DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg
Campus Friedrichshafen

SCHRIFTENREIHE DER FAKULTÄT FÜR TECHNIK DER DUALEN HOCHSCHULE BADEN-WÜRTTEMBERG RAVENSBURG

2016/01

A Manipulation and Conversion Toolbox for POPINDA-Formatted, Elliptic Hexahedral Meshes for the Use with MegaCads and OpenFOAM

Martin Lichtmes

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg
Campus Friedrichshafen

SCHRIFTENREIHE DER FAKULTÄT FÜR TECHNIK DER DUALEN HOCHSCHULE BADEN-WÜRTTEMBERG RAVENSBURG

2016/01

A Manipulation and Conversion Toolbox for POPINDA-Formatted, Elliptic Hexahedral Meshes for the Use with MegaCads and OpenFOAM

Martin Lichtmes

IMPRESSUM

Schriftenreihe der Fakultät für Technik
der Dualen Hochschule Baden-Württemberg Ravensburg

Herausgeber

Prof. Dr. Martin Freitag
Prorektor

Duale Hochschule Baden-Württemberg Ravensburg
Baden-Wuerttemberg Cooperative State University
Marienplatz 2
88212 Ravensburg
Deutschland

http://www.dhbw-ravensburg.de

2016/01, Dezember 2016

ISBN 978-3-945557-02-0
ISSN 2199-238X
DOI 10.12903/DHBW_RV_FN_01_2016_LICHTMES

Druck und Verarbeitung

Gestaltung
Nicole Stuepp
DHBW Ravensburg
Marienplatz 2, 88212 Ravensburg

Druck
Frick Kreativbüro & Onlinedruckerei e.K.
Brühlstraße 6
86381 Krumbach

A Manipulation and Conversion Toolbox for POPINDA-Formatted, Elliptic Hexahedral Meshes for the Use with MegaCads and OpenFOAM

Martin Lichtmes[1]

ABSTRACT

A conversion and manipulation toolbox for POPINDA-formatted hexahedral meshes, as may be used in Computational Fluid Dynamics (CFD) is presented. With a focus on ongoing research activities regarding IC-engine aerodynamics carried out at the Faculty of Technology of the Baden-Württemberg Cooperative State University Ravensburg, the toolbox has mainly been developed to make elliptically smoothed hexahedral meshes conveniently available to the OpenFOAM CFD framework. The Software MegaCads is capable of producing meshes of elliptic, hexahedral type even for complex geometries. Therefore, and because of its free availability, MegaCads has been chosen as mesh design basis in the given scope. The toolbox in its current version deals with ASCII encoded POPINDA mesh files and offers several manipulation routines such as scaling, extruding etc. to use in conjunction with MegaCads as well as the intended mesh conversion mechanisms. The paper gives an overview regarding the currently implemented toolset and briefly introduces the reader especially into how to use it as mesh conversion interface between MegaCads and OpenFOAM.

Keywords: Elliptic, Hexahedral, Structured, Mesh, Grid, Manipulation, Conversion, Computational Fluid Dynamics, CFD, OpenFOAM, MegaCads, POPINDA, FLOWer

[1] Research Scientist (Fluid Dynamics), Research Division: „*Motorische Verbrennung inhomogener Gasgemische*“ (IHGG; engl.: *Internal Combustion of Inhomogeneous Gas Mixtures*), Faculty of Technology, DHBW Ravensburg

NOMENCLATURE

Symbol	SI Base Unit	Description
b		Base node index
c		Cell index
d		Cartesian direction variable
e		Extrusion length
i, j, k		Direction indices
l	$[\mathrm{mm}]*$	Edge length
n_i, n_j, n_k		Number of points in ijk-direction
N_i, N_j, N_k		Number of cells in ijk-direction
ϕ	$[°]$	Rotation/Revolution angle
s		Scaling factor
$\mathbf{t}$		Translation vector
t_x, t_y, t_z	$[\mathrm{mm}]*$	Cartesian translation components
T		Transpose
τ	$[\mathrm{mm}]*$	Absolute distance tolerance
v		Vertex node index
x, y, z		Cartesian coordinates

Abbrevations

DHBW	Baden-Württemberg Cooperative State University (Duale Hochschule Baden-Württemberg)
DLR	Deutsches Zentrum für Luft- und Raumfahrt (engl.: German Aerospace Center)
DNS	Direct Numerical Simulation
GUI	Graphic User Interface
Hex	Hexahedral
IC	Internal Combustion
IHGG	Motorische Verbrennung inhomogener Gasgemische (engl.: Internal Combustion of Inhomogeneous Gas Mixtures)
LES	Large Eddy Simulation
POPINDA	Portable Parallelisation of Industrial Aerodynamic Applications
RANS	Reynolds-Averaged Navier-Stokes
STL	Stereolithography

1 INTRODUCTION

Elliptic hexahedral meshes (or grids) are of great use in Computational Fluid Dynamics (CFD), since they are known for being eminently suited for the numerical, discrete solution of the Navier-Stokes equations which represent a nonlinear system of second order partial differential equations [1]. Besides their good numerical fitness, structured hexahedral meshes can be stored in a very simple and compact 'nested-points' (cf. chapter 3) arrangement, what makes it easy to store, manipulate or convert them in manifold ways. Whilst the open-source CFD framework OpenFOAM (*Open Field Operation and Manipulation)* is being shipped with advanced mesh generation utilities BlockMesh and SnappyHexMesh it does not offer the desirable elliptical smoothing for hexahedral meshes [2,3]. In contrast, creating this type of meshes on complex geometries is what MegaCads (*Multiblock Elliptic Grid-Generation and Computer Aided Design System*) is capable of [4,5,6]. MegaCads is freeware and comes with an intuitive GUI (Graphic User Interface). The meshes can be parametrised extensively and for experienced or at least ambitious users, there is practically no hard limitation in geometric or topological complexity of the resulting mesh. MegaCads has been developed by the DLR (*German Aerospace Centre*) and may be acquired via their website *http://www.megacads.dlr.de* [5]. For the below presented work, MegaCads has been used in version 2.5.2. At present, MegaCads is not undergoing any major development [5] but it is highly usable in its current stage[2]. It offers manifold basic geometric operations to create parametric polygons, splines, vectors, curves or surfaces as well as nonlinear (e.g. Poisson-type) point distributions on curves, surfaces and volumes. The herein presented toolbox has been given the name BLoOMYBOXX which loosely relates to the CFD code FLOWer of the DLR – within MegaCads, the POPINDA mesh format is referred to as 'FLOWer format' – that utilises POPINDA-formatted, structured meshes and to the box-like shape of hexahedral cells. POPINDA stands for *Portable Parallelisation of Industrial Aerodynamic Applications* [7].

[2] Not all periphery functionalities are fully functional or operating stable (i.e. ANSYS output, IGES conversion etc.). But to our knowledge all meshing and design features (geometric and mathematical operators) are working as intended/expected.

OpenFOAM is property of the *OpenFOAM Foundation Ltd.* and released under GNU *General Public License* [2]. Herein, OpenFOAM has been used in version 3.0.1. BLoOMYBOXX is a Linux executable written in C++ and is in its current version 0.2 released as freeware. It has been developed and tested under Ubuntu 15.10 (64 bit). To acquire a copy of BLoOMYBOXX please contact the *Baden-Württemberg Cooperative State University* (DHBW) *Ravensburg* using the author's Email address (*lichtmes@dhbw-ravensburg.de*) or visit the website *http://www.ravensburg.dhbw.de.*

2 MOTIVATION AND RELATED RESEARCH

Nowadays, semi- or fully-automatic mesh generation (e.g. triangular, tetra- or polyhedral) has become more important in general industrial CFD than the rather manual hexahedral meshing because of flexibility and performance speed (cf. figure 1). But among several known drawbacks, unstructured or hybrid (partially unstructured and structured) meshes may introduce a significant lack of computational accuracy and stability alongside higher computational efforts compared to more uniform and 'cleaner' fully-structured hexahedral (hex) meshes. Hex meshes therefore still receive great appreciation especially in applications where, for instance, high accuracy or solution smoothness is mandatory (i.e. fundamental research topics etc.). Since manually generated meshes can be controlled more precisely by the user, the numerical quality may be driven to a maximum even regarding single selected grid cells easily. In CFD such high quality meshes are not only well suited for smooth and accurate RANS (*Reynolds-Averaged Navier-Stokes*) simulations but also for LES (*Large Eddy Simulations*) and even DNS (*Direct Numerical Simulation*). Of course, manual hex mesh generation often appears to be very time consuming. One must therefore be aware whether the manual or a rather automatic (unstructured) approach is the most suitable for their intended purposes.

