4 Using T-Rex

Usage of T-Rex can be summarised in the following steps

- Generation of the computational grid with Generator,
- (optional) partitioning the grid with Divisor if the user wants to run the program in parallel
- Running the CFD simulation with Processor,
- (optional) connecting the result files from different processors (if Processor) was run in parallel,
- visualisation of results.

4.1 Using Generator

Generator is used to divide the problem domain into a finite number of cells, or control volumes. As an input it takes a special ASCII file, so called the *domain file*, which gives the instructions to Generator how to generate the mesh. The domain file must be named as <NAME>.d where <NAME> is any string allowed by the operating system, whereas .d is obligatory extension. As the output, Generator gives the following:

- <NAME>.inp - ASCII format; grid in AVS data format, used for visualisation and optionally later partitioning of the grid,
- <NAME>.gmv - ASCII format; grid in GMV data format, used only for later visualisation,
- <NAME>.cns - binary format; topology of the grid and boundary condition markers,
- <NAME>.geo - binary format; geometrical quantities: cell volumes, cell centre coordinates etc.
- <NAME>.x.eps, <NAME>.y.eps, <NAME>.z.eps Encapsulated Postscript cuts through the domain for quick visualisation of the generated grid,
- <NAME>.eps, Encapsulated Postscript showing the shaded 3D view of the grid. quick visualisation of the generated grid.

4.1.1 Running the Generator

Compilation of Generator is explained in section **??**. When the executable file for Generator is created, the user has to go to the directory where the domain file resides and run it. When the Generator starts, it writes it's logo on the screen and prompts the user with the following question:

```
Input problem name:
```

At this point user has to enter the name of the domain file (<NAME>) *without* extension. Generator will continue to work and will write some messages on the standard output. When it finishes the generation of the grid, it will prompt the user with following questions:

```
Enter the x coordinate for cutting:
Enter the y coordinate for cutting:
Enter the z coordinate for cutting:
```

The user has to enter the position of cutting planes in x,y and z directions to each of these questions. Those planes are used just for creation of Encapsulated Postscript cuts of the domain. Finally, the Generator will prompt the user with the question:

```
Enter the x, y and z coordinate of the camera:
```

To that question user has to enter three numbers which are the x, y and z coordinates of the viewpoint (camera). This, camera position, will be used in the creation of the 3D figure in Encapsulated Postscript format. Note that the camera coordinate is also the source of light and be careful to place the camera outside the domain.

4.1.2 Creating the domain file

Domain file consists of following sections, and all of them (except the comments) must be present in the file:

1. Memory requirements - defines maximal number of nodes (and cells) [3], boundary cells and cell faces.
2. Points - defines number of points and their x, y and z coordinates
3. Blocks - defines hexahedronal blocks. Each block is defined with 8 points from the above list of points. It also defines the resolution of each block and it's weight in local i, j and k direction respectively.
4. Lines - defines distribution of points on the lines weather by explicitly defining coordinates of each point on th line, or by setting the weight of the line.
5. Surfaces - defines weight of each surface.
6. Boundary conditions - defines boundary markers for later insertion of boundary conditions.

[3] For xexahedronal cells, then number of cells is just slightly smaller then the number of nodes. Therefore the memory for nodes and cells is allocated with a single constnt.

7. Periodic boundaries - defines periodic boundaries.
8. Copy boundaries - defines copy boundaries.
9. Refinement - defines regions of local refinement by cell splitting. It can define rectangular, ellipsoidal region or region defined by a plane.
10. Smoothing - defines the region for Laplacian smoothing of the domain.

- Comments - each line which begins with a character # or % is regarded as a comment line and skipped by the Generator. Comment lines can appear anywhere in the input file.

