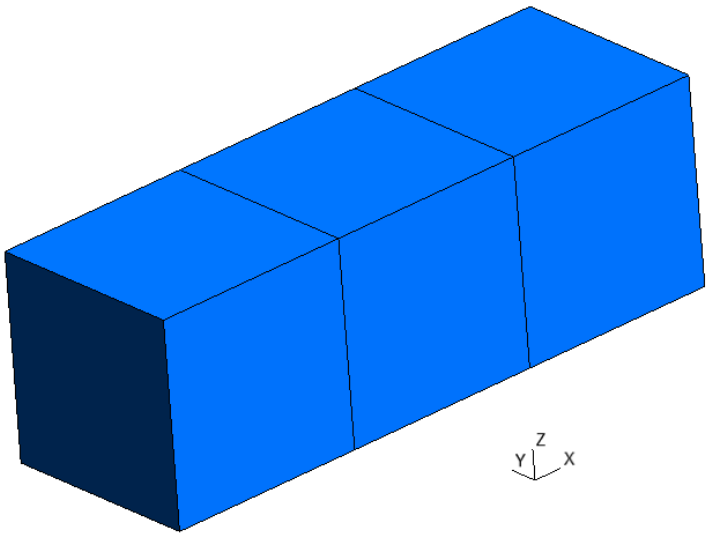

Fig. 3: Parent

One can add children to parent specifying `data.children` with list of file names of children started with `/` character.

```
metadata:
  run:
    factory: geo
    strategy:
      class: strategy.NoBoolean
data:
  class: block.Matrix
  matrix: [ [ 0;;2, 1;;2, 2;;2, 3;;2 ],
            [ 0;;2, 1;;2 ],
            [ 0;;2, 1;;2 ] ]
  items_zone: [ PARENT ]
  children: [
    /child_1.yaml,
    /child_2.yaml
```

(continues on next page)

(continued from previous page)

```
  ]
  items_do_structure_map: 1
  items_do_quadrate_map: 1
```

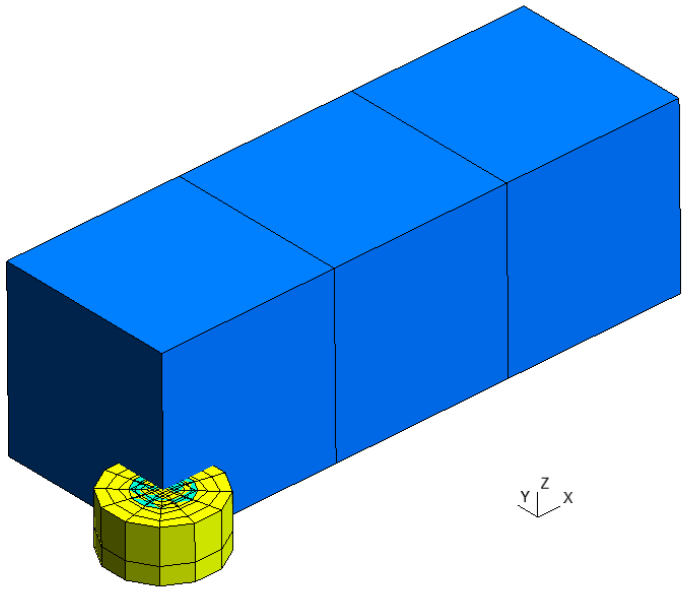

Fig. 4: Parent with `children` without `children_transforms`

As one can see, children are placed in their origin `(0, 0, 0)`. To move them to another location use `data.children_transforms` field that contains `transforms` for each children according with their position in `data.children` field.

In this example, we move first child by -1 and second child by 2 along Y-axis.

```
metadata:
  run:
    factory: geo
    strategy:
      class: strategy.NoBoolean
data:
  class: block.Matrix
  matrix: [ [ 0;;2, 1;;2, 2;;2, 3;;2 ],
            [ 0;;2, 1;;2 ],
            [ 0;;2, 1;;2 ] ]
  items_zone: [ PARENT ]
  children: [
    /child_1.yaml,
    /child_2.yaml
  ]
  children_transforms: [
    [ [ 0, -1, 0] ],
    [ [ 0, 2, 0] ]
  ]
```

(continues on next page)

(continued from previous page)

```
20  items_do_structure_map: 1
21  items_do_quadrate_map: 1
```

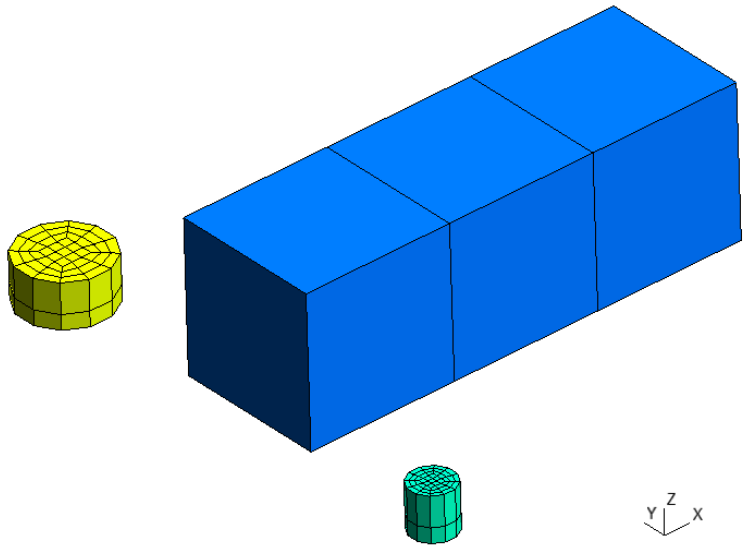

Fig. 5: Parent with `children` with `children_transforms`

1.1.3 Items Children

Adding items children is a little trickier: first we need specify children for each item, then apply transformations to each child of each item.

Working with `items` consists of 2 steps: creating `main field` with options and then creating `addressing field`, that ends with `_map` suffix with exception of fields with `do_` prefix, they don't need `main field`. Each value in `addressing field` is an index in `main field` and position of value indicates to which `item` of *Matrix* or *Layer* should be assigned option from `main_field`.

In this example, parent has 3 items, so addressing fields will have length of 3. First we need to define main field for children `items_children` then addressing field `items_children_map`.

`items_children` consists of 3 options, first and second assign one child to an item and third - two:

1. `/child_1.yaml`,
2. `/child_2.yaml`,
3. `/child_1.yaml` and `/child_2.yaml`.

`items_children_map` assigns first option from `items_children` to first item, second option from `items_children` to second item, and third to third: `[0, 1, 2]`.

Similarly fields `items_children_transforms` and `items_children_transforms_map` are set.

`items_children_transforms` consists of 2 options, first assign no transforms to one child and third one transform to two children:

1. `[ [ ] ]`,
2. `[ [ [ 0, 0, 0.1 ] ], [ [ 0, 0, -0.3 ] ] ]`.

`items_children_transforms_map` assigns first option from `items_children` to first and second items and second option to third: `[0, 0, 1]`.

It's convenient to create only geometry instead of mesh while working with children and their transformations.

```yaml
metadata:
  run:
    factory: geo
data:
  class: block.Matrix
  matrix: [ [ 0;;2, 1;;2, 2;;2, 3;;2 ],
            [ 0;;2, 1;;2 ],
            [ 0;;2, 1;;2 ] ]
  items_zone: [ PARENT ]
  items_children: [
    [ /child_1.yaml ],
    [ /child_2.yaml ],
    [ /child_1.yaml, /child_2.yaml ],
  ]
  items_children_map: [
    0, 1, 2
  ]
  items_children_transforms: [
    [ [ ] ],
    [ [ [ 0, 0, 0.1 ] ], [ [ 0, 0, -0.3 ] ] ],
  ]
  items_children_transforms_map: [
    0, 0, 1
  ]
  items_do_structure_map: 1
  items_do_quadrate_map: 1
```

```
python -m gmsh_scripts parent_items.yaml
```

After that we could add strategy to create the mesh.

```yaml
metadata:
  run:
    factory: geo
    strategy:
      class: strategy.NoBoolean
data:
  class: block.Matrix
  matrix: [ [ 0;;2, 1;;2, 2;;2, 3;;2 ],
            [ 0;;2, 1;;2 ],
            [ 0;;2, 1;;2 ] ]
  items_zone: [ PARENT ]
  items_children: [
    [ /child_1.yaml ],
    [ /child_2.yaml ],
    [ /child_1.yaml, /child_2.yaml ],
```

(continues on next page)

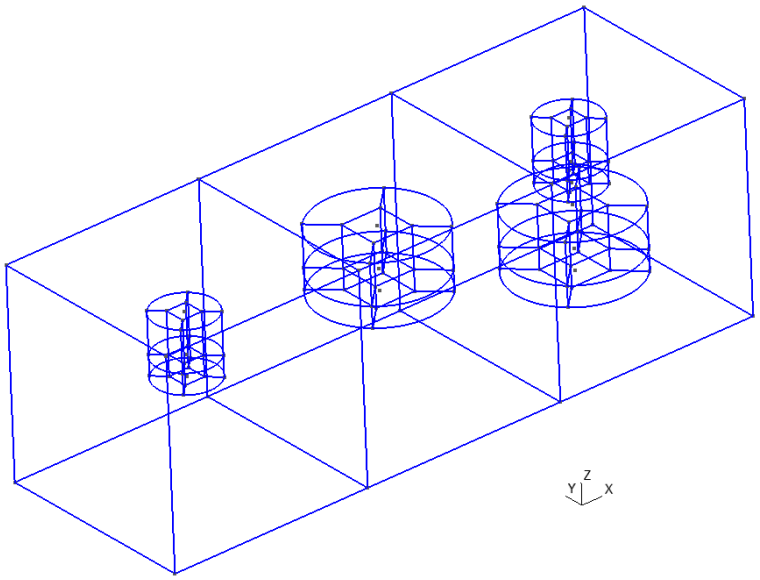

Fig. 6: Parent with `items_children` in `geo_unrolled` output format

(continued from previous page)

```
  ]
  items_children_map: [
    0, 1, 2
  ]
  items_children_transforms: [
    [ [ ] ],
    [ [ [ 0, 0, 0.1 ] ], [ [ 0, 0, -0.3 ] ] ],
  ]
  items_children_transforms_map: [
    0, 0, 1
  ]
  items_do_structure_map: 1
  items_do_quadrate_map: 1
```

We could supress generation of items (not their children) by setting `items_do_register_map` to 0.

```
metadata:
  run:
    factory: geo
    strategy:
      class: strategy.NoBoolean
data:
  class: block.Matrix
  matrix: [ [ 0;;2, 1;;2, 2;;2, 3;;2 ],
            [ 0;;2, 1;;2 ],
            [ 0;;2, 1;;2 ] ]
  items_zone: [ PARENT ]
  items_children: [
    [ /child_1.yaml ],
```

(continues on next page)

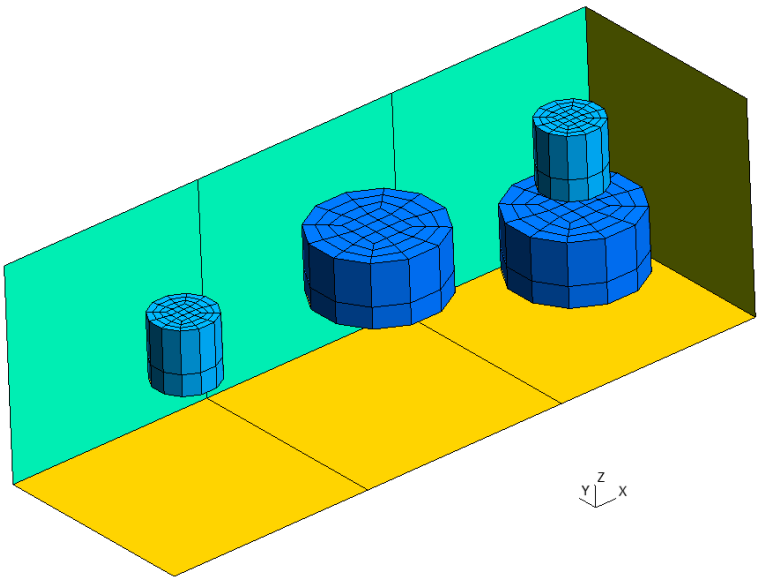

Fig. 7: Parent with `items_children`

(continued from previous page)

```
    [ /child_2.yaml ],
    [ /child_1.yaml, /child_2.yaml ],
  ]
  items_children_map: [
    0, 1, 2
  ]
  items_children_transforms: [
    [ [ ] ],
    [ [ [ 0, 0, 0.1 ] ], [ [ 0, 0, -0.3 ] ] ],
  ]
  items_children_transforms_map: [
    0, 0, 1
  ]
  items_do_structure_map: 1
  items_do_quadrate_map: 1
  items_do_register_map: 0
```

We could also create many copies of children inside an item adding additional parameter after coordinate with `:` separator, e.g. `5:4` divides last item into 3 parts (with 4 nodes) and creates children inside each part.

```
metadata:
  run:
    factory: geo
    strategy:
      class: strategy.NoBoolean
data:
  class: block.Matrix
  matrix: [ [ 0;;2, 1;;2, 2;;2, 5:4;;2 ],
            [ 0;;2, 1;;2 ],
            [ 0;;2, 1;;2 ] ]
```

(continues on next page)

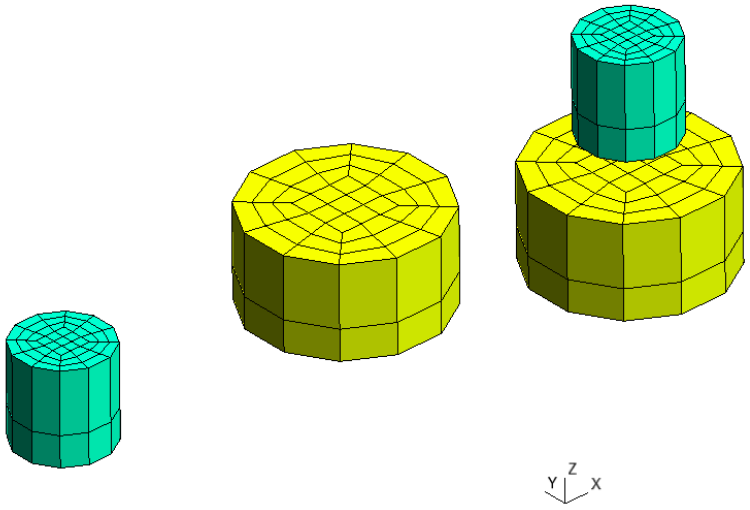

Fig. 8: Parent with `items_children` without `items`

(continued from previous page)

```
  items_zone: [ PARENT ]
  items_children: [
    [ /child_1.yaml ],
    [ /child_2.yaml ],
    [ /child_1.yaml, /child_2.yaml ],
  ]
  items_children_map: [
    0, 1, 2
  ]
  items_children_transforms: [
    [ [ ] ],
    [ [ [ 0, 0, 0.1 ] ], [ [ 0, 0, -0.3 ] ] ],
  ]
  items_children_transforms_map: [
    0, 0, 1
  ]
  items_do_structure_map: 1
  items_do_quadrate_map: 1
  items_do_register_map: 0
```

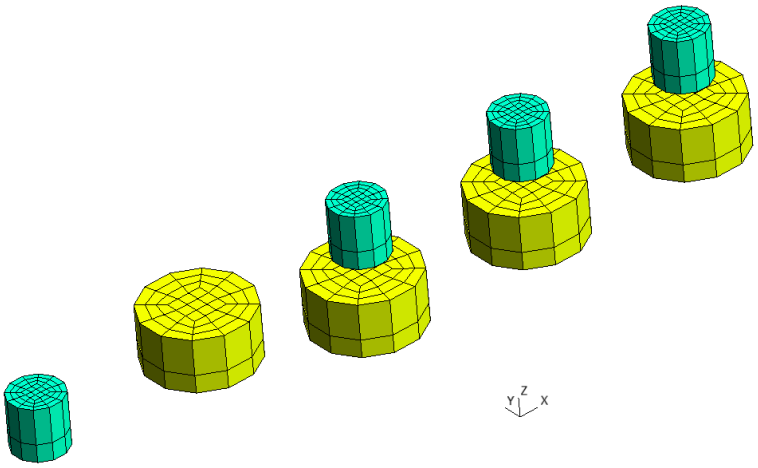

Fig. 9: Parent with `items_children` with extended third item

1.1.4 All together

One could combine `children` and `items_children`, e.g. to add arbitrarily and regularly located children.

```
metadata:
  run:
    factory: occ
    strategy:
      class: strategy.NoBoolean
data:
  class: block.Matrix
  matrix: [ [ 0;;2, 1;;2, 2;;2, 5:4;;2 ],
            [ 0;;2, 1;;2 ],
            [ 0;;2, 1;;2 ] ]
  items_zone: [ PARENT ]
  children: [
    /child_1.yaml,
    /child_2.yaml
  ]
  children_transforms: [
    [ [ 0, -1, 0 ] ],
    [ [ 0, 2, 0 ] ]
  ]
  items_children: [
    [ /child_1.yaml ],
    [ /child_2.yaml ],
    [ /child_1.yaml, /child_2.yaml ],
  ]
  items_children_map: [
    0, 1, 2
  ]
  items_children_transforms: [
    [ [ ] ],
```

(continues on next page)

(continued from previous page)

```
    [ [ [ 0, 0, 0.1 ] ], [ [ 0, 0, -0.3 ] ] ],
  ]
  items_children_transforms_map: [
    0, 0, 1
  ]
  items_do_structure_map: 1
  items_do_quadrate_map: 1
  items_do_register_map: 0
```

```
python -m gmsh_scripts parent_all.yaml
```

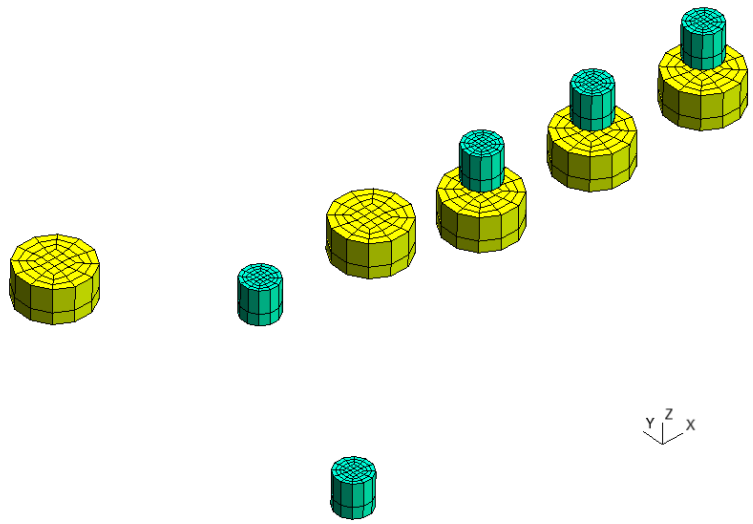

Fig. 10: Parent with `children` and `items_children`

1.1.5 Mesh

To create mesh we need boolean operations that are available in `occ` factory.

Warning: It's recommended to disable `items_do_structure_map` and `items_do_quadrate_map` fields, i.e. to create unstructured tetrahedral mesh while using boolean operations to achieve better stability of mesh generation.

```
data:
  class: block.Layer
  layer: [ [ 0.05;;4, 0.15;;4 ],
           [ 0.1;;2, 0.3;;2 ] ]
  layer_curves: [ [ line, circle_arc ],
                  [ line, line ] ]
  items_zone: [ CHILD_1 ]
  items_do_structure_map: 0
  items_do_quadrate_map: 0
```

```yaml
data:
  class: block.Layer
  layer: [ [ 0.1;;4, 0.3;;4 ],
           [ 0.1;;2, 0.3;;2 ] ]
  layer_curves: [ [ line, circle_arc ],
                  [ line, line ] ]
  items_zone: [ CHILD_2 ]
  items_do_structure_map: 0
  items_do_quadrate_map: 0
```

```yaml
metadata:
  run:
    factory: occ
data:
  class: block.Matrix
  matrix: [ [ 0;;2, 1;;2, 2;;2, 5:4;;2 ],
            [ 0;;2, 1;;2 ],
            [ 0;;2, 1;;2 ] ]
  items_zone: [ PARENT ]
  children: [
    /child_1.yaml,
    /child_2.yaml
  ]
  children_transforms: [
    [ [ 0, -1, 0 ] ],
    [ [ 0, 2, 0 ] ]
  ]
  items_children: [
    [ /child_1.yaml ],
    [ /child_2.yaml ],
    [ /child_1.yaml, /child_2.yaml ],
  ]
  items_children_map: [
    0, 1, 2
  ]
  items_children_transforms: [
    [ [ ] ],
    [ [ [ 0, 0, 0.1 ] ], [ [ 0, 0, -0.3 ] ] ],
  ]
  items_children_transforms_map: [
    0, 0, 1
  ]
  items_do_structure_map: 0
  items_do_quadrate_map: 0
  items_do_register_map: 1
```

```
python -m gmsh_scripts parent_all.yaml
```

One could customize mesh quality using `run.options` fields, e.g:

1. `Mesh.MeshSizeFactor` - factor applied to all mesh element sizes,
2. `Mesh.MeshSizeMin` - minimum mesh element size,

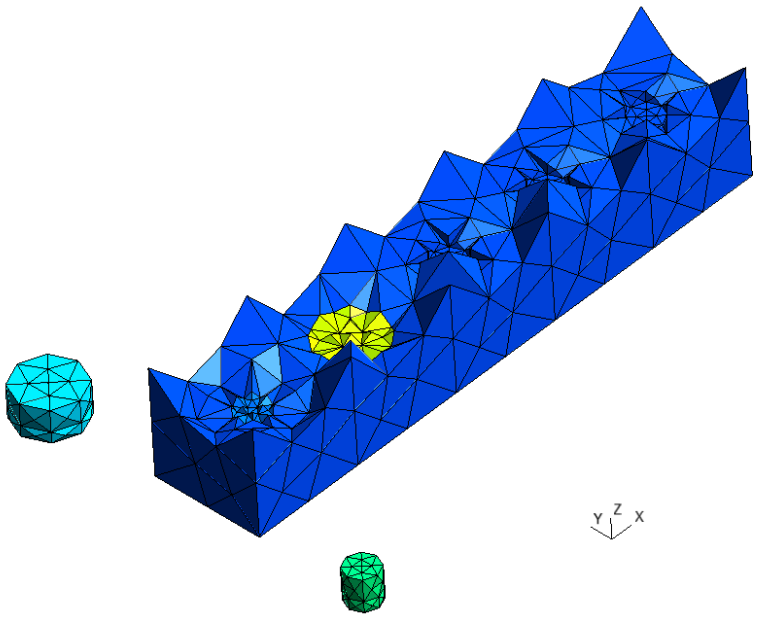

Fig. 11: Mesh generated with `occ factory`

3. `Mesh.MeshSizeMax` - maximum mesh element size,
4. `Mesh.MeshSizeExtendFromBoundary` - extend computation of mesh element sizes from the boundaries into the interior (`0`: never; `1`: for surfaces and volumes; `2`: for surfaces and volumes, but use smallest surface element edge length instead of longest length in 3D Delaunay; `-2`: only for surfaces; `-3`: only for volumes),
5. `Mesh.MeshSizeFromCurvature` - automatically compute mesh element sizes from curvature, using the value as the target number of elements per 2 * Pi radians.

Note: See documentation of gmsh for more information about options and their description.

```
metadata:
  run:
    factory: occ
    options:
      Mesh.MeshSizeFactor: 1.0
      Mesh.MeshSizeMin: 0.1
      Mesh.MeshSizeMax: 0.3
      Mesh.MeshSizeExtendFromBoundary: 2
      Mesh.MeshSizeFromCurvature: 12
data:
  class: block.Matrix
  matrix: [ [ 0;;2, 1;;2, 2;;2, 5:4;;2 ],
            [ 0;;2, 1;;2 ],
            [ 0;;2, 1;;2 ] ]
  items_zone: [ PARENT ]
  children: [
    /child_1.yaml,
    /child_2.yaml
```

(continues on next page)

(continued from previous page)

```
  ]
  children_transforms: [
    [ [ 0, -1, 0 ] ],
    [ [ 0, 2, 0 ] ]
  ]
  items_children: [
    [ /child_1.yaml ],
    [ /child_2.yaml ],
    [ /child_1.yaml, /child_2.yaml ],
  ]
  items_children_map: [
    0, 1, 2
  ]
  items_children_transforms: [
    [ [ ] ],
    [ [ [ 0, 0, 0.1 ] ], [ [ 0, 0, -0.3 ] ] ],
  ]
  items_children_transforms_map: [
    0, 0, 1
  ]
  items_do_structure_map: 0
  items_do_quadrate_map: 0
  items_do_register_map: 1
```

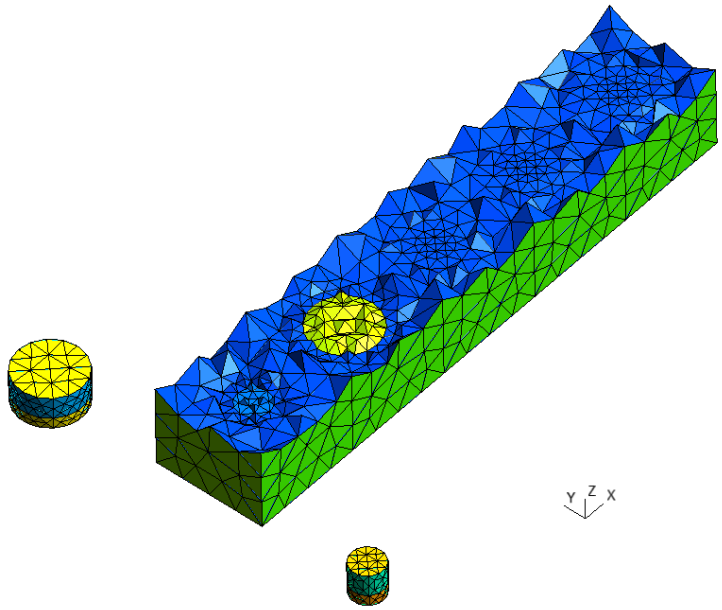

Fig. 12: Mesh configured with `metadata.run.options`

1.2 Simple Layer

1.2.1 Concept

Class *Layer* is a successor of class *Matrix*, which is a collection of *Block* in a structure of 3D regular grid.

Class *Layer* removes *Block* from *Matrix* that do not have 0 coordinate by X or Y axes and applies transformations to remaining blocks *Block* that are connect their sides.