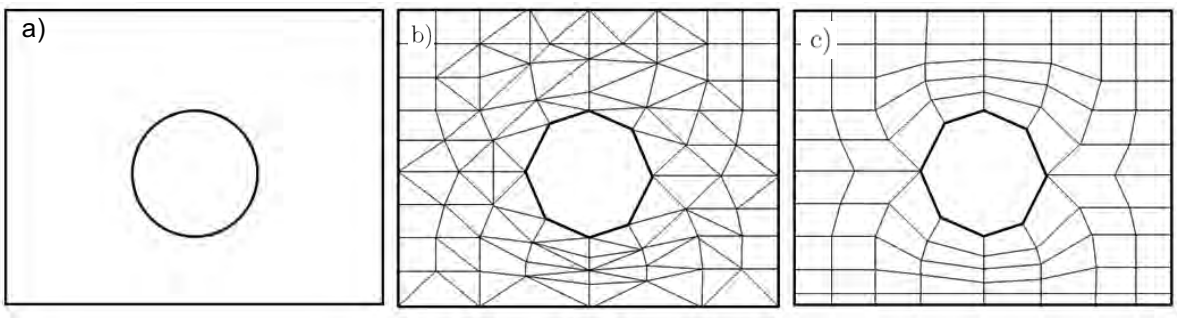

Figure 1: *On unstructured and structured meshing. a) A physical 2D domain (e.g. air around a cylinder) b) An unstructured approximation of 'a)' by unordered triangles and quadrilaterals c) A structured approximation of 'a)' by ordered quadrilaterals only*

The research division *Motorische Verbrennung inhomogener Gasgemische* (IHGG, engl.: *Internal Combustion of Inhomogeneous Gas Mixtures*) at the *Faculty of Technology* of the DHBW are undertaking RANS and LES simulations regarding medium scale four-valve gas engines (cylinder volume approx. 4.8 litres). The quality requirements of the simulations, the exceptional geometric and topologic complexity[3] of IC-engine (IC: internal combustion) models as well as the need to model moving and morphing meshes at comparatively low costs, led to the decision of harnessing fully hexahedral meshes as they can be generated using MegaCads. Since all CFD simulations within that scope are to be performed using OpenFOAM and there is no native connection or transfer mechanism between MegaCads and OpenFOAM, it was obviously necessary to create such an interface. Another reason for developing certain 'tooltips' (such as extrusion or rotation) is the fundamental need of simple manipulation routines for the quick creation of regular but high quality mesh regions (e.g. straight inlet pipes, valve wakes, piston bowls). It turned out, that corresponding tasks within MegaCads or OpenFOAM often suffer from 'clumsiness' or less handy feasibility. Basically, all of these actions might also either be taken directly within MegaCads and OpenFOAM or via dedicated workarounds. However, as a matter of convenience and processing speed, it is desirable to have such tooltips available alongside the mesh conversion utilities when preparing a mesh for the intended purposes.

Initially, (at least for the use in RANS simulations) the unstructured/hybrid mesh generators GMSH [8], NETGEN [9] and EnGrid[4] [10] have been utilised for benchmarking purposes but did not perform sufficiently in the above mentioned use case, as either the local quality of the resulting mesh was not satisfactory or the generators crashed, while creating the base mesh in critical regions of the highly irregular surface geometry. It must be noted, that these issues occurred mainly due to ill-conditioned or faulty underlying surface geometry (disjoint or duplicate surfaces etc.) which originates from reverse-engineering data. The OpenFOAM utility SnappyHexMesh has also been tried out for meshing the above-mentioned IC-engine geometry. It succeeded to create an automatic mesh for the provided geometry but unfortunately failed in terms of usability regarding control of the local mesh quality due to the geometric complexity. For instance, during flow computations based on SnappyHexMesh grids, numerical stability issues occurred that have been traced back to a region within the flow domain where large gradients of velocity, pressure and density alongside strong geometric non-uniformity and curvature

[3] Geometric complexity is given by strong and alternating local surface curvature as well as the presence of many very small and highly non-uniform surface regions.
[4] GMSH and EnGrid internally make use of NETGEN and TETGEN [11].

occurred. Hence, SnappyHexMesh was found not being flexible enough in the desired locations to obtain sufficient mesh quality under acceptable efforts. BlockMesh is considered not to be of practical relevance as mesh-design utility within the mentioned scope due to its intrinsic methodologic limitations (cf. [2]). Table 1 summarises the considered mesh generators/utilities. The term 'polymorph' means that the meshes may consist of different types of elements, i.e. tetrahedrals, pyramids, hexahedrals and more.

Generator	Mesh Type	Observation
GMSH	unstructured (polymorph)	crashed
NETGEN	unstructured (polymorph)	crashed
EnGrid	unstructured (polymorph)	crashed
SnappyHexMesh	unstructured (polymorph)	success (long processing time), local quality insufficient
BlockMesh	unstructured (polymorph)	method not suited
MegaCads	structured (quadrilateral and hexahedral)	success, high quality, elaborate

Table 1: *Mesh generators/utilities that have been considered for meshing*

3 ELLIPTIC, STRUCTURED MESHES IN MEGACADS

As already mentioned, elliptic meshes are well suited for the use in many CFD applications. The term 'elliptic' denotes the type of system of governing partial differential equations that is solved to distribute the grid points. The reader is referred to e.g. [1] for more detailed information. A common technique – as harnessed by MegaCads – is to generate an algebraic base grid from boundary point-distributions and to redistribute the resulting off-boundary (internal) points in an 'elliptic manner'. The underlying algebraic grids in general show a rather 'unruled' distribution of points with at best weak respect to smoothness or cell quality requirements, whereas the grid points of elliptic meshes are arranged much smoother and the grid lines tend to be more orthogonal resulting in higher quality cells, as is depicted in figure 2.

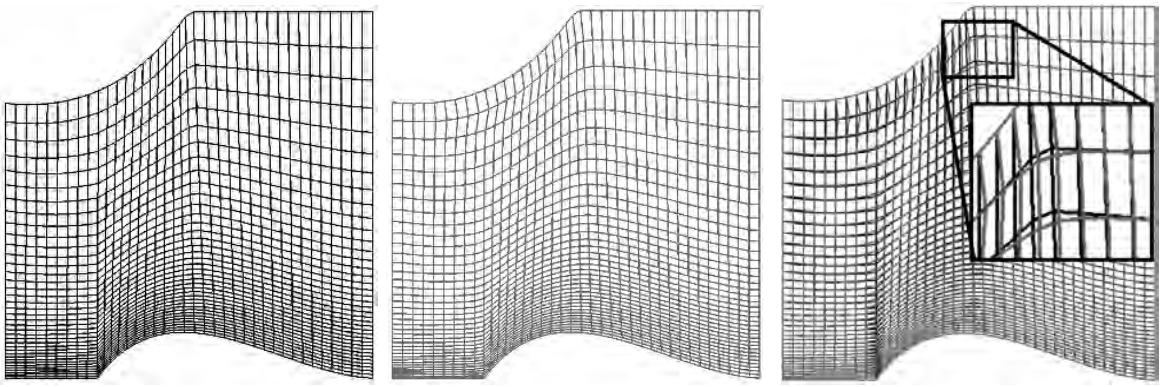

Figure 2: *Comparison between algebraic (black) and elliptic (red) 2D sample grids.*
Left: Algebraic grid. Middle: Elliptic grid. Right: Both grids overlaid with close-up section.