The templated format of the domain file follows. The numbers on the left corresponds to the above descriptions, and are not part of the input file. Entities in this <> brackets are variables, whereas entities in this >< brackets denote keywords. A list of allowed keywords is enclosed in this [,,] brackets and is separated by commas. Entities entered without any brackets are obligatory, and cannot be altered. The same nomenclature rules apply as well to description of all the other file formats in this manual.

```
1. <max_nodes>  <max_boundary_cells>  <max_cell_faces>
2. <num_points>
        <point_id 1>  <x>  <y>  <z>
           .
           .
           .
        <point_id num_points>  <x>  <y>  <z>
3. <num_blocks>
        <block_id 1>   <num_i>  <num_j>  <num_k>
           <weigth_i>  <weigth_j>  <weigth_k>
           <block_point 1> ... <block_point 8>
           .
           .
           .
        <block_id num_blocks>   <num_i>  <num_j>  <num_k>
           <weigth_i>  <weigth_j>  <weigth_k>
           <block_point 1> ... <block_point 8>
4. <num_lines>
        <line_id 1>  <line_point 1>  <line_point 2>
           <node_id 1>  <x>  <y>  <z>
           .
           .
           .
           <node_id num_nodes>  <x>  <y>  <z>
         [OR]
       -<line_id 1>  <line_point 1>  <line_point 2>
           <weigth>
           .
           .
           .
```

```
        <line_id num_lines>  <line_point 1>  <line_point 2>
           <node_id 1>  <x>  <y>  <z>
           .
           .
           .
           <node_id num_nodes>  <x>  <y>  <z>
         [OR]
       -<line_id num_lines>  <line_point 1>  <line_point 2>
           <weigth>
5. <num_surfaces>
        <surface_id 1>  <surf_point 1> ... <surf_point 4>
           <weigth_i>  <weigth_j>  <weigth_k>
           .
           .
           .
        <surface_id num_surfaces>  <surf_point 1> ... <surf_point 4>
           <weigth_i>  <weigth_j>  <weigth_k>
6. <num_bc>
        <bc_id 1> <i_start> <j_start> <k_start> <i_end> <j_end> <k_end>
           <block_id>  <marker>
        [OR]
        <bc_id 1>  >position<  [IMIN,IMAX,JMIN,JMAX,KMIN,KMAX]
           <block_id>  <marker>
           .
           .
           .
        <bc_id num_bc> <i_start> <j_start> <k_start> <i_end> <j_end> <k_end>
           <block_id>  <marker>
        [OR]
        <bc_id num_bc> >position< [IMIN,IMAX,JMIN,JMAX,KMIN,KMAX]
           <block_id>  <marker>
7. <num_per>
        <per_id 1>  <point_surf_1 1> ... <point_surf_1 4>
                    <point_surf_2 1> ... <point_surf_2 4>
           .
           .
           .
        <per_id num_per>  <point_surf_1 1> ... <point_surf_1 4>
                          <point_surf_2 1> ... <point_surf_2 4>
8. <num_copy>
        <copy_id 1>  <point_surf_1 1> ... <point_surf_1 4>
                     <point_surf_2 1> ... <point_surf_2 4>
           .
           .
           .
        <copy_id num_per>  <point_surf_1 1> ... <point_surf_1 4>
                           <point_surf_2 1> ... <point_surf_2 4>
9. <num_levels>
        <level_id 1>  <num_regions>
           <region_id 1>  >shape<  [RECTANGLE,PLANE,ELIPSOID]
```

