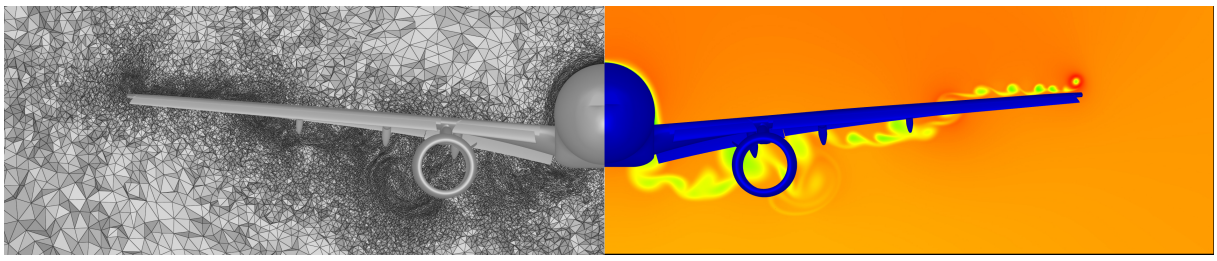
Mesh and solution interactive visualization tool

ViZiR 4: User guide



Inria, Gamma Project Team
Contact: Adrien.Loseille@inria.fr
Matthieu.Maunoury@inria.fr
September 12, 2024

Contents

1 Introduction

1.1 About ViZiR 4

ViZiR 4 is a light, simple and interactive high-order mesh and solution visualization software using OpenGL 4 graphic pipeline. The main features of ViZiR 4 are:

- Light, simple and interactive visualization software.
- Surface and volume (tetrahedra, pyramids, prisms, hexahedra) meshes.
- Pixel exact rendering of high-order solutions on straight elements.
- Almost pixel exact rendering on curved elements (high-order meshes).
- Post-processing tools, such as picking, display of isolines, clipping, capping.

Visit http://vizir.inria.fr to download and try ViZiR 4!
A mailing list https://sympa.inria.fr/sympa/info/vizir-news exists with news and updates on the software. In addition to this user guide, details on ViZiR 4 can be found in [Loseille and Feuillet, 2018, Feuillet et al., 2021] and other examples in [Maunoury et al., 2022].

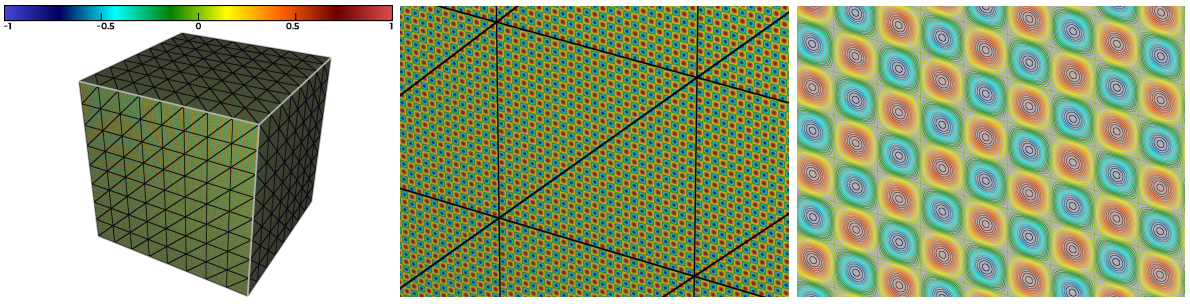
1.2 Why ViZiR 4

At the time, two versions of ViZiR exist:

- A first version (ViZiR Legacy) based on OpenGL Legacy. It supports the visualization of linear and quadratic meshes with solutions on it.
- An other version (ViZiR 4) based on OpenGL 4. Unlike the first one, it is able to display high-order solutions and high-order meshes with pixel exact fidelity.

Should I use ViZiR 4 or ViZiR Legacy ? The answer of this question depends on the use of ViZiR. To visualize high order (more exactly of degree not equal to one) meshes and/or solutions, ViZiR 4 should be used as ViZiR Legacy is not able to do that. Otherwise, to visualize P^1 meshes with (or without) P^1 or P^0 solutions, both ViZiR Legacy and ViZiR 4 can be used. The shortcuts are the same.

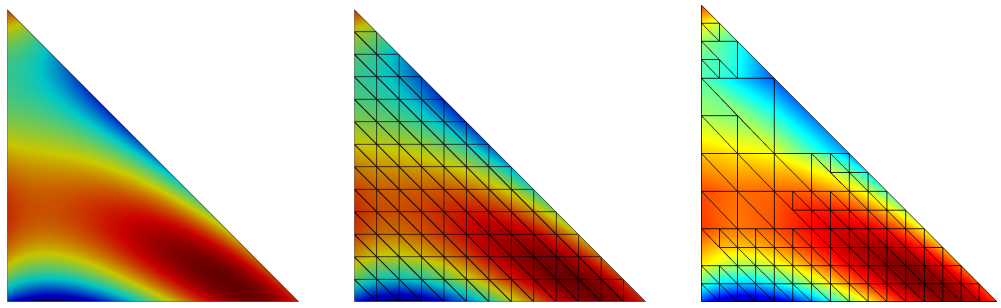
Let now focus on ViZiR 4. OpenGL 4 graphic pipeline flexibility allows to compute on the fly the solution. It leads to a pixel exact rendering when straight elements (of degree one) are considered regardless of the degree of the solution. This recent language (GLSL) enables ViZiR 4 to certify a faithful and interactive rendering. High order solutions are natively handled by ViZiR 4 on surface and volume (tetrahedra, pyramids, prisms, hexahedra) meshes which can naturally be hybrid. An example of pixel exact rendering on straight elements is presented in Figure 1 where a wave mode is rendered.



(a) Far view representing a continuous function (b) Zoom on the rendering (c) Isolines

Figure 1: Pixel exact rendering of a wave mode in a mesh of straight elements (3832 tetrahedra).

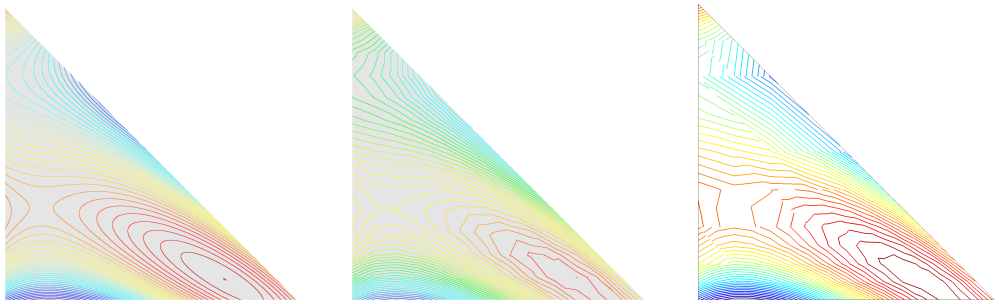
Figure 2 compares pixel exact rendering obtained with ViZiR 4 to affine representations for a polynomial function of degree 3 on a single element. Even with an adaptive subdivision, which allows a better approximation than with an uniform one, there is still a visualization error which is particularly visible when isolines are displayed (see Figure 2 right and middle).