A mesh is called 'structured' when all cells are arranged in a countable order, whereas 'unstructured' meshes in general have a highly irregular cell ordering. In structured hex meshes the 'countability' is established by introducing ijk-directions which refer to the normals of the three pairs of opposing faces of a hex cell. Hence, the indices of all neighbouring cells are known a priori by knowing the index of the cell in the centre and the total number of cells N_i, N_j, N_k in ijk-direction of a grid block. Figure 3 describes that relation for 2D cells (extension to 3D is straight-forward). This is obviously not the case for unstructured meshes. Any off-boundary hex cell in a fully hexahedral mesh always has exactly six adjacent, neighbouring cells with regularly assigned indices. Structured hex cells can therefore be denoted by a single list index that refers to its position in the grid. The same is true for the cell's vertices (nodes).

Because any first order[5] hex cell as shown in figure 4 can be described by the coordinates of its eight vertex nodes (3D points), the structured cell arrangement allows for storing the cells simply by means of ordered node lists. These lists implicitly contain all cell connectivity information that is needed for further processing (e.g. cell boundary coupling for flux computation). The block connectivity of multi-block meshes is also included. Except for 'hanging nodes'[6] block boundaries can be considered connected, when two cells of separate block boundaries share exactly four identical nodes. Block boundaries may either belong to two distinct blocks or one and the same block.

[5] 'First order' means that there are no additional nodes placed on the cell's boundary faces or boundary edges and the nodes are connected by straight lines only.

[6] 'Hanging nodes' do not perfectly match a counter-part node on the opposing block boundary or even have none.

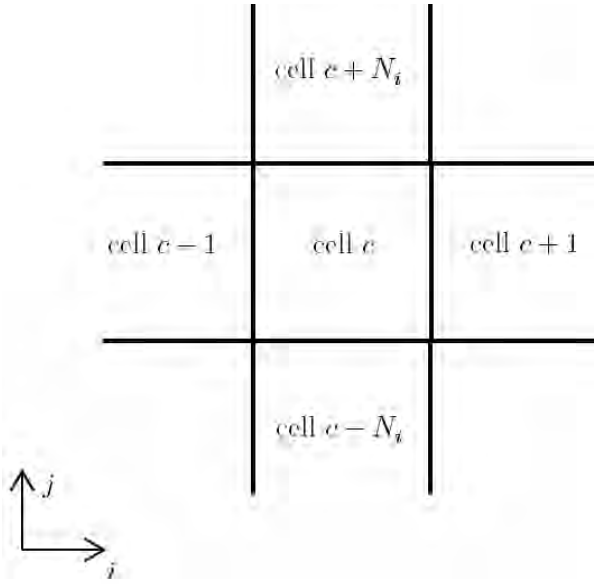

Figure 3: *On the ordering of cells*

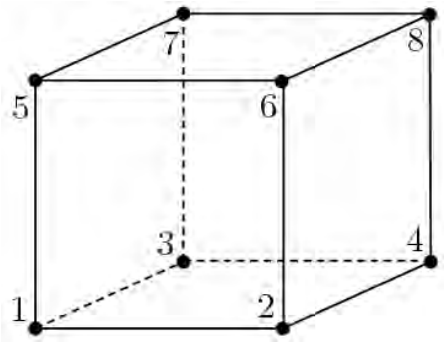

Figure 4: *A hexahedral cell defined by ordered vertex nodes*

As mentioned above, MegaCads is capable of producing elliptic meshes in 2D (quadrilateral cells) and 3D (hexahedral cells) space. The POPINDA output of MegaCads also shows the favourable points ordering. Furthermore, with a focus on OpenFOAM the intuitive GUI (cf. figure 5) makes the mesh design process a lot more vivid and transparent compared to the command-line based solutions BlockMesh or SnappyHexMesh.

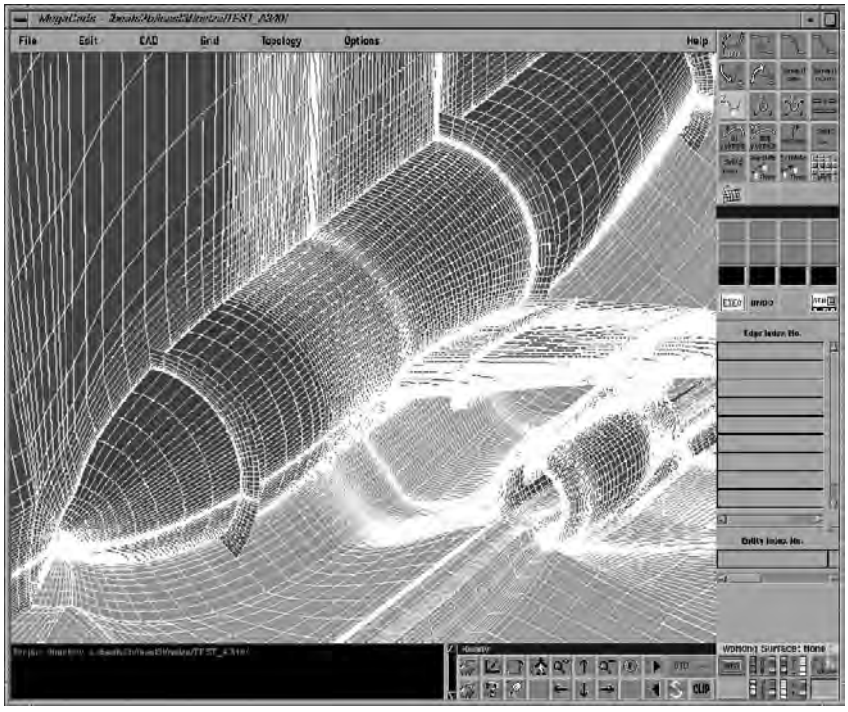

Figure 5: *The MegaCads GUI [5]*

4 THE POPINDA MESH FORMAT

The mesh output of MegaCads that is harnessed by the herein presented toolbox is shaped in POPINDA-format. The POPINDA format utilises the idea of points ordering as described above. A typical POPINDA hex mesh file is illustrated in figure 6. The '$$' symbol marks commentary lines. The first integer in the first non-commentary line is the total number of blocks in the mesh file. The second and the third integer are at least within the named focus of no importance7 and always set to '1'. The next non-commentary line contains the number of points n_i, n_j, n_k of the first block and a fourth integer which is here again of no importance8 and always set to '0'. Any block consists of

$$n = n_i \times n_j \times n_k \tag{1}$$

points, which form a total of

$$N = \underbrace{(n_i\text{-}1)}_{N_i} \times \underbrace{(n_j\text{-}1)}_{N_j} \times \underbrace{(n_k\text{-}1)}_{N_k} \tag{2}$$

hexahedral cells simply by their known ordering. Given any cell in accordance to figure 4, the indices of the eight vertex nodes $v_1 \dots v_8$ of each cell of a mesh block are computed as described in (3) and (4). While looping over i, j and k, the list indices of each node of any cell can be derived from the list index of the cell's base node b (node 1 in accordance to figure 4):

$$b = i + (j\text{-}1)n_i + (k\text{-}1)n_i n_j \tag{3}$$

$$\begin{aligned} v_1(b) &= b \\ v_2(b) &= b + 1 \\ v_3(b) &= b + n_i \\ v_4(b) &= b + n_i + 1 \\ v_5(b) &= b + n_i n_j \end{aligned} \tag{4}$$

[7] The second integer denotes the number of multigrid levels of the mesh. The third integer indicates the type of coordinate system used [5].