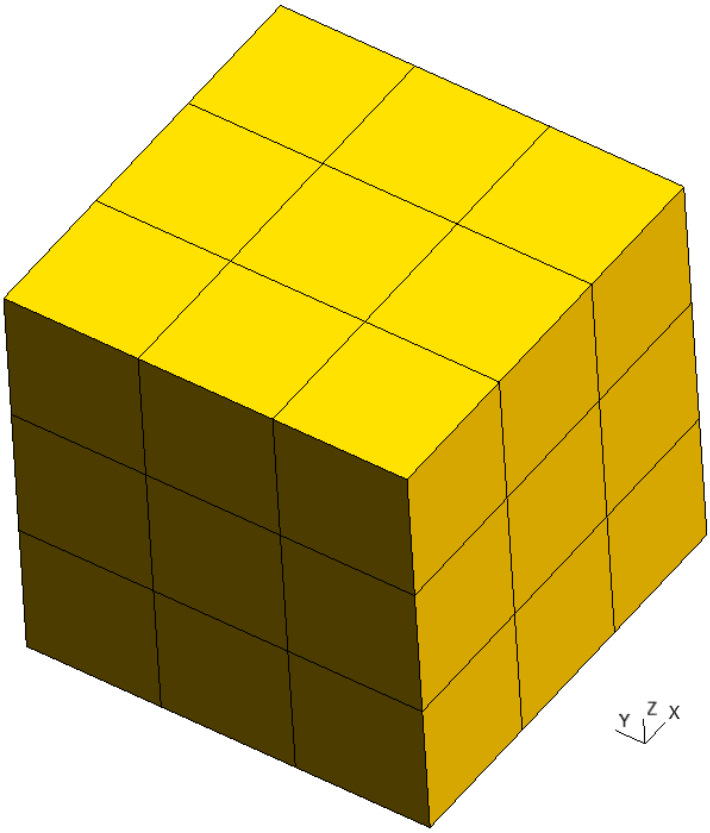

Fig. 13: Matrix

We can change type of surfaces by X and Y axes to curved ones. Number of nodes by circumferential direction is determined by last number `0.5;;8` at first radial `layer`:

```
metadata:
  run:
    factory: geo
    strategy:
      class: strategy.NoBoolean
data:
  class: block.Layer
  layer: [ [ 0.5;;8, 1.5;;2 ],
           [ 1;;2, 2;;2, 3;;2 ] ]
  "layer_curves": [ [ line, circle_arc ],
                    [ line, line, line ] ]
```

(continues on next page)

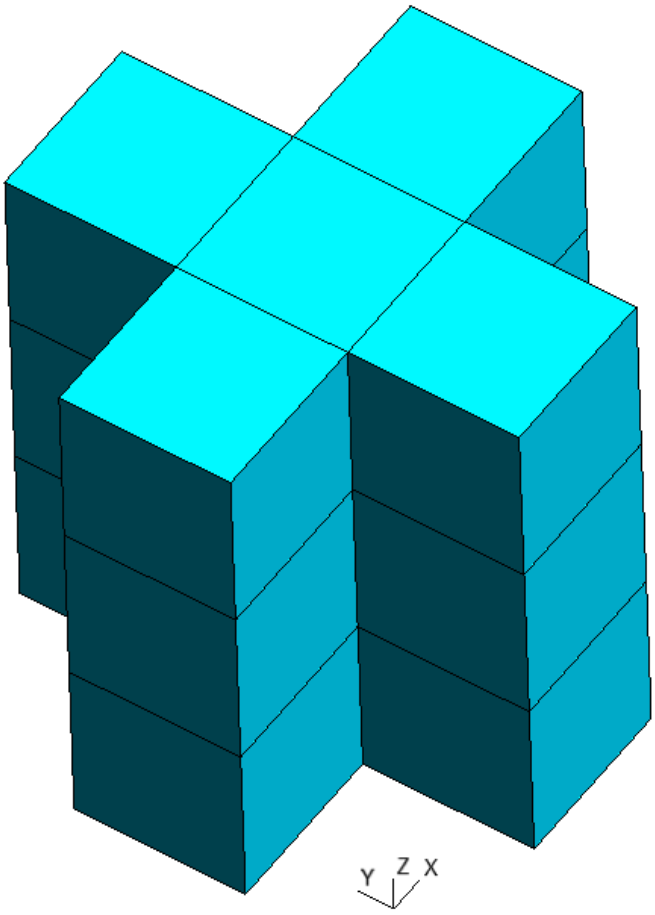

Fig. 14: Filtered Matrix

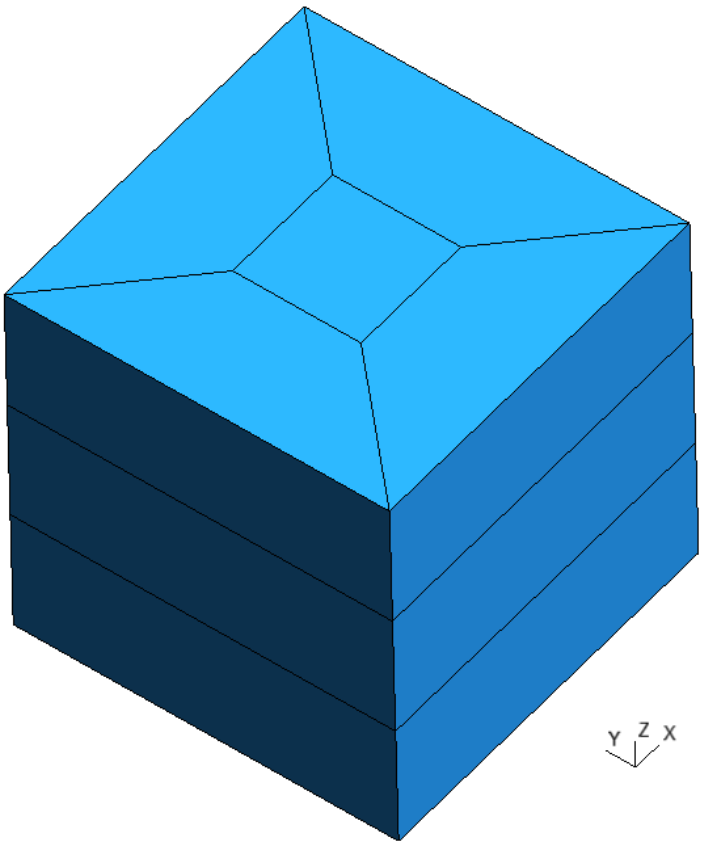

Fig. 15: Layer

(continued from previous page)

```
  items_do_structure_map: 1
  items_do_quadrate_map: 1
```

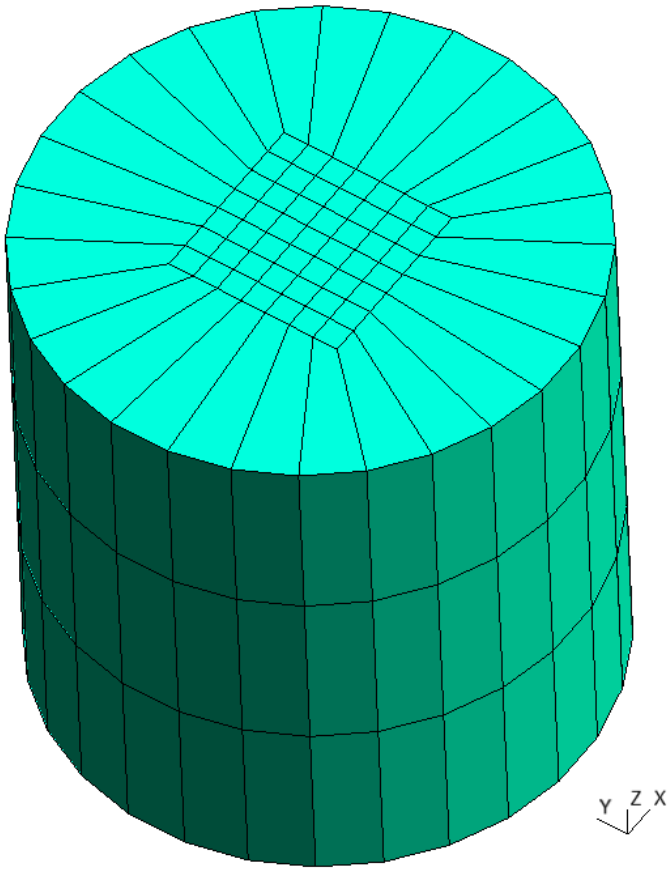

Fig. 16: Curved Layer

One can also change number of nodes by radial/height layer changing last number of subfields of first/second `layer`:

```
metadata:
  run:
    factory: geo
    strategy:
      class: strategy.NoBoolean
data:
  class: block.Layer
  layer: [ [ 0.5;;8, 1.5;;4 ],
           [ 1;;4, 2;;8, 3;;16 ] ]
  "layer_curves": [ [ line, circle_arc ],
                    [ line, line, line ] ]
  items_do_structure_map: 1
  items_do_quadrate_map: 1
```

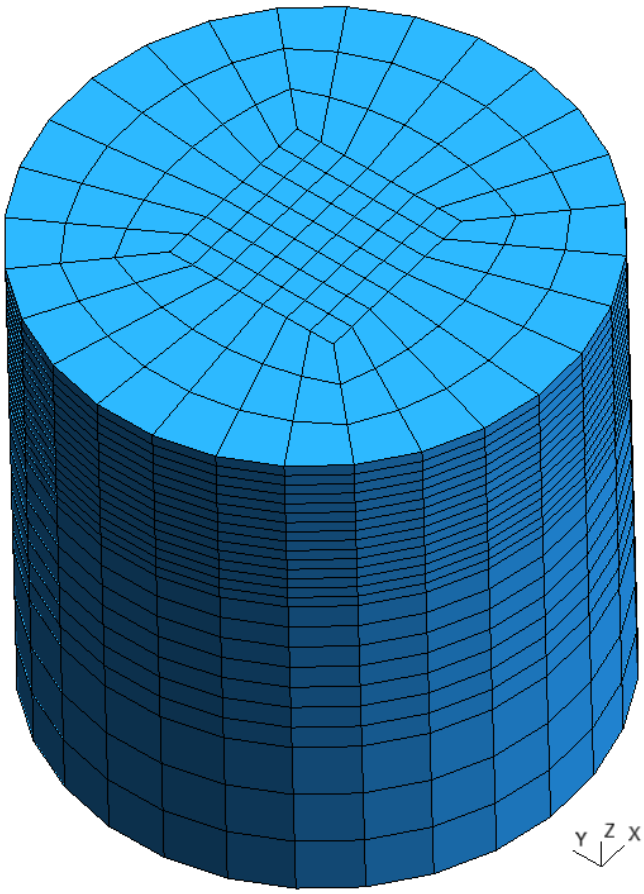

Fig. 17: Layer with different number of nodes by radial/height layers

1.2.2 Number of layers

One can add radial layers by appending additional items in the first `layer` subfield:

```
metadata:
run:
  factory: geo
  strategy:
    class: strategy.NoBoolean
data:
class: block.Layer
layer: [ [ 0.5;;8, 1.5;;2, 2.5;;2, 4;;2, 5;;2 ],
         [ 1;;2, 2;;2, 3;;2 ] ]
"layer_curves": [ [ line, circle_arc, line, circle_arc, line ],
                  [ line, line, line ] ]
items_do_structure_map: 1
items_do_quadrate_map: 1
```

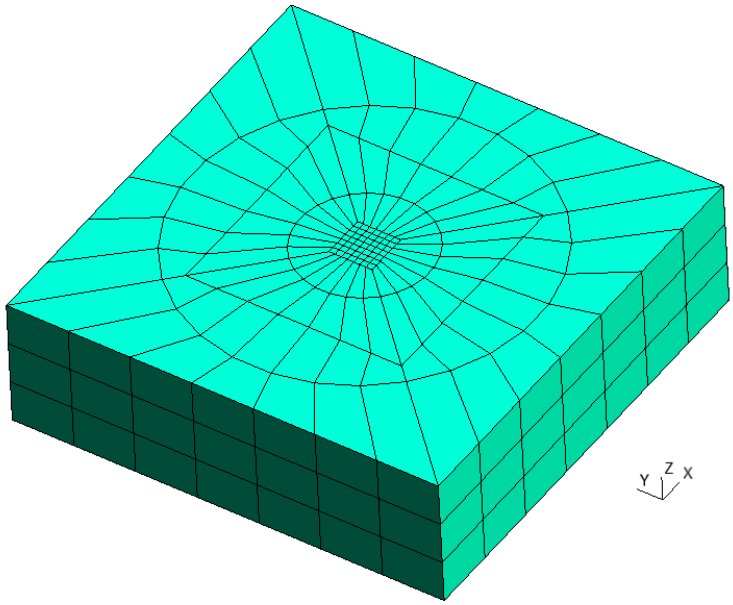

Fig. 18: Multi Layer

One can add height layers by appending additional items in the second `layer` subfield:

```
metadata:
run:
  factory: geo
  strategy:
    class: strategy.NoBoolean
data:
class: block.Layer
layer: [ [ 0.5;;8, 1.5;;2, 2.5;;2, 4;;2, 5;;2 ],
         [ 1;;2, 2;;2, 3;;2, 6;;2, 10;;2 ] ]
"layer_curves": [ [ line, circle_arc, line, circle_arc, line ],
```

(continues on next page)

(continued from previous page)

```
                    [ line, line, line, line, line ] ]
items_do_structure_map: 1
items_do_quadrate_map: 1
```

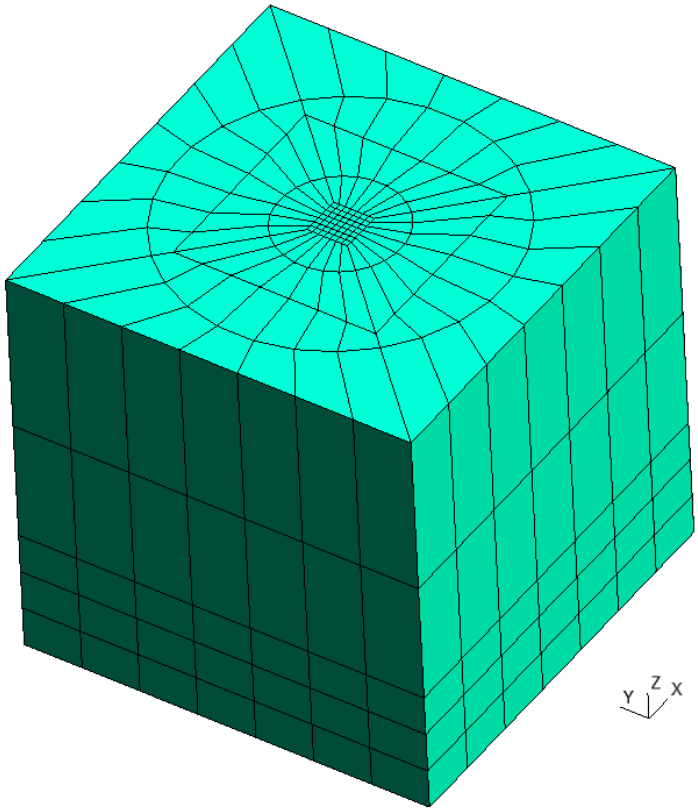

Fig. 19: Multi Height Layer

1.2.3 Zones

To specify zones one needs add `items_zones` and `items_zones_map` fields. Where `items_zones` is a list of pairs of volume and six surfaces names, e.g. `[ Volume, [ NX, X, NY, Y, NZ, Z ] ]`:

1. `Volume` - volume name
2. `[ NX, X, NY, Y, NZ, Z ]` - surfaces names:
    - `NX` - surface pointing in the opposite direction of X-axis
    - `X` - surface pointing in the direction of X-axis
    - `NY` - surface pointing in the opposite direction of Y-axis
    - `Y` - surface pointing in the direction of Y-axis
    - `NZ` - surface pointing in the opposite direction of Z-axis
    - `Z` - surface pointing in the direction of Z-axis

`items_zones_map` (and all fields that ends with `_map`) is an addressing array between items (Blocks in Layer) and corresponding index in some list with properties (e.g. `items_zones`) and has shape `number-of-height` by `number-of-radial` layers.

If one want to assign zone names with index `1` from `items_zones` (`[ B, [ NX_B, X_B, NY_B, Y_B, NZ_B, Z_B ] ]`) to 3th height layer and 5th radial layer one can set `1` to `items_zones_map` at `[3, 5]` location.

In this example we set:

1. `0` index of `items_zones` to middle (3th) height layer by all radial layers except last (5th);
2. `1` index of `items_zones` to 2nd and 4th height layer by all radial layers and also last (5th) radial layer in the middle (3th) height layer;
3. `2` index of `items_zones` to bottom (1st) and top (5th) height layer by all radial layers.

```
metadata:
  run:
    factory: geo
    strategy:
      class: strategy.NoBoolean
data:
  class: block.Layer
  layer: [ [ 0.5;;8, 1.5;;2, 2.5;;2, 4;;2, 5;;2 ],
           [ 1;;2, 2;;2, 3;;2, 6;;2, 10;;2 ] ]
  "layer_curves": [ [ line, circle_arc, line, circle_arc, line ],
                    [ line, line, line, line, line ] ]
  items_do_structure_map: 1
  items_do_quadrate_map: 1
  items_zone: [
    [ Red, [ NX, X, NY, Y, NZ, Z ] ],
    [ Green, [ NX, X, NY, Y, NZ, Z ] ],
    [ Blue, [ NX, X, NY, Y, NZ, Z ] ]
  ]
  items_zone_map: [
    [ 2, 2, 2, 2, 2],
    [ 1, 1, 1, 1, 1],
    [ 0, 0, 0, 0, 1],
    [ 1, 1, 1, 1, 1],
    [ 2, 2, 2, 2, 2]
  ]
```

1.3 Cross section 2

Let's consider a cross section (See examples/matrix/cross_section_2)

We can use *`Matrix`* class to create this type of geometry. First we need to decompose geometry into blocks and then modify each of them according to the dimensions.

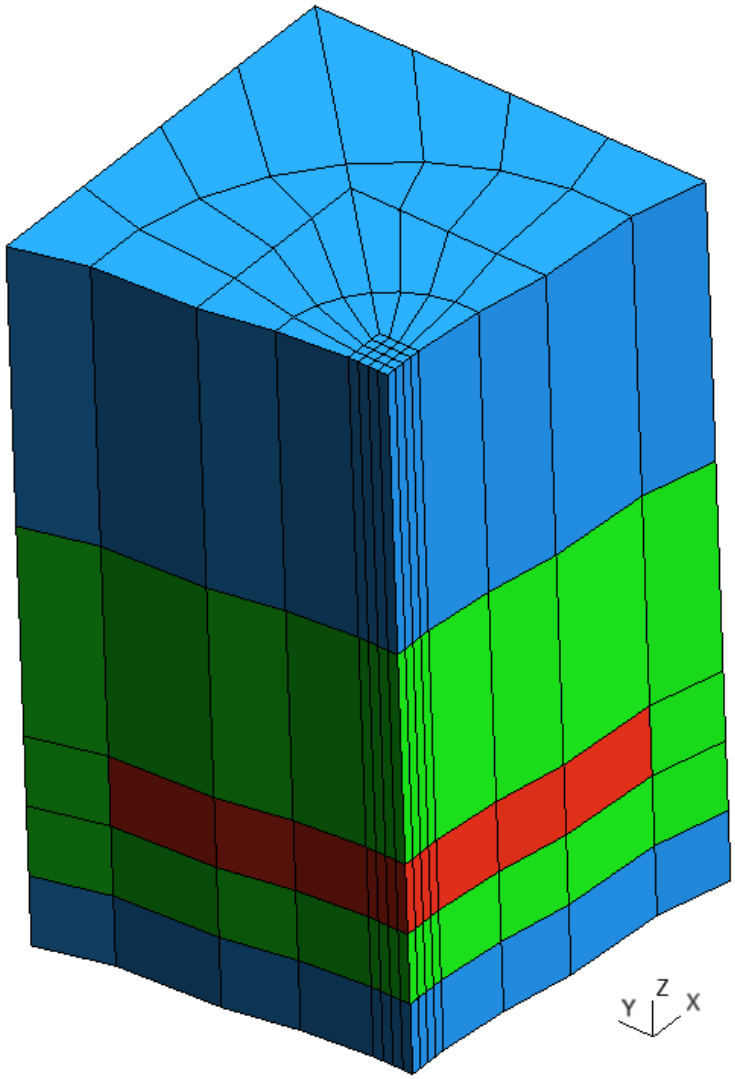

Fig. 20: Section by X and Y axes of Layer with zones: red - 0, green - 1, blue - 2

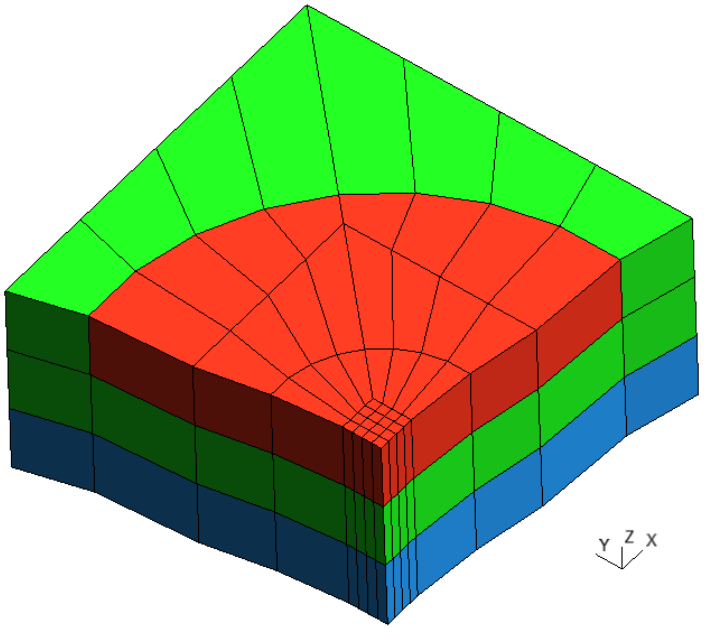

Fig. 21: Section by X, Y and Z axes of Layer with zones: red - `0`, green - `1`, blue - `2`

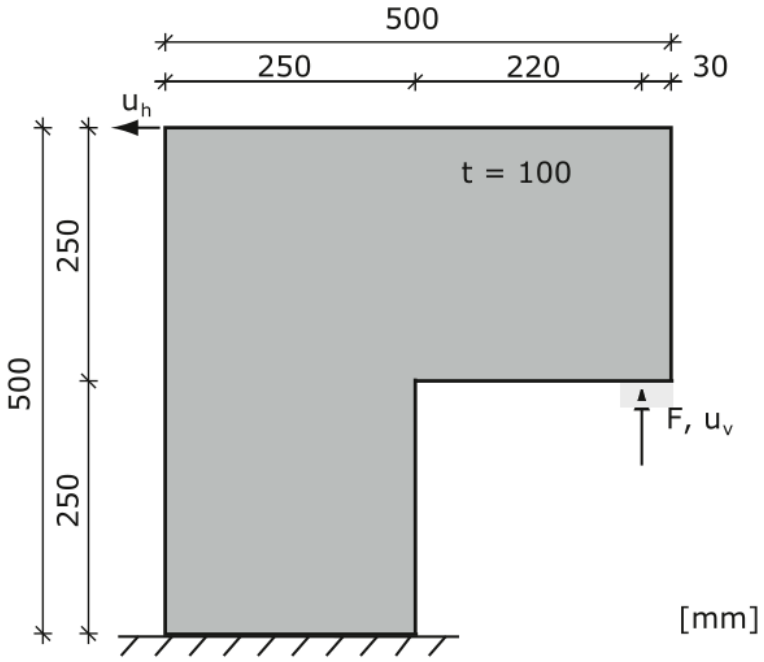

Fig. 22: Cross section

1.3.1 Decomposition

Geometry could be decomposed into 4 blocks: `BOTTOM`, `TOP1`, `TOP2` and `TOP3`, where `TOP3` part is required for a small surface with F, u_v boundary condition at the end of the upper part of the geometry. Without the boundary condition, 3 blocks would be enough. Each of these blocks will be described by *`Matrix`* class.

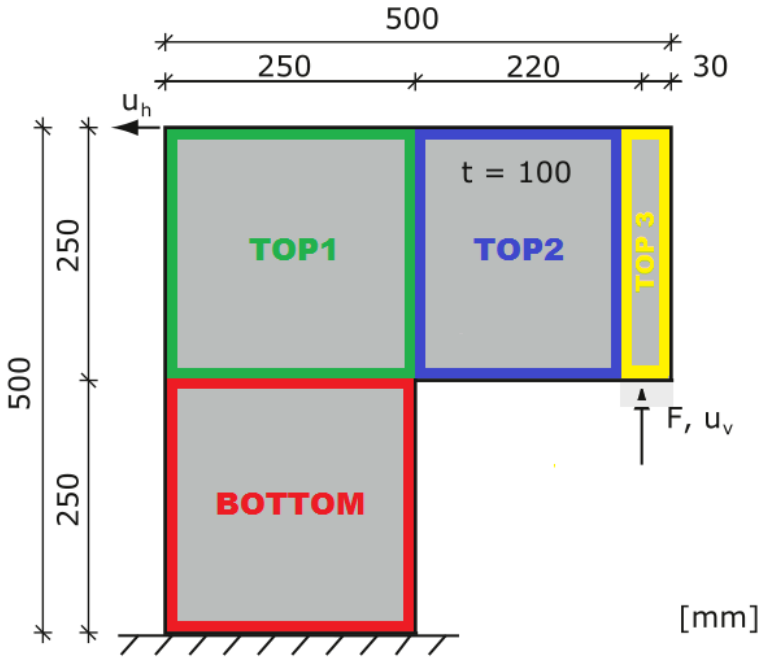

Fig. 23: Decomposition of geometry

1.3.2 Geometry

Let's define that X-axis is directed to the right, Y - in the depth and Z - upward, i.e cross section is symmetric along Y-axis.

First we should create a separate file for each of the blocks:

- `top_1.yaml`

```
data:
  class: block.Matrix
  matrix: [ [ 0, 0.250 ], [ 0, 1 ], [ 0, 0.250 ] ]
```

- `top_2.yaml`

```
 data:
   class: block.Matrix
   matrix: [ [ 0, 0.220 ], [ 0, 1 ], [ 0, 0.250 ] ]
```

- `top_3.yaml`

```
data:
  class: block.Matrix
  matrix: [ [ 0, 0.030 ], [ 0, 1 ], [ 0, 0.250 ] ]
```

- bottom.yaml

```
data:
  class: block.Matrix
  matrix: [ [ 0, 0.250 ], [ 0, 1 ], [ 0, 0.250 ] ]
```

Each of the files consists of one high level field `data` whose has 2 fields: 1. `class` - name of the class of the block 2. `matrix` - lists of point coordinates by axes

For example Matrix has 2 points by X-axis with coordinates `0` and `0.250`.

Matrix also has 2 points by Y-axis with `0` and `1` coordinates and 2 points by Z-axis with `0` and `0.250`. Thus Matrix is a box with dimensions: `0.250`, `1` and `0.250` by X, Y and Z axis respectively and origin at point `(0, 0, 0)`.

We could generate geometry of `bottom.yaml` into `bottom.geo_unrolled` file:

```
python -m gmsh_scripts bottom.yaml
```

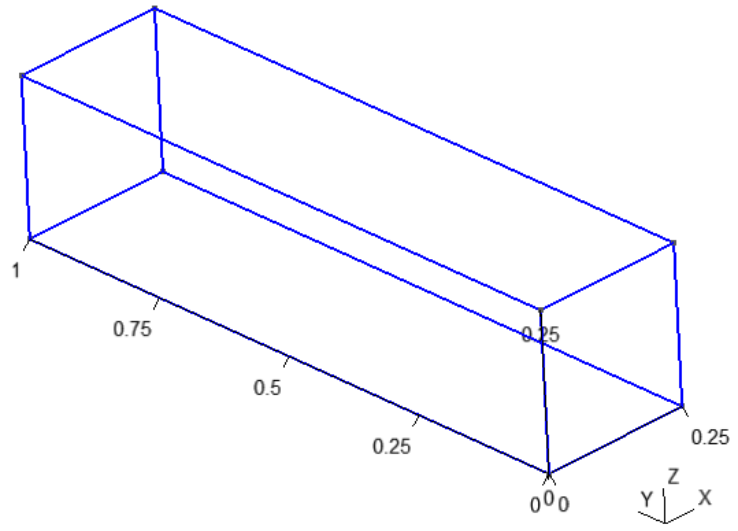

Fig. 24: Geometry of the the `BOTTOM` block

Now we should create main file `main.yaml` with all blocks:

```
data:
  class: block.Block
  do_register: 0
  children: [
    /bottom.yaml,
    /top_1.yaml,
    /top_2.yaml,
    /top_3.yaml
  ]
  children_transforms: [
    [ ],
    [ [ 0, 0, 0.250 ] ],
    [ [ 0.250, 0, 0.250 ] ],
    [ [ 0.470, 0, 0.250 ] ]
  ]
```

File also has one high level field `data` with 4 sub-fields:

1. `class` - name of the class of the block
2. `do_register` - create this block? (set 0 because we don't need this block itself, i.e. it's only a container for other blocks)
3. `children` - references to other block files (should start with `/` character)
4. `children_transforms` - transforms of other blocks

Field `children_transforms` is a list of *Transform* for each `children`. In this tutorial we only need simple *Translate* that are given by 3 numbers - offset along X, Y ans Z axes respectively.