(a) Rendering with ViZiR 4

(b) Rendering with uniform subdivision in a classic visualization software

(c) Rendering with adaptive subdivision in a classic visualization software



(d) Isolines with ViZiR 4

(e) Isolines with uniform subdivision in a classic visualization software

(f) Isolines with adaptive subdivision in a classic visualization software

Figure 2: Rendering (top) and isolines (bottom) of a P^3-solution. Left: pixel exact rendering with ViZiR 4. Middle: uniform subdivision with 13×13 triangles with a classic visualization software. Right: adaptive subdivision of a similar number of triangles (169 triangles) with a classic visualization software.

1.3 Installing ViZiR 4

According to the OS, the process to install ViZiR 4 differs and is described below.

1.3.1 On Mac

Mount the .dmg and copy the application (e.g. the vizir.app file generated) in your Applications directory. Add the following line to your ~/.bash_profile file:

```
alias vizir4='/Applications/vizir4.app/Contents/MacOS/vizir4'
```

Do not forget to source the bash profile. Open a new terminal window, ViZiR 4 can now be launched by entering the command below:

```
vizir4
```

1.3.2 On Linux

An archive containing the executable is provided in the website.
You may need to launch a script to define the good environment variables:

```
./vizir4.sh
```

1.3.3 On Windows

Download ViZiR 4 archive for Windows. Extract the archive, double-click on the executable (file .exe) to launch ViZiR 4, in this case a terminal should open. A terminal, such as Windows PowerShell, can also be used to launch ViZiR 4.

1.4 Input formats

In general, one or two files are given as input to ViZiR 4: a mesh file and eventually a solution file. It is possible to have only a mesh file without solution. It follows the Gamma Mesh Format (GMF) and the associated libMeshb[1] library. Details can be found in the PDF[2] of the library. ViZiR 4 imports meshes stored in .mesh or .meshb files. Files ending by .mesh are ASCII files, and thus need more time to be loaded than .meshb binary files. Section 5 is devoted to meshes files formats. In the same way, solution files end with .sol (ASCII) or .solb (binary). Section 6 is dedicated to solutions files formats. Some examples are given in Section 9. Many examples of mesh and/or solution files can be downloaded in the website[3] of ViZiR 4.

[1] https://github.com/LoicMarechal/libMeshb
[2] https://github.com/LoicMarechal/libMeshb/blob/master/Documentation/libMeshb7.pdf
[3] http://vizir.inria.fr

2 Very basic use of ViZiR 4

First, we describe the very basic commands to launch ViZiR 4.

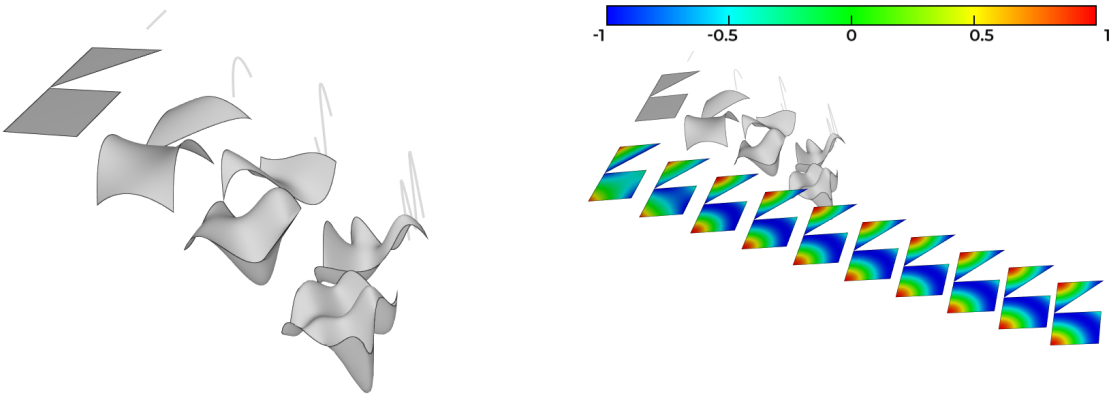
2.1 Launch the demo

To have a first overview of ViZiR 4, the demo can be launched with -demo to see different elements types and degrees

```
vizir4 -demo
```

or -demofull to see more solution degrees

```
vizir4 -demofull
```



(a) Rendering of elements (degree 1 to 4) (b) Rendering of solutions (degree 1 to 10)

Figure 3: Views of the demo

2.2 Read a mesh

To read a mesh, the keyword is -in following by the name of the mesh:

```
vizir4 -in name_of_the_mesh
```

The name of the mesh is supposed to have the extension .mesh or .meshb. Note that ViZiR 4 will also try to add the extension .mesh and .meshb to open the mesh file. For instance, if you run vizir4 -in xxx, ViZiR 4 will try to find xxx.meshb then xxx.mesh and open it if it exists.
Thus, a mesh file named xxx.mesh[b] can be opened with

```
vizir4 -in xxx.mesh[b]
```

or

```
vizir4 -in xxx
```

2.3 Read a solution

To read a solution, a mesh needs to be imported (with -in). To import a solution, whose extension is supposed to be .sol or .solb, the keyword is -sol:

```
vizir4 -in name_of_the_mesh -sol name_of_the_solution
```

If the mesh and solution files have the same name, excepting the extension, the solution can also directly imported with

```
vizir4 -in name_of_the_mesh
```

2.4 Read several meshes

Several meshes (with eventually solutions) can be opened together:

```
vizir4 -in mesh1 mesh2
```

The interest is that views and rendering options are linked. In fact, the propagation of events (modification of the view, the cut plane, rendering options or the center of the scene) can be enable or disable in the preferences (File > Open preferences).

2.5 Important information

When a volume mesh is considered, the surfaces of the elements should be given in the mesh (and solution if any) file. If some surfaces are missing, a pre-processing step of reconstruction of these surfaces and their traces should be done following Section 11.3. If this pre-processing has not been done and surfaces are missing, a keyword can be used (see Section 7.10).

3 Graphical user interface

The graphical user interface of ViZiR 4 (see Figure 4) is composed of several parts:

- Main window with the mesh / solution.
- Bottom bar to select a solution, pick an element from its number, open a menu.
- Inside terminal to print information (more information in the external terminal).
- Menu bar to open menus: File, Tools, State, Utilities, Filters, Help...

