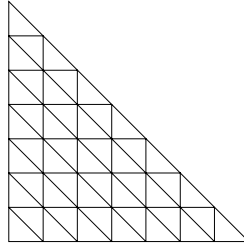
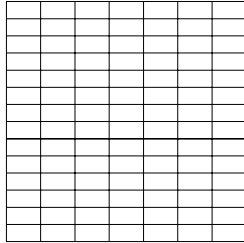


Unstructured Block Grid Code



1 Elements

```
enum element_typ { corner_el, edge_el,  quadrangle_el, triangle_el, hexahedron_el, ... };

class Point {
    double_3D coordinate;
};

class Edge {
    // information about element
    double_3D (transform*)(double); // transformation of edge

    int id_corner_W;
    int id_corner_E;

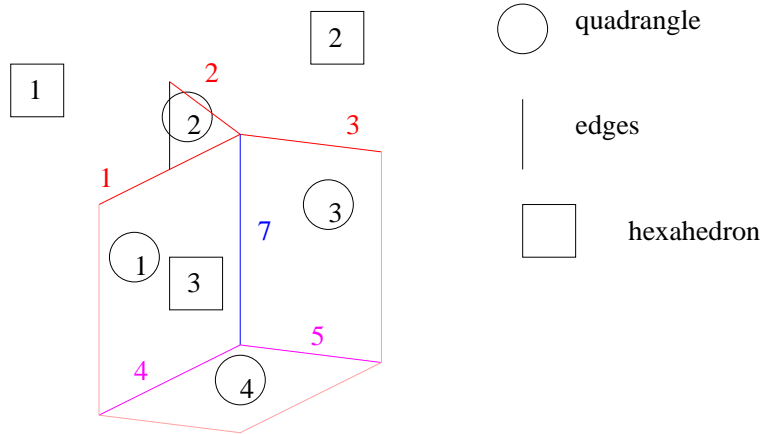
    // information about neighbour
    int number_neighbors; //

    map (element_typ, id_element) // quadrangle_el, hexahedron_el
                                     // triangle_el
                                     // 0 \leq num \leq number_neighbors

    int id_M_el[number_neighbors]; // id number of corresponding middle el
    int id_S_el[number_neighbors]; // id number of corresponding south el
    int id_N_el[number_neighbors]; // id number of corresponding north el

};
```

Example 1 *The edge in the following picture*



has the neighbours

hexahedron_el, quadrangle_el, hexahedron_el, quadrangle_el,
hexahedron_el, quadrangle_el

These neighbours have the id_M_el id number's:

3, 1, 1, 2, 2, 3

The id_S_el id number's are:

4, 4, ?, ?, ?, 5

and the id_N_el id number's are:

?, 1, ?, 2, ?, 3

Algorithm for setting these data:

- Iterate all volumne elements and
- k

```
class Quadrangle {
    // information about element
    int id_corner_SW;
    int id_corner_SE;
```

```

    int id_corner_NW;
    int id_corner_NE;

    int id_edge_S;
    int id_edge_N;
    int id_edge_E;
    int id_edge_W;

    // information about neighbour
    int number_neighbors; // 1 or 2
    int id_neighbor_hexahedron[2];
};

class Hexahedron {
    int id_corners[8];
    int id_corners_periodic[8];

    int id_faces[6];
    int id_edges[12];
};

```

2 Unstructured Grid

The unstructured grid is described by:

1. an array of grid points with 3D coordinates (id number)
2. an array of edges with 2 corners and transformation mapping (id number)
3. an array of volume elements (hexahedron, (prism, ...)) with corners (8, (6, ...)).
(do consistency check for edges of elements)

```

class Unstructured_grid {
public:
    Unstructured_grid(); // first puts num_poi and ... zero

```

```

void set_number_points(int num_poi, int num_poi_plus_periodic);

void set_coordinate_point(int i);
void set_coordinate_point_periodic(int i, int i_per);
                                // i_per must be larger than num_poi

void set_hexahedron(int i_SW, int i_SE, ...);
void set_hexahedron_periodic(int i_SW, int i_SE, ... ,
                             int i_SW_periodic, int i_SE_periodic, ...);

void construction_hexahedron_points_done();

void set_transformation_edge(int i, int j, double_3D (transform*)(double));
                                // ‘transform’ must be zero at 0,1
void set_transformation_face(int i, int j, double_3D (transform*)(double, d
                                // ‘transform’ must be zero at (0,0), ...

void construction_done();

private:
    bool construction_hexahedron_points_done;
    bool construction_done;

    bool periodic;    // is num_poi == num_poi_plus_periodic

    int num_poi;
    int num_poi_plus_periodic;
    int num_hexahedra;

    Hexahedron hexahedra[num_poi_hexahedra];
    Point      points[num_poi];
};

class Cylinder : public Unstructured_grid {
    Cylinder(double radius, double length);

```

};

Additional informations:

1. Volume:

3 Block Data Structure

The orientation of the data structure of every block depends on the id number of corners and edges

1. Edge: from corner with low id-number to corner with high id-number
2. Face: Corner with lowest id-number is SW point.

Now consider the two adjacent edges.

x-axis is edge with lower id-number.

y-axis is edge with high id-number.

3. Volume: Lowest id-number is SWD point.

Now consider the three adjacent edges.

x-axis is edge with lowest id-number.

y-axis is edge with middle id-number.

z-axis is edge with highest id-number.

The discretization of the unstructured grid is based on numbers

$$n_1, \dots, n_d,$$

where d is the degree of freedom.

3.1 Variable_Point

1. Hexahedron: array of length $(n_x + 1) * (n_y + 1) * (n_z + 1)$.
2. Quadrangle: array of length $(n_x + 1) * (n_y + 1)$.

and

array of length $(n_x + 1) * (n_y + 1) * \text{number_neighbors}$

3. Edge: array of length $n_x + 1$.
and
array of length $(n_x + 1) * \text{number_neighbors}$
4. Corner: array of length 1.
and
array of length `number_neighbors`

3.2 Copying Data