For example:

1. Child `bottom.yaml` has no transforms
2. Child `top_1.yaml` has one *Translate* `[ 0, 0, 0.250 ]` with offset 0.250 by Z-axis and no offsets by X and Y (we just need to elevate to the `bottom.yaml`)
3. Child `top_2.yaml` has one *Translate* `[ 0.250, 0, 0.250 ]`
4. Child `top_3.yaml` has one *Translate* `[ 0.470, 0, 0.250 ]`

Let's generate geometry with all blocks into `main.geo_unrolled`:

```
python -m gmsh_scripts main.yaml
```

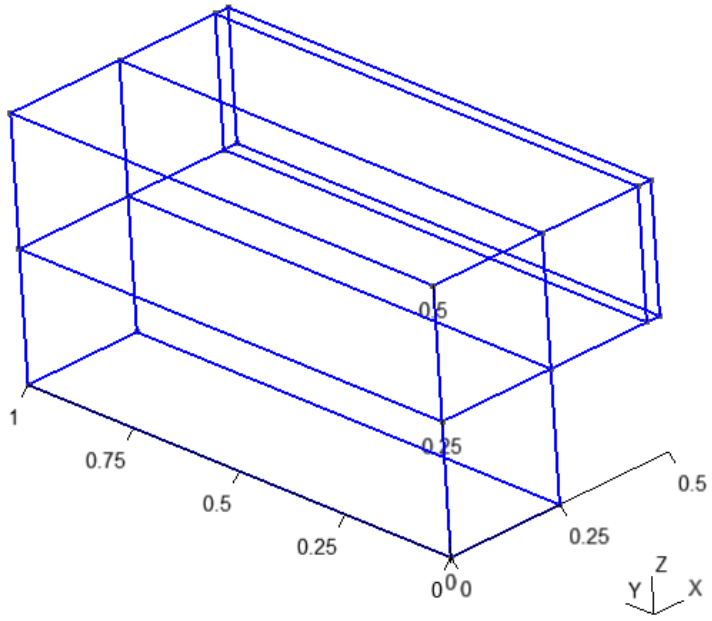

Fig. 25: Geometry with all blocks

1.3.3 Mesh

To generate mesh we should add `metadata` field to the `main.yaml` file:

```yaml
metadata:
  run:
    factory: geo
    strategy:
      class: strategy.NoBoolean
data:
  class: block.Block
  do_register: 0
  children: [
    /bottom.yaml,
    /top_1.yaml,
    /top_2.yaml,
    /top_3.yaml
  ]
  children_transforms: [
    [ ],
    [ [ 0, 0, 0.250 ] ],
    [ [ 0.250, 0, 0.250 ] ],
    [ [ 0.470, 0, 0.250 ] ]
  ]
```

File `metadata` has `run` sub-field with fields:

1. `factory` - Which kernel of gmsh to use for mesh generation? Currently, gmsh has two kernels: `geo` and `occ`. We use `geo` because it's faster
2. `strategy` - *`Strategy`* of mesh generation
3. `strategy.class` - Class of the strategy. We use *`NoBoolean`* because we don't need boolean operations

> **Warning:** If we need boolean operations we MUST use `occ` factory with default strategy (just don't set it in the `metadata`)

Now mesh generator will return mesh into `main.msh2` file (it also returns `main.geo_unrolled` as before). Generator creates unstructured tetrahedral mesh by default.

```
python -m gmsh_scripts main.yaml
```

1.3.3.1 Unstructured Tetrahedral

We can customize unstructured mesh with parameters in input files.

First type of parameters aka `point parameters` is set in `matrix` field (e.g. `bottom.yaml`):

```yaml
data:
  class: block.Matrix
  matrix: [ [ 0;0.01, 0.250;0.1 ], [ 0;0.01, 1;0.1 ], [ 0;0.01, 0.250;0.1 ] ]
```

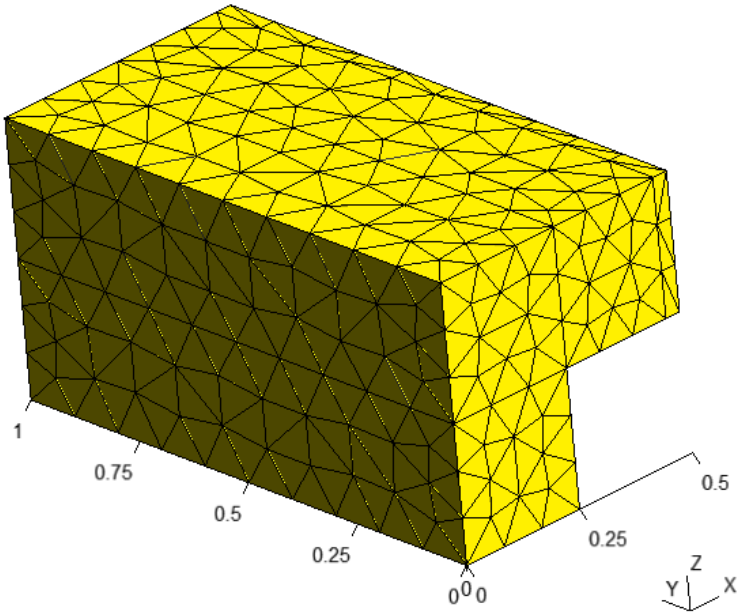

Fig. 26: Default mesh

As one can see, for each point a new parameter have been added with `;` separator, e.g. `0;0.01` for first point by X-axis or `0.250;0.1` for second point by Z-axis. Parameters `0.01` or `0.1` are approximate sizes of the mesh near the corresponding points.

In this example, mesh is finer near the `(0, 0, 0)` point with size `0.01` and coarser near the `(0.250, 1, 0.250)` point with size `0.1`.

Let's add `metadata` field to `bottom.yaml` and generate mesh:

```
python -m gmsh_scripts bottom.yaml
```

```
metadata:
  run:
    factory: geo
    strategy:
      class: strategy.NoBoolean
data:
  class: block.Matrix
  matrix: [ [ 0;0.01, 0.250;0.1 ], [ 0;0.01, 1;0.1 ], [ 0;0.01, 0.250;0.1 ] ]
```

One could fix mesh size along one of the axis (e.g. Y with `0.01`):

```
metadata:
  run:
    factory: geo
    strategy:
      class: strategy.NoBoolean
data:
  class: block.Matrix
  matrix: [ [ 0;0.01, 0.250;0.1 ], [ 0;0.01, 1;0.01 ], [ 0;0.01, 0.250;0.1 ] ]
```

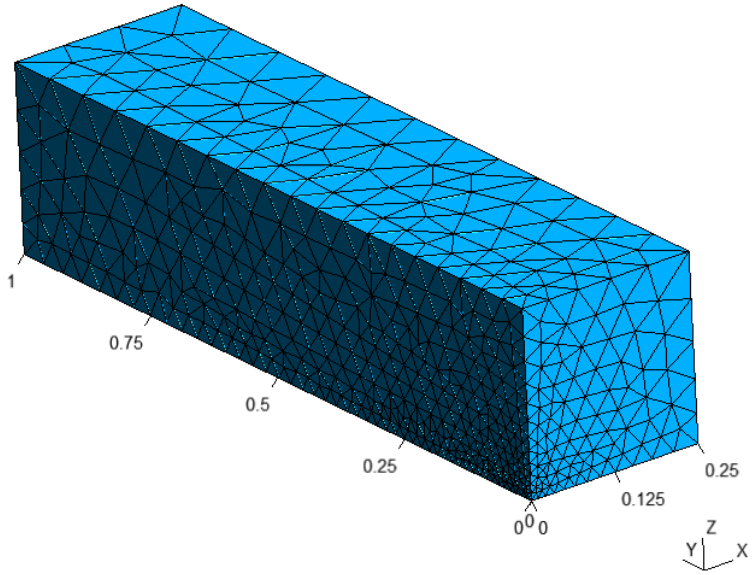

Fig. 27: Unstructured tetrahedral mesh of the `BOTTOM` block

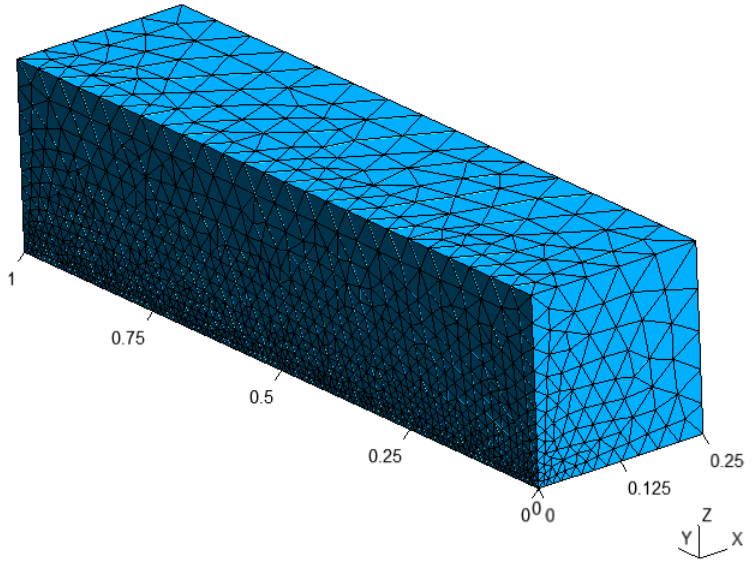

Fig. 28: Unstructured tetrahedral mesh with fixed size along Y-axis of the `BOTTOM` block

To generate all blocks, one needs to specify point parameters at all blocks and run generator:

```
python -m gmsh_scripts main.yaml
```

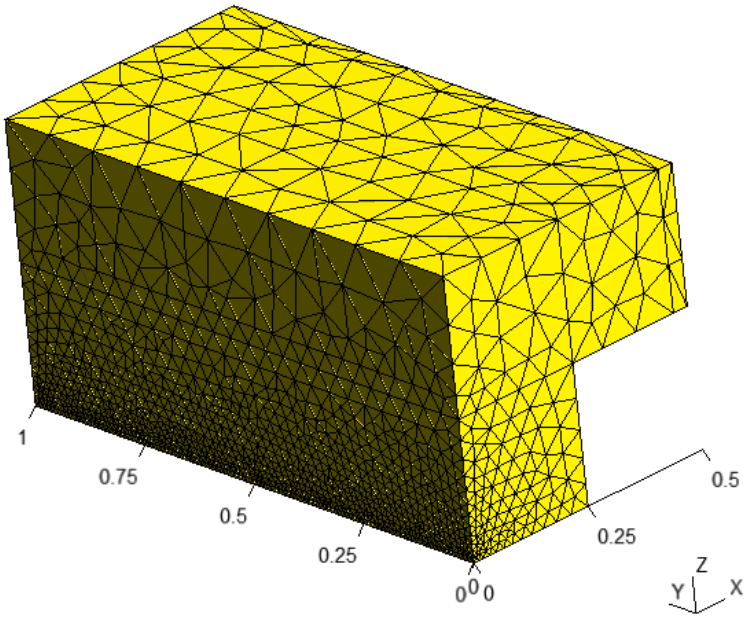

Fig. 29: Unstructured tetrahedral mesh with fixed size along Y-axis at `BOTTOM` block

Second type of parameters aka `global parameters` is set in `metadata.run.options` field (e.g. `bottom.yaml`):

```yaml
metadata:
  run:
    factory: geo
    strategy:
      class: strategy.NoBoolean
    options:
      Mesh.MeshSizeFactor: 0.5
      Mesh.MeshSizeMin: 0
      Mesh.MeshSizeMax: 1.0e+22
      Mesh.MeshSizeFromPoints: 1
data:
  class: block.Matrix
  matrix: [ [ 0;0.01, 0.250;0.1 ], [ 0;0.01, 1;0.01 ], [ 0;0.01, 0.250;0.1 ] ]
```

Here are 4 options (many other options available, see gmsh documentation): 1. `Mesh.MeshSizeFactor` - factor applied to all mesh element sizes 2. `Mesh.MeshSizeMin` - minimum mesh element size 3. `Mesh.MeshSizeMax` - maximum mesh element size 4. `Mesh.MeshSizeFromPoints` - compute mesh element sizes from values given at geometry points (e.g. in `matrix` field)

In this example `Mesh.MeshSizeFactor` is set to `0.5` that generate mesh that is twice as fine.

One could disable `Mesh.MeshSizeFromPoints` (set to `0`) to create uniform mesh whose size is controlled only by `global parameters`.

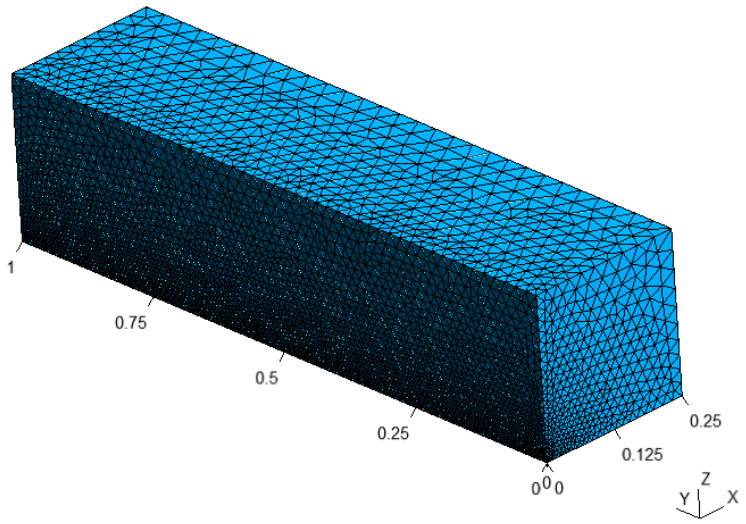

Fig. 30: Unstructured tetrahedral mesh with `Mesh.MeshSizeFactor = 0.5`

```
metadata:
  run:
    factory: geo
    strategy:
      class: strategy.NoBoolean
    options:
      Mesh.MeshSizeFactor: 1
      Mesh.MeshSizeMin: 0
      Mesh.MeshSizeMax: 1.0e+22
      Mesh.MeshSizeFromPoints: 0
data:
  class: block.Matrix
  matrix: [ [ 0;0.01, 0.250;0.1 ], [ 0;0.01, 1;0.01 ], [ 0;0.01, 0.250;0.1 ] ]
```

Then we could use to control mesh size, e.g with `Mesh.MeshSizeMax = 0.1`:

```
metadata:
  run:
    factory: geo
    strategy:
      class: strategy.NoBoolean
    options:
      Mesh.MeshSizeFactor: 1
      Mesh.MeshSizeMin: 0
      Mesh.MeshSizeMax: 0.1
      Mesh.MeshSizeFromPoints: 0
data:
  class: block.Matrix
  matrix: [ [ 0, 0.250 ], [ 0, 1 ], [ 0, 0.250 ] ]
```

To generate all blocks, one needs to specify `global parameters` in `main.yaml`:

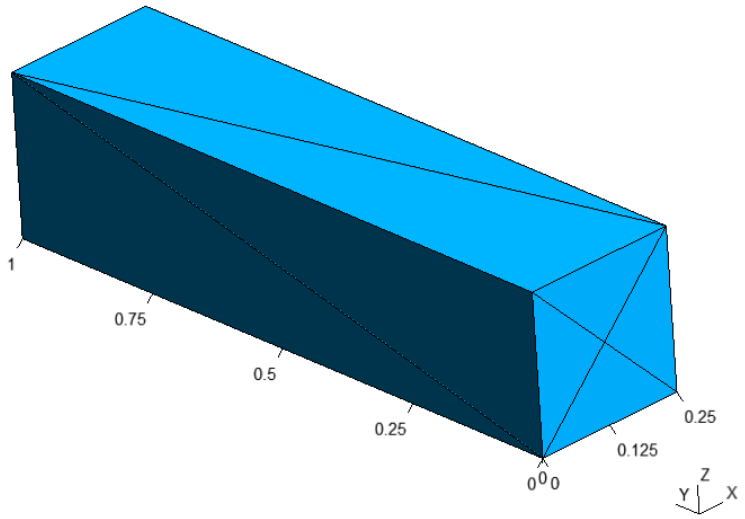

Fig. 31: Unstructured tetrahedral mesh with `Mesh.MeshSizeFromPoints = 0`

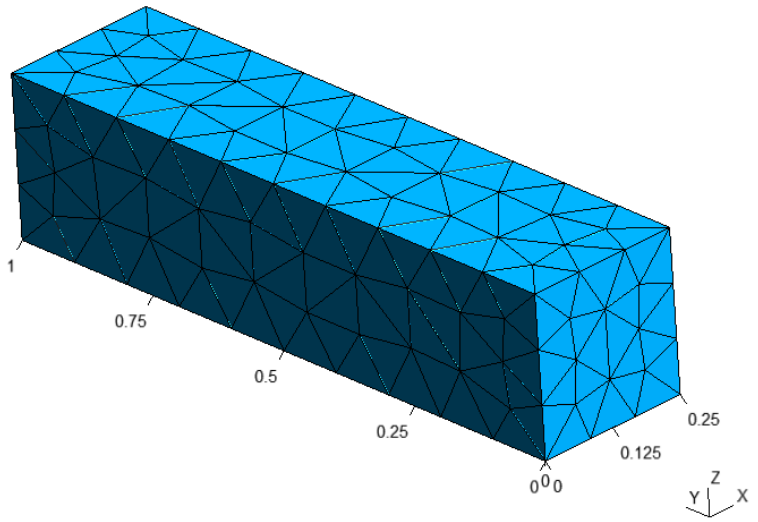

Fig. 32: Unstructured mesh with `Mesh.MeshSizeMax = 0.1`

```yaml
metadata:
  run:
    factory: geo
    strategy:
      class: strategy.NoBoolean
    options:
      Mesh.MeshSizeFactor: 1
      Mesh.MeshSizeMin: 0
      Mesh.MeshSizeMax: 0.1
      Mesh.MeshSizeFromPoints: 0
data:
  class: block.Block
  do_register: 0
  children: [
    /bottom.yaml,
    /top_1.yaml,
    /top_2.yaml,
    /top_3.yaml
  ]
  children_transforms: [
    [ ],
    [ [ 0, 0, 0.250 ] ],
    [ [ 0.250, 0, 0.250 ] ],
    [ [ 0.470, 0, 0.250 ] ]
  ]
```

```
python -m gmsh_scripts main.yaml
```

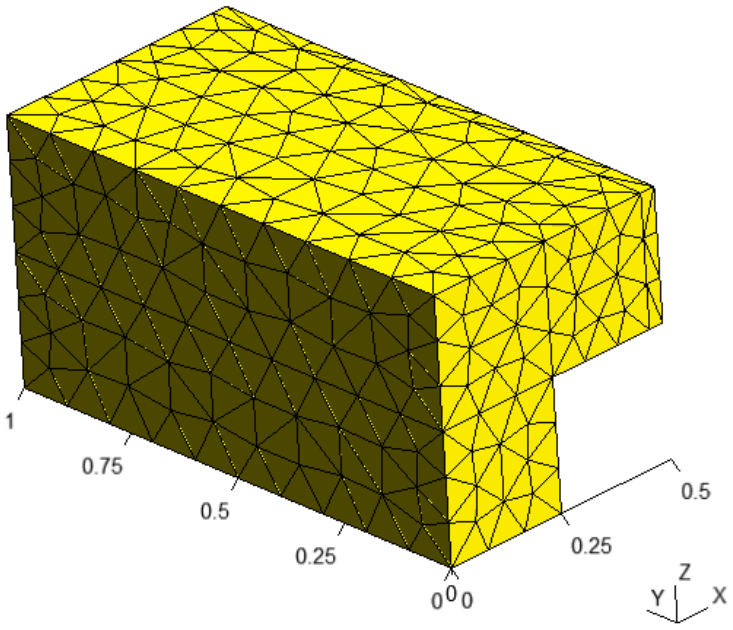

Fig. 33: Unstructured tetrahedral mesh controlled by `global parameters`

1.3.3.2 Unstructured Hexahedral

> **Warning:** Generation of hexahedral unstructured mesh is experimental so not always creates a quality mesh, it depends on the complexity of the geometry.

Unstructured hexahedral parameters are set in `metadata.run.options` field and have `Recombine` in their names (see gmsh options)

To generate unstructured hexahedral mesh parameter `` `Mesh.SubdivisionAlgorithm `` should be set greater than 1 (see tutorial 11 of gmsh for more information)

```yaml
metadata:
  run:
    factory: geo
    strategy:
      class: strategy.NoBoolean
    options:
      Mesh.MeshSizeFactor: 1
      Mesh.MeshSizeMin: 0
      Mesh.MeshSizeMax: 0.1
      Mesh.MeshSizeFromPoints: 0
      Mesh.SubdivisionAlgorithm: 2
data:
  class: block.Matrix
  matrix: [ [ 0, 0.250 ], [ 0, 1 ], [ 0, 0.250 ] ]
```

```
python -m gmsh_scripts bottom.yaml
```

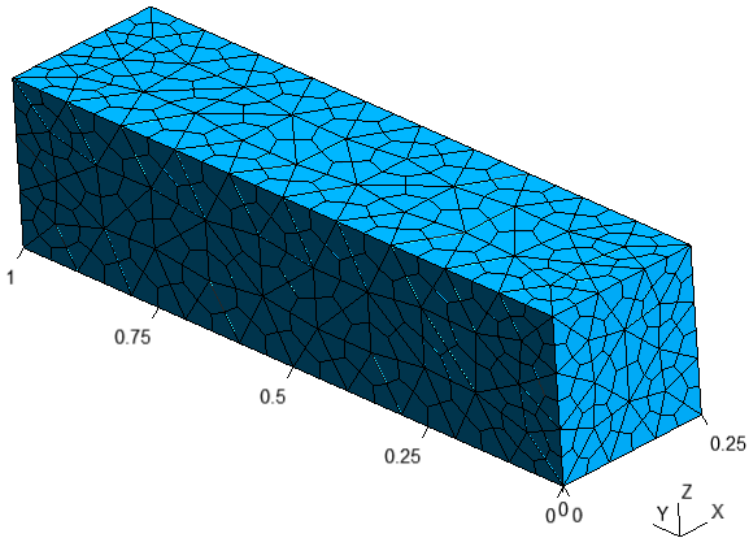

Fig. 34: Unstructured hexahedral mesh of `bottom.yaml`

To generate unstructured hexahedral mesh of all blocks add parameters to `metadata` of `main.yaml`:

```yaml
metadata:
  run:
    factory: geo
    strategy:
      class: strategy.NoBoolean
    options:
      Mesh.MeshSizeFactor: 1
      Mesh.MeshSizeMin: 0
      Mesh.MeshSizeMax: 0.1
      Mesh.MeshSizeFromPoints: 0
      Mesh.SubdivisionAlgorithm: 2
data:
  class: block.Block
  do_register: 0
  children: [
    /bottom.yaml,
    /top_1.yaml,
    /top_2.yaml,
    /top_3.yaml
  ]
  children_transforms: [
    [ ],
    [ [ 0, 0, 0.250 ] ],
    [ [ 0.250, 0, 0.250 ] ],
    [ [ 0.470, 0, 0.250 ] ]
  ]
```

```
python -m gmsh_scripts main.yaml
```

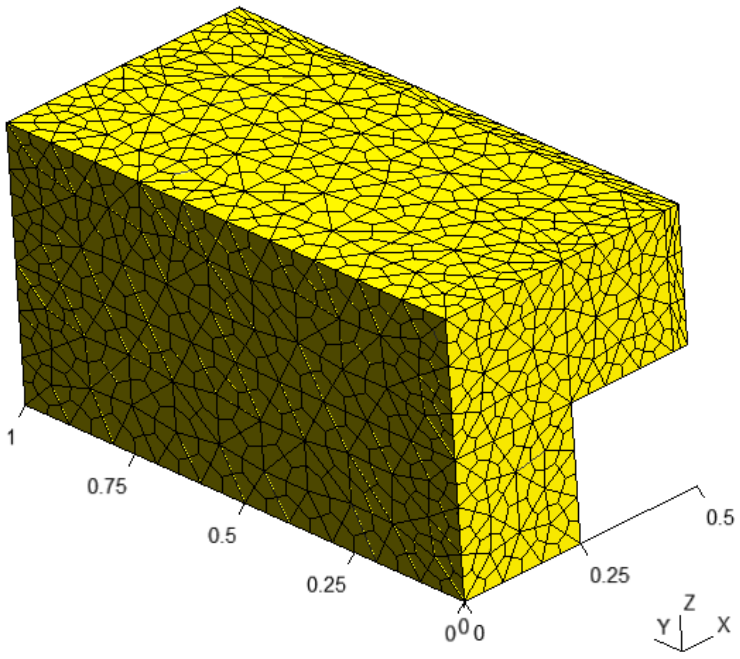

Fig. 35: Unstructured hexahedral mesh

1.3.3.3 Structured Tetrahedral

To create structured tetrahedral mesh one should add third parameter to the points at `matrix` field with `;` separator, e.g. in `bottom.yaml`:

```
metadata:
   run:
     factory: geo
     strategy:
       class: strategy.NoBoolean

data:
  class: block.Matrix
  matrix: [ [ 0;0.01, 0.250;0.1;4 ], [ 0;0.01, 1;0.1;8 ], [ 0;0.01, 0.250;0.1;16 ] ]
```

Third argument should be set only for second point and specifies number of nodes along corresponding direction. E.g. `4` nodes by X-axis, `8` nodes by Y and `16` by Z.

```
python -m gmsh_scripts bottom.yaml
```

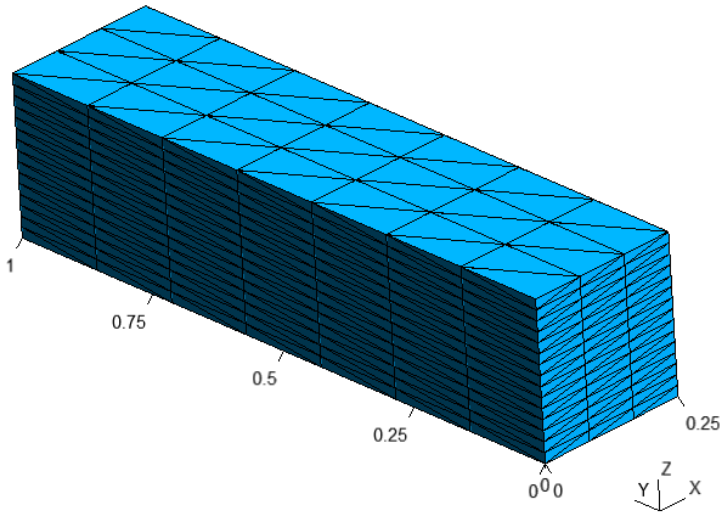

Fig. 36: Structured tetrahedral mesh of the `BOTTOM` block

One could disable structured mesh generation by setting `items_do_structure_map` to `0` (`1` by default) in the `data` field:

```
metadata:
   run:
     factory: geo
     strategy:
       class: strategy.NoBoolean

data:
  class: block.Matrix
```

(continues on next page)

(continued from previous page)

```
  matrix: [ [ 0;0.01, 0.250;0.1;4 ], [ 0;0.01, 1;0.1;8 ], [ 0;0.01, 0.250;0.1;16 ] ]
  items_do_structure_map: 0
```

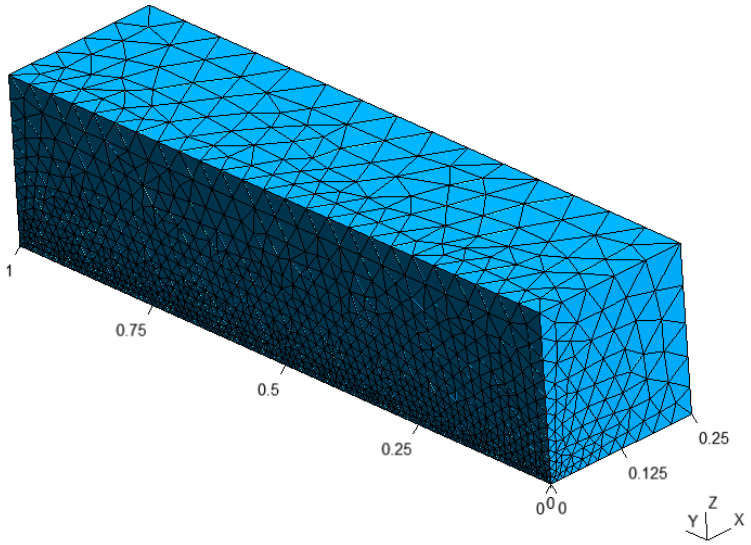

Fig. 37: Unstructured tetrahedral mesh of the BOTTOM block with items_do_structure_map = 0

To create structured tetrahedral mesh with all blocks one should set third parameter in each of the blocks and run main.yaml

Warning: Number of nodes MUST be consistent between adjacent blocks, e.g. all TOP blocks should have the same number of nodes by Y and Z axis

- top_1.yaml

```
data:
  class: block.Matrix
  matrix: [ [ 0, 0.250;0.1;8 ], [ 0, 1;0.1;8 ], [ 0, 0.250;0.1;8 ] ]
```

- top_2.yaml

```
 data:
   class: block.Matrix
   matrix: [ [ 0, 0.220;0.1;8 ], [ 0, 1;0.1;8 ], [ 0, 0.250;0.1;8 ] ]
```

- top_3.yaml

```
data:
  class: block.Matrix
  matrix: [ [ 0, 0.030;0.1;8 ], [ 0, 1;0.1;8 ], [ 0, 0.250;0.1;8 ] ]
```

- bottom.yaml

```
data:
  class: block.Matrix
  matrix: [ [ 0, 0.250;0.1;8 ], [ 0, 1;0.1;8 ], [ 0, 0.250;0.1;8 ] ]
```

```
python -m gmsh_scripts main.yaml
```