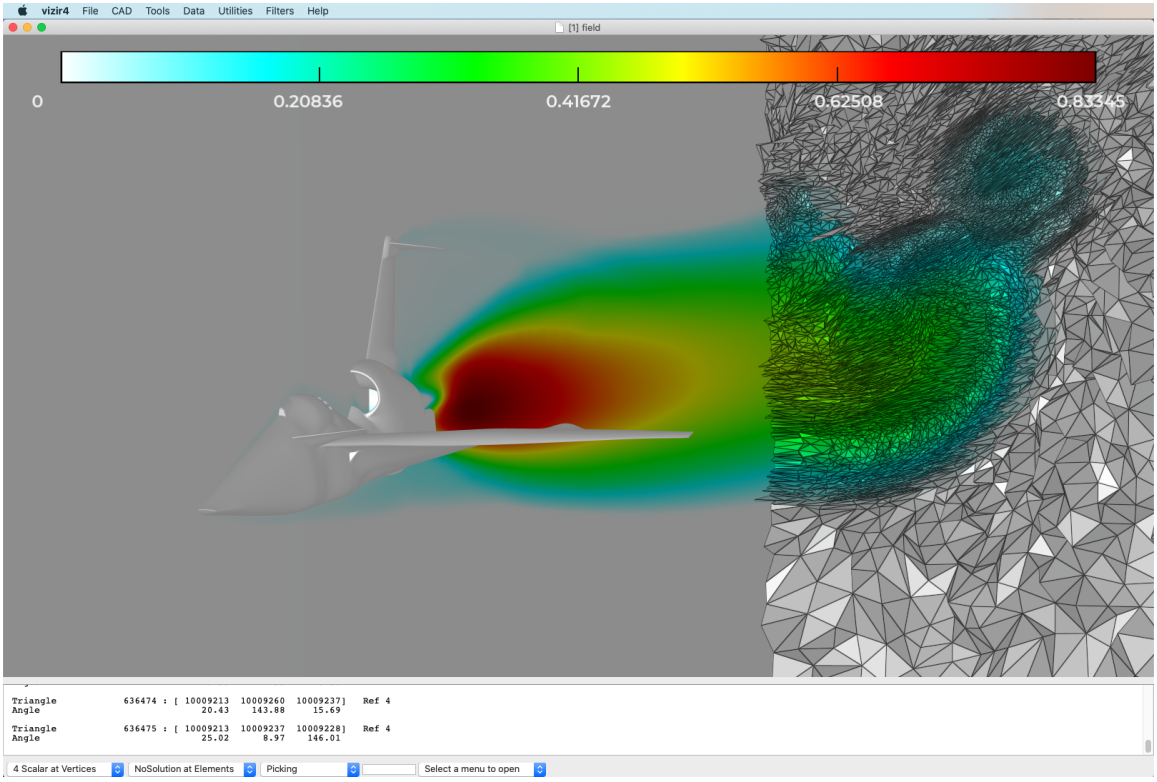


Figure 4: ViZiR 4 graphical interface

This section presents now the menus of ViZiR 4.

3.1 File menu

3.1.1 Open Mesh File / Open Solution File

The mesh must be in the GMF format (mesh or mesh file). The solution must be in the GMF format (sol or solb file).

3.1.2 Save Image (screenshot)

Save the current image. The extension of the file name must be one of these format: png, jpg, jpeg, bmp.

3.1.3 Write Mesh file / Write Solution file

Write mesh/solution files. The GMF format is used. If the extension is not given, binary will be used.

3.1.4 Preferences

All modifications in the preferences menu will change the default options (from now and for next use of ViZiR 4).

3.2 Tools menu

3.2.1 Set Window Size

Modify the window size.

3.2.2 Cut Plane Visualization

Modify the parameters of the cut plane. The plane equation is :

$$ax + by + cz + d = 0 \tag{1}$$

where (abc) is a normal.

3.2.3 Modify View

Modify the view parameters.

- Center : coordinates of the scene center (in the real physical space).
- Rotation : quaternion to handle the rotation view.
- Translation : vector to define the translation in the window coordinates.
- Fovy : level of zoom.

Several buttons allow to set the view along an axis.

3.2.4 Mesh Visualization

Modify the mesh parameters.

- Display elements (lines). Shortcut `l`.
- Display edges that are stored in the mesh file. Shortcut `E`.
- Display lines on surfaces. Shortcut `Y`.
- Draw surface normals. For all surface elements.
- Hide surface elements.
- Hide volume elements. Shortcut `v`.
- Display all metrics. For all vertices. Shortcut `M`.
- Display axis. Shortcut `a`.
- Display bounding box. Shortcut `B`.
- Display all vertices. That may be expensive. Shortcut `D`.
- Display all vertices numbers. That may be expensive. Shortcut `P`.

The shading and the rendering modes can also be modified.
The shrink is a contraction toward the center of gravity of the elements.
For high-order elements, a tessellation is used to display the elements with triangles. The level of tessellation is the adaptive level of details.
Transparency can be set or unset with shortcut `e`. The percentage of opacity can be modified: 1 (100 %) means that it is opaque while 0 means it is transparent (not visible then). The check box "Test depth" is taken into account only when transparency is active. If depth testing is enabled, `OpenGL` tests the depth value of a fragment (i.e. pixel) against the existing content of the depth buffer. If the depth test fails, the fragment is discarded. Thus, when the test depth is not active, all elements are plotted regardless the distance to the camera. See [Sellers et al., 2013] for more details.

3.2.5 Solution Visualization

Solution Visualization Modify the solution parameters.

- Display solution modes. No solution, exact or filled rendering. Shortcut `m`.
- Display isolines modes. No isoline, in colors or in black. Shortcut `o`.
- Display the palette. Shortcut `p`.
- Save palette. Palette will be saved and used in the state file.
- Modify the current solution field (if several).
- The palette is defined by 5 values.
- Update the palette with the 5 values.
- Reset the palette.
- Linear interpolation of the palette from the extrema.
- Set a logarithmic interpolation of the palette.
- Modify the size of the isolines.
- The number of subisolines is the number of isolines between two values of the palette. There are `4 * subisolines` isolines.
- Set a cutoff below and / or above a value.

Vector Visualization 3 modes exist to display vector arrows:

- All units.
- Scaled by norms.
- Scaled by elements sizes.

By default, they are rendered in black but they can be rendered in colors.

3.2.6 Mesh Modifications

Possible changes:

- Merge duplicate vertices. Remove points and modify the connectivity of the elements with the remaining vertices.
- Modify references. Choose a type of elements. Elements can be filled after multi-picking. A list of number of elements is needed as well as the new reference integer. The button Analyse Changes only counts the number of modifications that would be done while the button Update Mesh modifies the mesh.
- Mesh deformations (warp). Choose a vector field. The factor scale (1 by default) can be modified.

3.3 State menu

3.3.1 Load State file / Write State file

Load the parameters of a state file / write the parameters in a state file.

3.3.2 State Manager

Modify all the options available in the state files. It contains information on the view (center of the view, rotation, translation...), the plane equation, the solution (on/off), the isolines (on/off), the level of tessellation for high-order elements, the lines thickness... See Section 8 for more details.

3.4 Utilities menu

3.4.1 Launch Movie Mode

Launch the movie mode which take screenshots from a list of meshes/solutions. See Section 11.5 for more details.

3.4.2 Launch Surface Reconstruction Mode

Launch the surface reconstruction mode which add the missing surfaces (mesh and solution). See Section 11.3 for more details.

3.4.3 Launch State Sequences Mode

Launch the state sequences mode to load several state files and generate the images. See Section 11.4 for more details.

3.4.4 Plot over line menu

A vertex picked can be used to fill the coordinates of one extremity of the line. Then, the plot over line can be used as shown in Section 11.12.