[8] The fourth integer denotes the number of 'ghost cells', Ghost cells are 'hypothetical' cells that are typically used for quantity extrapolations that reach beyond the mesh boundaries [5].

$$v_6(b) = b + n_i n_j + 1$$

$$v_7(b) = b + n_i n_j + n_i$$

$$v_8(b) = b + n_i n_j + n_i + 1$$

Thus, the entire hex mesh can immediately be constructed from the POPINDA points list purely. Because OpenFOAM only 'knows' unstructured meshes, the memory or linear solver related advantages of structured meshing are not accessible to it. However, the manipulation and conversion procedures of the herein presented software make intensive use of the nested data arrangement.

```
$$  nblock  nlevel  icoord
     1       1       1
$$ PROJECT= "/$$PROJECT_DIRECTORY"
$$ SCRIPT= "/$$SCRIPT_DIRECTORY"
$$  physical block dimensions:
$$   i    j    k    iwidth
    39      39       1       0
$$  coordinates:
$$       x                   y                   z
4.0000000000e+01 0.0000000000e+00 0.0000000000e+00
4.0000000000e+01 4.0000000596e-01 0.0000000000e+00
4.0000000000e+01 7.9388964176e-01 0.0000000000e+00
        .                   .                   .
        .                   .                   .
        .                   .                   .

$$  physical block dimensions:
$$   i    j    k    iwidth
    15      39       1       0
$$  coordinates:
$$       x                   y                   z
3.8877437592e+01 6.1882553101e+00 0.0000000000e+00
3.8875961304e+01 6.6861419678e+00 0.0000000000e+00
3.8874488831e+01 7.2089881897e+00 0.0000000000e+00
3.8873203278e+01 7.7604265213e+00 0.0000000000e+00
3.8872085571e+01 8.3445634842e+00 0.0000000000e+00
3.8871135712e+01 8.9660692215e+00 0.0000000000e+00
3.8870483398e+01 9.6302776337e+00 0.0000000000e+00
```

Figure 6: *A snippet of an ASCII encoded POPINDA/FLOWer mesh file.*

5 MESH MANIPULATION TOOLTIPS

BLoOMYBOXX comes with a text-based interactive menu which enables the user to select the wished manipulation or conversion procedure to be performed on the user-prescribed mesh. The interface displays some basic mesh properties, i.e. mesh development information as well as the number of points, blocks, cells and some additional data that might be helpful for further processing. It interactively guides the user through mesh manipulation or conversion as illustrated in figure 7. The available manipulation tooltips at the current stage of development are:

- Clean up mesh
- Scale mesh
- Translate mesh
- Extrude 2D mesh along cartesian axis
- Rotate mesh
- Revolve 2D mesh around cartesian axis
- Make 2D mesh axisymmetric
- Mirror mesh over xy-plane

The meshes are kept in POPINDA format during all manipulation procedures, which means they can be manipulated and reread into the MegaCads environment as fully processible grids. To perform any sort of manipulation or conversion, one must specify a POPINDA shaped ASCII mesh file. The software is simply run by typing[9]

```
$ bloomyboxx_alias <mesh-file>
```

in a terminal window from the working directory where <mesh-file> refers to the mesh file's path. BLoOMYBOXX handles multi-block grids in 2D and 3D. It is clear that some tooltips exclusively apply to 2D or 3D meshes (e.g. 'extrusion').

[9] The alias '`bloomyboxx_alias`' must point to the location of the executable file.

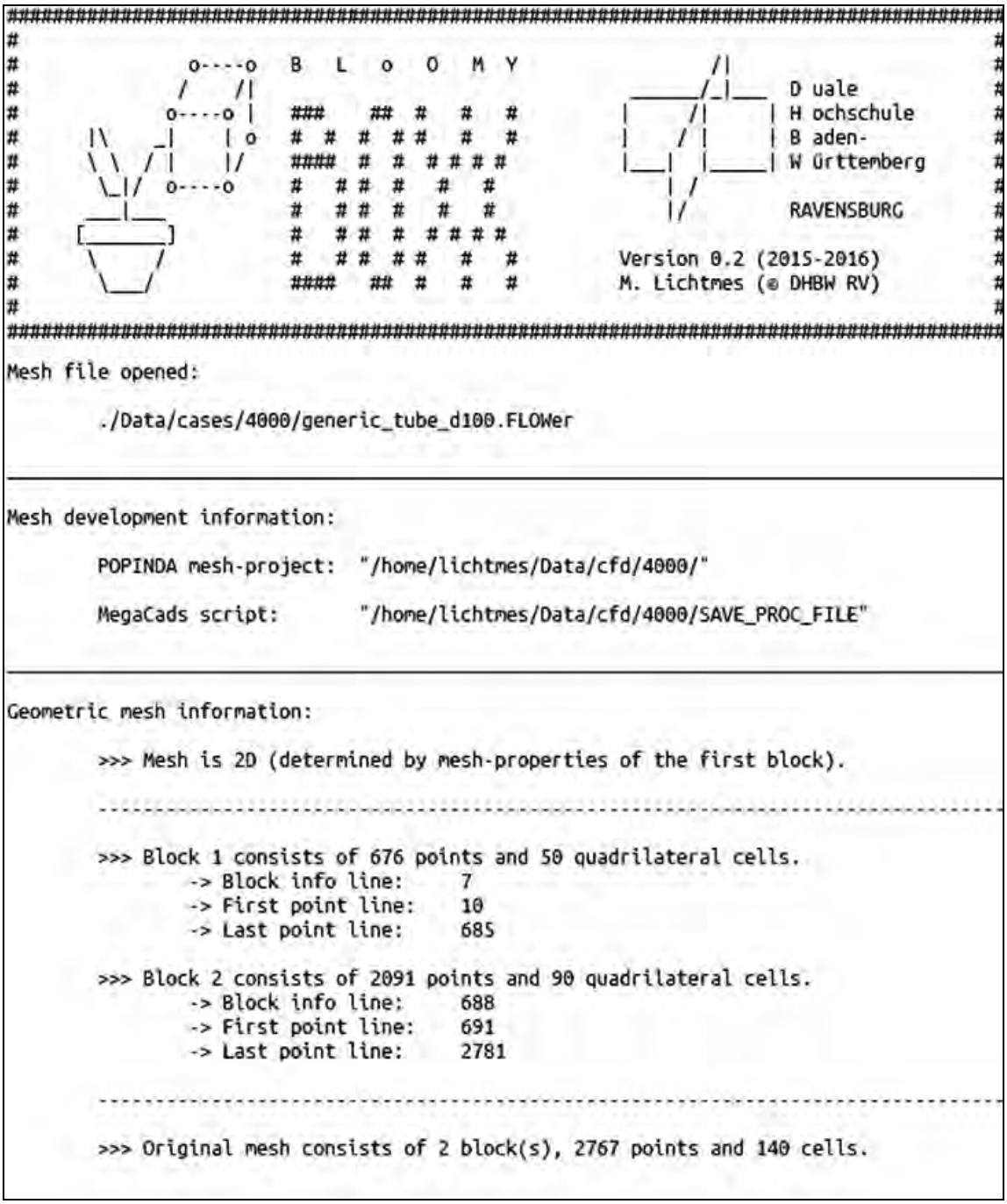

Figure 7a: *A sample BloOMYBOXX run (terminal view part 1). The mesh reading sequence.*