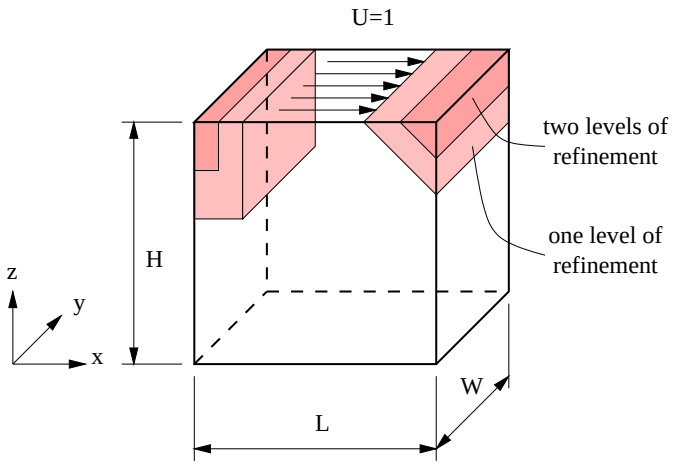

Figure 1: Problem domain for the cavity flow

```
    <x1>  <y1>  <z1>  <x2>  <y2>  <z2>
  .
  .
  .
  <region_id num_regions>  >shape<  [RECTANGLE,PLANE,ELIPSOID]
    <x1>  <y1>  <z1>  <x2>  <y2>  <z2>
<level_id num_levels>  <num_regions>
  <region_id 1>  >shape<  [RECTANGLE,PLANE,ELIPSOID]
    <x1>  <y1>  <z1>  <x2>  <y2>  <z2>
  .
  .
  .
  <region_id num_regions>  >shape<  [RECTANGLE,PLANE,ELIPSOID]
    <x1>  <y1>  <z1>  <x2>  <y2>  <z2>
```

So far, we have introduced several issues which demand further explanation. Here we mean the weights, boundary conditions, refinement levels. We believe that they are best explained with few examples which now follow.

Example 1: A cavity We begin with probably the simplest possible test case which is the lid driven cavity flow. This example illustrates the following:

- definition of a simple, single-block domain,
- definition of boundary condition markers,
- local cell refinement.

The domain of interest is shown in figure (**??**). Let's set the dimension of the domain to $L \times W \times H = 1 \times 1 \times 1$, and let's define two levels of local grid

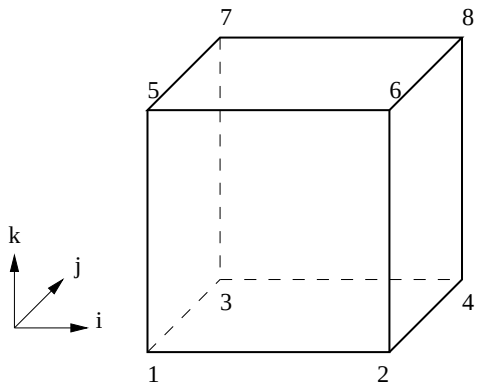

Figure 2: Problem domain for the cavity flow

refinement in the regions where high gradients of pressure can be expected. A regions with one level of refinement is emphasised in light gray and regions with two levels of refinement are emphasised with dark gray in figure (**??**). Three different types of boundary conditions are required: prescribed velocity, wall and symmetry. At this stage, however, it is enough to know that three boundary markers are needed.

Each block is defined by it's local coordinate system (i,j,k), and local points. The correspondence between them is illustrated in figure (**??**). In this case, local point numbers coincide with global numbers.

The domain file begins with section for memory allocation. Maximal number of nodes (and cells), boundary cells and cell faces has to be given. For a locally driven cavity shown in figure (**??**), we might expect 30000 nodes. A good rule of thumb says that it is necessary to reserve the one third of number of nodes for boundary cells and three times the number of nodes for cell faces. Therfore, the section for memory

```
#-----------------------------------------#
#  Nodes (cells), boundary cells and sides  #
#-----------------------------------------#
  30000 10000 90000
```

The domain file continues with section for points, which is nothing more then a list of points with their corresponding x, y and z coordinates:

```
#----------#
#  Points  #
#----------#
  8
    1   0.0   0.0   0.0
    2   1.0   0.0   0.0
    3   0.0   1.0   0.0
    4   1.0   1.0   0.0
```

```
5   0.0   0.0   1.0
6   1.0   0.0   1.0
7   0.0   1.0   1.0
8   1.0   1.0   1.0
```

This is followed by a section for blocks. In this case, only one block is defined:

```
#----------#
#  Blocks  #
#----------#
  1
    1  11   11   11
       1.0  1.0  1.0
       1  2  3  4  5  6  7  8
```

It is important to note that the initial grid will have $11 \times 11 \times 11$ nodes which means $10 \times 10 \times 10$ cells. Further, weights in all three local directions (i, j and k) are equal to one, which means that no stretching of the grid lines will be performed. In this case, we will not define anything special for lines and surfaces, so this two sections look like:

```
#---------#
#  Lines  #
#---------#
  0
#------------#
#  Surfaces  #
#------------#
  0
```