3.4.5 Non stationary solutions menu

The goal of this menu is to navigate through several solution files with a unique mesh file. A list of solution file needs to be defined for instance:

```
Simu_Vizir.00000.solb
Simu_Vizir.00001.solb
Simu_Vizir.00002.solb
...
```

This file can be loaded with "Load Files" in the menu or with the parameter `-sols`. For instance

```
vizir4 -in Simu_Vizir.meshb -sols vizir.sols
```

Then, buttons are used to change the solution. The button "Play" loads all the solution until the end Shortcuts `F9` and `F10` go to the previous or next solution while `F8` launches a loop to play all solutions infinitely until `F8` is pressed again (connected to a signal).

3.5 Filters menu

3.5.1 Quality

Computation of quality is defined only for tetrahedra. Extrema can be computed. Then, it is possible to filter with a range and show only elements that belong to this range.

3.5.2 Jacobian

Two types of jacobian computation are possible: minimal jacobian where the minimal value of each element is stored and the exact jacobian where a high order polynomial jacobian is displayed. Extrema are computed. Then, it is possible to filter with a range and show only elements that belong to this range.

3.5.3 Solution

Extrema of the solution fields are computed and for solutions at vertices the indices of the vertex where the extrema are computed are printed. Then, it is possible to filter with a range and show only elements that belong to this range.

3.6 Help menu

3.6.1 List of shortcuts

Give the list of shortcuts.

4 List of shortcuts

ViZiR 4 offers shortcuts allowing a fast interaction with the scene through the mouse and the keyboard. These shortcuts are summarized in the following tables.

4.1 Mouse interactions

Action	Function
Mouse wheel	Zoom in/out (if activated in the preference menu)
Left button pressed + Move	Rotate the view around the mesh center
Middle button pressed + Move	Translate the view
shift + left click or Double-click	Select a face element (picking)
shift + middle click or alt + middle click	Select a vertex element (picking)
alt + left click	Select an edge element (picking)

4.2 Keyboard interactions

All the Keyboard interactions are case sensitive and are summarized below.

Shortcuts	Function
Arrows	Translate the view
Space	Screenshot the current view
Escape	Quit the window
+/-	Increase/decrease size of edges/points
</>	Display previous/next solution field if any
a	Show axis
A	Show FPS (animate)
b	Change background
B	Show bounding box
c	Show scene center
C	Enable/disable capping
D	Show all points numbers
e	Activate/deactivate transparency
E	Show edges
f	Change rendering of solid faces
F	Change rendering of edges
g	Show corners and ridges if any
i	Return to initial view
j	Show minimal jacobian
J	Show normalized jacobian
k	Change view dimension (2/3)
l	Show mesh edges
L	Show tessellation for non-simplex elements
m	Show solution on surface entities if any
M	Show metrix solution if any
n	Change type of solution (vertices/elements)
N	Show in/out orientation
o	Show solution isolines if any
p	Show the palette if a solution is loaded
q	Quit the window
r	Change shading rendering
s/S	Suppress/un-suppress picked reference entities
t/T	Increase/decrease tessellation level for curved elements
u	Show text
U	Show cut plane
v	Hide/Show volume elements
V	Center scene to the picked entity
w	Save current visualization configuration in vizir.state
Y	Show mesh edges in surfaces
z/Z	Zoom in/out
F1	Activate/deactivate plane
F2	Clipping : enable/disable modifications of the plane (position/angle)
F3	Reset plane
F4	Capping (slice) : enable/disable modifications of the plane (position/angle)
F5	Activate shrink
F6/F7	Increase/Decrease shrink value
F8	Play all solutions if any
F9/F10	Go to previous/next solution file if any

4.3 Detailed explanations

We now give explanations to the most common interactions and queries.

4.3.1 Controlling the view

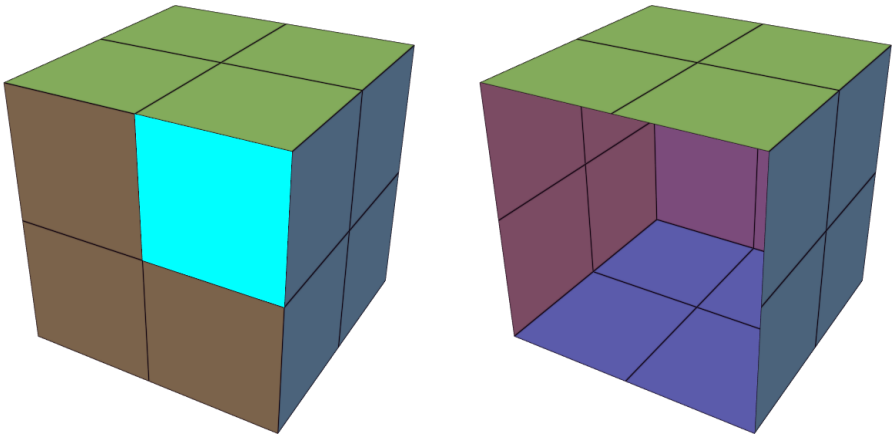
The view can both be controlled by the mouse and the keyboard:

- To zoom (in or out), the `mouse wheel` (if you change the preference in the dedicated menu) can be used as well as the keys `z` (zoom in) or `Z` (zoom out).

- To translate the view, press the `middle button` of the mouse and move it in the desired direction. The `arrows` of the keyboard (`left`, `right`, `up`, `down`) can also be used to translate the view.
- To rotate the view, press the `left` button of the mouse and move it in the desired direction.

4.3.2 Pick and hide elements

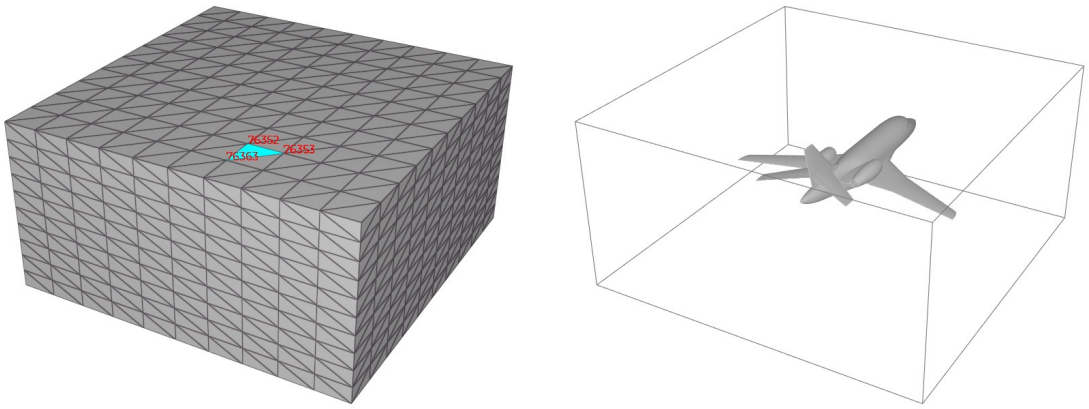
An element can be picked with a double click or with `shift` + click on it, as shown in Figure 5.a for the mesh `cube_struct.mesh`. The reference of this element is the number 20 (color brown in this example) and the selected item became light blue. The color reference can be shown with the button `f`: several face renderings are possible. Then, all the elements whose reference is 20 (color brown in this example) can be hidden with button `s`. To re-display the last elements, press `S`. A second example of hidden elements by reference is shown in Figure 6.