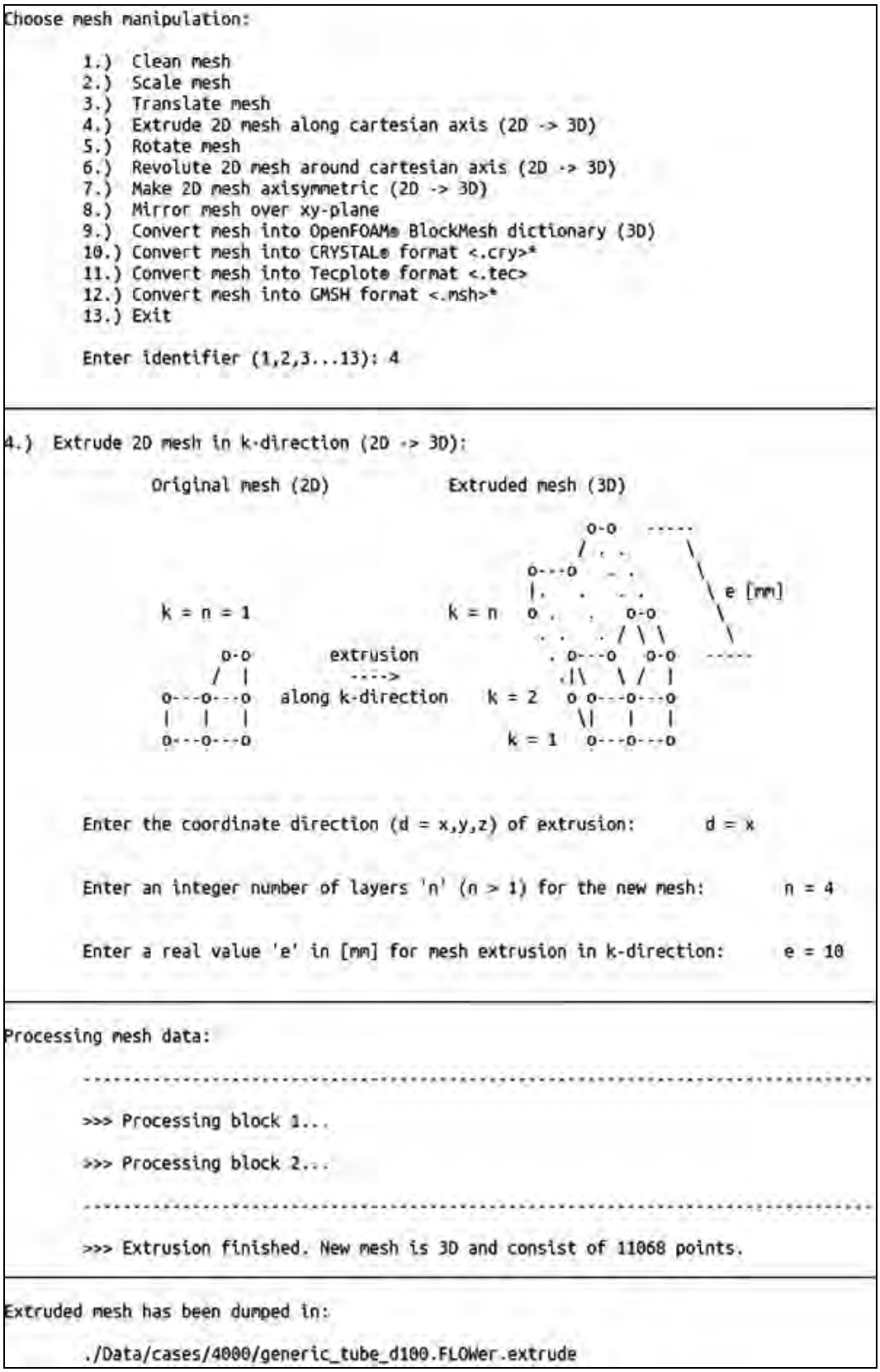

Figure 7b: *A sample BloOMYBOXX run (terminal view part 2). The processing sequence. A 2D mesh is being extruded in x-direction at a new total of 4 k-layers and an extrusion length* $e = 10$ mm.

Clean up mesh

The 'clean up mesh' manipulator corrects slight non-zero inaccuracies of grid point coordinates in the mesh. For instance, such inaccuracies occur often when points have been rotationally shifted or revolved, since those displacements cannot be calculated exactly due to numerical error. But also allegedly clean meshes dumped by MegaCads may contain very small but unwanted offsets that were introduced during mesh generation. The user can then prescribe an absolute correction tolerance τ (e.g. $\tau := 10^{-6}\ \mathrm{mm}$), which the software will use to check whether a point's (P) coordinate component $x_\mathrm{P}, y_\mathrm{P}, z_\mathrm{P}$ is to be set to zero or kept as it is. The tooltip is applicable to both, 2D and 3D meshes. Consequently, it may also be utilised to project slightly uneven surfaces onto cartesian planes.

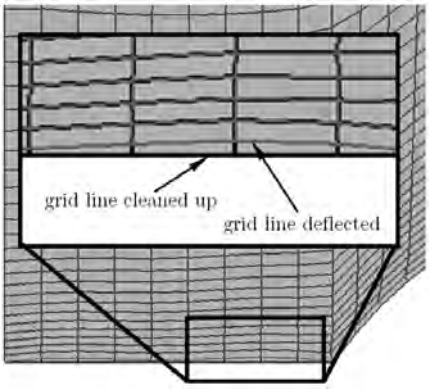

Figure 8: *Mesh cleaned up. Original mesh: Red. The base mesh shows slight inaccuracies in the bottom region (see close-up). The cartesian points that lie within the user-prescribed distance tolerance τ from any axis are shifted towards the corresponding one.*

Scale Mesh

As the name suggests, this tooltip simply scales 2D and 3D meshes by a user prescribed scaling factor s. Scaling can yet only be achieved uniformly, which means the grid is being stretched or shrunk in all three cartesian directions by the same factor.

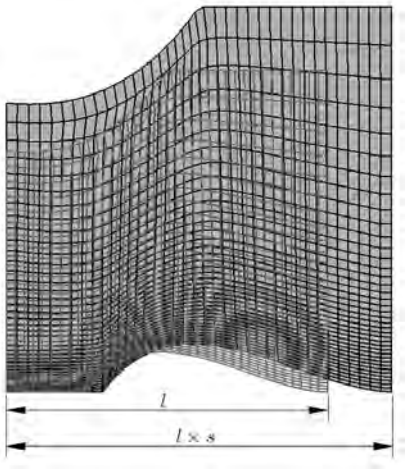

Figure 9: *Mesh scaling. Original mesh: Red. The base mesh has been uniformly scaled by a factor of $s = 1.2$.*

Translate Mesh

2D and 3D meshes will be translated along a user prescribed 3D Cartesian vector $\mathbf{t} = \left(t_x, t_y, t_z\right)^{\mathrm{T}}$.

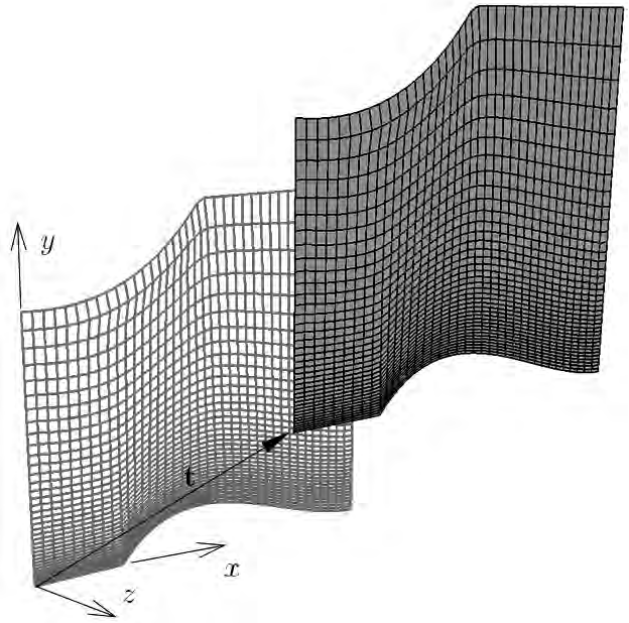

Figure 10: *Mesh translation. Original mesh: Red. The base mesh has been translated by a vector* $\mathbf{t} = (l/2, l/2, l/2)^{\mathrm{T}}$*, where* l *is the edge length of the quadratic bounding box of the 2D base grid.*

Extrude 2D mesh along cartesian axis

When generating 2D meshes with MegaCads there will only be one k-layer ($n_k = 1$). BLoOMYBOXX offers the possibility to extrude a 2D mesh in the k-direction. The user can prescribe the new number of resulting k-layers, the length of extrusion e and the cartesian direction $d = x, y, z$ of extrusion.