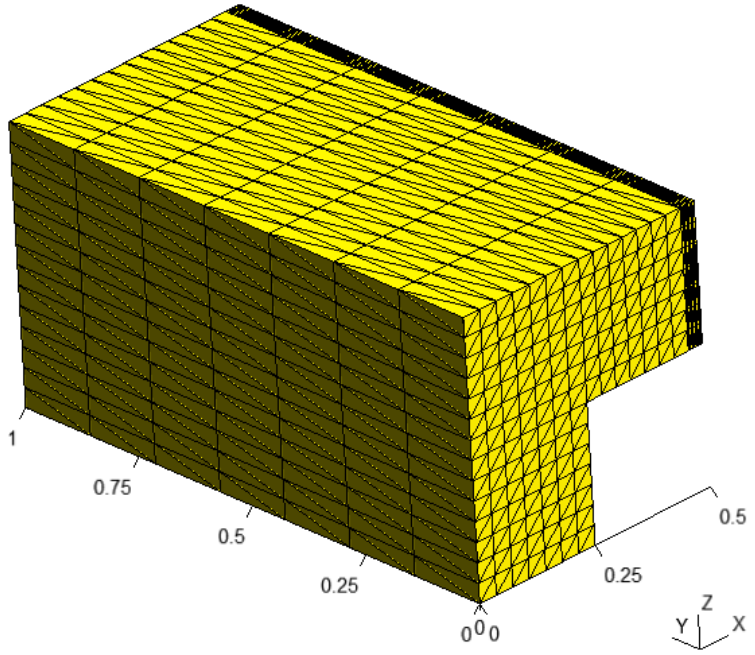

Fig. 38: Structured tetrahedral mesh

To disable generation of structured mesh for all blocks at once one should set `children_items_do_structure_map = [0, ..., number of children]` at parent block, e.g. for `main.yaml`:

```yaml
metadata:
  run:
    factory: geo
    strategy:
      class: strategy.NoBoolean
data:
  class: block.Block
  do_register: 0
  children: [
    /bottom.yaml,
    /top_1.yaml,
    /top_2.yaml,
    /top_3.yaml
  ]
  children_transforms: [
    [ ],
    [ [ 0, 0, 0.250 ] ],
    [ [ 0.250, 0, 0.250 ] ],
    [ [ 0.470, 0, 0.250 ] ]
  ]
  children_items_do_structure_map: [0, 0, 0, 0]
```

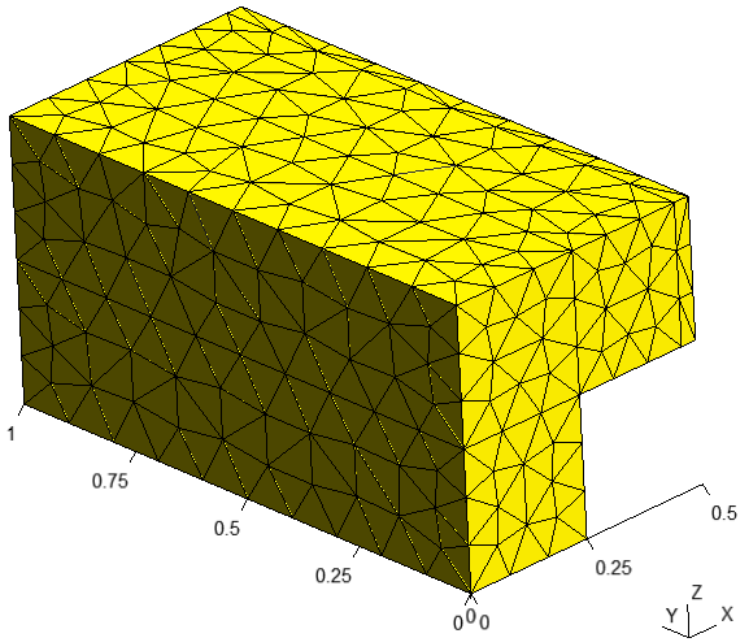

Fig. 39: Structured tetrahedral mesh with disabled `children_items_do_structure_map`

1.3.3.4 Structured Hexahedral

For creating structured hexahedral mesh one could do the same steps as for *Structured Tetrahedral* but with `items_do_quadrate_map` = 1 (`0` by default) in the `data` field:

```
metadata:
   run:
     factory: geo
     strategy:
       class: strategy.NoBoolean

data:
  class: block.Matrix
  matrix: [ [ 0;0.01, 0.250;0.1;4 ], [ 0;0.01, 1;0.1;8 ], [ 0;0.01, 0.250;0.1;16 ] ]
  items_do_quadrate_map: 1
```

```
python -m gmsh_scripts bottom.yaml
```

One could change positions of nodes along axes using one of the two methods:

1. `progression` - increase/decrease space between nodes from start point to end point
2. `bump` - increase/decrease space between node from center to points

To use `progression` we should specify 2 additional sub-parameters to the third parameter separated by `:`:

1. The first one is `0` (which choose `progression` type)
2. The second is a coefficient of the `progression` - if coefficient > 1 space will be increasing from first point to second else decreasing

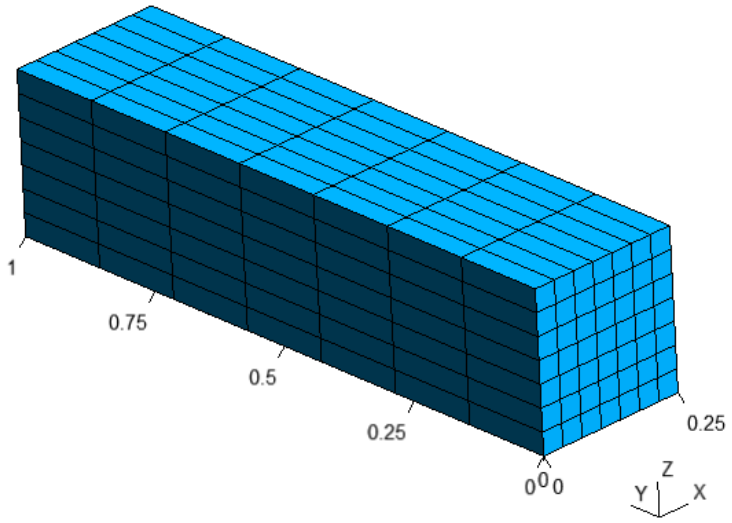

Fig. 40: Structured hexahedral mesh of the `BOTTOM` block

For example, `progression` sub-parameters `0:1.5` for Y-axis:

```
metadata:
   run:
     factory: geo
     strategy:
       class: strategy.NoBoolean

data:
  class: block.Matrix
  matrix: [ [ 0;0.01, 0.250;0.1;4 ], [ 0;0.01, 1;0.1;8:0:1.5 ], [ 0;0.01, 0.250;0.1;16 ]␣
↪]
  items_do_quadrate_map: 1
```

For example, `progression` sub-parameters `0:0.75` for Y-axis:

```
metadata:
   run:
     factory: geo
     strategy:
       class: strategy.NoBoolean

data:
  class: block.Matrix
  matrix: [ [ 0;0.01, 0.250;0.1;4 ], [ 0;0.01, 1;0.1;8:0:0.75 ], [ 0;0.01, 0.250;0.1;16␣
↪] ]
  items_do_quadrate_map: 1
```

To use `bump` we should specify 2 additional sub-parameters to the third parameter separated by `::`

1. The first one is `1` (which choose `bump` type)
2. The second is a coefficient of the `bump` - if coefficient > 1 space will be increasing from the center else decreasing

For example, `bump` sub-parameters `1:2.0` for Y-axis:

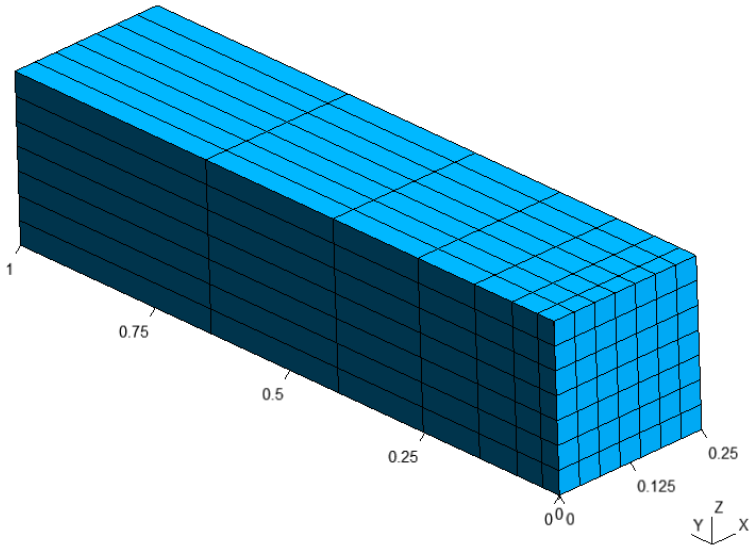

Fig. 41: Structured hexahedral mesh with `progression` = *1.5*

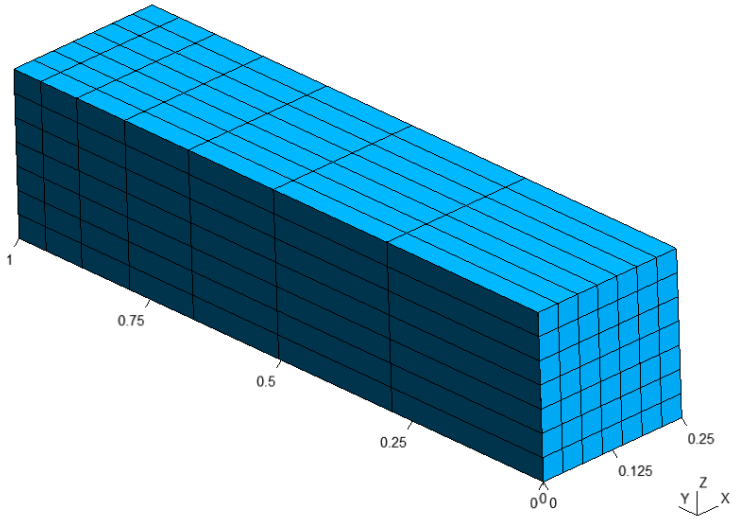

Fig. 42: Structured hexahedral mesh with `progression` = *0.75*

```
metadata:
   run:
     factory: geo
     strategy:
       class: strategy.NoBoolean

data:
  class: block.Matrix
  matrix: [ [ 0;0.01, 0.250;0.1;4 ], [ 0;0.01, 1;0.1;8:1:2.0 ], [ 0;0.01, 0.250;0.1;16 ]␣
↪]
  items_do_quadrate_map: 1
```

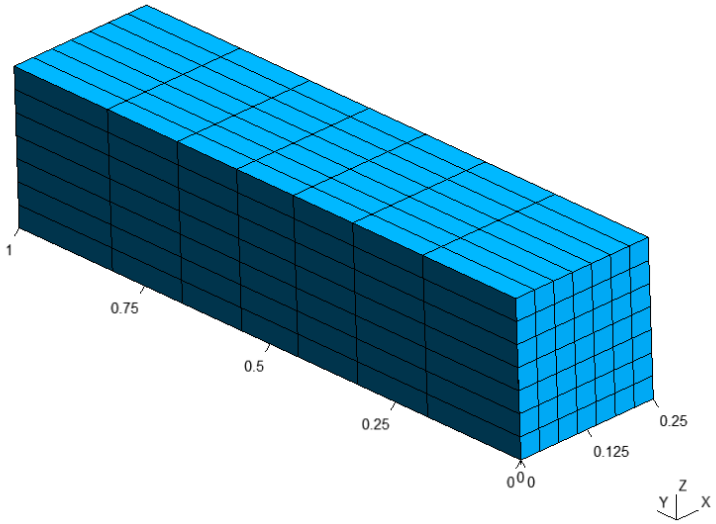

Fig. 43: Structured hexahedral mesh with bump = *2.0*

For example, bump sub-parameters `1:0.5` for Y-axis:

```
metadata:
   run:
     factory: geo
     strategy:
       class: strategy.NoBoolean

data:
  class: block.Matrix
  matrix: [ [ 0;0.01, 0.250;0.1;4 ], [ 0;0.01, 1;0.1;8:1:0.5 ], [ 0;0.01, 0.250;0.1;16 ]␣
↪]
  items_do_quadrate_map: 1
```

To generate structured hexahedral mesh of all blocks one could set `items_do_quadrate_map = 0` at each of the blocks or set `children_items_do_quadrate_map = [0, ..., number of children]` at parent block, e.g. for `main.yaml`:

```
metadata:
  run:
    factory: geo
```

(continues on next page)

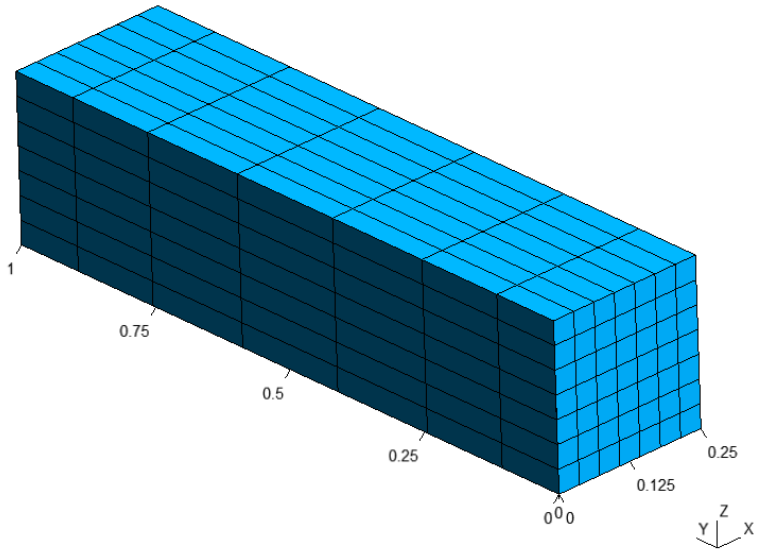

Fig. 44: Structured hexahedral mesh with bump = *0.5*

(continued from previous page)

```
    strategy:
      class: strategy.NoBoolean
data:
  class: block.Block
  do_register: 0
  children: [
    /bottom.yaml,
    /top_1.yaml,
    /top_2.yaml,
    /top_3.yaml
  ]
  children_transforms: [
    [ ],
    [ [ 0, 0, 0.250 ] ],
    [ [ 0.250, 0, 0.250 ] ],
    [ [ 0.470, 0, 0.250 ] ]
  ]
  children_items_do_quadrate_map: [1, 1, 1, 1]
```

```
python -m gmsh_scripts main.yaml
```

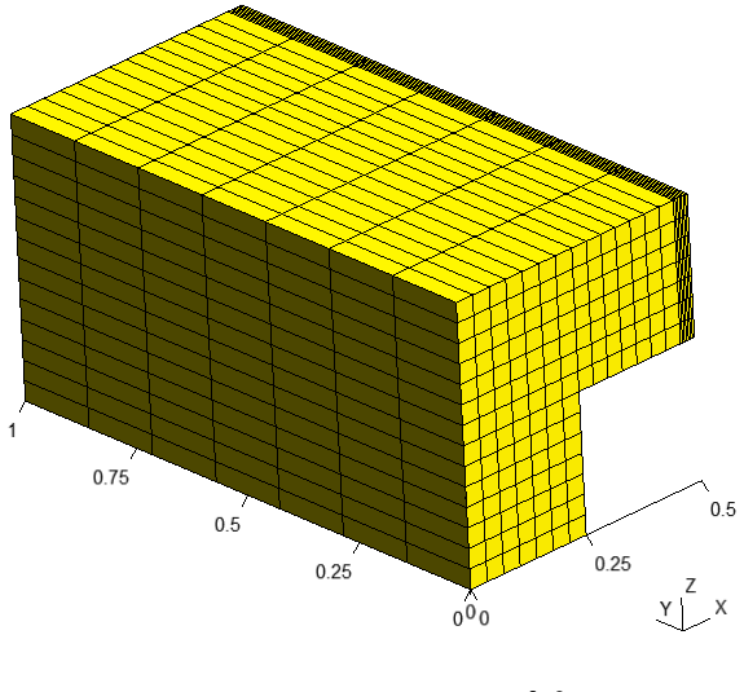

Fig. 45: Structured hexahedral mesh

1.3.4 Zones

If we want to add names to entities of the mesh (e.g. volumes of surfaces) we should set additional field `items_zones` in the `data` field

For example, we can add `[ [ Volume, [ NX, X, NY, Y, NZ, Z ] ] ]` in the``bottom.yaml``, where:

1. `Volume` - volume name
2. `[ NX, X, NY, Y, NZ, Z ]` - surfaces names:
   - `NX` - surface pointing in the opposite direction of X-axis
   - `X` - surface pointing in the direction of X-axis
   - `NY` - surface pointing in the opposite direction of Y-axis
   - `Y` - surface pointing in the direction of Y-axis
   - `NZ` - surface pointing in the opposite direction of Z-axis
   - `Z` - surface pointing in the direction of Z-axis

```
metadata:
   run:
     factory: geo
     strategy:
       class: strategy.NoBoolean

data:
  class: block.Matrix
  matrix: [ [ 0;0.01, 0.250;0.1;4 ], [ 0;0.01, 1;0.1;8 ], [ 0;0.01, 0.250;0.1;16 ] ]
```

(continues on next page)

(continued from previous page)

```
  items_zone: [ [ Volume, [ NX, X, NY, Y, NZ, Z ] ] ]
  items_do_quadrate_map: 1
```

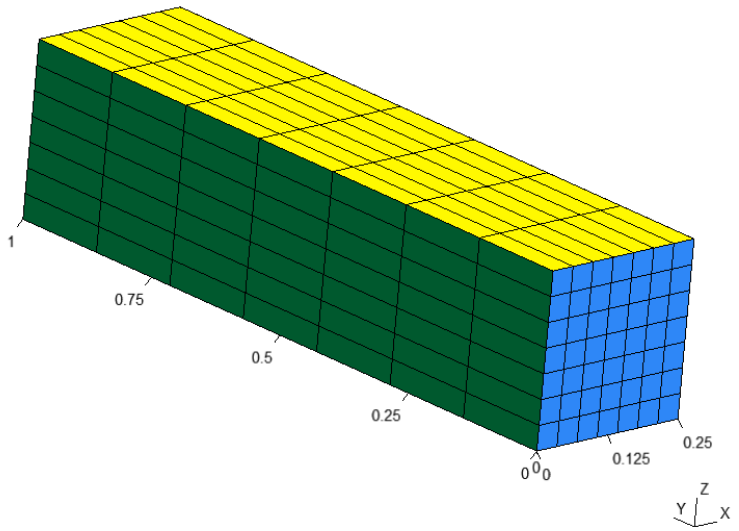

Fig. 46: Structured hexahedral with zones

1.3.5 Result

Note: We should define different zone names of bottom surfaces of `TOP2` and `TOP3`, e.g. `Top2NZ` and `Top3NZ` respectively

- top_1.yaml

```
data:
  class: block.Matrix
  matrix: [ [ 0;0.1, 0.250;0.1;8 ], [ 0;0.1, 1;0.1;8 ], [ 0;0.1, 0.250;0.1;8 ] ]
  items_zone: [ [ Volume, [ NX, X, NY, Y, NZ, Z ] ] ]
```

- top_2.yaml

```
data:
  class: block.Matrix
  matrix: [ [ 0;0.1, 0.220;0.1;8 ], [ 0;0.1, 1;0.1;8 ], [ 0;.1, 0.250;0.1;8 ] ]
  items_zone: [ [ Volume, [ NX, X, NY, Y, Top2NZ, Z ] ] ]
```

- top_3.yaml

```
data:
  class: block.Matrix
  matrix: [ [ 0;0.1, 0.030;0.1;8 ], [ 0;0.1, 1;0.1;8 ], [ 0;.1, 0.250;0.1;8 ] ]
  items_zone: [ [ Volume, [ NX, X, NY, Y, Top3NZ, Z ] ] ]
```

- bottom.yaml

```yaml
data:
  class: block.Matrix
  matrix: [ [ 0;.1, 0.250;.1;8 ], [ 0;.1, 1;.1;8 ], [ 0;.1, 0.250;.1;8 ] ]
  items_zone: [ [ Volume, [ NX, X, NY, Y, NZ, Z ] ] ]
```

- main.yaml

```yaml
metadata:
  run:
    factory: geo
    strategy:
      class: strategy.NoBoolean
data:
  class: block.Block
  do_register: 0
  children: [
    /bottom.yaml,
    /top_1.yaml,
    /top_2.yaml,
    /top_3.yaml
  ]
  children_transforms: [
    [ ],
    [ [ 0, 0, 0.250 ] ],
    [ [ 0.250, 0, 0.250 ] ],
    [ [ 0.470, 0, 0.250 ] ]
  ]
  children_items_do_quadrate_map: [ 1, 1, 1, 1 ]
  children_items_do_structure_map: [ 1, 1, 1, 1 ]
```

```
python -m gmsh_scripts main.yaml
```

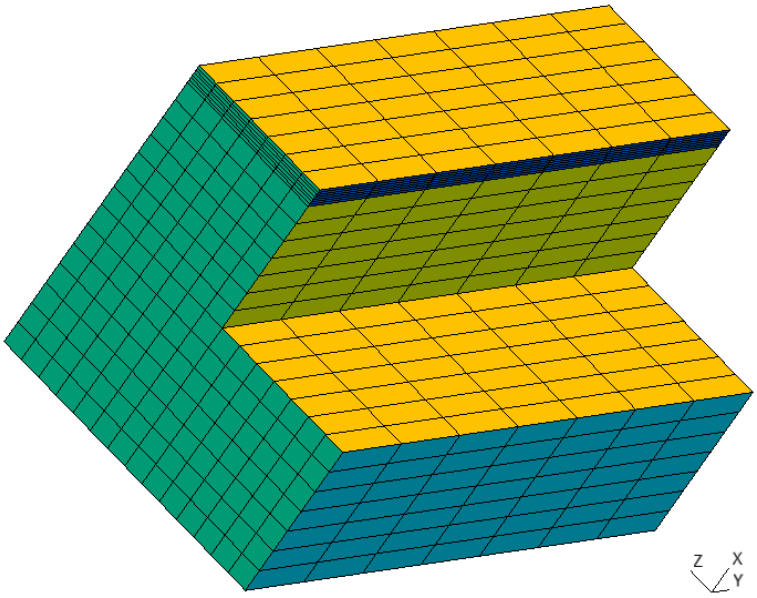

Fig. 47: Mesh

CHAPTER

TWO

API REFERENCE

This page contains auto-generated API reference documentation[1].

2.1 gmsh_scripts

2.1.1 Subpackages

2.1.1.1 gmsh_scripts.block

Submodules

gmsh_scripts.block.block

Module Contents

Classes

Block	Basic building block of the mesh

Attributes

str2obj

class gmsh_scripts.block.block.**Block**(*points=None*, *curves=None*, *surfaces=None*, *volume=None*, *do_register=True*, *do_register_children=True*, *do_unregister=False*, *do_unregister_children=True*, *do_unregister_boolean=False*, *transforms=None*, *self_transforms=None*, *do_quadrate=False*, *do_structure=True*, *structure=None*, *structure_type=None*, *zone=None*, *boolean_level=None*, *path=None*, *parent=None*, *children=None*, *children_transforms=None*)

Basic building block of the mesh

Block is a cuboid with 8 points, 12 curves, 6 surfaces and 1 volume.

[1] Created with sphinx-autoapi

Axes
Y
Z X
NX, NY and NZ are negative X, Y and Z directions

Points
NZ:
P1 P0
P2 P3
Z:
P5 P4
P6 P7

Curves
X direction curves from P0 by right-hand rule:
C0: P1 -> P0
C1: P5 -> P4
C2: P6 -> P7
C3: P2 -> P3
Y direction curves from P0 by right-hand rule:
C4: P3 -> P0
C5: P2 -> P1
C6: P6 -> P5
C7: P7 -> P4
Z direction curves from P0 by right-hand rule:
C8: P0 -> P4
C9: P1 -> P5
C10: P2 -> P6
C11: P3 -> P7

Surfaces
NX surface
S0: C5 -> C9 -> -C6 -> -C10
X surface
S1: -C4 -> C11 -> C7 -> -C8
NY surface
S2: -C3 -> C10 -> C2 -> -C11
Y surface
S3: C0 -> C8 -> -C1 -> -C9
NZ surface
S4: -C0 -> -C5 -> C3 -> C4
Z surface
S5: C1 -> -C7 -> -C2 -> C6

Parameters

- **points** (*list of dict, list of list, list*) – 8 corner points of the block
- **curves** (*list of dict, list of list, list, list of* Curve) – 12 edge curves of the block
- **surfaces** (*list of dict, list of list, list, list of* Surface) – 6 boundary surfaces of the block
- **volume** (*list of dict, list of list, list, list of* Volume) – volumes of the block (1 by now, TODO several volumes)
- **do_register** (*bool or int*) – register Block in the registry
- **do_unregister** (*bool*) – unregister Block from the registry
- **do_register_children** (*bool*) – invoke register for children
- **do_unregister_children** (*bool*) – invoke unregister for children 0 - not unregister 1 - unregister in any case if you are owner 2 - decide to unregister by all owners 3 - decide to unregister by all members
- **transforms** (*list of dict, list of list, list of* Transform) – points and curves points transforms (Translation, Rotation, Coordinate Change, etc)
- **do_quadrate** (*list of dict, bool*) – transform triangles to quadrangles for surfaces and tetrahedra to hexahedra for volumes
- **structure** (*list of dict, list of list, list of* Transform) – make structured mesh instead of unstructured by some rule
- **parent** (Block) – parent of the Block
- **children** (*list of* Block) – children of the Block
- **children_transforms** (*list of list of dict, list of list of list, list of list of* Transform) – transforms for children Blocks
- **boolean_level** (*int*) – Block boolean level, if the Block level > another Block level, then intersected volume joins to the Block, if levels are equal third Block is created, if None - don't do boolean

curves_points = [[1, 0], [5, 4], [6, 7], [2, 3], [3, 0], [2, 1], [6, 5], [7, 4], [0, 4], [1, 5], [2, 6], [3, 7]]

surfaces_curves = [[5, 9, 6, 10], [4, 11, 7, 8], [10, 2, 11, 3], [0, 8, 1, 9], [0, 5, 3, 4], [7, 2, 6, 1]]

surfaces_curves_signs = [None, None, None, None, None, None]

static parse_points(*points*)

static parse_curves(*curves*)

static parse_surfaces(*surfaces*)

static parse_volumes(*volumes*)

static parse_transforms(*transforms*, *parent*)

static parse_structure(*structure*, *do_structure*)

static parse_do_quadrate(*do_quadrate*)

static parse_zone(*zone*)

static parse_structure_type(*structure_type*)

Parse structure type

https://gitlab.onelab.info/gmsh/gmsh/-/blob/master/Mesh/meshGRegionTransfinite.cpp

Transfinite surface meshes

s4 +——c3——+ s3

|

|

c4 c2

|

|

s1 +——c1——+ s2

f(u,v) = (1-u) c4(v) + u c2(v) + (1-v) c1(u) + v c3(u) - [(1-u)(1-v) s1 + u(1-v) s2 + uv s3 + (1-u)v s4]

Transfinite volume meshes

a0 s0 s1 f0 s0 s1 s5 s4 s6

s7 s6 a1 s1 s2 f1 s1 s2 s6 s5 *

——- a2 s3 s2 f2 s3 s2 s6 s7 /| **|\s4 |\ a3 s0 s3 f3 s0 s3 s7 s4 / | | *-------* s5 a4 s4**

s5 f4 s0 s1 s2 s3 s7/s4/ |s2 | | s2| | a5 s5 s6 f5 s4 s5 s6 s7 ——* s5

s3 -|—— | a6 s7 s6 | / |

| | a7 s4 s7 | / |

——- a8 s0 s4 |/ |

v w s0 s1 a9 s1 s5 ——-

| a10 s2 s6 v w s3/s0 s1

***–u a11 s3 s7 |**

*–u

TODO How to create other types? (RLL, LRL, LLR, RRR)

Tried to rotate volume_points = [0, 1, 2, 3, 4, 5, 6, 7] # LLL volume_points = [1, 2, 3, 0, 5, 6, 7, 4] # LRR volume_points = [2, 3, 0, 1, 6, 7, 4, 5] # RRL volume_points = [3, 0, 1, 2, 7, 4, 5, 6] # RLR Tried to swap top and bottom volume_points = [4, 5, 6, 7, 0, 1, 2, 3] # RRL volume_points = [5, 6, 7, 4, 1, 2, 3, 0] # RLR volume_points = [6, 7, 4, 5, 2, 3, 0, 1] # LLL volume_points = [7, 4, 5, 6, 3, 0, 1, 2] # LRR Tried to reverse volume_points = [3, 2, 1, 0, 7, 6, 5, 4] # RLR volume_points = [0, 3, 2, 1, 4, 7, 6, 5] # LLL volume_points = [1, 0, 3, 2, 5, 4, 7, 6] # LRR volume_points = [2, 1, 0, 3, 6, 5, 4, 7] # RRL Tried to swap top and bottom with reverse after volume_points = [7, 6, 5, 4, 3, 2, 1, 0] # LRR volume_points = [4, 7, 6, 5, 0, 3, 2, 1] # RRL volume_points = [5, 4, 7, 6, 1, 0, 3, 2] # RLR volume_points = [6, 5, 4, 7, 2, 1, 0, 3] # LLL