(a) An element is picked (light blue)

(b) All elements with the same reference are hidden with the button `s`

Figure 5: Illustration of picking



(a) Select an element

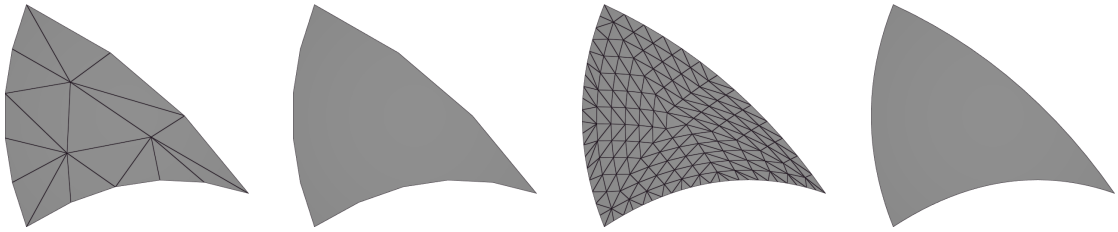
(b) Press `s` to hide elements sharing the same reference

Figure 6: Hide elements by reference

4.3.3 Tessellation for non-simplicial elements

When an element is a simplex (segment, triangle or tetrahedron), the representation is pixel exact and no tessellation shader is involved. Otherwise, the element is subdivided as in Figure 7 (see OpenGL references [Wolff, 2011, Sellers et al., 2013] and `ViZiR 4` references [Loseille and Feuillet, 2018, Feuillet et al., 2021]). There is a default subdivision parameter, and it is possible to refine this tessellation with `t` or to unrefined

it with T. To display (or un-display) this tessellation mesh, the button is L. Note that the button l is used to display the mesh. The file used for this example is TriangleP2.mesh.



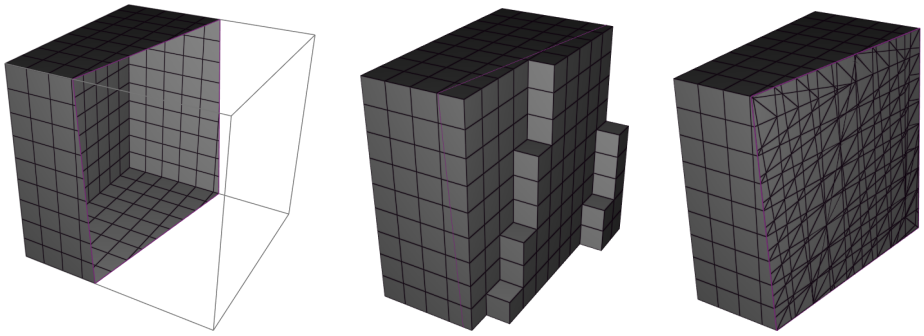
(a) Usual Tessellation (b) Rendering of tessellation defined in (a) (c) Tessellation refined with t (d) Rendering of tessellation defined in (c)

Figure 7: Tessellation for non-simplicial elements (here a P^2-Triangle)

4.3.4 Clipping and capping

When volume elements are considered, it is of particular interest to use cross sections to investigate the interior of the mesh or the solution. Two kinds of cross sections are available: clipping (the entire elements are shown) or capping (the elements are cut with respect to the cut plane). To enable one of them, press first F1. Then, to use clipping (resp. capping), press F2 (resp. F4). There are two modes (to change press again F2 or F4): one to modify the plane, the second to activate the clipping or capping. To rotate (resp. translate) the plane, use the left (resp. middle) button of the mouse. Figure 8 shows examples of clipping and capping. When clipping, only all the elements in the half-space considered (according to the cut plane), are displayed (Figure 8.b). When clipping is considered, the intersection between the elements of the mesh and the cut plane is computed and only the restriction of these elements in the cut plane is shown (Figure 8.c).

Remark. *Capping is available for **volume** elements of **degree 1** (P^1 or Q^1) only.*



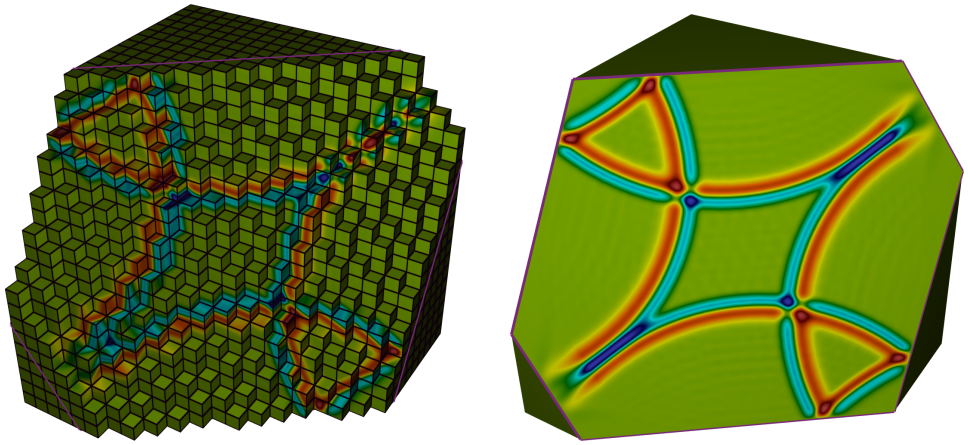
(a) Interactive selection of a plane (F1, then F2 or F4) (b) Clipping (F2) (c) Capping (F4)

Figure 8: Clipping and capping

Figure 9 shows solution clipping and capping. Here, the solution is again rendered with m.

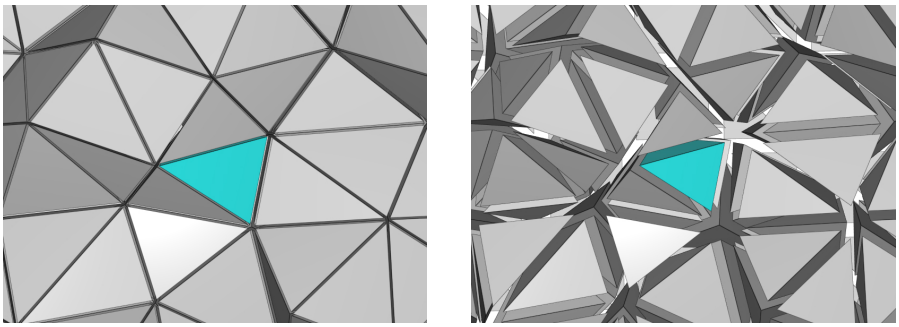
4.3.5 Shrinking

Shrink allows to reduce the size of the elements and enhance the faces of the elements. To activate shrink, use F5 as in shown in Figure 10.a. To increase or decrease shrink value, use F6 or F7 as shown in Figure 10.b.