After surface section, we came to boundary conditions, which looks like:

```
#-----------------------#
#  Boundary conditions  #
#-----------------------#
  3
    1     Kmax
          1   2
    2     Jmin
          1   3
    3     Jmax
          1   3
```

This section needs special attention. Three surfaces are marked, the top surface (`KMAX`) for prescribed velocity with marker 2, and two surfaces for symmetry boundary conditions with marker 3. *All the remaining surfaces, which are not defined in this section, will have the default marker 1.* There are no periodic boundary conditions, so the section for them looks similar to one for lines and surfaces:

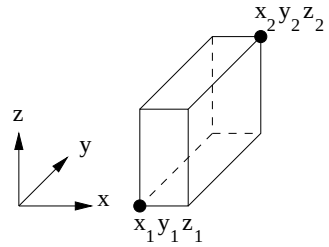

Figure 3: Definition of rectangular region for refinement

```
#-----------------------#
#  Periodic boundaries  #
#-----------------------#
  0
```

Copy boundaries are not present neither, therefore the section for copy boundaries looks like:

```
#-------------------#
#  Copy boundaries  #
#-------------------#
  0
```

Now follows the section for local cell refinement. As it was shown in figure/ (**??**), we have two levels of refinement, and each of them has two refined regions: a rectangle-shaped one in the upper right and the triangular-shaped one in the upper right corner of the domain. Rectangular regions are defined with keywords: `FILL` and `RECTANGLE` and six real numbers which represent x, y and z coordinate of the lower-left-front and x, y and z coordinates of the upper-right-back corner of the domain. A rectangular region is illustrated in figure (**??**). Triangular regions are defined with `FILL` and `PLANE` keywords and six more real numbers. First three numbers are the coordinates of the point belonging to that plane and the three remaining ones define the normal of the plane in the direction in which the refinement will take place. [4] A planar region is illustrated in figure (**??**).

```
#------------#
# Refinement #
#------------#
  2
# level 1 #
    1  2
       1 Fill Rectangle
         0.0  0.0  0.6  0.2  1.0  1.0
```

[4] Apart from `RECTANGLE` and `PLANE` a keyword `ELIPSOID` can also be used. If used it is also followed by six real numbers, where first three are the coordinates of the ellipsoid centre, and the three remaining ones are its principal axes in x, y and z direction.

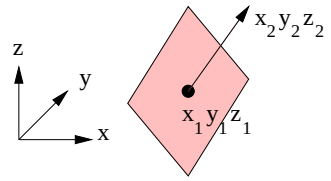

Figure 4: Definition of planar region for refinement

Figure 5: Final grid for the cavity flow

```
        2 Fill Plane
          0.75 0.75 0.75 1.0  0.0  1.0
# level 2 #
    2  2
        1 Fill Rectangle
          0.0  0.0  0.8  0.1  1.0  1.0
        2 Fill Plane
        0.85 0.85 0.85 1.0  0.0  1.0
```

The domain file finishes with smoothing section. For this domain, no smoothing is needed, so we just have to set the number of smoothing regions to 0.

```
#-----------#
# Smoothing #
#-----------#
  0
```

The final grid for this example is shown on figure (**??**). Few things have to be added at this point. Boundary markers are only integers. On the surfaces where no boundary marker is specified, a default value 1 will be set. All the keywords for the Generator, as well as for all the other principal programs are case *insensitive*.

Example 2: A dune We continue with a slightly more complex problem domain shown in figure (**??**). The purpose of this example is to illustrate the following:

- definition of a more complex, multi-block domain,
- stretching of the grid lines toward the walls,
- definition of the periodic boundary conditions.