Parameters

structure_type (*str*) – LLL, LRR, LRR or RRL, L/R - Left/Right triangles arrangement

of X (NX), Y (NY), Z (NZ) surfaces respectively, e.g. LRR - Left arrangement for X and NX surfaces, Right for Y and NY, Right for Z and NZ

Returns

tuple of:

surfaces_arrangement (list of str): surfaces arrangement:
Left or Right (AlternateLeft and AlternateRight are incompatible with structured meshes)

surfaces_points (list of list of int): surfaces points tags
(s1, s2, s3, s4)

volume_points (list of int): volume points tags
(s0, s1, s2, s3, s4, s5, s6, s7)

Return type
tuple

register()

add_child(*child*, *transforms=None*)

transform()

register_points()

register_curve_points()

register_curves()

register_curves_loops()

register_surfaces()

register_surfaces_loops()

register_volumes()

register_structure()

register_quadrate()

pre_unregister()

unregister()

make_tree()
Tree of blocks

Returns
children of blocks

Return type
dict

gmsh_scripts.block.block.**str2obj**

gmsh_scripts.block.layer

Module Contents

Classes

Layer	Matrix

Attributes

str2obj

class gmsh_scripts.block.layer.**Layer**(*layer=None*, *layer_curves=None*, *layer_types=None*, *items_do_register_map=None*, *items_do_register_children_map=None*, *items_do_register_mask=None*, *items_do_unregister_map=None*, *items_do_unregister_children_map=None*, *items_do_unregister_boolean_map=None*, *items_do_quadrate_map=None*, *items_do_structure_map=None*, *items_boolean_level_map=None*, *items_zone=None*, *items_zone_map=None*, *items_transforms=None*, *items_transforms_map=None*, *items_self_transforms=None*, *items_self_transforms_map=None*, *items_children=None*, *items_children_map=None*, *items_children_transforms=None*, *items_children_transforms_map=None*, *points=None*, *curves=None*, *surfaces=None*, *volume=None*, *do_register=False*, *do_unregister=False*, *do_register_children=True*, *do_unregister_children=True*, *do_unregister_boolean=False*, *transforms=None*, *self_transforms=None*, *do_quadrate=False*, *do_structure=True*, *structure=None*, *structure_type='LLL'*, *zone=None*, *parent=None*, *children=None*, *children_transforms=None*, *boolean_level=None*, *path=None*)

Bases: *gmsh_scripts.block.matrix.Matrix*

Matrix

Args:

static parse_layers_map(*old_layers*, *n2o_l2l_l2l*, *default=None*)

static parse_layers_block_map(*m*, *default*, *new2old*, *item_types=()*)

static parse_layers_block_mask(*m*, *default*, *new2old*, *item_types=()*)

static get_layers_curves(*parsed_layers_curves*, *parsed_g2l_b2b_l2l*, *parsed_layers_coordinates*)

Parse layers curves

Parameters

- **parsed_layers_curves** (*list of list*) – curves of layers

- **parsed_g2l_b2b_l2l** (*dict*) – local index of a block of the matrix to local index of a block of the parsed layer
- **parsed_layers_coordinates** (*list of list*) – layers

Returns
curves (list) curves map of the matrix (list)

Return type
tuple

static get_structure_type(*parsed_g2l_b2b_l2l*)

gmsh_scripts.block.layer.**str2obj**

gmsh_scripts.block.matrix

Module Contents

Classes

Matrix	Matrix

Attributes

str2obj

class gmsh_scripts.block.matrix.**Matrix**(*matrix=None*, *items_curves=None*, *items_curves_map=None*, *items_do_register_map=None*, *items_do_register_children_map=None*, *items_do_unregister_map=None*, *items_do_unregister_children_map=None*, *items_do_unregister_boolean_map=None*, *items_transforms=None*, *items_transforms_map=None*, *items_self_transforms=None*, *items_self_transforms_map=None*, *items_do_quadrate_map=None*, *items_do_structure_map=None*, *items_structure_type=None*, *items_structure_type_map=None*, *items_zone=None*, *items_zone_map=None*, *items_boolean_level_map=None*, *items_children=None*, *items_children_map=None*, *items_children_transforms=None*, *items_children_transforms_map=None*, *points=None*, *curves=None*, *surfaces=None*, *volume=None*, *do_register=False*, *do_register_children=True*, *do_unregister=False*, *do_unregister_children=True*, *do_unregister_boolean=False*, *transforms=None*, *self_transforms=None*, *do_quadrate=False*, *do_structure=True*, *structure=None*, *structure_type='LLL'*, *zone=None*, *boolean_level=None*, *path=None*, *parent=None*, *children=None*, *children_transforms=None*)

Bases: *gmsh_scripts.block.block.Block*

Matrix

Args:

static evaluate_items_values(*values*, *b2ids*, *gm=0.0*, *gs=None*, *gcs='Cartesian'*)

static parse_matrix_items_map(*m*, *default*, *new2old*, *item_types=(bool, str, int, float)*)

gmsh_scripts.block.matrix.**str2obj**

gmsh_scripts.block.polyhedron

Module Contents

Classes

Polyhedron

Attributes

str2obj

class gmsh_scripts.block.polyhedron.**Polyhedron**(*points=None*, *polygons=None*, *do_register=True*, *do_register_children=True*, *do_unregister=False*, *do_unregister_children=True*, *do_unregister_boolean=False*, *transforms=None*, *self_transforms=None*, *zone=None*, *boolean_level=None*, *path=None*, *parent=None*, *children=None*, *children_transforms=None*)

Bases: *gmsh_scripts.block.Block*

static parse_points(*points*)

static parse_polygons(*polygons*)

static parse_curves(*curves*)

static parse_surfaces(*surfaces*)

static parse_volumes(*volumes*)

static parse_structure(*structure*, *do_structure*)

static parse_do_quadrate(*do_quadrate*)

register_curves_loops()

register_surfaces()

gmsh_scripts.block.polyhedron.**str2obj**

Package Contents

Classes

Block	Basic building block of the mesh
Layer	Matrix
Matrix	Matrix

class gmsh_scripts.block.**Block**(*points=None*, *curves=None*, *surfaces=None*, *volume=None*, *do_register=True*, *do_register_children=True*, *do_unregister=False*, *do_unregister_children=True*, *do_unregister_boolean=False*, *transforms=None*, *self_transforms=None*, *do_quadrate=False*, *do_structure=True*, *structure=None*, *structure_type=None*, *zone=None*, *boolean_level=None*, *path=None*, *parent=None*, *children=None*, *children_transforms=None*)

Basic building block of the mesh

Block is a cuboid with 8 points, 12 curves, 6 surfaces and 1 volume.

Axes
Y
Z X
NX, NY and NZ are negative X, Y and Z directions

Points
NZ:
P1 P0
P2 P3
Z:
P5 P4
P6 P7

Curves
X direction curves from P0 by right-hand rule:
C0: P1 -> P0
C1: P5 -> P4
C2: P6 -> P7
C3: P2 -> P3
Y direction curves from P0 by right-hand rule:
C4: P3 -> P0
C5: P2 -> P1
C6: P6 -> P5
C7: P7 -> P4
Z direction curves from P0 by right-hand rule:
C8: P0 -> P4
C9: P1 -> P5

C10: P2 -> P6
C11: P3 -> P7

Surfaces
NX surface
S0: C5 -> C9 -> -C6 -> -C10
X surface
S1: -C4 -> C11 -> C7 -> -C8
NY surface
S2: -C3 -> C10 -> C2 -> -C11
Y surface
S3: C0 -> C8 -> -C1 -> -C9
NZ surface
S4: -C0 -> -C5 -> C3 -> C4
Z surface
S5: C1 -> -C7 -> -C2 -> C6

Parameters

- **points** (*list of dict, list of list, list*) – 8 corner points of the block
- **curves** (*list of dict, list of list, list, list of* Curve) – 12 edge curves of the block
- **surfaces** (*list of dict, list of list, list, list of* Surface) – 6 boundary surfaces of the block
- **volume** (*list of dict, list of list, list, list of* Volume) – volumes of the block (1 by now, TODO several volumes)
- **do_register** (*bool or int*) – register Block in the registry
- **do_unregister** (*bool*) – unregister Block from the registry
- **do_register_children** (*bool*) – invoke register for children
- **do_unregister_children** (*bool*) – invoke unregister for children 0 - not unregister 1 - unregister in any case if you are owner 2 - decide to unregister by all owners 3 - decide to unregister by all members
- **transforms** (*list of dict, list of list, list of* Transform) – points and curves points transforms (Translation, Rotation, Coordinate Change, etc)
- **do_quadrate** (*list of dict, bool*) – transform triangles to quadrangles for surfaces and tetrahedra to hexahedra for volumes
- **structure** (*list of dict, list of list, list of* Transform) – make structured mesh instead of unstructured by some rule
- **parent** (Block) – parent of the Block
- **children** (*list of* Block) – children of the Block
- **children_transforms** (*list of list of dict, list of list of list, list of list of* Transform) – transforms for children Blocks

- **boolean_level** (*int*) – Block boolean level, if the Block level > another Block level, then intersected volume joins to the Block, if levels are equal third Block is created, if None - don't do boolean

curves_points = [[1, 0], [5, 4], [6, 7], [2, 3], [3, 0], [2, 1], [6, 5], [7, 4], [0, 4], [1, 5], [2, 6], [3, 7]]

surfaces_curves = [[5, 9, 6, 10], [4, 11, 7, 8], [10, 2, 11, 3], [0, 8, 1, 9], [0, 5, 3, 4], [7, 2, 6, 1]]

surfaces_curves_signs = [None, None, None, None, None, None]

static parse_points(*points*)

static parse_curves(*curves*)

static parse_surfaces(*surfaces*)

static parse_volumes(*volumes*)

static parse_transforms(*transforms*, *parent*)

static parse_structure(*structure*, *do_structure*)

static parse_do_quadrate(*do_quadrate*)

static parse_zone(*zone*)

static parse_structure_type(*structure_type*)

Parse structure type

https://gitlab.onelab.info/gmsh/gmsh/-/blob/master/Mesh/meshGRegionTransfinite.cpp

Transfinite surface meshes

s4 +——c3——+ s3

|

|

c4 c2

|

|

s1 +——c1——+ s2

f(u,v) = (1-u) c4(v) + u c2(v) + (1-v) c1(u) + v c3(u) - [(1-u)(1-v) s1 + u(1-v) s2 + uv s3 + (1-u)v s4]

Transfinite volume meshes

a0 s0 s1 f0 s0 s1 s5 s4 s6

s7 s6 a1 s1 s2 f1 s1 s2 s6 s5 *
——- a2 s3 s2 f2 s3 s2 s6 s7 /| **|\s4 |\ a3 s0 s3 f3 s0 s3 s7 s4 / | | *-------* s5 a4 s4 s5 f4 s0 s1 s2 s3 s7/s4/ |s2 | | s2|** | a5 s5 s6 f5 s4 s5 s6 s7 ——* s5

s3 -|—— | a6 s7 s6 | / |

| | a7 s4 s7 | / |
——- a8 s0 s4 |/ |

v w s0 s1 a9 s1 s5 ——-

| a10 s2 s6 v w s3/s0 s1

***–u a11 s3 s7 |**
*–u

TODO How to create other types? (RLL, LRL, LLR, RRR)
Tried to rotate volume_points = [0, 1, 2, 3, 4, 5, 6, 7] # LLL volume_points = [1, 2, 3, 0, 5, 6, 7, 4] # LRR volume_points = [2, 3, 0, 1, 6, 7, 4, 5] # RRL volume_points = [3, 0, 1, 2, 7, 4, 5, 6] # RLR Tried to swap top and bottom volume_points = [4, 5, 6, 7, 0, 1, 2, 3] # RRL volume_points = [5, 6, 7, 4, 1, 2, 3, 0] # RLR volume_points = [6, 7, 4, 5, 2, 3, 0, 1] # LLL volume_points = [7, 4, 5, 6, 3, 0, 1, 2] # LRR Tried to reverse volume_points = [3, 2, 1, 0, 7, 6, 5, 4] # RLR volume_points = [0, 3, 2, 1, 4, 7, 6, 5] # LLL volume_points = [1, 0, 3, 2, 5, 4, 7, 6] # LRR volume_points = [2, 1, 0, 3, 6, 5, 4, 7] # RRL Tried to swap top and bottom with reverse after volume_points = [7, 6, 5, 4, 3, 2, 1, 0] # LRR volume_points = [4, 7, 6, 5, 0, 3, 2, 1] # RRL volume_points = [5, 4, 7, 6, 1, 0, 3, 2] # RLR volume_points = [6, 5, 4, 7, 2, 1, 0, 3] # LLL

Parameters
structure_type (*str*) – LLL, LRR, LRR or RRL, L/R - Left/Right triangles arrangement of X (NX), Y (NY), Z (NZ) surfaces respectively, e.g. LRR - Left arrangement for X and NX surfaces, Right for Y and NY, Right for Z and NZ

Returns
tuple of:
surfaces_arrangement (list of str): surfaces arrangement:
Left or Right (AlternateLeft and AlternateRight are incompatible with structured meshes)
surfaces_points (list of list of int): surfaces points tags
(s1, s2, s3, s4)
volume_points (list of int): volume points tags
(s0, s1, s2, s3, s4, s5, s6, s7)

Return type
tuple

register()

add_child(*child*, *transforms=None*)

transform()

register_points()

register_curve_points()

register_curves()

register_curves_loops()

register_surfaces()

register_surfaces_loops()

register_volumes()

register_structure()

register_quadrate()

pre_unregister()

unregister()

make_tree()

Tree of blocks

Returns
children of blocks

Return type
dict

class gmsh_scripts.block.**Layer**(*layer=None*, *layer_curves=None*, *layer_types=None*, *items_do_register_map=None*, *items_do_register_children_map=None*, *items_do_register_mask=None*, *items_do_unregister_map=None*, *items_do_unregister_children_map=None*, *items_do_unregister_boolean_map=None*, *items_do_quadrate_map=None*, *items_do_structure_map=None*, *items_boolean_level_map=None*, *items_zone=None*, *items_zone_map=None*, *items_transforms=None*, *items_transforms_map=None*, *items_self_transforms=None*, *items_self_transforms_map=None*, *items_children=None*, *items_children_map=None*, *items_children_transforms=None*, *items_children_transforms_map=None*, *points=None*, *curves=None*, *surfaces=None*, *volume=None*, *do_register=False*, *do_unregister=False*, *do_register_children=True*, *do_unregister_children=True*, *do_unregister_boolean=False*, *transforms=None*, *self_transforms=None*, *do_quadrate=False*, *do_structure=True*, *structure=None*, *structure_type='LLL'*, *zone=None*, *parent=None*, *children=None*, *children_transforms=None*, *boolean_level=None*, *path=None*)

Bases: *gmsh_scripts.block.matrix.Matrix*

Matrix

Args:

static parse_layers_map(*old_layers*, *n2o_l2l_l2l*, *default=None*)

static parse_layers_block_map(*m*, *default*, *new2old*, *item_types=()*)

static parse_layers_block_mask(*m*, *default*, *new2old*, *item_types=()*)

static get_layers_curves(*parsed_layers_curves*, *parsed_g2l_b2b_l2l*, *parsed_layers_coordinates*)

Parse layers curves

Parameters

- **parsed_layers_curves** (*list of list*) – curves of layers
- **parsed_g2l_b2b_l2l** (*dict*) – local index of a block of the matrix to local index of a block of the parsed layer
- **parsed_layers_coordinates** (*list of list*) – layers

Returns
curves (list) curves map of the matrix (list)

Return type
tuple

static get_structure_type(*parsed_g2l_b2b_l2l*)

class gmsh_scripts.block.**Matrix**(*matrix=None*, *items_curves=None*, *items_curves_map=None*, *items_do_register_map=None*, *items_do_register_children_map=None*, *items_do_unregister_map=None*, *items_do_unregister_children_map=None*, *items_do_unregister_boolean_map=None*, *items_transforms=None*, *items_transforms_map=None*, *items_self_transforms=None*, *items_self_transforms_map=None*, *items_do_quadrate_map=None*, *items_do_structure_map=None*, *items_structure_type=None*, *items_structure_type_map=None*, *items_zone=None*, *items_zone_map=None*, *items_boolean_level_map=None*, *items_children=None*, *items_children_map=None*, *items_children_transforms=None*, *items_children_transforms_map=None*, *points=None*, *curves=None*, *surfaces=None*, *volume=None*, *do_register=False*, *do_register_children=True*, *do_unregister=False*, *do_unregister_children=True*, *do_unregister_boolean=False*, *transforms=None*, *self_transforms=None*, *do_quadrate=False*, *do_structure=True*, *structure=None*, *structure_type='LLL'*, *zone=None*, *boolean_level=None*, *path=None*, *parent=None*, *children=None*, *children_transforms=None*)

Bases: *gmsh_scripts.block.block.Block*

Matrix

Args:

static evaluate_items_values(*values*, *b2ids*, *gm=0.0*, *gs=None*, *gcs='Cartesian'*)

static parse_matrix_items_map(*m*, *default*, *new2old*, *item_types=(bool, str, int, float)*)

2.1.1.2 gmsh_scripts.boolean

Submodules

gmsh_scripts.boolean.boolean

Module Contents

Classes

Boolean

NoBoolean

BooleanAllBlock

Attributes

str2obj

class gmsh_scripts.boolean.boolean.**Boolean**

class gmsh_scripts.boolean.boolean.**NoBoolean**

Bases: *Boolean*

class gmsh_scripts.boolean.boolean.**BooleanAllBlock**

Bases: *Boolean*

gmsh_scripts.boolean.boolean.**str2obj**

2.1.1.3 gmsh_scripts.coordinate_system

Submodules

gmsh_scripts.coordinate_system.coordinate_system

Module Contents

Classes

CoordinateSystem	Abstract
Affine	Affine coordinate system
Cartesian	Affine coordinate system
Cylindrical	Cylindrical coordinate system
Spherical	Spherical coordinate system
Toroidal	Abstract
Tokamak	Abstract
Hexahedral	Natural Hexahedral Coordinate System
Block	Block Coordinate System
Path	Path coordinate system
Layer	Different layers by X, Y and Z axis
QuarterLayer	Quarter of Layer by +X and +Y directions

Attributes

str2obj

class gmsh_scripts.coordinate_system.coordinate_system.**CoordinateSystem**(*dim=None*, *origin=None*, ***kwargs*)

Abstract

Parameters

- **dim** (*int*) – Dimension
- **origin** (*np.ndarray or list*) – Origin

class gmsh_scripts.coordinate_system.coordinate_system.**Affine**(*origin=np.zeros(3)*, *vs=np.eye(3, 3)*, ***kwargs*)

Bases: *CoordinateSystem*

Affine coordinate system

Parameters

- **origin** (*np.ndarray or list*) – Origin
- **vs** (*np.ndarray or list of list*) – Basis vectors

class gmsh_scripts.coordinate_system.coordinate_system.**Cartesian**(***kwargs*)

Bases: *Affine*

Affine coordinate system

Parameters

- **origin** (*np.ndarray or list*) – Origin
- **vs** (*np.ndarray or list of list*) – Basis vectors

class gmsh_scripts.coordinate_system.coordinate_system.**Cylindrical**(*origin=np.zeros(3)*, ***kwargs*)

Bases: *CoordinateSystem*

Cylindrical coordinate system

- r - radius [0, inf)
- phi - azimuthal angle or longitude [0, 2pi) (counterclockwise from X to Y)
- z - height

class gmsh_scripts.coordinate_system.coordinate_system.**Spherical**(*origin=np.zeros(3)*, ***kwargs*)

Bases: *CoordinateSystem*

Spherical coordinate system

- r - radius [0, inf)
- phi - azimuthal angle [0, 2pi) (counterclockwise from X to Y)
- **theta - polar (zenith) angle or colatitude = pi/2 - latitude [0, pi]**
  (from Z to -Z, i.e XY-plane is pi/2)

class gmsh_scripts.coordinate_system.coordinate_system.**Toroidal**(*origin=np.zeros(4)*, ***kwargs*)

Bases: *CoordinateSystem*

Abstract

Parameters

- **dim** (*int*) – Dimension
- **origin** (*np.ndarray or list*) – Origin

class gmsh_scripts.coordinate_system.coordinate_system.**Tokamak**(*origin=np.zeros(6)*, ***kwargs*)

Bases: *CoordinateSystem*

Abstract

Parameters

- **dim** (*int*) – Dimension
- **origin** (*np.ndarray or list*) – Origin

class gmsh_scripts.coordinate_system.coordinate_system.**Hexahedral**(*origin=np.zeros(3)*, *ps=None*, *order=None*, ***kwargs*)

Bases: *CoordinateSystem*

Natural Hexahedral Coordinate System

- xi [-1, 1]
- eta [-1, 1]
- zeta [-1, 1]

Parameters

- **ps** (*np.ndarray or list of list*) – points coordinates of hexahedron
- **order** (*np.ndarray or list of list*) – order of points
- **origin** (*np.ndarray or list*) – Origin

class gmsh_scripts.coordinate_system.coordinate_system.**Block**(*origin=np.zeros(3)*, *ps=None*, *order=None*, ***kwargs*)

Bases: *CoordinateSystem*

Block Coordinate System

- xi [-1, 1]
- eta [-1, 1]
- zeta [-1, 1]

Parameters

- **ps** (*np.ndarray or list of list*) – points coordinates of the block
- **order** (*np.ndarray or list of list*) – order of points
- **origin** (*np.ndarray or list*) – Origin

class gmsh_scripts.coordinate_system.coordinate_system.**Path**(*origin=np.zeros(3)*, *curves=None*, *orientations=None*, *transforms=None*, *weights=None*, *local_weights=None*, *do_normalize=False*, *normalize_kind=1*, *normalize_local_kind=1*, ***kwargs*)

Bases: *CoordinateSystem*

Path coordinate system

- xi - X, transverse, right axis, axis of pitch rotation (-inf, inf)
- eta - Y, vertical, down axis, axis of yaw rotation (-inf, inf)

- zeta - Z, longitudinal, front axis, axis of roll rotation [0, 1]

Parameters

- **origin** (*np.ndarray or list*) – Origin
- **curves** (*list of Curve or list of Curve*) – Curves
- **orientations** (*list*) – Orientation in curves points (number of curves + 1)
- **transforms** (*list of list*) – Curves points transforms
- **weights** (*list of float*) – Curves weights in global path coordinate system
- **local_weights** (*list of list*) – Curves weights in local curve coordinate system

parse_orientations(*orientations*, *do_deg2rad*)

register()

transform()

evaluate_bounds()

get_value_derivative_orientation(*u*)

get_local_coordinate_system(*u*)

class gmsh_scripts.coordinate_system.coordinate_system.**Layer**(*origin=np.zeros(3)*, *layers=None*, *layers_curves=None*, *layers_types=None*, ***kwargs*)

Bases: *CoordinateSystem*

Different layers by X, Y and Z axis

Parameters

- **layers** (*list*) – [[c_x_1, c_x_2, …, c_x_NX], [c_y_1, c_y_2, …, c_y_NY], [c_nx_1, c_nx_2, …, c_nx_NNX], [c_ny_1, c_ny_2, …, c_ny_NNY]], [c_z_1, c_z_2, …, c_z_NZ]], [c_nz_1, c_nz_2, …, c_nz_NNZ]], where N - number of layers, c (float): coordinate (0, inf).
- **layers_curves** (*list*) – [[name_x_1, name_x_2, …, name_x_NX], [name_y_1, name_y_2, …, name_y_NY], [name_nx_1, name_nx_2, …, name_nx_NNX], [name_ny_1, name_ny_2, …, name_ny_NNY]], [name_z_1, name_z_2, …, name_z_NZ]], [name_nz_1, name_nz_2, …, name_nz_NNZ]], where N - number of layers, name - curve name (see py:class:*curve.Curve* class)
- **layers_types** (*list*) –

  [[type_x_1, type_x_2, …, type_x_NX],
  [type_y_1, type_y_2, …, type_y_NY], [type_z_1, type_z_2, …, type_z_NZ]], [type_nz_1, type_nz_2, …, type_nz_NNZ]],

  where type (str): ‘in’ - inscribed, ‘out’ - circumscribed.

class gmsh_scripts.coordinate_system.coordinate_system.**QuarterLayer**(*origin=np.zeros(3)*, *layers=None*, *layers_curves=None*, *layers_types=None*, ***kwargs*)

Bases: *CoordinateSystem*

Quarter of Layer by +X and +Y directions

Parameters

- **layers** (*list*) –

  [[c_x_1, c_x_2, …, c_x_NX],
  [c_y_1, c_y_2, …, c_y_NY], [c_z_1, c_z_2, …, c_z_NZ]], [c_nz_1, c_nz_2, …, c_nz_NNZ]],

  where N - number of layers, c (float): coordinate (0, inf).