(a) Clipping (b) Capping

Figure 9: Solution clipping and capping



(a) Activate shrink with F5 (b) Increase/decrease shrink value with F6/F7

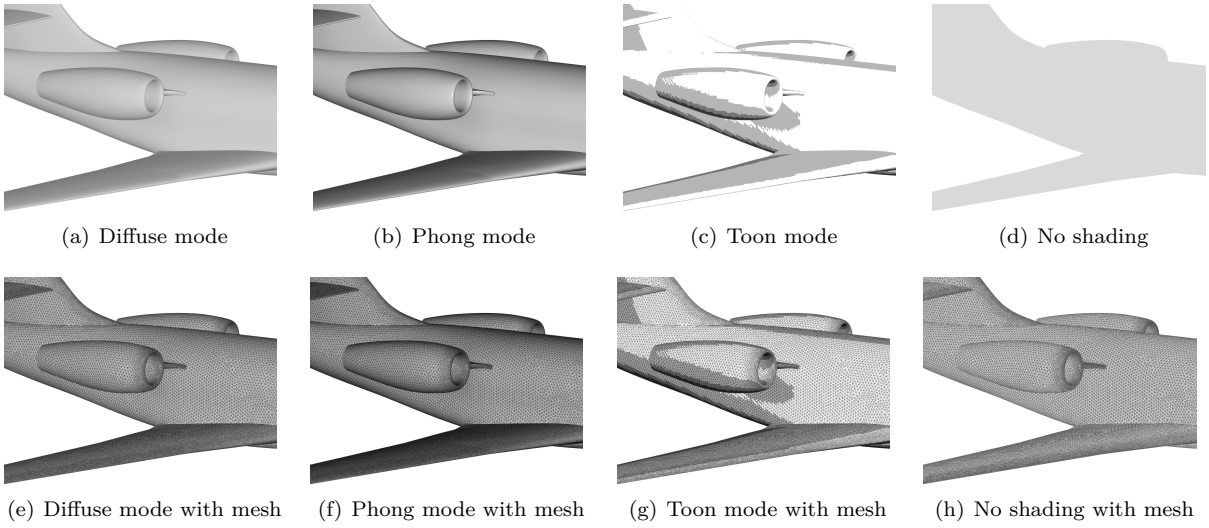
Figure 10: Shrink

4.3.6 Shading rendering modes

The button `r` is used to change the shading rendering mode. As shown in Figure 11 there are 4 shading rendering modes:

- Diffuse lighting mode.
- Phong mode: this model includes the ambient, diffuse and specular components.
- Toon mode.
- No shading mode.

By default, the diffuse mode is used.



(a) Diffuse mode (b) Phong mode (c) Toon mode (d) No shading

(e) Diffuse mode with mesh (f) Phong mode with mesh (g) Toon mode with mesh (h) No shading with mesh

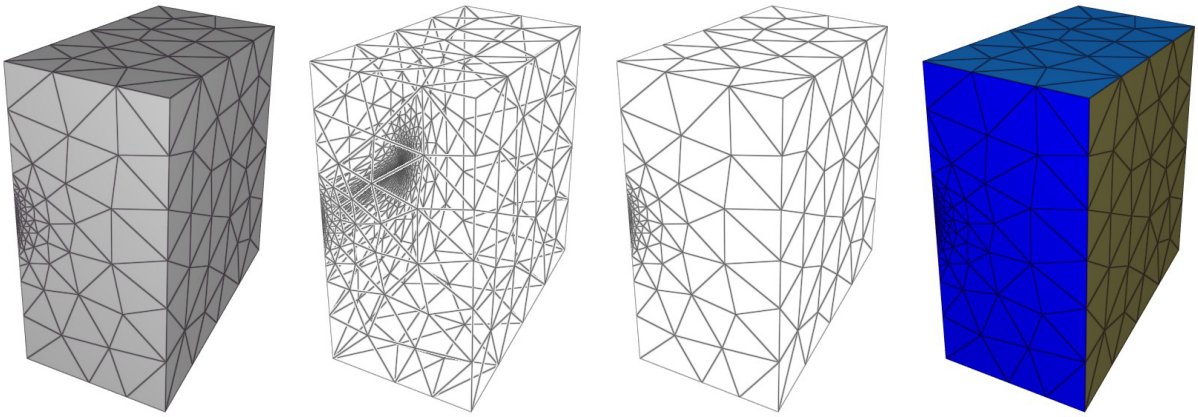
Figure 11: Shading rendering modes

4.3.7 Rendering modes

The button `f` is used to change the shading rendering mode. As shown in Figure 12 there are 4 rendering modes:

- Grey mode.
- Off mode.
- Back mode.
- Reference mode.

By default, the diffuse mode is used.



(a) Grey mode (b) Off mode (c) Back mode (d) Reference mode

Figure 12: Rendering modes

5 Mesh files

ViZiR 4 is able to display most of the elements until degree 4. In practice, many high order meshes are limited to degree 2 or 3 at the moment. There is no technical limitation to render elements of degree greater than 4. Some examples of meshes are given in the Tutorial directory and in Section 9.1.

5.1 Structure of a mesh file

The meshes used in ViZiR 4 follows the libMeshb format as explained in Section 1.4. Writing a mesh is a two-step scheme:

1. create an empty mesh file with the right version and dimension.
2. write every field (vertices, elements...).

The version, defined with MeshVersionFormatted, is equal to 1, 2 or 3 and is set such that:

1. real numbers are written in single precision (32 bits).
2. real numbers are written in double precision (64 bits).
3. same as 2 but file size may be greater than 2 GBytes.

The dimension, defined with Dimension, is equal to 2 or 3.
To define the vertices, the keyword Vertices, is followed by the number of vertices and then for each vertex, x, y, z in case of dimension 3 and a reference are written.
To define the elements, the corresponding keyword (see the next subsections), is followed by the number of such elements and then for each element, its connectivity and a reference tag are given.
Some examples are given in Section 9.

5.2 List of 0D elements

We give a (non-exhaustive) list of 0D elements:

Keyword 0D elements	
data	description
Corners	
1 integer	list of vertex indices considered as corners

5.3 List of 1D elements

We give a (non-exhaustive) list of 1D elements:

Keyword 1D elements	
data	description
Edges	
3 integers	2 vertex indices and a reference
EdgesP2	
4 integers	3 node indices and a reference
EdgesP3	
5 integers	4 node indices and a reference
EdgesP4	
6 integers	5 node indices and a reference
Ridges	
1 integer	list of Edges considered as ridges

5.4 List of 2D elements

We give a (non-exhaustive) list of 2D elements:

Keyword 2D elements	
data	description
Triangles	
4 integers	3 vertex indices and a reference
TrianglesP2	
7 integers	6 node indices and a reference
TrianglesP3	
11 integers	10 node indices and a reference
TrianglesP4	
16 integers	15 node indices and a reference
Quadrilaterals	
5 integers	4 vertex indices and a reference
QuadrilateralsQ2	
10 integers	9 node indices and a reference
QuadrilateralsQ3	
17 integers	16 node indices and a reference
QuadrilateralsQ4	
26 integers	25 node indices and a reference