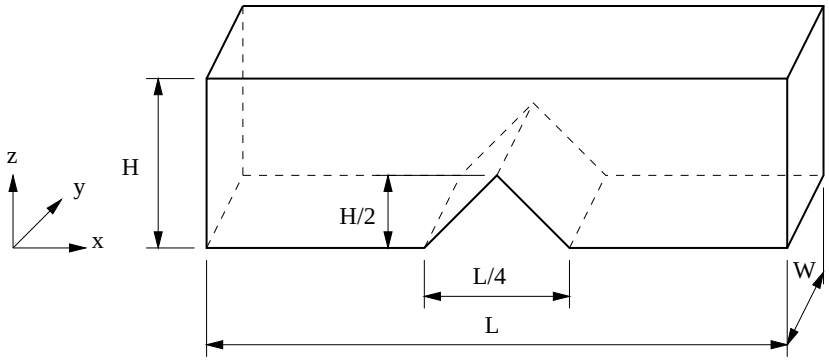

Figure 6: Problem domain for the dune flow

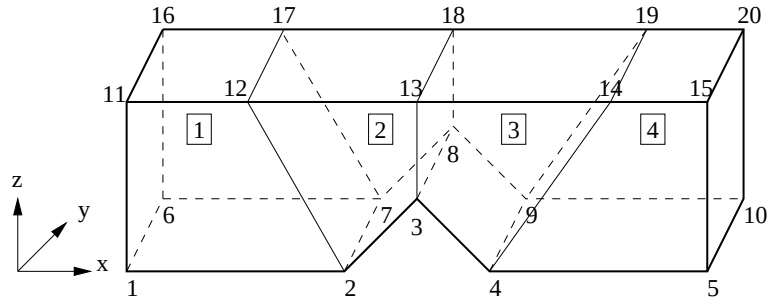

Figure 7: Global points for the dune

To generate the grid for this test case, the domain first has to be decomposed into four blocks [5]. One possible division and global point numbering is shown in figure (**??**). By setting $L = 8$ and $H = 2$, we can define the points section of the domain as following:

```
#========#
# Points #
#--------#
20

 1   0.0   0.0   0.0
 2   3.0   0.0   0.0
 3   4.0   0.0   1.0
 4   5.0   0.0   0.0
 5   8.0   0.0   0.0

 6   0.0   0.5   0.0
 7   3.0   0.5   0.0
```

[5] This grid could have been generated with a single block, but we wanted to demonstrate the procedure for a multi block grid

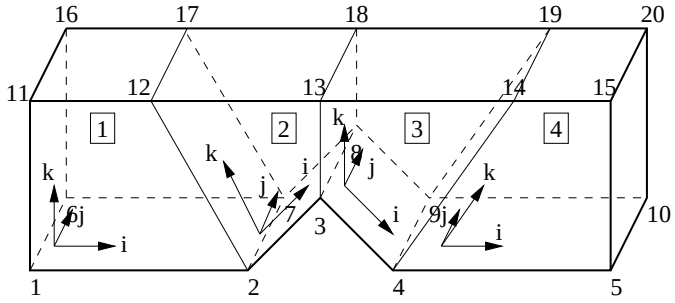

Figure 8: Local block coordinates for the dune

```
 8   4.0   0.5   1.0
 9   5.0   0.5   0.0
10   8.0   0.5   0.0

11   0.0   0.0   2.0
12   1.5   0.0   2.0
13   4.0   0.0   2.0
14   6.5   0.0   2.0
15   8.0   0.0   2.0

16   0.0   0.5   2.0
17   1.5   0.5   2.0
18   4.0   0.5   2.0
19   6.5   0.5   2.0
20   8.0   0.5   2.0
```

To continue with block definition, we must decide on the local coordinates within each block. These can be defined in any sense, but we have adopted the one shown in figure (**??**). Following the local coordinate systems within each block, the block section of the domain file looks as follows:

```
#========#
# Blocks #
#--------#
 4
  1  20  5  30
    1.00  1.00  -0.9
    1  2  6  7 11 12 16 17
  2  16  5  30
    4.00  1.00  -0.9
    2  3  7  8 12 13 17 18
  3  16  5  30
    0.25  1.00  -0.9
    3  4  8  9 13 14 18 19
  4  20  5  30
    1.00  1.00  -0.9
```

weigth:	meaning:
-0.999 to -0.751	hyperbolical stretch in both directions
-0.749 to -0.501	hyperbolical stretch to the origin of local coordinate
-0.499 to -0.251	hyperbolical stretch apart from origin of local coordinate
-0.249 to 0.000	undefined
0.001 to 0.999	linear stretching to the origin of local coordinate
1.001 to 1.000	linear stretching apart from origin of local coordinate