- **layers_curves** (*list*) –

  [[name_x_1, name_x_2, …, name_x_NX],
  [name_y_1, name_y_2, …, name_y_NY], [name_z_1, name_z_2, …, name_z_NZ]], [name_nz_1, name_nz_2, …, name_nz_NNZ]],

  where N - number of layers, name - curve name (see py:class:*curve.Curve* class)

- **layers_types** (*list*) –

  [[type_x_1, type_x_2, …, type_x_NX],
  [type_y_1, type_y_2, …, type_y_NY], [type_z_1, type_z_2, …, type_z_NZ]], [type_nz_1, type_nz_2, …, type_nz_NNZ]],

  where type (str): ‘in’ - inscribed, ‘out’ - circumscribed.

gmsh_scripts.coordinate_system.coordinate_system.**str2obj**

2.1.1.4 gmsh_scripts.entity

Submodules

gmsh_scripts.entity.curve

Module Contents

Classes

Curve	Curve

Attributes

str2obj

class gmsh_scripts.entity.curve.**Curve**(*tag=None*, *name='line'*, *zone=None*, *points=None*, *structure=None*, ***kwargs*)

Curve

Parameters

- **tag** (*int*) – unique id
- **name** (*str*) – type of the curve: line, polyline, circle_arc, ellipse_arc, spline, bspline, bezier

- **zone** (*str*) – zone
- **points** (*list of* Point) – curve points
- **structure** (Structure) – curve structure
- **kwargs** (*dict*) – other keyword arguments

gmsh_scripts.entity.curve.**str2obj**

gmsh_scripts.entity.curve_loop

Module Contents

Classes

CurveLoop	

Attributes

str2obj

class gmsh_scripts.entity.curve_loop.**CurveLoop**(*tag=None*, *name=None*, *zone=None*, *curves=None*, *curves_signs=None*, ***kwargs*)

gmsh_scripts.entity.curve_loop.**str2obj**

gmsh_scripts.entity.point

Module Contents

Classes

Point	Point

Attributes

str2obj

class gmsh_scripts.entity.point.**Point**(**args*, *tag=None*, *zone=None*, *coordinate_system=None*, *coordinates=None*, ***kwargs*)

Point :param tag: unique id :type tag: int or None :param zone: zone :type zone: str or None :param coordinate_system: coordinate system :type coordinate_system: str or dict or CoordinateSystem or None :param coordinates: coordinates values :type coordinates: list of float or np.ndarray or None

tag

unique id

Type
int or None

zone

zone

Type
str or None

coordinate_system

coordinate system

Type
CoordinateSystem

coordinates

coordinates values

Type
np.ndarray

kwargs

other keyword arguments (e.g. meshSize)

Type
dict or None

static parse_coordinate_system(*coordinate_system*, *default=Cartesian()*, *name_key='name'*)

static parse_coordinates(*coordinates*, *coordinate_system*)

static parse_args(*args*)

Parse list Patterns: 1. [coordinates] 2. [coordinates, meshSize] 3. [coordinates, coordinate_system] 4. [coordinates, zone] 5. [coordinates, meshSize, coordinate_system] 5. [coordinates, coordinate_system, zone] 5. [coordinates, meshSize, zone] 5. [coordinates, meshSize, coordinate_system, zone]

Parameters
args (*list*) –

Return type
int or None, str or None, *CoordinateSystem* or None, list of float, dict

static parse_points(*points=None*, *do_deg2rad=False*)

Parse list of raw points Patterns 1. [[], [], [], ...] 2. [[], [], [], ..., meshSize] 3. [[], [], [], ..., coordinate_system] 4. [[], [], [], ..., zone] 5. [[], [], [], ..., meshSize, coordinate_system] 6. [[], [], [], ..., coordinate_system, zone] 7. [[], [], [], ..., meshSize, zone] 8. [[], [], [], ..., meshSize, coordinate_system, zone] 9. [{}, {}, {}, ...] 10. [{}, {}, {}, ..., meshSize] 11. [{}, {}, {}, ..., coordinate_system] 12. [{}, {}, {}, ..., zone] 13. [{}, {}, {}, ..., meshSize, coordinate_system] 14. [{}, {}, {}, ..., coordinate_system, zone] 15. [{}, {}, {}, ..., meshSize, zone] 16. [{}, {}, {}, ..., meshSize, coordinate_system, zone]

Parameters

- **points** (*list or None*) – raw points
- **do_deg2rad** (*bool*) – do degrees to radians conversion

Returns
points objects

Return type
list of *Point*

gmsh_scripts.entity.point.**str2obj**

gmsh_scripts.entity.surface

Module Contents

Classes

Surface

Attributes

str2obj

class gmsh_scripts.entity.surface.**Surface**(*tag=None*, *name='line'*, *zone=None*, *curves_loops=None*, *structure=None*, *quadrate=None*, ***kwargs*)

gmsh_scripts.entity.surface.**str2obj**

gmsh_scripts.entity.surface_loop

Module Contents

Classes

SurfaceLoop

Attributes

str2obj

class gmsh_scripts.entity.surface_loop.**SurfaceLoop**(*tag=None*, *name=None*, *zone=None*, *surfaces=None*, ***kwargs*)

gmsh_scripts.entity.surface_loop.**str2obj**

gmsh_scripts.entity.volume

Module Contents

Classes

Volume

Attributes

str2obj

class gmsh_scripts.entity.volume.**Volume**(*tag=None*, *name=None*, *zone=None*, *surfaces_loops=None*, *structure=None*, *quadrate=None*, ***kwargs*)

gmsh_scripts.entity.volume.**str2obj**

Package Contents

Classes

Point	Point
Curve	Curve
CurveLoop	
Surface	
SurfaceLoop	
Volume	

class gmsh_scripts.entity.**Point**(**args*, *tag=None*, *zone=None*, *coordinate_system=None*, *coordinates=None*, ***kwargs*)

Point :param tag: unique id :type tag: int or None :param zone: zone :type zone: str or None :param coordinate_system: coordinate system :type coordinate_system: str or dict or CoordinateSystem or None :param coordinates: coordinates values :type coordinates: list of float or np.ndarray or None

tag

unique id

Type
int or None

zone

zone

Type
str or None

coordinate_system

coordinate system

Type
CoordinateSystem

coordinates

coordinates values

Type
np.ndarray

kwargs

other keyword arguments (e.g. meshSize)

Type
dict or None

static parse_coordinate_system(*coordinate_system*, *default=Cartesian()*, *name_key='name'*)

static parse_coordinates(*coordinates*, *coordinate_system*)

static parse_args(*args*)

Parse list Patterns: 1. [coordinates] 2. [coordinates, meshSize] 3. [coordinates, coordinate_system] 4. [coordinates, zone] 5. [coordinates, meshSize, coordinate_system] 5. [coordinates, coordinate_system, zone] 5. [coordinates, meshSize, zone] 5. [coordinates, meshSize, coordinate_system, zone]

Parameters
args (*list*) –

Return type
int or None, str or None, *CoordinateSystem* or None, list of float, dict

static parse_points(*points=None*, *do_deg2rad=False*)

Parse list of raw points Patterns 1. [[], [], [], ...] 2. [[], [], [], ..., meshSize] 3. [[], [], [], ..., coordinate_system] 4. [[], [], [], ..., zone] 5. [[], [], [], ..., meshSize, coordinate_system] 6. [[], [], [], ..., coordinate_system, zone] 7. [[], [], [], ..., meshSize, zone] 8. [[], [], [], ..., meshSize, coordinate_system, zone] 9. [{}, {}, {}, ...] 10. [{}, {}, {}, ..., meshSize] 11. [{}, {}, {}, ..., coordinate_system] 12. [{}, {}, {}, ..., zone] 13. [{}, {}, {}, ..., meshSize, coordinate_system] 14. [{}, {}, {}, ..., coordinate_system, zone] 15. [{}, {}, {}, ..., meshSize, zone] 16. [{}, {}, {}, ..., meshSize, coordinate_system, zone]

Parameters

- **points** (*list or None*) – raw points
- **do_deg2rad** (*bool*) – do degrees to radians conversion

Returns
points objects

Return type
list of *Point*

class gmsh_scripts.entity.**Curve**(*tag=None*, *name='line'*, *zone=None*, *points=None*, *structure=None*, ***kwargs*)

Curve

Parameters

- **tag** (*int*) – unique id
- **name** (*str*) – type of the curve: line, polyline, circle_arc, ellipse_arc, spline, bspline, bezier
- **zone** (*str*) – zone
- **points** (*list of Point*) – curve points
- **structure** (*Structure*) – curve structure
- **kwargs** (*dict*) – other keyword arguments

class gmsh_scripts.entity.**CurveLoop**(*tag=None*, *name=None*, *zone=None*, *curves=None*, *curves_signs=None*, ***kwargs*)

class gmsh_scripts.entity.**Surface**(*tag=None*, *name='line'*, *zone=None*, *curves_loops=None*, *structure=None*, *quadrate=None*, ***kwargs*)

class gmsh_scripts.entity.**SurfaceLoop**(*tag=None*, *name=None*, *zone=None*, *surfaces=None*, ***kwargs*)

class gmsh_scripts.entity.**Volume**(*tag=None*, *name=None*, *zone=None*, *surfaces_loops=None*, *structure=None*, *quadrate=None*, ***kwargs*)

2.1.1.5 gmsh_scripts.optimize

Submodules

gmsh_scripts.optimize.optimize

Module Contents

Classes

Optimize	Abstract
NoOptimize	Abstract
OptimizeOne	Optimize once
OptimizeMany	Optimize several times

Attributes

str2obj

n

class gmsh_scripts.optimize.optimize.**Optimize**

Abstract

class gmsh_scripts.optimize.optimize.**NoOptimize**

Bases: *Optimize*

Abstract

class gmsh_scripts.optimize.optimize.**OptimizeOne**(*method=None*, *force=False*, *n_iterations=1*, *threshold=0.3*)

Bases: *Optimize*

Optimize once

Parameters

- **method** (*str*) – Optimization method: "" for default tetrahedral mesh optimizer "Netgen" for Netgen optimizer, "HighOrder" for direct high-order mesh optimizer, "HighOrderElastic" for high-order elastic smoother, "HighOrderFastCurving" for fast curving algorithm, "Laplace2D" for Laplace smoothing, "Relocate2D" and "Relocate3D" for node relocation
- **force** (*bool*) – Apply to discrete entities
- **n_iterations** (*int*) – Number of iterations
- **threshold** (*float*) – Optimize tetrahedra with quality below threshold

class gmsh_scripts.optimize.optimize.**OptimizeMany**(*optimizers=None*, *force=False*, *threshold=0.3*)

Bases: *Optimize*

Optimize several times

Parameters

- **optimizers** (*list*) – [[opt1, n_iter1], [opt2, n_iter2], ...]
- **force** (*bool*) – Apply to discrete entities
- **threshold** (*float*) – Optimize tetrahedra with quality below threshold

gmsh_scripts.optimize.optimize.**str2obj**

gmsh_scripts.optimize.optimize.**n**

2.1.1.6 gmsh_scripts.parse

Submodules

gmsh_scripts.parse.parse

Module Contents

Functions

parse_row(row[, sep_i, sep_si])	Parse row of grid or layers
parse_row_item(item, item_t, pc, pm, ps, sep_i, sep_ii)	
parse_row_item_coordinate(c, pc, item_t, sep)	Parse coordinate item of grid row
parse_row_item_mesh_size(m, pm, cs, pc, sep)	Parse mesh size item of grid row
parse_row_item_structure(s, cs[, sep])	Parse structure col of grid row
parse_grid(grid[, sep_i, sep_si])	Parse grid
parse_layers2grid(layers[, sep_i, sep_si])	Parse layers
correct_layers(layers)	Correct layers
parse_layers(layers[, sep_i, sep_si])	Parse layers

gmsh_scripts.parse.parse.**parse_row**(*row*, *sep_i=';'*, *sep_si=':'*)

Parse row of grid or layers

Item types:

1. 'v' - value
2. 'i' - increment

Row types:

1. 'p' - product
2. 's' - slice

Types (first item of the row):

1. 'v;p' - item value, row product
2. 'v;s' - item value, row slice
3. 'i;p' - item increment, row product
4. 'i;s' - item increment, row slice

Parameters

- **row** (*list*) – row to parse, [t, i0 (origin), i1, i2, ..., iN], where t - type, i - item
- **sep_i** (*str*) – items separator
- **sep_si** (*str*) – sub-items separator

Returns

row (list): corrected row, values (list): [coordinates, mesh sizes, structures], maps (list): [o_b2is, o_is2b, n_b2is, n_is2b, n2o_i2i, n2o_b2b],

where 2 - to, o - old, n - new, b - block, is - indices

Return type
tuple

gmsh_scripts.parse.parse.**parse_row_item**(*item*, *item_t*, *pc*, *pm*, *ps*, *sep_i*, *sep_ii*)

gmsh_scripts.parse.parse.**parse_row_item_coordinate**(*c*, *pc*, *item_t*, *sep*)

Parse coordinate item of grid row

Parameters

- **c** (*str*) – col to parse, variants: 1. "c:n" (uniform): coordinate (float), number of steps (int) 2. "c:n:a:b" (Beta CDF): coordinate (float), number of steps (int),

  alpha (float), beta (float)

- **pc** (*float*) – previous coordinate
- **item_t** (*str*) – item type
- **sep** (*str*) – values separator

Returns
new coordinates

Return type
list of float

gmsh_scripts.parse.parse.**parse_row_item_mesh_size**(*m*, *pm*, *cs*, *pc*, *sep*)

Parse mesh size item of grid row

Parameters

- **m** (*str*) – col to parse, variants: 1. "m" (direct): mesh size (float) 2. "m:w:a:b" (Beta PDF): mesh size (float), weight (float),

  alpha (float), beta (float)

- **pm** (*float*) – previous mesh size
- **cs** (*list of float*) – coordinates
- **pc** (*float*) – previous coordinate
- **sep** (*str*) – values separator

Returns
mesh size for each coordinate

Return type
list of float

gmsh_scripts.parse.parse.**parse_row_item_structure**(*s*, *cs*, *sep=':'*)

Parse structure col of grid row

TODO interpolation of number of nodes between coordinates? :param s: col to parse, variants:

1. "n": number of nodes (int)
2. **"n:t": number of nodes (int), type (int):**
   0 - Progression (default), 1 - Bump, 2 - Beta
3. **"n:t:k": number of nodes (int), type (int):**
   0 - Progression (default), 1 - Bump, 2 - Beta, coefficient (float) from 0 to inf, default 1.0

Parameters

- **cs** (*list of float*) – coordinates,
- **sep** (*str*) – values separator

Returns
[[number of nodes, type, coefficient], …, n coordinates]

Return type
list of list

gmsh_scripts.parse.parse.**parse_grid**(*grid*, *sep_i=';'*, *sep_si=':'*)

Parse grid

Parameters

- **grid** (*list of list*) – list of rows
- **sep_i** (*str*) – items separator
- **sep_si** (*str*) – sub-items separator

Returns
grid, values, maps

Return type
tuple

gmsh_scripts.parse.parse.**parse_layers2grid**(*layers*, *sep_i=';'*, *sep_si=':'*)

Parse layers

0. **From list of rows (2D ragged array 6xNI), where**
   NI - number of items/blocks by each direction (type item excluded!). First item of Z/NZ rows implicitly set to 0 [

   [0 1 2 3] X [4 5 6] Y [7 8] NX [9] NY [10 11] Z [12 13] NZ

   ] layers
1. Corrected layers
2. **To layers block map (4D ragged array 4x2xNJxNI), where**
   NJ - number of items/blocks by Z/NZ respectively, NI - number of items/blocks by X/Y/NX/NY respectively

[

[

[

[0 1 2 3] Z0 [4 5 6 7] Z1

X0 X1 X2 X3

] Z [

[8 9 10 11] NZ0 [12 13 14 15] NZ1

X0 X1 X2 X3

] NZ

] X [

[

[16 17 18] Z0 [19 20 21] Z1

```
            Y0 Y1 Y2
        ] Z [
            [22 23 24] NZ0 [25 26 27] NZ1
            Y0 Y1 Y2
        ] NZ
    ] Y [
        [
            [ 28 29] Z0 [ 30 31] Z1
            NX0 NX1
        ] Z [
            [ 32 33] NZ0 [ 34 35] NZ1
            NX0 NX1
        ] - NZ
    ] NX [
        [
            [ 36] Z0 [ 37] Z1
            NY0
        ] Z [
            [ 38] NZ0 [ 39] NZ1
            NY0
        ] NZ
    ] NY
```

] - layers block map

To grid (3D array, 3xNI), NI - number of items by X/Y/Z respectively [

NX + X - X, where NX is negated and reversed NY + Y - Y, where NY is negated and reversed NZ + [0] + Z - Z, where NZ is negated and reversed

] - grid

To block map of the grid (3D array, NZN+ZN x NYN+YN-1 x NXN+XN-1) [

```
    [
        [ 0 1 2 3 4] NY0/Y0 [ 5 6 7 8 9] Y1 [ 10 11 12 13 14] Y2
        NX1 NX0/X0 X1 X2 X3
    ] NZ1 [
        [ 15 16 17 18 19] NY0/Y0 [ 20 21 22 23 24] Y1 [ 25 26 27 28 29] Y2
        NX1 NX0/X0 X1 X2 X3
    ] NZ0 [
        [ 30 31 32 33 34] NY0/Y0 [ 35 36 37 38 39] Y1 [ 40 41 42 43 44] Y2
        NX1 NX0/X0 X1 X2 X3
```

] Z0 [

[45 46 47 48 49] NY0/Y0 [50 51 52 53 54] Y1 [55 56 57 58 59] Y2

NX1 NX0/X0 X1 X2 X3

] Z1

] grid block map

Global indexes of layers block map at grid block map [

[

[35 12/25/34/39 13 14 15] NY0/Y0 [- 26 - - -] Y1 [- 27 - - -] Y2

NX1 NX0/X0 X1 X2 X3

] NZ1 [

[33 8/22/32/38 9 10 11] NY0/Y0 [- 23 - - -] Y1 [- 24 - - -] Y2

NX1 NX0/X0 X1 X2 X3

] NZ0 [

[29 0/16/28/36 1 2 3] NY0/Y0 [- 17 - - -] Y1 [- 18 - - -] Y2

NX1 NX0/X0 X1 X2 X3

] Z0 [

[31 4/19/30/37 5 6 7] NY0/Y0 [- 20 - - -] Y1 [- 21 - - -] Y2

NX1 NX0/X0 X1 X2 X3

] Z1

] block map of the grid

Parameters

- **layers** (*list of list*) – list of rows
- **sep_i** (*str*) – items separator
- **sep_si** (*str*) – sub-items separator

Returns
new_layers, values, maps

Return type
tuple

gmsh_scripts.parse.parse.**correct_layers**(*layers*)

Correct layers

[

[0 1 2 3] X [4 5 6] Y [7 8] NX [9] NY [10 11] Z [12 13] NZ

] layers with variants:

1. **X/Z (2xN)**

   [

   [0 1 2 3] X [4 5] Z

   ] layers

2. **X/Y/Z (3xN)**

[
[0 1 2 3] X [4 5 6] Y [7 8] Z
] layers

3. **X/Y/Z/NZ (4xN)**

[
[0 1 2 3] X [4 5 6] Y [7 8] Z [9 10] NZ
] layers

4. **X/Y/NX/NY/Z (5xN)**

[
[0 1 2 3] X [4 5 6] Y [7 8] NX [9] NY [10 11] Z
] layers

Parameters
layers (*list of list*) – layers

Returns:

gmsh_scripts.parse.parse.**parse_layers**(*layers*, *sep_i=';'*, *sep_si=':'*)

Parse layers

2.1.1.7 gmsh_scripts.quadrate

Submodules

gmsh_scripts.quadrate.quadrate

Module Contents

Classes

Quadrate
NoQuadrate
QuadrateBlock

Attributes

str2obj

class gmsh_scripts.quadrate.quadrate.**Quadrate**(*name=None*, ***kwargs*)

class gmsh_scripts.quadrate.quadrate.**NoQuadrate**

class gmsh_scripts.quadrate.quadrate.**QuadrateBlock**

gmsh_scripts.quadrate.quadrate.**str2obj**

2.1.1.8 gmsh_scripts.refine

Submodules

gmsh_scripts.refine.refine

Module Contents

Classes

Refine	Abstract
NoRefine	Abstract
RefineBySplit	**param n_iterations** Number of iterations

Attributes

str2obj

class gmsh_scripts.refine.refine.**Refine**

Abstract

class gmsh_scripts.refine.refine.**NoRefine**

Bases: *Refine*

Abstract

class gmsh_scripts.refine.refine.**RefineBySplit**(*n_iterations=1*)

Bases: *Refine*

Parameters
n_iterations (*int*) – Number of iterations

gmsh_scripts.refine.refine.**str2obj**

2.1.1.9 gmsh_scripts.size

Submodules

gmsh_scripts.size.size

Module Contents

Classes

Size
NoSize
BooleanPoint
BooleanEdge
Bagging

Attributes

str2obj

class gmsh_scripts.size.size.**Size**

evaluate_map(*block*)

class gmsh_scripts.size.size.**NoSize**

Bases: *Size*

class gmsh_scripts.size.size.**BooleanPoint**(*intra_function='min'*, *inter_function='min'*, *factor=1.0*, *min_size=0.0*, *max_size=1e+22*)

Bases: *Size*

evaluate_map(*block*)

class gmsh_scripts.size.size.**BooleanEdge**(*intra_function='min'*, *inter_function='min'*, *factor=1.0*, *min_size=0.0*, *max_size=1e+22*)

Bases: *Size*

evaluate_map(*block*)

class gmsh_scripts.size.size.**Bagging**(*sizes=()*, *inter_function='mean'*)

Bases: *Size*

evaluate_map(*block*)

gmsh_scripts.size.size.**str2obj**

2.1.1.10 gmsh_scripts.smooth

Submodules

gmsh_scripts.smooth.smooth

Module Contents

Classes

Smooth	Abstract
NoSmooth	
SmoothByDim	**param dims** Dims to smooth

Attributes

str2obj

class gmsh_scripts.smooth.smooth.**Smooth**

Abstract

class gmsh_scripts.smooth.smooth.**NoSmooth**

class gmsh_scripts.smooth.smooth.**SmoothByDim**(*dims=None*, *n_iterations=1*, *smooth_normals=False*, *smooth_cross_field=False*)

Bases: *Smooth*

Parameters

- **dims** (*list*) – Dims to smooth
- **n_iterations** (*int*) – Number of iterations
- **smooth_normals** (*bool*) – Do smooth normals?
- **smooth_cross_field** (*bool*) – Do smooth cross fields?

gmsh_scripts.smooth.smooth.**str2obj**

2.1.1.11 gmsh_scripts.strategy

Submodules

gmsh_scripts.strategy.strategy

Module Contents

Classes

Strategy	Abstract strategy
Base	Default strategy
Fast	Generates geometry only (.geo_unrolled file)
NoBoolean	Doesn't use boolean operations

Attributes

str2obj

class gmsh_scripts.strategy.strategy.**Strategy**(*factory=None*, *model_name=None*, *output_path=None*, *output_formats=None*)

Abstract strategy

class gmsh_scripts.strategy.strategy.**Base**(*factory=None*, *model_name=None*, *output_path=None*, *output_formats=None*, *boolean_function=BooleanAllBlock()*, *zone_function=DirectionByNormal()*, *size_function=NoSize()*, *structure_function=StructureBlock()*, *quadrate_function=NoQuadrate()*, *optimize_function=OptimizeOne()*, *refine_function=NoRefine()*, *smooth_function=NoSmooth()*)

Bases: *Strategy*

Default strategy

class gmsh_scripts.strategy.strategy.**Fast**(*factory=None*, *model_name=None*, *output_path=None*, *output_formats=None*)

Bases: *Strategy*

Generates geometry only (.geo_unrolled file)

class gmsh_scripts.strategy.strategy.**NoBoolean**(*factory=None*, *model_name=None*, *output_path=None*, *output_formats=None*, *zone_function=BlockZone()*, *size_function=NoSize()*, *structure_function=StructureBlock()*, *quadrate_function=NoQuadrate()*, *optimize_function=OptimizeOne()*, *refine_function=NoRefine()*, *smooth_function=NoSmooth()*)

Bases: *Strategy*

Doesn't use boolean operations

gmsh_scripts.strategy.strategy.**str2obj**

2.1.1.12 gmsh_scripts.structure

Submodules

gmsh_scripts.structure.structure

see https://gitlab.onelab.info/gmsh/gmsh/blob/gmsh_4_8_4/tutorial/python/x2.py # Set this to True to build a fully hex mesh: #transfinite = True transfinite = False transfiniteAuto = False

if transfinite:
NN = 30 for c in gmsh.model.getEntities(1):

gmsh.model.mesh.setTransfiniteCurve(c[1], NN)

for s in gmsh.model.getEntities(2):
gmsh.model.mesh.setTransfiniteSurface(s[1]) gmsh.model.mesh.setRecombine(s[0], s[1]) gmsh.model.mesh.setSmoothing(s[0], s[1], 100)

gmsh.model.mesh.setTransfiniteVolume(v1)

elif transfiniteAuto:
gmsh.option.setNumber('Mesh.MeshSizeMin', 0.5) gmsh.option.setNumber('Mesh.MeshSizeMax', 0.5) # setTransfiniteAutomatic() uses the sizing constraints to set the number # of points gmsh.model.mesh.setTransfiniteAutomatic()

else:
gmsh.option.setNumber('Mesh.MeshSizeMin', 0.05) gmsh.option.setNumber('Mesh.MeshSizeMax', 0.05)

def setTransfiniteAutomatic(dimTags=[], cornerAngle=2.35, recombine=True):

Set transfinite meshing constraints on the model entities in `dimTag'. Transfinite meshing constraints are added to the curves of the quadrangular surfaces and to the faces of 6-sided volumes. Quadragular faces with a corner angle superior to `cornerAngle' (in radians) are ignored. The number of points is automatically determined from the sizing constraints. If `dimTag' is empty, the constraints are applied to all entities in the model. If `recombine' is true, the recombine flag is automatically set on the transfinite surfaces.

Module Contents

Classes

Structure	
NoStructure	
StructureAuto	**param corner_angle** Quadrangular faces with a corner angle superior
StructureBlock	

Attributes

str2obj

class gmsh_scripts.structure.structure.**Structure**(*name=None*, ***kwargs*)

class gmsh_scripts.structure.structure.**NoStructure**

class gmsh_scripts.structure.structure.**StructureAuto**(*corner_angle=np.rad2deg(2.35)*, *quadrate=True*)

Parameters

- **corner_angle** (*float*) – Quadrangular faces with a corner angle superior to corner_angle (in degree) are ignored
- **quadrate** (*bool*) – Quadrate surfaces

class gmsh_scripts.structure.structure.**StructureBlock**(*do_quadrate=True*, *do_structure=True*)

gmsh_scripts.structure.structure.**str2obj**

2.1.1.13 gmsh_scripts.support

Submodules

gmsh_scripts.support.support

Module Contents

Classes

LoggingDecorator	Decorator for logger initialization
GmshDecorator	Decorator fot gmsh initialization
GmshOptionsDecorator	Decorator for setting gmsh options
DataTree	Volumes data tree

Functions

flatten(iterable[, types])	Flatten iterable through types
plot_quality()	
plot_statistics()	
timeit(f)	
beta_function(xs, a, b[, n])	Beta function
beta_pdf(xs, a, b[, n])	Beta probability density function
beta_cdf(xs, a, b[, n])	Beta cumulative distribution function
check_on_file(path)	Check path on the file
volumes_surfaces_to_volumes_groups_surfaces(...)	For Environment object. For each distinct inner volume in Environment

class gmsh_scripts.support.support.**LoggingDecorator**(*filename=None*, *filemode='a'*, *fmt=None*, *datefmt=None*, *level='INFO'*)

Decorator for logger initialization

Parameters

- **filename** (*str*) – path to log file
- **filemode** (*str*) – mode of log file
- **fmt** (*str*) – format of messages See https://docs.python.org/3/library/logging.html#logrecord-attributes
- **datefmt** (*str*) – format of the date
- **level** (*int or str*) – CRITICAL = 50, FATAL = CRITICAL, ERROR = 40, WARNING = 30, WARN = WARNING, INFO = 20, DEBUG = 10, NOTSET = 0

class gmsh_scripts.support.support.**GmshDecorator**

Decorator fot gmsh initialization

1. Initialize
2. Call function
3. Finalize

class gmsh_scripts.support.support.**GmshOptionsDecorator**(*options=None*)

Decorator for setting gmsh options

1. Set options
2. Call function

Options:
https://gmsh.info/doc/texinfo/gmsh.html#Options

Unstructured mesh:
Gmsh provides a choice between several 2D and 3D unstructured algorithms. Each algorithm has its own advantages and disadvantages.

For all 2D unstructured algorithms a Delaunay mesh that contains all the points of the 1D mesh is initially constructed using a divide-and-conquer algorithm5. Missing edges are recovered using edge swaps6. After this initial step several algorithms can be applied to generate the final mesh:

The "MeshAdapt" algorithm7 is based on local mesh modifications. This technique makes use of edge swaps, splits, and collapses: long edges are split, short edges are collapsed, and edges are swapped if a better geometrical configuration is obtained. The "Delaunay" algorithm is inspired by the work of the GAMMA team at INRIA8. New points are inserted sequentially at the circumcenter of the element that has the largest adimensional circumradius. The mesh is then reconnected using an anisotropic Delaunay criterion. The "Frontal-Delaunay" algorithm is inspired by the work of S. Rebay9. Other experimental algorithms with specific features are also available. In particular, "Frontal-Delaunay for Quads"10 is a variant of the "Frontal-Delaunay" algorithm aiming at generating right-angle triangles suitable for recombination; and "BAMG"11 allows to generate anisotropic triangulations. For very complex curved surfaces the "MeshAdapt" algorithm is the most robust. When high element quality is important, the "Frontal-Delaunay" algorithm should be tried. For very large meshes of plane surfaces the "Delaunay" algorithm is the fastest; it usually also handles complex mesh size fields better than the "Frontal-Delaunay". When the "Delaunay" or "Frontal-Delaunay" algorithms fail, "MeshAdapt" is automatically triggered. The "Automatic" algorithm uses "Delaunay" for plane surfaces and "MeshAdapt" for all other surfaces.

Several 3D unstructured algorithms are also available:

The "Delaunay" algorithm is split into three separate steps. First, an initial mesh of the union of all the volumes in the model is performed, without inserting points in the volume. The surface mesh is then recovered using H. Si's boundary recovery algorithm Tetgen/BR. Then a three-dimensional version of the 2D Delaunay algorithm described above is applied to insert points in the volume to respect the mesh size constraints. The "Frontal" algorithm uses J. Schoeberl's Netgen algorithm 12. The "HXT" algorithm13 is a new efficient and parallel reimplementaton of the Delaunay algorithm. Other experimental algorithms with specific features are also available. In particular, "MMG3D"14 allows to generate anisotropic tetrahedralizations. The "Delaunay" algorithm is currently the most robust and is the only one that supports the automatic generation of hybrid meshes with pyramids. Embedded model entities and the Field mechanism to specify element sizes (see Specifying mesh element sizes) are currently only supported by the "Delaunay" and "HXT" algorithms.

If your version of Gmsh is compiled with OpenMP support (see Compiling the source code), most of the meshing steps can be performed in parallel:

1D and 2D meshing is parallelized using a coarse-grained approach, i.e. curves (resp. surfaces) are each meshed sequentially, but several curves (resp. surfaces) can be meshed at the same time. 3D meshing using HXT is parallelized using a fine-grained approach, i.e. the actual meshing procedure for a single volume is done is parallel. The number of threads can be controlled with the -nt flag on the command line (see Command-line options), or with the General.NumThreads, Mesh.MaxNumThreads1D, Mesh.MaxNumThreads2D and Mesh.MaxNumThreads3D options (see General options list and Mesh options list). To determine the size of mesh elements, Gmsh locally computes the minimum of

1) the size of the model bounding box;
2) if `Mesh.MeshSizeFromPoints' is set, the mesh size specified at geometrical points;
3) if `Mesh.MeshSizeFromCurvature' is positive, the mesh size based on curvature (the value specifying the number of elements per 2 * pi rad);
4) the background mesh size field;
5) any per-entity mesh size constraint.