5.5 List of 3D elements

We give a (non-exhaustive) list of 3D elements:

Keyword 3D elements	
data	description
Tetrahedra	
5 integers	4 vertex indices and a reference
TetrahedraP2	
11 integers	10 node indices and a reference
TetrahedraP3	
21 integers	20 node indices and a reference
TetrahedraP4	
36 integers	35 node indices and a reference
Hexahedra	
9 integers	8 vertex indices and a reference
HexahedraQ2	
28 integers	27 node indices and a reference
HexahedraQ3	
65 integers	64 node indices and a reference
HexahedraQ4	
126 integers	125 node indices and a reference
Prisms	
7 integers	6 vertex indices and a reference
PrismsP2	
19 integers	18 node indices and a reference
PrismsP3	
41 integers	40 node indices and a reference
PrismsP4	
76 integers	75 node indices and a reference
Pyramids	
6 integers	5 vertex indices and a reference
PyramidsP2	
15 integers	14 node indices and a reference
PyramidsP3	
31 integers	30 node indices and a reference
PyramidsP4	
56 integers	55 node indices and a reference

5.6 Numbering of the elements

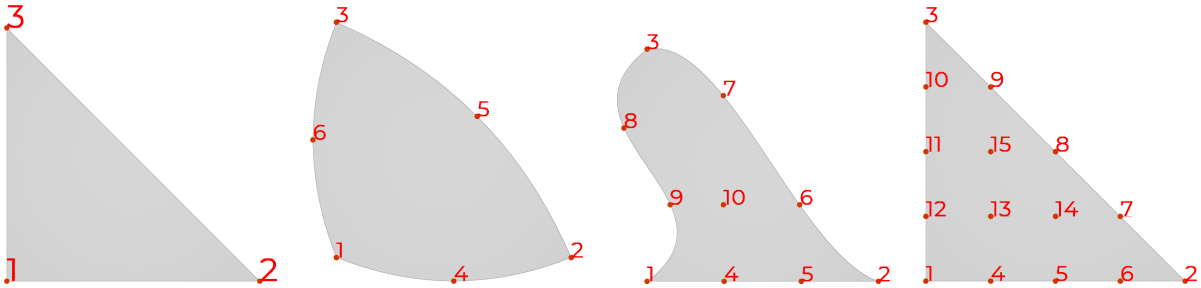
The default numbering follows [George et al., 2019] and an automatic generation procedure is proposed in the appendix of this thesis: [Feuillet, 2019].
Basically, the numbering is done as follows :

1. Number the points which are vertices in the degree one case,
2. Number the points which lie in the edges,
3. Number the points which lie in the faces,
4. Number the points which lie in the volumes.

The following figures show this default numbering.

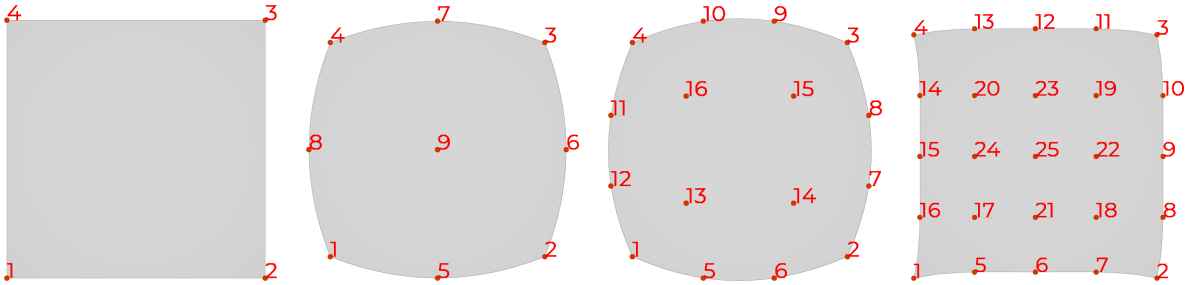
5.7 Bézier numbering

In this section, the default Bézier numbering is given for all elements. Note that in Section 5.8, it is explained how to give its own numbering.



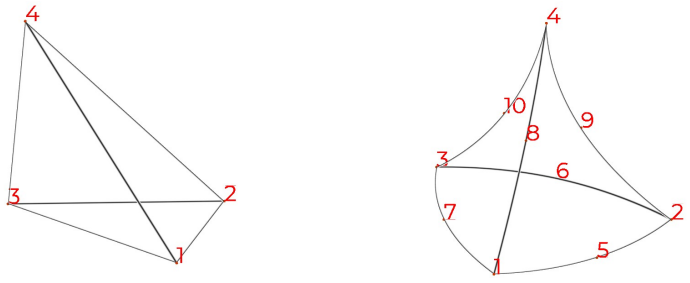
(a) P^1-triangle (b) P^2-triangle (c) P^3-triangle (d) P^4-triangle

Figure 13: Numbering of triangles



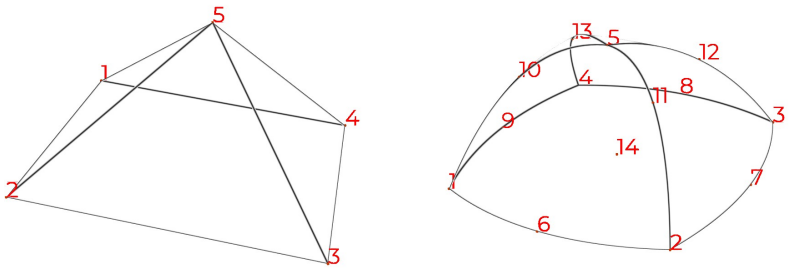
(a) Q^1-quadrilateral (b) Q^2-quadrilateral (c) Q^3-quadrilateral (d) Q^4-quadrilateral