Table 1: Allowed weight values and their meaning

```
4  5  9 10 14 15 19  20
```

For this example domain, the block weights are not all equal to one. For example, sub-domain 2 has weigth in i direction equal to 4.0. This tells the mesh generator to linearly stretch the cells towards the end in i direction by a factor of 4. The stretching factor is the ratio of the cell sizes in the beginning and the end of the local direction. For all blocks the weigth in k direction is -0.9. Weights which are negative mean hyperbolic stretching. Further, weights in the range from -0.75 to -0.99 are for stretching toward both sides, whereas the weights from -0.25 to -0.75 are for hyperbolic stretching in one direction only. All the weigth values and their meanings are summarised in table (**??**). No lines and surfaces need explicit definition for this case, so the next two sections are as follows:

```
#---------#
#  Lines  #
#---------#
  0
#------------#
#  Surfaces  #
#------------#
  0
```

For this test case we need only one boundary marker, and that is for solid walls in the top and in bottom of the domain. These do not have to be specified, because by default they will be set to 1. Therefore, the section for boundary conditions is:

```
#---------------------#
# Boundary conditions #
# (only the default)  #
#---------------------#
  0
```

For this problem, periodic boundaries have to be specified in both stream-wise (x) and span-wise (y) direction. To define the periodic boundaries we have to identify all pairs of block surfaces which have connected and specify them by

Figure 9: Final grid for the dune flow

their points in domain file. For example, to connect the domain in stream-wise direction, we have to connect the surface 1-6-16-11 with the surface 5-10-20-15 (see figure (**??**)). To connect the domain in the span-wise direction, we have to specify four more pairs of surfaces. The section for periodic boundary conditions in the domain file, looks as following:

```
#====================#
# Periodic Boundaries #
#--------------------#
  5
    1  1  6 16 11
       5 10 20 15
    2  1  2 12 11
       6  7 17 16
    3  2  3 13 12
       7  8 18 17
    4  3  4 14 13
       8  9 19 18
    5  4  5 15 11
       9 10 20 19
```

For this example, we will not perform any local cell refinement so the next section is:

```
#============#
# Refinement #
#------------#
  0
```

The final grid for the flow around the cylinder is shown in figure (**??**).

Example 3: A cylinder Next example is the domain for the calculation of the flow around the circular cylinder. It will demonstrate the following:

- definition of a more complex shape via the line section of the input file,
- refinement of an ellipsoidal region of the domain.

The problem domain is shown in figure (**??**). If we set the dimensions of the domain to: $L = 9, H = 3, W = 3$, and if select the nodes as it is shown in figure (**??**), the point section of the input file might look as following:

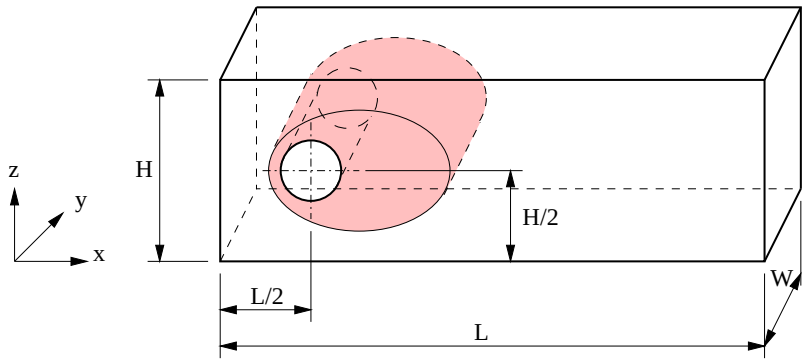

Figure 10: Problem domain for the flow around the cylinder

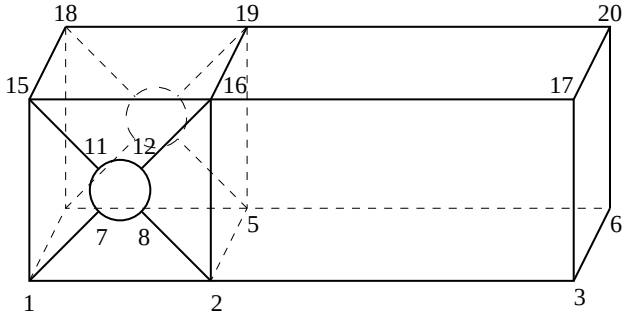

Figure 11: Problem domain for the flow around the cylinder with specified point numbers. Some point numbers are omitted for the sake of clarity.

```
%==========%
%  Points  %
%==========%
20
%-----------%
%   Floor   %
%-----------%
  1   0.0   0.0   0.0
  2   3.0   0.0   0.0
  3   9.0   0.0   0.0
  4   0.0   3.0   0.0
  5   3.0   3.0   0.0
  6   9.0   3.0   0.0
%--------------%
%   Cylinder   %
%--------------%
  7   1.25   0.0   1.25
  8   1.75   0.0   1.25
  9   1.25   3.0   1.25
 10   1.75   3.0   1.25
 11   1.25   0.0   1.75
 12   1.75   0.0   1.75
 13   1.25   3.0   1.75
 14   1.75   3.0   1.75
%----------%
%   Roof   %
%----------%
 15   0.0   0.0   3.0
 16   3.0   0.0   3.0
 17   9.0   0.0   3.0
 18   0.0   3.0   3.0
 19   3.0   3.0   3.0
 20   9.0   3.0   3.0
```

After the point section, we come to the line section, which is very important for this case, since the lines connecting the points around the cylinder (7-8-12-11 and 9-10-14-13) are not straight lines, but rather arcs. In the line section we must define 8 lines, i.e. 7-8, 8-12, 12-11, 11-8 (see figure (**??**)) and 9-10, 10-14, 14-13, 13-9 (omitted in figure (**??**)). So, the line section for this problem domain looks as:

```
%%%%%%%%%%%%%%%%%%%%
%   line section   %
%%%%%%%%%%%%%%%%%%%%
  8
*---- line connecting points 8 and 12
  1    8  12
    1    1.853554     .000000    1.146447
    2    1.880203     .000000    1.175276
    3    1.904509     .000000    1.206107
```

```
 4    1.926320     .000000    1.238751
 5    1.945503     .000000    1.273005
 6    1.961940     .000000    1.308658
 7    1.975528     .000000    1.345492
 8    1.986185     .000000    1.383277
 9    1.993844     .000000    1.421783
10    1.998459     .000000    1.460770
11    2.000000     .000000    1.500000
12    1.998459     .000000    1.539230
13    1.993844     .000000    1.578217
14    1.986185     .000000    1.616723
15    1.975528     .000000    1.654508
16    1.961940     .000000    1.691342
17    1.945503     .000000    1.726995
18    1.926320     .000000    1.761249
19    1.904508     .000000    1.793893
20    1.880203     .000000    1.824724
21    1.853553     .000000    1.853553
 .
 .
```

We have shown here just one part of the line section, just the definition of the line 8-12. There are seven more lines defined in the same way. It is clear from this example that Generator is quite cumbersome to use. Theoretically it is capable of generating any domain, but in practice, that is far from an easy task. The user is therefore strongly advised to use some more sophisticated mesh generator instead of Generator. The surface section for this case is empty, and the block section is formed in the similar way to the block section of previous example (dune). and is not shown here in order to keep this user guide short. Boundary condition and periodic boundary sections are skipped from the same reason. In the refinement region, there is a feature which didn't occur in the previous examples, namely an ellipsoidal refinement region. It is described in the refinement section of the domain file in the following way:

```
%%%%%%%%%%%%%%%%%
%   refinement    %
%%%%%%%%%%%%%%%%%
 1
    1  1
       1   FILL ELIPSOID
           2.3  1.5  1.5  1.5  1000.0  1.0
```

The number of refinement levels is one (first line after the comments). In that level, only one region is refined (second line after the comment). The ellipsoidal region is specified by the keywords: `FILL` and `RECTANGLE` which is followed by six real numbers, first three of them specifying the x, y and z coordinates of ellipsoid, and the remaining three specifying the ellipsoid axes in x, y and z directions respectively.

The final grid for the flow around the cylinder is shown in figure (**??**).