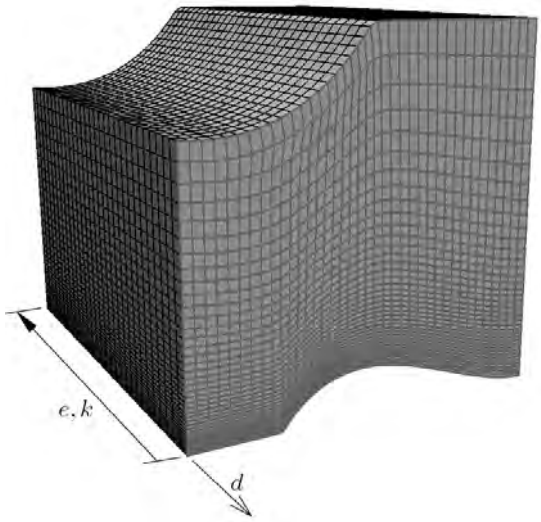

Figure 11: *Mesh extrusion. Original mesh: Red. The 2D base grid has been extruded by 30* k*-layers at a negative extrusion length matching twice the edge length of the quadratic bounding box (*$e = -2l$*) of the base grid. The direction of extrusion is* $d = z$.

Rotate mesh

Similarly to the translation tooltip, it is possible to rotate a 2D or 3D mesh. The user has to choose an axis $d = x, y, z$ of rotation and a rotation angle ϕ in degrees.

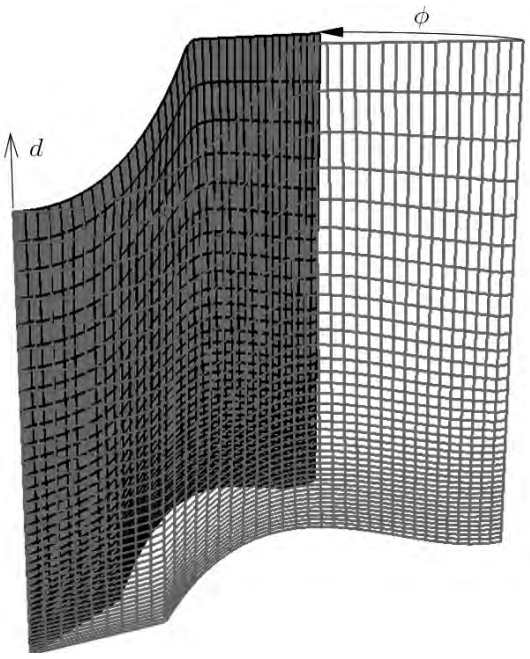

Figure 12: *Mesh rotation. Original mesh: Red. The mesh has been rotated by $\phi = 25\,°$ around the cartesian y-axis ($d = y$).*

Revolve 2D mesh around cartesian axis

Analogous to extrusion additional k-layers are introduced to create a rotational 3D extrusion of a 2D base mesh. The user must specify a new number of k-layers n_k, a revolution angle ϕ in degrees and the cartesian axis d of revolution x, y, z.

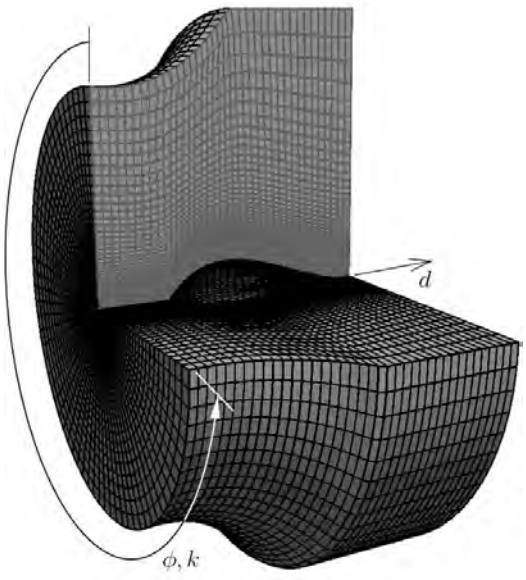

Figure 13: *Mesh revolution. Original mesh: Red. Volume mesh created by revolution of the 2D base grid by $\phi = -270\,°$ around the x-axis ($d = x$).*

Make 2D mesh axisymmetric

Basically, this manipulator works in the same way as revolution except the fact that the new number of k-layers is restricted to being $n_k := 2$ and the revolution angle ϕ is limited to a maximum value of 5 ° for computational quality of the CFD solution. Additionally, this transformation is bounded to the x-axis as centre of revolution.

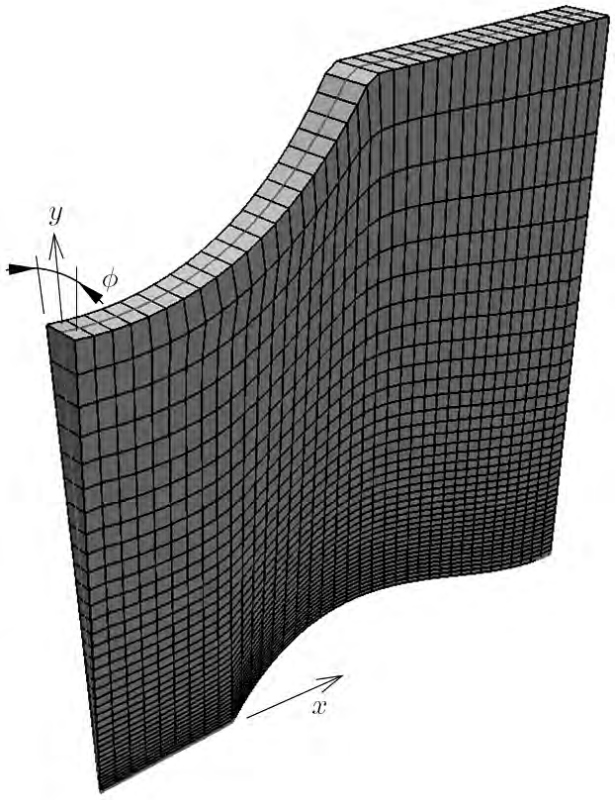

Figure 14: *Mesh made axisymmetric. Original mesh: Red. The mesh has inversely been revolved by a single additional k-layer.*

Since different solvers handle axisymmetry in different ways, one must be aware of the here adopted method:

When revolved, the original 2D mesh points are expected to lie perfectly in the xy-plane ($z_\mathrm{P} = 0$; one may want to run 'Clean up mesh' in advance). The new z-coordinates of the 3D grid points are computed by duplicating the 2D points and symmetrically shifting them at an angle of $\pm\phi/2$ around the x-axis. To address the requirements of the below described conversion to OpenFOAM [3] the duplicate points at $y_\mathrm{P} \equiv 0$ (points on the axis of symmetry) collapse perfectly[10], resulting in wedge or prism elements instead of hex cells (cf. figure 15 and [3]). Wedge or prism elements are also created when revolution is applied and points are located at the axis of revolution.

[10] Some solvers expect a small non-zero distance between those 'axis points'.

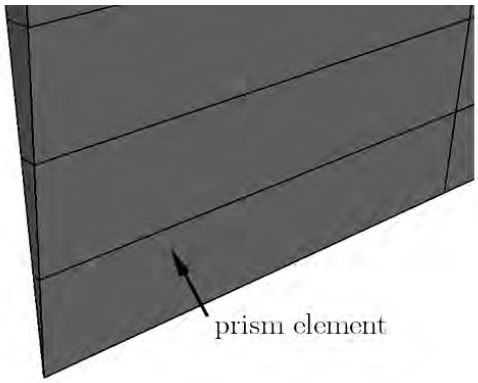

Figure 15: *Prism type cells at the axis of revolution.*

Mirror mesh over xy-plane

When mirrored, a 2D or 3D mesh's points are being inversely duplicated. Yet, only mirroring over the xy-plane is implemented which means there are no parameters to be specified by the user. The mirror image and the base mesh are not being block-wise unified. This means, with respect to the original mesh, the mirror image is represented by one or more additional block(s).