This value is then constrained in the interval [`Mesh.MeshSizeMin', `Mesh.MeshSizeMax'] and multiplied by `Mesh.MeshSizeFactor'. In addition, boundary mesh sizes (on curves or surfaces) are interpolated inside the enclosed entity (surface or volume, respectively) if the option `Mesh.MeshSizeExtendFromBoundary' is set (which is the case by default).

When the element size is fully specified by a background mesh size field (as it is in this example), it is thus often desirable to set

Mesh.MeshSizeExtendFromBoundary = 0; Mesh.MeshSizeFromPoints = 0; Mesh.MeshSizeFromCurvature = 0;

This will prevent over-refinement due to small mesh sizes on the boundary.

Quadrate:

To generate quadrangles instead of triangles, we can simply add gmsh.model.mesh.setRecombine(2, pl) If we'd had several surfaces, we could have used the global option "Mesh.RecombineAll":

gmsh.option.setNumber("Mesh.RecombineAll", 1)

The default recombination algorithm is called "Blossom": it uses a minimum cost perfect matching algorithm to generate fully quadrilateral meshes from triangulations. More details about the algorithm can be found in the following paper: J.-F. Remacle, J. Lambrechts, B. Seny, E. Marchandise, A. Johnen and C. Geuzaine, "Blossom-Quad: a non-uniform quadrilateral mesh generator using a minimum cost perfect matching algorithm", International Journal for Numerical Methods in Engineering 89, pp. 1102-1119, 2012.

For even better 2D (planar) quadrilateral meshes, you can try the experimental "Frontal-Delaunay for quads" meshing algorithm, which is a triangulation algorithm that enables to create right triangles almost everywhere: J.-F. Remacle, F. Henrotte, T. Carrier-Baudouin, E. Bechet, E. Marchandise, C. Geuzaine and T. Mouton. A frontal Delaunay quad mesh generator using the L^inf norm. International Journal for Numerical Methods in Engineering, 94, pp. 494-512, 2013. Uncomment the following line to try the Frontal-Delaunay algorithms for quads:

gmsh.option.setNumber("Mesh.Algorithm", 8)

The default recombination algorithm might leave some triangles in the mesh, if recombining all the triangles leads to badly shaped quads. In such cases, to generate full-quad meshes, you can either subdivide the resulting hybrid mesh (with `Mesh.SubdivisionAlgorithm' set to 1), or use the full-quad recombination algorithm, which will automatically perform a coarser mesh followed by recombination, smoothing and subdivision. Uncomment the following line to try the full-quad algorithm:

gmsh.option.setNumber("Mesh.RecombinationAlgorithm", 2) # or 3

You can also set the subdivision step alone, with

gmsh.option.setNumber("Mesh.SubdivisionAlgorithm", 1)

gmsh.model.mesh.generate(2)

Note that you could also apply the recombination algorithm and/or the subdivision step explicitly after meshing, as follows:

gmsh.model.mesh.generate(2) gmsh.model.mesh.recombine() gmsh.option.setNumber("Mesh.SubdivisionAlgorithm", 1) gmsh.model.mesh.refine()

gmsh_scripts.support.support.**flatten**(*iterable*, *types=(list,)*)

Flatten iterable through types

Parameters

- **iterable** – some iterable
- **types** – types to recurse

Returns

elements

Return type
generator of object

class gmsh_scripts.support.support.**DataTree**(*vs_dt*)

Volumes data tree # TODO add surfaces loops :param vs_dt: Volumes dim-tags :type vs_dt: list of tuple

vs_dt

Volumes dim-tags

Type
list of tuple

vs_ss_dt

Surfaces dim-tags of volumes

Type
list of tuple

vs_ss_cs_dt

Curves dim-tags of surfaces of volumes

Type
list of tuple

vs_ss_cs_ps_dt

Points dim-tags of curves of surfaces of volumes

Type
list of tuple

ps_dt_to_cs

Points dim-tags to coordinates

Type
dict

static evaluate_boundary_surfaces(*vs_ss_dt*)

static evaluate_global_surfaces_loops(*vs_ss_dt*, *vs_ss_cs_dt*)

gmsh_scripts.support.support.**plot_quality**()

gmsh_scripts.support.support.**plot_statistics**()

gmsh_scripts.support.support.**timeit**(*f*)

gmsh_scripts.support.support.**beta_function**(*xs*, *a*, *b*, *n=10000*)

Beta function

https://en.wikipedia.org/wiki/Beta_function#Incomplete_beta_function

Parameters

- **xs** (*float, np.ndarray*) – argument(s)
- **a** (*float*) – alpha
- **b** (*float*) – beta
- **n** (*int*) – number of integration steps

Returns
value

Return type
float, np.ndarray

gmsh_scripts.support.support.**beta_pdf**(*xs*, *a*, *b*, *n=10000*)

Beta probability density function

https://en.wikipedia.org/wiki/Beta_distribution#Probability_density_function

Parameters

- **xs** (*float, np.ndarray*) – argument(s)
- **a** (*float*) – alpha
- **b** (*float*) – beta
- **n** (*int*) – number of integration steps

Returns
value

Return type
float, np.ndarray

gmsh_scripts.support.support.**beta_cdf**(*xs*, *a*, *b*, *n=10000*)

Beta cumulative distribution function

https://en.wikipedia.org/wiki/Beta_distribution#Cumulative_distribution_function https://en.wikipedia.org/wiki/Beta_function#Incomplete_beta_function

Parameters

- **xs** (*float, np.ndarray*) – argument(s)
- **a** (*float*) – alpha
- **b** (*float*) – beta
- **n** (*int*) – number of integration steps

Returns
value [0, 1]

Return type
float, np.ndarray

gmsh_scripts.support.support.**check_on_file**(*path*)

Check path on the file

In the order: 1. If file at absolute path 2. Else if file at relative to current working directory path 3. Else if file at relative to running script directory path 4. Else if file at relative to real running script directory path (with eliminating all symbolics links) 0. Else no file

Parameters
path (*str*) – path to check

Returns
result, expanded path to file or None

Return type
tuple

gmsh_scripts.support.support.**volumes_surfaces_to_volumes_groups_surfaces**(*volumes_surfaces*)

For Environment object. For each distinct inner volume in Environment should exist the surface loop. If inner volumes touch each other they unite to volume group and have common surface loop. :param volumes_surfaces: [[v1_s1, ..., v1_si], ..., [vj_s1, ..., vj_si]] :return: volumes_groups_surfaces [[vg1_s1, ..., vg1_si], ...]

2.1.1.14 gmsh_scripts.transform

Submodules

gmsh_scripts.transform.transform

Module Contents

Classes

Transform	General transformation of Point coordinates.
Translate	Translate coordinates of the Point by the displacement
Rotate	TODO only for 3D coordinate systems
CartesianToCartesian	Convert coordinates of the Point from Cartesian to Cartesian system.
CylindricalToCartesian	Convert coordinates of the Point from Cylindrical to Cartesian system.
SphericalToCartesian	Convert coordinates of the Point from Spherical to Cartesian system.
ToroidalToCartesian	Convert coordinates of the Point from Toroidal to Cartesian system.
TokamakToCartesian	Convert coordinates of the Point from Tokamak to Cartesian system.
BlockToCartesian	Convert coordinates of the Point from Block to Cartesian system.
CartesianToCartesianByBlock	Convert coordinates of the Point from Cartesian to Cartesian system by Block coordinates.
AffineToAffine	Convert coordinates of the Point from Affine to Affine system.
AffineToCartesian	Convert coordinates of the Point from Affine to Cartesian system.
PathToCartesian	Convert coordinates of the Point from Path to Cartesian system.
LayerToCartesian	General transformation of Point coordinates.
QuarterLayerToCartesian	General transformation of Point coordinates.
AnyAsSome	Treat any coordinate system as some coordinate system without coordinates transformation
TransformationMatrix	Affine transformation matrix

Functions

reduce_transforms(transforms, point)	Apply transformations on the point.

Attributes

str2obj

gmsh_scripts.transform.transform.**reduce_transforms**(*transforms*, *point*)

Apply transformations on the point.

Parameters

- **transforms** (*list*) – Transformations
- **point** (Point) – Point

Returns
Transformed point.

Return type
Point

class gmsh_scripts.transform.transform.**Transform**(*cs_from=None*, *cs_to=None*, *cs_self=None*, ***kwargs*)

General transformation of Point coordinates.

Parameters

- **cs_from** (CoordinateSystem) – CoordinateSystem from
- **cs_to** (CoordinateSystem) – CoordinateSystem to
- **cs_self** (CoordinateSystem) – CoordinateSystem self

class gmsh_scripts.transform.transform.**Translate**(*delta*, ***kwargs*)

Bases: *Transform*

Translate coordinates of the Point by the displacement

Parameters
delta (*list or np.ndarray*) – displacement

class gmsh_scripts.transform.transform.**Rotate**(*origin*, *direction*, *angle*, ***kwargs*)

Bases: *Transform*

TODO only for 3D coordinate systems :param origin: origin of rotation :type origin: list or np.ndarray :param direction: rotation axis :type direction: list or np.ndarray :param angle: counterclockwise angle of rotation in radians [0, 2*pi) :type angle: float

class gmsh_scripts.transform.transform.**CartesianToCartesian**(***kwargs*)

Bases: *Transform*

Convert coordinates of the Point from Cartesian to Cartesian system.

For compatibility.

class gmsh_scripts.transform.transform.**CylindricalToCartesian**(**kwargs)

Bases: *Transform*

Convert coordinates of the Point from Cylindrical to Cartesian system.

[r, phi, z] -> [x, y, z] r - radius [0, inf), phi - azimuthal angle [0, 2*pi) (counterclockwise from X to Y), z - height

class gmsh_scripts.transform.transform.**SphericalToCartesian**(**kwargs)

Bases: *Transform*

Convert coordinates of the Point from Spherical to Cartesian system.

[r, phi, theta] -> [x, y, z]

- r - radius [0, inf)
- phi - azimuthal angle [0, 2*pi) (counterclockwise from X to Y)
- theta - polar angle [0, pi] [from top to bottom, i.e XY-plane is pi/2]

class gmsh_scripts.transform.transform.**ToroidalToCartesian**(**kwargs)

Bases: *Transform*

Convert coordinates of the Point from Toroidal to Cartesian system.

[r, phi, theta, r2] -> [x, y, z]

r - inner radius (r < r2)

phi - inner angle [0, 2*pi)

theta - outer angle [0, 2*pi)

r2 - outer radius

class gmsh_scripts.transform.transform.**TokamakToCartesian**(**kwargs)

Bases: *Transform*

Convert coordinates of the Point from Tokamak to Cartesian system.

[r, phi, theta, r2, kxy, kz] -> [x, y, z]

r - inner radius (r < r2)

phi - inner angle [0, 2*pi)

theta - outer angle [0, 2*pi)

r2 - outer radius

kxy - inner radius XY scale coefficient in positive outer radius direction

kz - inner radius Z scale coefficient

class gmsh_scripts.transform.transform.**BlockToCartesian**(*cs_from=None*, **kwargs)

Bases: *Transform*

Convert coordinates of the Point from Block to Cartesian system.

[xi, eta, zeta] -> [x, y, z]

xi, eta, zeta - local block coordinates

Parameters

cs_from (coordinate_system.Block) – Block Coordinate System

class gmsh_scripts.transform.transform.**CartesianToCartesianByBlock**(*block=None*, *block_coordinates=None*, ***kwargs*)

Bases: *Transform*

Convert coordinates of the Point from Cartesian to Cartesian system by Block coordinates.

[x, y, z] -> [x, y, z] + [dx, dy, dz]

[dx, dy, dz] = [xi, eta, zeta] - block_coordinates in Cartesian system

Parameters

- **block** (block.Block) – Block object
- **block_coordinates** (*list or np.ndarray*) – xi, eta, zeta - block local coordinates

class gmsh_scripts.transform.transform.**AffineToAffine**(*cs_to*, ***kwargs*)

Bases: *Transform*

Convert coordinates of the Point from Affine to Affine system.

[x0, y0, z0] -> [x1, y1, z1]

Parameters

cs_to (Affine) – Affine coordinate system

class gmsh_scripts.transform.transform.**AffineToCartesian**(***kwargs*)

Bases: *Transform*

Convert coordinates of the Point from Affine to Cartesian system.

[x0, y0, z0] -> [x, y, z]

class gmsh_scripts.transform.transform.**PathToCartesian**(***kwargs*)

Bases: *Transform*

Convert coordinates of the Point from Path to Cartesian system.

[x, y, u] -> [x, y, z]

u - relative path coordinate [0, 1], where 0 - start of the path, 1 - end.

x, y - Cartesian coordinates [-inf, inf] on the normal plane to direction of the path derivative at the point with relative path coordinate u.

class gmsh_scripts.transform.transform.**LayerToCartesian**(***kwargs*)

Bases: *Transform*

General transformation of Point coordinates.

Parameters

- **cs_from** (CoordinateSystem) – CoordinateSystem from
- **cs_to** (CoordinateSystem) – CoordinateSystem to
- **cs_self** (CoordinateSystem) – CoordinateSystem self

static update_coordinate(*p0*, *pn*, *n0*, *nn*, *l00*, *l0n*, *ln0*, *lnn*, *lt*)

class gmsh_scripts.transform.transform.**QuarterLayerToCartesian**(***kwargs*)

Bases: *Transform*

General transformation of Point coordinates.

Parameters

- **cs_from** (CoordinateSystem) – CoordinateSystem from
- **cs_to** (CoordinateSystem) – CoordinateSystem to
- **cs_self** (CoordinateSystem) – CoordinateSystem self

static update_coordinate(*p0*, *pn*, *n0*, *nn*, *l00*, *l0n*, *ln0*, *lnn*, *lt*)

class gmsh_scripts.transform.transform.**AnyAsSome**(*cs_to*, ***kwargs*)

Bases: *Transform*

Treat any coordinate system as some coordinate system without coordinates transformation

Parameters

cs_to (CoordinateSystem) – some coordinate system

class gmsh_scripts.transform.transform.**TransformationMatrix**(*matrix*, ***kwargs*)

Bases: *Transform*

Affine transformation matrix

Matrix could describe translation, rotation, reflection and dilation

See also:

https://en.wikipedia.org/wiki/Affine_transformation#Representation

Parameters

matrix (*list or list of list or np.ndarray*) – transformation matrix

gmsh_scripts.transform.transform.**str2obj**

2.1.1.15 gmsh_scripts.utils

Submodules

gmsh_scripts.utils.obj2gmsh

Module Contents

Functions

main(lines)

Attributes

parser

gmsh_scripts.utils.obj2gmsh.**main**(*lines*)

gmsh_scripts.utils.obj2gmsh.**parser**

2.1.1.16 gmsh_scripts.zone

Submodules

gmsh_scripts.zone.zone

Module Contents

Classes

Zone

Block	Name from zone field of entity
NoZone	Name from zone field of entity
DirectionByNormal	Construct zones from boolean map and Block's zones
Mesh	Construct zones from mesh instead of Block's zones

Attributes

str2obj

class gmsh_scripts.zone.zone.**Zone**

evaluate_map(*block*)

class gmsh_scripts.zone.zone.**Block**(*dims=None*, *dims_interfaces=None*, *inter_separator='|'*)

Bases: *Zone*

Name from zone field of entity

evaluate_map(*block*)

class gmsh_scripts.zone.zone.**NoZone**

Bases: *Zone*

Name from zone field of entity

class gmsh_scripts.zone.zone.**DirectionByNormal**(*dims=None*, *dims_interfaces=None*, *join_interfaces=None*, *self_separator='_'*, *intra_separator='-'*, *inter_separator='|'*, *zones=None*, *zones_directions=None*)

Bases: *Zone*

Construct zones from boolean map and Block's zones

Parameters

- **dims** (*list*) – Which dims to zones
- **dims_interfaces** (*list*) – Which dims to interfaces zones
- **join_interfaces** (*list*) – Which interfaces should be joined together, Variants:
  1. [[volumes zones], [volumes zones], ...]
  2. [[[volumes zones], [entities zones]], ...]

  If the zone set with * symbol then join all zones which includes the zone else only zones that fully match the zone.
- **self_separator** (*str*) – separator in volume, surface, curve and point
- **intra_separator** (*str*) – separator between volume, surface, curve and point
- **inter_separator** (*str*) – separator between interfaces

evaluate_map(*block*)

class gmsh_scripts.zone.zone.**Mesh**(*dims=(0, 1, 2, 3)*, *dims_interfaces=(0, 1, 2, 3)*, *join_interfaces='all'*, *entities_separator='_'*, *join_separator='-'*)

Bases: *Zone*

Construct zones from mesh instead of Block's zones

Parameters

- **dims** (*tuple of int*) – Which dims to zones
- **dims_interfaces** (*tuple of int*) – Which dims to interfaces zones
- **join_interfaces** (*str*) – Join interfaces zones 'all' - join all in alphabetical order 'first' - get first only in alphabetical order None - don't join
- **entities_separator** (*str*) – separator between volume, surface, curve and point
- **join_separator** (*str*) – separator used in join_interfaces

evaluate_map(*block*)

gmsh_scripts.zone.zone.**str2obj**

2.1.2 Submodules

2.1.2.1 gmsh_scripts.factory

Module Contents

Classes

Factory

Attributes

FACTORY

exception gmsh_scripts.factory.**FactoryClassError**(*value*)

Bases: Exception

Common base class for all non-exit exceptions.

exception gmsh_scripts.factory.**FactoryKeyError**(*value*)

Bases: Exception

Common base class for all non-exit exceptions.

exception gmsh_scripts.factory.**FactoryValueError**(*value*)

Bases: Exception

Common base class for all non-exit exceptions.

class gmsh_scripts.factory.**Factory**

gmsh_scripts.factory.**FACTORY**

2.1.2.2 gmsh_scripts.load

Module Contents

Functions

load(path)

gmsh_scripts.load.**load**(*path*)

2.1.2.3 gmsh_scripts.plot

Module Contents

Functions

plot_action(a[, output, height, width, title, ...])	
main()	

gmsh_scripts.plot.**plot_action**(*a*, *output=None*, *height='600px'*, *width='600px'*, *title=''*, *options=False*, *no_hierarchy=False*, *background='white'*, *font='black'*)

gmsh_scripts.plot.**main**()

2.1.2.4 gmsh_scripts.registry

Module Contents

Functions

reset([factory, point_tol])
correct_kwargs(entity, name)
register_point(point)
register_curve(curve)
register_curve_loop(curve_loop)
register_surface(surface)
register_surface_loop(surface_loop)
register_volume(volume)
register_quadrate_surface(surface)
register_structure_curve(curve)
register_structure_surface(surface)
register_structure_volume(volume)
pre_unregister_volume(volume)
unregister_volumes()
get_unregistered_volumes()
register_curve_structure(points, structure)
register_surface_structure(points, structure)
register_volume_structure(tag, structure)
register_surface_quadrate(points, quadrate)
get_curve_structure(points)
get_surface_structure(points)
get_volume_structure(tag)
get_surface_quadrate(points)
register_boolean_new2olds(m)
register_boolean_old2news(m)
get_boolean_new2olds()
get_boolean_old2news()
register_volume2block(volume_tag, block)

Attributes

FACTORY
POINT_TOL
POINTS
CURVES
CURVES_LOOPS
SURFACES
SURFACES_LOOPS
VOLUMES
POINT_TAG
CURVE_TAG
CURVE_LOOP_TAG
SURFACE_TAG
SURFACE_LOOP_TAG
VOLUME_TAG
QUADRATED_SURFACES
STRUCTURED_CURVES
STRUCTURED_SURFACES
STRUCTURED_VOLUMES
CURVE_STRUCTURE
SURFACE_STRUCTURE
VOLUME_STRUCTURE
SURFACE_QUADRATE
BOOLEAN_NEW2OLDS
BOOLEAN_OLD2NEWS

continues on next page

Table 1 – continued from previous page

VOLUME2BLOCK
USE_REGISTRY_TAG
UNREGISTERED_VOLUMES
POINT_KWARGS
CURVE_KWARGS
CURVE_LOOP_KWARGS
SURFACE_KWARGS
SURFACE_LOOP_KWARGS
VOLUME_KWARGS
RECOMBINE_KWARGS
TRANSFINITE_CURVE_KWARGS
TRANSFINITE_CURVE_TYPES
TRANSFINITE_SURFACE_KWARGS
TRANSFINITE_VOLUME_KWARGS
name2kwargs
add_point
add_curve
add_curve_loop
add_surface
add_surface_loop
add_volume
add_quadrate
add_structure_curve
add_structure_surface
add_structure_volume

gmsh_scripts.registry.**FACTORY** = 'geo'

gmsh_scripts.registry.**POINT_TOL = 12**

gmsh_scripts.registry.**POINTS**

gmsh_scripts.registry.**CURVES**

gmsh_scripts.registry.**CURVES_LOOPS**

gmsh_scripts.registry.**SURFACES**

gmsh_scripts.registry.**SURFACES_LOOPS**

gmsh_scripts.registry.**VOLUMES**

gmsh_scripts.registry.**POINT_TAG = 1**

gmsh_scripts.registry.**CURVE_TAG = 1**

gmsh_scripts.registry.**CURVE_LOOP_TAG = 1**

gmsh_scripts.registry.**SURFACE_TAG = 1**

gmsh_scripts.registry.**SURFACE_LOOP_TAG = 1**

gmsh_scripts.registry.**VOLUME_TAG = 1**

gmsh_scripts.registry.**QUADRATED_SURFACES**

gmsh_scripts.registry.**STRUCTURED_CURVES**

gmsh_scripts.registry.**STRUCTURED_SURFACES**

gmsh_scripts.registry.**STRUCTURED_VOLUMES**

gmsh_scripts.registry.**CURVE_STRUCTURE**

gmsh_scripts.registry.**SURFACE_STRUCTURE**

gmsh_scripts.registry.**VOLUME_STRUCTURE**

gmsh_scripts.registry.**SURFACE_QUADRATE**

gmsh_scripts.registry.**BOOLEAN_NEW2OLDS**

gmsh_scripts.registry.**BOOLEAN_OLD2NEWS**

gmsh_scripts.registry.**VOLUME2BLOCK**

gmsh_scripts.registry.**USE_REGISTRY_TAG = True**

gmsh_scripts.registry.**UNREGISTERED_VOLUMES**

gmsh_scripts.registry.**POINT_KWARGS**

gmsh_scripts.registry.**CURVE_KWARGS**

gmsh_scripts.registry.**CURVE_LOOP_KWARGS**

gmsh_scripts.registry.**SURFACE_KWARGS**

gmsh_scripts.registry.**SURFACE_LOOP_KWARGS**

gmsh_scripts.registry.**VOLUME_KWARGS**

gmsh_scripts.registry.**RECOMBINE_KWARGS**

gmsh_scripts.registry.**TRANSFINITE_CURVE_KWARGS**

gmsh_scripts.registry.**TRANSFINITE_CURVE_TYPES = ['Progression', 'Bump', 'Beta']**

gmsh_scripts.registry.**TRANSFINITE_SURFACE_KWARGS**

gmsh_scripts.registry.**TRANSFINITE_VOLUME_KWARGS**

gmsh_scripts.registry.**name2kwargs**

gmsh_scripts.registry.**reset**(*factory='geo'*, *point_tol=8*)

gmsh_scripts.registry.**correct_kwargs**(*entity*, *name*)

gmsh_scripts.registry.**add_point**

gmsh_scripts.registry.**add_curve**

gmsh_scripts.registry.**add_curve_loop**

gmsh_scripts.registry.**add_surface**

gmsh_scripts.registry.**add_surface_loop**

gmsh_scripts.registry.**add_volume**

gmsh_scripts.registry.**add_quadrate**

gmsh_scripts.registry.**add_structure_curve**

gmsh_scripts.registry.**add_structure_surface**

gmsh_scripts.registry.**add_structure_volume**

gmsh_scripts.registry.**register_point**(*point*)

gmsh_scripts.registry.**register_curve**(*curve*)

gmsh_scripts.registry.**register_curve_loop**(*curve_loop*)

gmsh_scripts.registry.**register_surface**(*surface*)

gmsh_scripts.registry.**register_surface_loop**(*surface_loop*)

gmsh_scripts.registry.**register_volume**(*volume*)

gmsh_scripts.registry.**register_quadrate_surface**(*surface*)

gmsh_scripts.registry.**register_structure_curve**(*curve*)

gmsh_scripts.registry.**register_structure_surface**(*surface*)

gmsh_scripts.registry.**register_structure_volume**(*volume*)

gmsh_scripts.registry.**pre_unregister_volume**(*volume*)

gmsh_scripts.registry.**unregister_volumes**()

gmsh_scripts.registry.**get_unregistered_volumes**()

gmsh_scripts.registry.**register_curve_structure**(*points*, *structure*)

gmsh_scripts.registry.**register_surface_structure**(*points*, *structure*)

gmsh_scripts.registry.**register_volume_structure**(*tag*, *structure*)

gmsh_scripts.registry.**register_surface_quadrate**(*points*, *quadrate*)

gmsh_scripts.registry.**get_curve_structure**(*points*)

gmsh_scripts.registry.**get_surface_structure**(*points*)

gmsh_scripts.registry.**get_volume_structure**(*tag*)

gmsh_scripts.registry.**get_surface_quadrate**(*points*)

gmsh_scripts.registry.**register_boolean_new2olds**(*m*)

gmsh_scripts.registry.**register_boolean_old2news**(*m*)

gmsh_scripts.registry.**get_boolean_new2olds**()

gmsh_scripts.registry.**get_boolean_old2news**()

gmsh_scripts.registry.**register_volume2block**(*volume_tag*, *block*)

gmsh_scripts.registry.**get_volume2block**()

gmsh_scripts.registry.**synchronize**()

2.1.2.5 gmsh_scripts.run

Module Contents

Functions

init_walk(obj[, prev_obj, prev_indices, prev_keys])	
set_parent(parent)	
parse_arguments()	
run(args)	
main()	

gmsh_scripts.run.**init_walk**(*obj*, *prev_obj=None*, *prev_indices=None*, *prev_keys=None*)

gmsh_scripts.run.**set_parent**(*parent*)

gmsh_scripts.run.**parse_arguments**()

gmsh_scripts.run.**run**(*args*)

gmsh_scripts.run.**main**()

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

g

gmsh_scripts, 53
gmsh_scripts.block, 53
gmsh_scripts.block.block, 53
gmsh_scripts.block.layer, 58
gmsh_scripts.block.matrix, 59
gmsh_scripts.block.polyhedron, 60
gmsh_scripts.boolean, 66
gmsh_scripts.boolean.boolean, 66
gmsh_scripts.coordinate_system, 67
gmsh_scripts.coordinate_system.coordinate_system, 67
gmsh_scripts.entity, 71
gmsh_scripts.entity.curve, 71
gmsh_scripts.entity.curve_loop, 72
gmsh_scripts.entity.point, 72
gmsh_scripts.entity.surface, 74
gmsh_scripts.entity.surface_loop, 74
gmsh_scripts.entity.volume, 75
gmsh_scripts.factory, 103
gmsh_scripts.load, 103
gmsh_scripts.optimize, 77
gmsh_scripts.optimize.optimize, 77
gmsh_scripts.parse, 79
gmsh_scripts.parse.parse, 79
gmsh_scripts.plot, 104
gmsh_scripts.quadrate, 84
gmsh_scripts.quadrate.quadrate, 84
gmsh_scripts.refine, 85
gmsh_scripts.refine.refine, 85
gmsh_scripts.registry, 104
gmsh_scripts.run, 110
gmsh_scripts.size, 86
gmsh_scripts.size.size, 86
gmsh_scripts.smooth, 87
gmsh_scripts.smooth.smooth, 87
gmsh_scripts.strategy, 88
gmsh_scripts.strategy.strategy, 88
gmsh_scripts.structure, 89
gmsh_scripts.structure.structure, 89
gmsh_scripts.support, 90
gmsh_scripts.support.support, 90
gmsh_scripts.transform, 96
gmsh_scripts.transform.transform, 96
gmsh_scripts.utils, 100
gmsh_scripts.utils.obj2gmsh, 100
gmsh_scripts.zone, 101
gmsh_scripts.zone.zone, 101