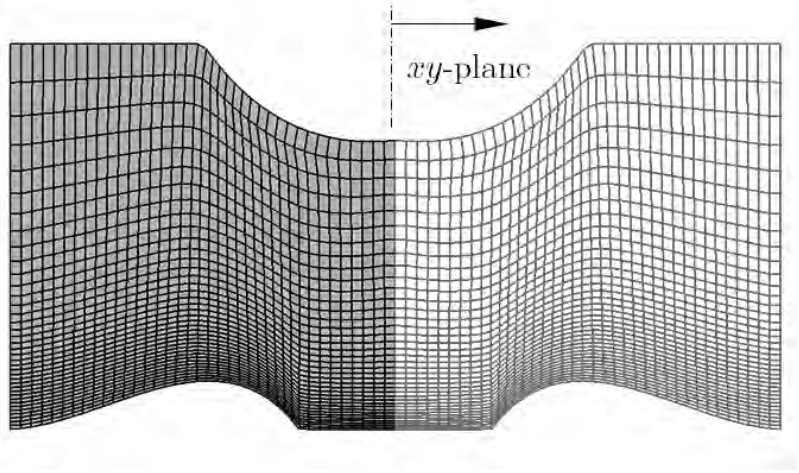

Figure 16: *Mesh mirroring. Original mesh: Red. The mirror image represents an additional mesh block. The arrow marks the xy-plane's normal vector. The original 2D mesh has been rotated by $90\,°$ around the y-axis before mirroring.*

6 MESH CONVERSION TOOLTIPS

The following mesh conversion tooltips are currently implemented in BLoOMYBOXX:

- Tecplot® DAT format
- OpenFOAM BlockMeshDict
- GMSH 2.0 format (yet experimental)

The Tecplot® DAT support is mainly intended to serve for mesh viewing purposes using established software, i.e. Tecplot®, ParaView or others. The two other formats immediately serve as input for CFD solvers or alternative preprocessing software.

As mentioned above, one of the main purposes of the toolbox initially was to convert POPINDA meshes into a grid format readable for OpenFOAM. Since also alternative mesh formats such as Tecplot® [12], VTK (Visual Toolkit) [13] or GMSH are expected to be useful within the mentioned research scope, a rather 'common' arrangement of the resulting mesh data has been chosen:

For instance, plain ordered (structured) Tecplot® files can easily be generated simply by adapting file and block header sequences of the POPINDA files, because the points are arranged in identical manner [14]. Instead, the polymorph unordered (unstructured) VTK, GMSH or OpenFOAM meshes cannot be created in the same way but explicitly require preassembling of the cells from point data. OpenFOAM itself utilises the 'face-addressing' PolyMesh format [3]. But BlockMesh files (BlockMeshDict) are apparently easier to derive from plain point data than PolyMesh grids, as they show strong similarity to the usual, cell based unstructured grid formats, such as VTK or GMSH. Furthermore, MegaCads meshes are purely hexahedral in 3D and because of their known ordering, it is reasonable to adopt a simpler, cell based conversion mechanism closer to common formats. This can be achieved by creating a BlockMeshDict shaped 'grid file'[11], consisting purely of hexahedral blocks with each block consisting of a single cell only. One can then simply command BlockMesh to take on the hard work of creating the face-addressing PolyMesh grid for OpenFOAM. BlockMesh will transform the one-cell hex blocks back into ordinary hex cells that – in terms of their geometry – perfectly represent the former POPINDA formatted mesh. The chosen method meaning simplicity and development speed, BLoOMYBOXX has been given the ability to construct BlockMeshDicts from points instead of PolyMesh grids. A snippet of a sample BlockMeshDict created from a POPINDA mesh file using BLoOMYBOXX is depicted in figure 16. The term 'hex' in the '`blocks(…)`' section stands for hexahedral blocks. The eight integers in the subsequent parentheses denote the list indices of the block defining vertices that are stored in the 'vertices' section. The second parentheses represents the numbers of cells within the corresponding block in all three ijk-directions and is according to the above-mentioned single-cell-block approach always set to '`(1 1 1)`'. The instruction '`simpleGrading (1 1 1)`' commands BlockMesh to use constant cell sizes across the corresponding block. The latter part of each block definition line is mandatory but obviously meaningless since the mesh will result in a single cell per block anyway.

[11] For BlockMesh a BlockMeshDict is not a 'grid' file but rather a grid generation script.

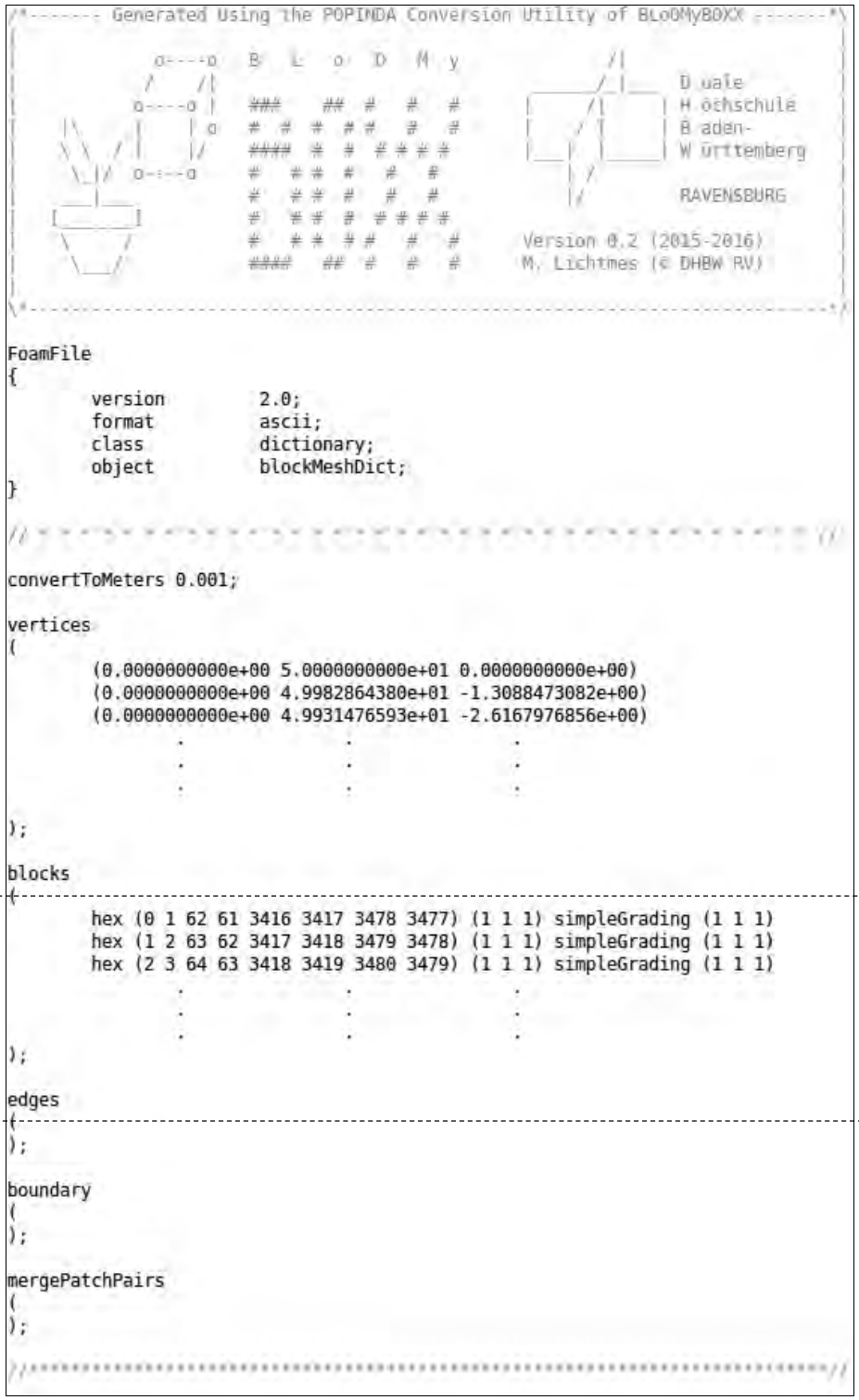

Figure 16: *A snippet of a sample BlockMeshDict file done with BLoOMYBOXX. The section where the single-cell-hex-blocks are defined is highlighted by a dashed box.*

At the current stage of development boundary patches at the outer block boundaries of the mesh are not created automatically, when converting the mesh into any other format. But regarding OpenFOAM, there is a general workaround available to do so, as shown in the following chapter.

7 MESH USE IN OPENFOAM

When an OpenFOAM mesh is to be created based on the described POPINDA-to-BlockMeshDict interface, one will have to specify boundary conditions on the mesh boundaries manually. As BLoOMYBOXX in its current version does not support automatic boundary patching, the following procedure is recommended to create boundary patches:

1. Create the desired BlockMeshDict file using BLoOMYBOXX.

2. Move the BlockMeshDict file into a properly set up OpenFOAM case directory's 'system' folder.

3. Run

   ```
   $ blockMesh
   ```

   from the top level of the case directory and monitor the output. If BlockMesh complains about negative cell volumes – which is likely to occur in cases of former extrusion or revolution – it is recommended to rerun extrusion/revolution in the opposite direction (extrusion: negative extrusion length e, revolution: negative revolution angle ϕ). BlockMesh will dump the new mesh in OpenFOAM's PolyMesh format in the case's 'constant/polyMesh' directory.

4. Now run OpenFOAM's automatic patching utility 'AutoPatch' using a sufficient patch detection angle <angle> in degrees:

   ```
   $ autoPatch –overwrite <angle>
   ```

   As the name implies, this will create patches (named 'auto0', 'auto1' ...) on the new PolyMesh's block boundaries.

5. ParaFOAM – OpenFOAM's software adaptation of the third-party resource ParaView [2,3,15] – may then be used to determine which of the generated patch names belongs to a specific block boundary. Either make sure, the working directory does not contain a time-step directory (i.e. '0' etc.) with 'U' and 'p' file in it or, after ParaFOAM has launched, uncheck the corresponding fields in the load options of the case before hitting 'Apply'. Otherwise, ParaFOAM will happen to crash because it cannot find the matching patch fields as they are not yet defined. ParaFOAM is launched from the case directory by typing:

   ```
   $ paraFoam
   ```

6. Resulting unwanted and duplicate internal block boundaries can be eliminated by applying the ‘StitchMesh’ utility tagged ‘–perfect’ on coinciding patches <1> and <2>:

```
$ stitchMesh <1> <2> -perfect
```

7. The ‘CreatePatch’ utility may then be harnessed to round up patch mapping by concatenating disjoint but coherent patches, creating patch groups, renaming them or even assigning adequate boundary-conditions (cf. [2]).

Since OpenFOAM also offers several backwards directed mesh conversion utilities one can afterwards (at least intermediately) transfer the former POPINDA meshes into almost any wished CFD or meshing environment (see [2] for further information).

8 CLOSING REMARKS AND OUTLOOK

The toolbox, as described herein, is being intensively used within the frame of the above named research regarding IC-engine CFD. All tooltips for manipulation and conversion are thus being steadily tested, improved and even extended. The software provides handy opportunities and shortcuts regarding multi-block based mesh generation and conversion.
Yet, there has not been observed any hard limitation concerning the use for manipulation or conversion, although it turned out that BlockMesh takes quite a while (up to some hours) to process larger grids of e.g. several million cells. There might be a slight increase in performance speed when using binary encoded files but until now, that assumption has not been tested or verified. Furthermore it is clear, that a direct ‘POPINDA-to-PolyMesh’ approach will result in faster conversion. The implementation of such a routine is therefore planned to take place in upcoming releases of BLoOMYBOXX.
Although MegaCads is older software, there is still a justified interest in highly controllable, customisable and high quality hex meshes. Thus, it meets many current technical needs just as good as any ‘up-to-date’ commercial structured mesh generator does. At least within the intended high quality CFD on IC-engines, the benefits of harnessing MegaCads in conjunction with OpenFOAM mean a massive research related relevance of both software products. BLoOMYBOXX will thus undergo further development to extend its capabilities and flexibility. Upcoming manipulation tooltips may include mesh-merging and mesh-decomposition, non-uniform extrusion, mesh-morphing (e.g. to model engine motion) or the construction of regular mesh patters (i.e. groups of turbine blades etc.), while future conversion utilities are most likely aiming on

implementation of a direct PolyMesh (to speed up mesh conversion) or VTK export, boundary patching as well as boundary patch based triangulation to derive STL (Stereolithography) surface representations and more.

Some functionalities of the toolbox are dedicated explicitly to OpenFOAM-specific topologic requirements such as the wedge/prism-approach for axisymmetric simulations. However, the toolbox is not restricted to be only of use when working with OpenFOAM but might also be helpful in conjunction with MegaCads meshing itself, FLOWer or – for instance, via OpenFOAM's mesh conversion utilities – any other CFD code.

ACKNOWLEDGEMENTS

The author gratefully acknowledges provision of the software MegaCads for teaching and research purposes by the DLR.

REFERENCE

[1] Thompson J. F., Soni B. K., Weatherill N. P.; „*Handbook of Grid Generation*“ (1999); *CRC Press*, ISBN 0-8493-2687-7

[2] *OpenFOAM* online presence; *OpenCFD Ltd. at OpenFOAM Foundation Ltd.*; Date of access: 27.06.2015; URL: *http://www.openfoam.com*

[3] *OpenFOAM Foundation Ltd.*; “*OpenFOAM – The Open Source CFD Toolbox; User Guide*” (2015)

[4] *German Aerospace Centre* (*DLR*) online presence; Date of access: 13.10.2015; URL: *http://www.dlr.de*

[5] *MegaCads* online presence; German Aerospace Centre (DLR); Date of access: 13.10.2015; URL: *http://www.megacads.de*

[6] Hepperle M., Brodersen O. Ronzheimer A. Schöning B., Rossow C. C.; „*The Parametric Grid Generation System MegaCads*“ (1996); *Proceedings of the 5th International Conference on Numerical Grid Generation in Computational Field Simulations*; Mississippi (USA)

[7] *ERCIM NEWS Online Edition* online presence; *European Research Consortium for Informatics and Mathematics*; Date of access: 01.07.2016; URL: *http://www.ercim.eu/publicaton/Ercim _News/enw32/schueller.html*

[8] *GMSH* online presence; Geuzaine C., Remacle J. F.; Date of access: 01.07.2016; URL: *http://www.gmsh.info*

[9] *NETGEN* online presence; Date of access: 01.07.2016; URL: *http://www.hpfem.jku.at/netgen/*

[10] *EnGrid* online presence; EnGits GmbH; Date of access: 01.07.2016; URL: *http://www.engits.eu*

[11] *TETGEN* online presence; Si H.; Date of access: 01.07.2016; URL: *http://www.wias-berlin.de/software/tetgen/*

[12] *Tecplot®* online presence; Tecplot Inc.; Date of access: 13.10.2015; URL: *http://www.tecplot.com*

[13] *VTK* online presence; Kitware Inc.; Date of access: 13.10.2015; URL: *http://www.vtk.org*

[14] *Tecplot Inc.*; “*Tecplot 360 – Data Format Guide*” (2008)

[15] *ParaView* online presence; Kitware Inc.; Date of access: 13.10.2015; URL: *http://www.paraview.org*

Herausgeber

Prof. Dr.-Ing. Martin Freitag
Dekan der Fakultät für Technik

Duale Hochschule Baden-Württemberg Ravensburg
Baden-Wuerttemberg Cooperative State University
Marienplatz 2
88212 Ravensburg

ISBN: 978-3-945557-02-0

ISSN: 2199-238X

DOI: 10.12903/DHBW_RV_FN_01_2016_Lichtmes