INDEX

A

add_child() (*gmsh_scripts.block.Block method*), 64
add_child() (*gmsh_scripts.block.block.Block method*), 57
add_curve (*in module gmsh_scripts.registry*), 109
add_curve_loop (*in module gmsh_scripts.registry*), 109
add_point (*in module gmsh_scripts.registry*), 109
add_quadrate (*in module gmsh_scripts.registry*), 109
add_structure_curve (*in module gmsh_scripts.registry*), 109
add_structure_surface (*in module gmsh_scripts.registry*), 109
add_structure_volume (*in module gmsh_scripts.registry*), 109
add_surface (*in module gmsh_scripts.registry*), 109
add_surface_loop (*in module gmsh_scripts.registry*), 109
add_volume (*in module gmsh_scripts.registry*), 109
Affine (*class in gmsh_scripts.coordinate_system.coordinate_system*), 68
AffineToAffine (*class in gmsh_scripts.transform.transform*), 99
AffineToCartesian (*class in gmsh_scripts.transform.transform*), 99
AnyAsSome (*class in gmsh_scripts.transform.transform*), 100

B

Bagging (*class in gmsh_scripts.size.size*), 86
Base (*class in gmsh_scripts.strategy.strategy*), 88
beta_cdf() (*in module gmsh_scripts.support.support*), 95
beta_function() (*in module gmsh_scripts.support.support*), 94
beta_pdf() (*in module gmsh_scripts.support.support*), 95
Block (*class in gmsh_scripts.block*), 61
Block (*class in gmsh_scripts.block.block*), 53
Block (*class in gmsh_scripts.coordinate_system.coordinate_system*), 69
Block (*class in gmsh_scripts.zone.zone*), 101
BlockToCartesian (*class in gmsh_scripts.transform.transform*), 98
Boolean (*class in gmsh_scripts.boolean.boolean*), 67
BOOLEAN_NEW2OLDS (*in module gmsh_scripts.registry*), 108
BOOLEAN_OLD2NEWS (*in module gmsh_scripts.registry*), 108
BooleanAllBlock (*class in gmsh_scripts.boolean.boolean*), 67
BooleanEdge (*class in gmsh_scripts.size.size*), 86
BooleanPoint (*class in gmsh_scripts.size.size*), 86

C

Cartesian (*class in gmsh_scripts.coordinate_system.coordinate_system*), 68
CartesianToCartesian (*class in gmsh_scripts.transform.transform*), 97
CartesianToCartesianByBlock (*class in gmsh_scripts.transform.transform*), 98
check_on_file() (*in module gmsh_scripts.support.support*), 95
coordinate_system (*gmsh_scripts.entity.Point attribute*), 76
coordinate_system (*gmsh_scripts.entity.point.Point attribute*), 73
coordinates (*gmsh_scripts.entity.Point attribute*), 76
coordinates (*gmsh_scripts.entity.point.Point attribute*), 73
CoordinateSystem (*class in gmsh_scripts.coordinate_system.coordinate_system*), 67
correct_kwargs() (*in module gmsh_scripts.registry*), 109
correct_layers() (*in module gmsh_scripts.parse.parse*), 83
Curve (*class in gmsh_scripts.entity*), 77
Curve (*class in gmsh_scripts.entity.curve*), 71
CURVE_KWARGS (*in module gmsh_scripts.registry*), 108
CURVE_LOOP_KWARGS (*in module gmsh_scripts.registry*), 108
CURVE_LOOP_TAG (*in module gmsh_scripts.registry*), 108
CURVE_STRUCTURE (*in module gmsh_scripts.registry*),

108
CURVE_TAG (*in module gmsh_scripts.registry*), 108
CurveLoop (*class in gmsh_scripts.entity*), 77
CurveLoop (*class in gmsh_scripts.entity.curve_loop*), 72
CURVES (*in module gmsh_scripts.registry*), 108
CURVES_LOOPS (*in module gmsh_scripts.registry*), 108
curves_points (*gmsh_scripts.block.Block attribute*), 63
curves_points (*gmsh_scripts.block.block.Block attribute*), 55
Cylindrical (*class in gmsh_scripts.coordinate_system.coordinate_system*), 68
CylindricalToCartesian (*class in gmsh_scripts.transform.transform*), 97

D

DataTree (*class in gmsh_scripts.support.support*), 94
DirectionByNormal (*class in gmsh_scripts.zone.zone*), 101

E

evaluate_boundary_surfaces() (*gmsh_scripts.support.support.DataTree static method*), 94
evaluate_bounds() (*gmsh_scripts.coordinate_system.coordinate_system.Path method*), 70
evaluate_global_surfaces_loops() (*gmsh_scripts.support.support.DataTree static method*), 94
evaluate_items_values() (*gmsh_scripts.block.Matrix static method*), 66
evaluate_items_values() (*gmsh_scripts.block.matrix.Matrix static method*), 60
evaluate_map() (*gmsh_scripts.size.size.Bagging method*), 86
evaluate_map() (*gmsh_scripts.size.size.BooleanEdge method*), 86
evaluate_map() (*gmsh_scripts.size.size.BooleanPoint method*), 86
evaluate_map() (*gmsh_scripts.size.size.Size method*), 86
evaluate_map() (*gmsh_scripts.zone.zone.Block method*), 101
evaluate_map() (*gmsh_scripts.zone.zone.DirectionByNormal method*), 102
evaluate_map() (*gmsh_scripts.zone.zone.Mesh method*), 102
evaluate_map() (*gmsh_scripts.zone.zone.Zone method*), 101

F

Factory (*class in gmsh_scripts.factory*), 103
FACTORY (*in module gmsh_scripts.factory*), 103
FACTORY (*in module gmsh_scripts.registry*), 107
FactoryClassError, 103
FactoryKeyError, 103
FactoryValueError, 103
Fast (*class in gmsh_scripts.strategy.strategy*), 88
flatten() (*in module gmsh_scripts.support.support*), 93

G

get_boolean_new2olds() (*in module gmsh_scripts.registry*), 110
get_boolean_old2news() (*in module gmsh_scripts.registry*), 110
get_curve_structure() (*in module gmsh_scripts.registry*), 110
get_layers_curves() (*gmsh_scripts.block.Layer static method*), 65
get_layers_curves() (*gmsh_scripts.block.layer.Layer static method*), 58
get_local_coordinate_system() (*gmsh_scripts.coordinate_system.coordinate_system.Path method*), 70
get_structure_type() (*gmsh_scripts.block.Layer static method*), 66
get_structure_type() (*gmsh_scripts.block.layer.Layer static method*), 59
get_surface_quadrate() (*in module gmsh_scripts.registry*), 110
get_surface_structure() (*in module gmsh_scripts.registry*), 110
get_unregistered_volumes() (*in module gmsh_scripts.registry*), 109
get_value_derivative_orientation() (*gmsh_scripts.coordinate_system.coordinate_system.Path method*), 70
get_volume2block() (*in module gmsh_scripts.registry*), 110
get_volume_structure() (*in module gmsh_scripts.registry*), 110
gmsh_scripts
    module, 53
gmsh_scripts.block
    module, 53
gmsh_scripts.block.block
    module, 53
gmsh_scripts.block.layer
    module, 58
gmsh_scripts.block.matrix
    module, 59
gmsh_scripts.block.polyhedron
    module, 60
gmsh_scripts.boolean

module, 66
gmsh_scripts.boolean.boolean
module, 66
gmsh_scripts.coordinate_system
module, 67
gmsh_scripts.coordinate_system.coordinate_system
module, 67
gmsh_scripts.entity
module, 71
gmsh_scripts.entity.curve
module, 71
gmsh_scripts.entity.curve_loop
module, 72
gmsh_scripts.entity.point
module, 72
gmsh_scripts.entity.surface
module, 74
gmsh_scripts.entity.surface_loop
module, 74
gmsh_scripts.entity.volume
module, 75
gmsh_scripts.factory
module, 103
gmsh_scripts.load
module, 103
gmsh_scripts.optimize
module, 77
gmsh_scripts.optimize.optimize
module, 77
gmsh_scripts.parse
module, 79
gmsh_scripts.parse.parse
module, 79
gmsh_scripts.plot
module, 104
gmsh_scripts.quadrate
module, 84
gmsh_scripts.quadrate.quadrate
module, 84
gmsh_scripts.refine
module, 85
gmsh_scripts.refine.refine
module, 85
gmsh_scripts.registry
module, 104
gmsh_scripts.run
module, 110
gmsh_scripts.size
module, 86
gmsh_scripts.size.size
module, 86
gmsh_scripts.smooth
module, 87
gmsh_scripts.smooth.smooth
module, 87
gmsh_scripts.strategy
module, 88
gmsh_scripts.strategy.strategy
module, 88
gmsh_scripts.structure
module, 89
gmsh_scripts.structure.structure
module, 89
gmsh_scripts.support
module, 90
gmsh_scripts.support.support
module, 90
gmsh_scripts.transform
module, 96
gmsh_scripts.transform.transform
module, 96
gmsh_scripts.utils
module, 100
gmsh_scripts.utils.obj2gmsh
module, 100
gmsh_scripts.zone
module, 101
gmsh_scripts.zone.zone
module, 101
GmshDecorator (*class in gmsh_scripts.support.support*), 91
GmshOptionsDecorator (*class in gmsh_scripts.support.support*), 91

H

Hexahedral (*class in gmsh_scripts.coordinate_system.coordinate_system*), 69

I

init_walk() (*in module gmsh_scripts.run*), 110

K

kwargs (*gmsh_scripts.entity.Point attribute*), 76
kwargs (*gmsh_scripts.entity.point.Point attribute*), 73

L

Layer (*class in gmsh_scripts.block*), 65
Layer (*class in gmsh_scripts.block.layer*), 58
Layer (*class in gmsh_scripts.coordinate_system.coordinate_system*), 70
LayerToCartesian (*class in gmsh_scripts.transform.transform*), 99
load() (*in module gmsh_scripts.load*), 103
LoggingDecorator (*class in gmsh_scripts.support.support*), 91

M

main() (*in module gmsh_scripts.plot*), 104

main() (*in module gmsh_scripts.run*), 110
main() (*in module gmsh_scripts.utils.obj2gmsh*), 101
make_tree() (*gmsh_scripts.block.Block method*), 65
make_tree() (*gmsh_scripts.block.block.Block method*), 57
Matrix (*class in gmsh_scripts.block*), 66
Matrix (*class in gmsh_scripts.block.matrix*), 59
Mesh (*class in gmsh_scripts.zone.zone*), 102
module
    gmsh_scripts, 53
    gmsh_scripts.block, 53
    gmsh_scripts.block.block, 53
    gmsh_scripts.block.layer, 58
    gmsh_scripts.block.matrix, 59
    gmsh_scripts.block.polyhedron, 60
    gmsh_scripts.boolean, 66
    gmsh_scripts.boolean.boolean, 66
    gmsh_scripts.coordinate_system, 67
    gmsh_scripts.coordinate_system.coordinate_system, 67
    gmsh_scripts.entity, 71
    gmsh_scripts.entity.curve, 71
    gmsh_scripts.entity.curve_loop, 72
    gmsh_scripts.entity.point, 72
    gmsh_scripts.entity.surface, 74
    gmsh_scripts.entity.surface_loop, 74
    gmsh_scripts.entity.volume, 75
    gmsh_scripts.factory, 103
    gmsh_scripts.load, 103
    gmsh_scripts.optimize, 77
    gmsh_scripts.optimize.optimize, 77
    gmsh_scripts.parse, 79
    gmsh_scripts.parse.parse, 79
    gmsh_scripts.plot, 104
    gmsh_scripts.quadrate, 84
    gmsh_scripts.quadrate.quadrate, 84
    gmsh_scripts.refine, 85
    gmsh_scripts.refine.refine, 85
    gmsh_scripts.registry, 104
    gmsh_scripts.run, 110
    gmsh_scripts.size, 86
    gmsh_scripts.size.size, 86
    gmsh_scripts.smooth, 87
    gmsh_scripts.smooth.smooth, 87
    gmsh_scripts.strategy, 88
    gmsh_scripts.strategy.strategy, 88
    gmsh_scripts.structure, 89
    gmsh_scripts.structure.structure, 89
    gmsh_scripts.support, 90
    gmsh_scripts.support.support, 90
    gmsh_scripts.transform, 96
    gmsh_scripts.transform.transform, 96
    gmsh_scripts.utils, 100
    gmsh_scripts.utils.obj2gmsh, 100
    gmsh_scripts.zone, 101
    gmsh_scripts.zone.zone, 101

N

n (*in module gmsh_scripts.optimize.optimize*), 78
name2kwargs (*in module gmsh_scripts.registry*), 109
NoBoolean (*class in gmsh_scripts.boolean.boolean*), 67
NoBoolean (*class in gmsh_scripts.strategy.strategy*), 88
NoOptimize (*class in gmsh_scripts.optimize.optimize*), 78
NoQuadrate (*class in gmsh_scripts.quadrate.quadrate*), 84
NoRefine (*class in gmsh_scripts.refine.refine*), 85
NoSize (*class in gmsh_scripts.size.size*), 86
NoSmooth (*class in gmsh_scripts.smooth.smooth*), 87
NoStructure (*class in gmsh_scripts.structure.structure*), 90
NoZone (*class in gmsh_scripts.zone.zone*), 101

O

Optimize (*class in gmsh_scripts.optimize.optimize*), 78
OptimizeMany (*class in gmsh_scripts.optimize.optimize*), 78
OptimizeOne (*class in gmsh_scripts.optimize.optimize*), 78

P

parse_args() (*gmsh_scripts.entity.Point static method*), 76
parse_args() (*gmsh_scripts.entity.point.Point static method*), 73
parse_arguments() (*in module gmsh_scripts.run*), 110
parse_coordinate_system() (*gmsh_scripts.entity.Point static method*), 76
parse_coordinate_system() (*gmsh_scripts.entity.point.Point static method*), 73
parse_coordinates() (*gmsh_scripts.entity.Point static method*), 76
parse_coordinates() (*gmsh_scripts.entity.point.Point static method*), 73
parse_curves() (*gmsh_scripts.block.Block static method*), 63
parse_curves() (*gmsh_scripts.block.block.Block static method*), 55
parse_curves() (*gmsh_scripts.block.polyhedron.Polyhedron static method*), 60
parse_do_quadrate() (*gmsh_scripts.block.Block static method*), 63
parse_do_quadrate() (*gmsh_scripts.block.block.Block static method*), 55

parse_do_quadrate() (*gmsh_scripts.block.polyhedron.Polyhedron static method*), 60
parse_grid() (*in module gmsh_scripts.parse.parse*), 81
parse_layers() (*in module gmsh_scripts.parse.parse*), 84
parse_layers2grid() (*in module gmsh_scripts.parse.parse*), 81
parse_layers_block_map() (*gmsh_scripts.block.Layer static method*), 65
parse_layers_block_map() (*gmsh_scripts.block.layer.Layer static method*), 58
parse_layers_block_mask() (*gmsh_scripts.block.Layer static method*), 65
parse_layers_block_mask() (*gmsh_scripts.block.layer.Layer static method*), 58
parse_layers_map() (*gmsh_scripts.block.Layer static method*), 65
parse_layers_map() (*gmsh_scripts.block.layer.Layer static method*), 58
parse_matrix_items_map() (*gmsh_scripts.block.Matrix static method*), 66
parse_matrix_items_map() (*gmsh_scripts.block.matrix.Matrix static method*), 60
parse_orientations() (*gmsh_scripts.coordinate_system.coordinate_system.Path method*), 70
parse_points() (*gmsh_scripts.block.Block static method*), 63
parse_points() (*gmsh_scripts.block.block.Block static method*), 55
parse_points() (*gmsh_scripts.block.polyhedron.Polyhedron static method*), 60
parse_points() (*gmsh_scripts.entity.Point static method*), 76
parse_points() (*gmsh_scripts.entity.point.Point static method*), 73
parse_polygons() (*gmsh_scripts.block.polyhedron.Polyhedron static method*), 60
parse_row() (*in module gmsh_scripts.parse.parse*), 79
parse_row_item() (*in module gmsh_scripts.parse.parse*), 80
parse_row_item_coordinate() (*in module gmsh_scripts.parse.parse*), 80
parse_row_item_mesh_size() (*in module gmsh_scripts.parse.parse*), 80
parse_row_item_structure() (*in module gmsh_scripts.parse.parse*), 80
parse_structure() (*gmsh_scripts.block.Block static method*), 63
parse_structure() (*gmsh_scripts.block.block.Block static method*), 55
parse_structure() (*gmsh_scripts.block.polyhedron.Polyhedron static method*), 60
parse_structure_type() (*gmsh_scripts.block.Block static method*), 63
parse_structure_type() (*gmsh_scripts.block.block.Block static method*), 56
parse_surfaces() (*gmsh_scripts.block.Block static method*), 63
parse_surfaces() (*gmsh_scripts.block.block.Block static method*), 55
parse_surfaces() (*gmsh_scripts.block.polyhedron.Polyhedron static method*), 60
parse_transforms() (*gmsh_scripts.block.Block static method*), 63
parse_transforms() (*gmsh_scripts.block.block.Block static method*), 55
parse_volumes() (*gmsh_scripts.block.Block static method*), 63
parse_volumes() (*gmsh_scripts.block.block.Block static method*), 55
parse_volumes() (*gmsh_scripts.block.polyhedron.Polyhedron static method*), 60
parse_zone() (*gmsh_scripts.block.Block static method*), 63
parse_zone() (*gmsh_scripts.block.block.Block static method*), 55
parser (*in module gmsh_scripts.utils.obj2gmsh*), 101
Path (*class in gmsh_scripts.coordinate_system.coordinate_system*), 69
PathToCartesian (*class in gmsh_scripts.transform.transform*), 99
plot_action() (*in module gmsh_scripts.plot*), 104
plot_quality() (*in module gmsh_scripts.support.support*), 94
plot_statistics() (*in module gmsh_scripts.support.support*), 94
Point (*class in gmsh_scripts.entity*), 75
Point (*class in gmsh_scripts.entity.point*), 72
POINT_KWARGS (*in module gmsh_scripts.registry*), 108
POINT_TAG (*in module gmsh_scripts.registry*), 108
POINT_TOL (*in module gmsh_scripts.registry*), 108
POINTS (*in module gmsh_scripts.registry*), 108
Polyhedron (*class in gmsh_scripts.block.polyhedron*), 60
pre_unregister() (*gmsh_scripts.block.Block method*), 65
pre_unregister() (*gmsh_scripts.block.block.Block method*), 57
pre_unregister_volume() (*in module*

gmsh_scripts.registry), 109
ps_dt_to_cs (*gmsh_scripts.support.support.DataTree attribute*), 94

Q

Quadrate (*class in gmsh_scripts.quadrate.quadrate*), 84
QuadrateBlock (*class in gmsh_scripts.quadrate.quadrate*), 84
QUADRATED_SURFACES (*in module gmsh_scripts.registry*), 108
QuarterLayer (*class in gmsh_scripts.coordinate_system.coordinate_system*), 70
QuarterLayerToCartesian (*class in gmsh_scripts.transform.transform*), 99

R

RECOMBINE_KWARGS (*in module gmsh_scripts.registry*), 109
reduce_transforms() (*in module gmsh_scripts.transform.transform*), 97
Refine (*class in gmsh_scripts.refine.refine*), 85
RefineBySplit (*class in gmsh_scripts.refine.refine*), 85
register() (*gmsh_scripts.block.Block method*), 64
register() (*gmsh_scripts.block.block.Block method*), 57
register() (*gmsh_scripts.coordinate_system.coordinate_system.Path method*), 70
register_boolean_new2olds() (*in module gmsh_scripts.registry*), 110
register_boolean_old2news() (*in module gmsh_scripts.registry*), 110
register_curve() (*in module gmsh_scripts.registry*), 109
register_curve_loop() (*in module gmsh_scripts.registry*), 109
register_curve_points() (*gmsh_scripts.block.Block method*), 64
register_curve_points() (*gmsh_scripts.block.block.Block method*), 57
register_curve_structure() (*in module gmsh_scripts.registry*), 110
register_curves() (*gmsh_scripts.block.Block method*), 64
register_curves() (*gmsh_scripts.block.block.Block method*), 57
register_curves_loops() (*gmsh_scripts.block.Block method*), 64
register_curves_loops() (*gmsh_scripts.block.block.Block method*), 57
register_curves_loops() (*gmsh_scripts.block.polyhedron.Polyhedron method*), 60
register_point() (*in module gmsh_scripts.registry*), 109
register_points() (*gmsh_scripts.block.Block method*), 64
register_points() (*gmsh_scripts.block.block.Block method*), 57
register_quadrate() (*gmsh_scripts.block.Block method*), 65
register_quadrate() (*gmsh_scripts.block.block.Block method*), 57
register_quadrate_surface() (*in module gmsh_scripts.registry*), 109
register_structure() (*gmsh_scripts.block.Block method*), 65
register_structure() (*gmsh_scripts.block.block.Block method*), 57
register_structure_curve() (*in module gmsh_scripts.registry*), 109
register_structure_surface() (*in module gmsh_scripts.registry*), 109
register_structure_volume() (*in module gmsh_scripts.registry*), 109
register_surface() (*in module gmsh_scripts.registry*), 109
register_surface_loop() (*in module gmsh_scripts.registry*), 109
register_surface_quadrate() (*in module gmsh_scripts.registry*), 110
register_surface_structure() (*in module gmsh_scripts.registry*), 110
register_surfaces() (*gmsh_scripts.block.Block method*), 64
register_surfaces() (*gmsh_scripts.block.block.Block method*), 57
register_surfaces() (*gmsh_scripts.block.polyhedron.Polyhedron method*), 60
register_surfaces_loops() (*gmsh_scripts.block.Block method*), 64
register_surfaces_loops() (*gmsh_scripts.block.block.Block method*), 57
register_volume() (*in module gmsh_scripts.registry*), 109
register_volume2block() (*in module gmsh_scripts.registry*), 110
register_volume_structure() (*in module gmsh_scripts.registry*), 110
register_volumes() (*gmsh_scripts.block.Block method*), 65

register_volumes() (*gmsh_scripts.block.block.Block method*), 57
reset() (*in module gmsh_scripts.registry*), 109
Rotate (*class in gmsh_scripts.transform.transform*), 97
run() (*in module gmsh_scripts.run*), 110

S

set_parent() (*in module gmsh_scripts.run*), 110
Size (*class in gmsh_scripts.size.size*), 86
Smooth (*class in gmsh_scripts.smooth.smooth*), 87
SmoothByDim (*class in gmsh_scripts.smooth.smooth*), 87
Spherical (*class in gmsh_scripts.coordinate_system.coordinate_system*), 68
SphericalToCartesian (*class in gmsh_scripts.transform.transform*), 98
str2obj (*in module gmsh_scripts.block.block*), 57
str2obj (*in module gmsh_scripts.block.layer*), 59
str2obj (*in module gmsh_scripts.block.matrix*), 60
str2obj (*in module gmsh_scripts.block.polyhedron*), 60
str2obj (*in module gmsh_scripts.boolean.boolean*), 67
str2obj (*in module gmsh_scripts.coordinate_system.coordinate_system*), 71
str2obj (*in module gmsh_scripts.entity.curve*), 72
str2obj (*in module gmsh_scripts.entity.curve_loop*), 72
str2obj (*in module gmsh_scripts.entity.point*), 74
str2obj (*in module gmsh_scripts.entity.surface*), 74
str2obj (*in module gmsh_scripts.entity.surface_loop*), 75
str2obj (*in module gmsh_scripts.entity.volume*), 75
str2obj (*in module gmsh_scripts.optimize.optimize*), 78
str2obj (*in module gmsh_scripts.quadrate.quadrate*), 85
str2obj (*in module gmsh_scripts.refine.refine*), 85
str2obj (*in module gmsh_scripts.size.size*), 86
str2obj (*in module gmsh_scripts.smooth.smooth*), 87
str2obj (*in module gmsh_scripts.strategy.strategy*), 89
str2obj (*in module gmsh_scripts.structure.structure*), 90
str2obj (*in module gmsh_scripts.transform.transform*), 100
str2obj (*in module gmsh_scripts.zone.zone*), 102
Strategy (*class in gmsh_scripts.strategy.strategy*), 88
Structure (*class in gmsh_scripts.structure.structure*), 90
StructureAuto (*class in gmsh_scripts.structure.structure*), 90
StructureBlock (*class in gmsh_scripts.structure.structure*), 90
STRUCTURED_CURVES (*in module gmsh_scripts.registry*), 108
STRUCTURED_SURFACES (*in module gmsh_scripts.registry*), 108
STRUCTURED_VOLUMES (*in module gmsh_scripts.registry*), 108
Surface (*class in gmsh_scripts.entity*), 77
Surface (*class in gmsh_scripts.entity.surface*), 74
SURFACE_KWARGS (*in module gmsh_scripts.registry*), 108
SURFACE_LOOP_KWARGS (*in module gmsh_scripts.registry*), 108
SURFACE_LOOP_TAG (*in module gmsh_scripts.registry*), 108
SURFACE_QUADRATE (*in module gmsh_scripts.registry*), 108
SURFACE_STRUCTURE (*in module gmsh_scripts.registry*), 108
SURFACE_TAG (*in module gmsh_scripts.registry*), 108
SurfaceLoop (*class in gmsh_scripts.entity*), 77
SurfaceLoop (*class in gmsh_scripts.entity.surface_loop*), 75
SURFACES (*in module gmsh_scripts.registry*), 108
surfaces_curves (*gmsh_scripts.block.Block attribute*), 63
surfaces_curves (*gmsh_scripts.block.block.Block attribute*), 55
surfaces_curves_signs (*gmsh_scripts.block.Block attribute*), 63
surfaces_curves_signs (*gmsh_scripts.block.block.Block attribute*), 55
SURFACES_LOOPS (*in module gmsh_scripts.registry*), 108
synchronize() (*in module gmsh_scripts.registry*), 110

T

tag (*gmsh_scripts.entity.Point attribute*), 76
tag (*gmsh_scripts.entity.point.Point attribute*), 73
timeit() (*in module gmsh_scripts.support.support*), 94
Tokamak (*class in gmsh_scripts.coordinate_system.coordinate_system*), 68
TokamakToCartesian (*class in gmsh_scripts.transform.transform*), 98
Toroidal (*class in gmsh_scripts.coordinate_system.coordinate_system*), 68
ToroidalToCartesian (*class in gmsh_scripts.transform.transform*), 98
TRANSFINITE_CURVE_KWARGS (*in module gmsh_scripts.registry*), 109
TRANSFINITE_CURVE_TYPES (*in module gmsh_scripts.registry*), 109
TRANSFINITE_SURFACE_KWARGS (*in module gmsh_scripts.registry*), 109
TRANSFINITE_VOLUME_KWARGS (*in module gmsh_scripts.registry*), 109
Transform (*class in gmsh_scripts.transform.transform*), 97
transform() (*gmsh_scripts.block.Block method*), 64
transform() (*gmsh_scripts.block.block.Block method*), 57
transform() (*gmsh_scripts.coordinate_system.coordinate_system.Path method*), 70

TransformationMatrix (*class in gmsh_scripts.transform.transform*), 100
Translate (*class in gmsh_scripts.transform.transform*), 97

U

unregister() (*gmsh_scripts.block.Block method*), 65
unregister() (*gmsh_scripts.block.block.Block method*), 57
unregister_volumes() (*in module gmsh_scripts.registry*), 109
UNREGISTERED_VOLUMES (*in module gmsh_scripts.registry*), 108
update_coordinate() (*gmsh_scripts.transform.transform.LayerToCartesian static method*), 99
update_coordinate() (*gmsh_scripts.transform.transform.QuarterLayerToCartesian static method*), 100
USE_REGISTRY_TAG (*in module gmsh_scripts.registry*), 108

V

Volume (*class in gmsh_scripts.entity*), 77
Volume (*class in gmsh_scripts.entity.volume*), 75
VOLUME2BLOCK (*in module gmsh_scripts.registry*), 108
VOLUME_KWARGS (*in module gmsh_scripts.registry*), 108
VOLUME_STRUCTURE (*in module gmsh_scripts.registry*), 108
VOLUME_TAG (*in module gmsh_scripts.registry*), 108
VOLUMES (*in module gmsh_scripts.registry*), 108
volumes_surfaces_to_volumes_groups_surfaces() (*in module gmsh_scripts.support.support*), 95
vs_dt (*gmsh_scripts.support.support.DataTree attribute*), 94
vs_ss_cs_dt (*gmsh_scripts.support.support.DataTree attribute*), 94
vs_ss_cs_ps_dt (*gmsh_scripts.support.support.DataTree attribute*), 94
vs_ss_dt (*gmsh_scripts.support.support.DataTree attribute*), 94

Z

Zone (*class in gmsh_scripts.zone.zone*), 101
zone (*gmsh_scripts.entity.Point attribute*), 76
zone (*gmsh_scripts.entity.point.Point attribute*), 73