Figure 14: Numbering of quadrilaterals



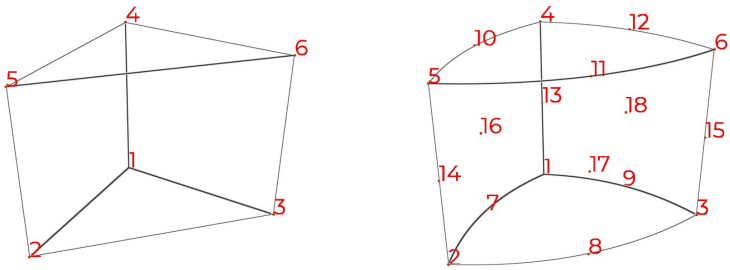
(a) P^1-tetrahedron (b) P^2-tetrahedron

Figure 15: Numbering of tetrahedra



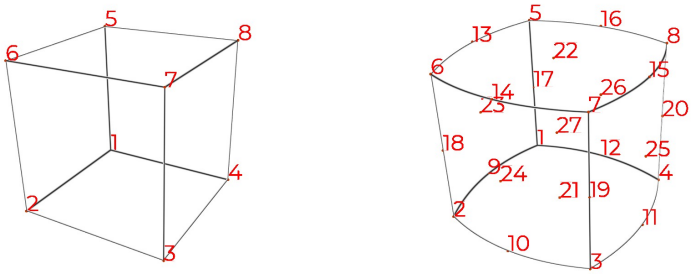
(a) P^1-pyramid (b) P^2-pyramid

Figure 16: Numbering of pyramids



(a) P^1-prism (b) P^2-prism

Figure 17: Numbering of prisms



(a) Q^1-hexahedron (b) Q^2-hexahedron

Figure 18: Numbering of hexahedra

```
P1EdgesOrdering[2] = { 0, 1 };
P2EdgesOrdering[3] = { 0, 2, 1 };
P3EdgesOrdering[4] = { 0, 3, 1, 2 };
P4EdgesOrdering[5] = { 0, 4, 1, 2, 3 };
```

```
P1TrianglesOrdering[3][3] = { { 1, 0, 0 },  { 0, 1, 0 },  { 0, 0, 1 } };

P2TrianglesOrdering[6][3] = { { 2, 0, 0 },  { 0, 2, 0 },  { 0, 0, 2 },
{ 1, 1, 0 },  { 0, 1, 1 },  { 1, 0, 1 } };

P3TrianglesOrdering[10][3] = { { 3, 0, 0 },  { 0, 3, 0 },  { 0, 0, 3 },
{ 2, 1, 0 },  { 1, 2, 0 },  { 0, 2, 1 },  { 0, 1, 2 },  { 1, 0, 2 },
{ 2, 0, 1 },  { 1, 1, 1 } };

P4TrianglesOrdering[15][3] = { { 4, 0, 0 },  { 0, 4, 0 },  { 0, 0, 4 },
{ 3, 1, 0 },  { 2, 2, 0 },  { 1, 3, 0 },  { 0, 3, 1 },  { 0, 2, 2 },
{ 0, 1, 3 },  { 1, 0, 3 },  { 2, 0, 2 },  { 3, 0, 1 },  { 2, 1, 1 },
{ 1, 2, 1 },  { 1, 1, 2 } };
```

```
Q1QuadrilateralsOrdering[4][2] = { { 0, 0 },  { 1, 0 },  { 1, 1 },  { 0, 1 } };

Q2QuadrilateralsOrdering[9][2] = { { 0, 0 },  { 2, 0 },  { 2, 2 },  { 0, 2 },
{ 1, 0 },  { 2, 1 },  { 1, 2 },  { 0, 1 },  { 1, 1 } };

Q3QuadrilateralsOrdering[16][2] = { { 0, 0 },  { 3, 0 },  { 3, 3 },  { 0, 3 },
{ 1, 0 },  { 2, 0 },  { 3, 1 },  { 3, 2 },  { 2, 3 },  { 1, 3 },  { 0, 2 },
{ 0, 1 },  { 1, 1 },  { 2, 1 },  { 2, 2 },  { 1, 2 } };

Q4QuadrilateralsOrdering[25][2] = { { 0, 0 },  { 4, 0 },  { 4, 4 },  { 0, 4 },
{ 1, 0 },  { 2, 0 },  { 3, 0 },  { 4, 1 },  { 4, 2 },  { 4, 3 },  { 3, 4 },
{ 2, 4 },  { 1, 4},  { 0, 3 },  { 0, 2 },  { 0, 1 },  { 1, 1 },  { 3, 1 },
{ 3, 3 },  { 1, 3 },  { 2, 1 },  { 3, 2 },  { 2, 3 },  { 1, 2 },  { 2, 2 } };
```

```
P1TetrahedraOrdering[4][4] = { { 1, 0, 0, 0 },  { 0, 1, 0, 0 },
{ 0, 0, 1, 0 },  { 0, 0, 0, 1 } };

P2TetrahedraOrdering[10][4] = { { 2, 0, 0, 0 },  { 0, 2, 0, 0 },
{ 0, 0, 2, 0 },  { 0, 0, 0, 2 },  { 1, 1, 0, 0 },  { 0, 1, 1, 0 },
{ 1, 0, 1, 0 },  { 1, 0, 0, 1 },  { 0, 1, 0, 1 },  { 0, 0, 1, 1 } };
```

```
P1PyramidsOrdering[5][3] = { { 0, 0, 0 },  { 1, 0, 0 },  { 1, 1, 0 },
{ 0, 1, 0 },  { 0, 0, 1 } };

P2PyramidsOrdering[14][3] = { { 0, 0, 0 },  { 2, 0, 0 },  { 2, 2, 0 },
{ 0, 2, 0 },  { 0, 0, 2 },  { 1, 0, 0 },  { 2, 1, 0 },  { 1, 2, 0 },
{ 0, 1, 0 },  { 0, 0, 1 },  { 1, 0, 1 },  { 1, 1, 1 },  { 0, 1, 1 },
{ 1, 1, 0 } };
```

```
P1PrismsOrdering[6][4] = { { 1, 0, 0, 0 },  { 0, 1, 0, 0 },  { 0, 0, 1, 0 },
{ 1, 0, 0, 1 },  { 0, 1, 0, 1 },  { 0, 0, 1, 1 } };

P2PrismsOrdering[18][4] = { { 2, 0, 0, 0 },  { 0, 2, 0, 0 },  { 0, 0, 2, 0 },
{ 2, 0, 0, 2 },  { 0, 2, 0, 2 },  { 0, 0, 2, 2 },  { 1, 1, 0, 0 },
{ 0, 1, 1, 0 },  { 1, 0, 1, 0 },  { 1, 1, 0, 2 },  { 0, 1, 1, 2 },
{ 1, 0, 1, 2 },  { 2, 0, 0, 1 },  { 0, 2, 0, 1 },  { 0, 0, 2, 1 },
{ 1, 1, 0, 1 },  { 0, 1, 1, 1 },  { 1, 0, 1, 1 } };
```

```
Q1HexahedraOrdering[8][3] = { { 0, 0, 0 },  { 1, 0, 0 },  { 1, 1, 0 },
{ 0, 1, 0 },  { 0, 0, 1 },  { 1, 0, 1 },  { 1, 1, 1 },  { 0, 1, 1 } };

static int1 Q2HexahedraOrdering[27][3] = { { 0, 0, 0 },  { 2, 0, 0 },
{ 2, 2, 0 },  { 0, 2, 0 },  { 0, 0, 2 },  { 2, 0, 2 },  { 2, 2, 2 },
{ 0, 2, 2 },  { 1, 0, 0 },  { 2, 1, 0 },  { 1, 2, 0 },  { 0, 1, 0 },
{ 1, 0, 2 },  { 2, 1, 2 },  { 1, 2, 2 },  { 0, 1, 2 },  { 0, 0, 1 },
{ 2, 0, 1 },  { 2, 2, 1 },  { 0, 2, 1 },  { 1, 1, 0 },  { 1, 1, 2 },
{ 1, 0, 1 },  { 2, 1, 1 },  { 1, 2, 1 },  { 0, 1, 1 },  { 1, 1, 1 } };
```

5.8 Reordering the geometrical elements

By default, a local numbering of nodes of high-order elements is chosen and follows the one defined by [George et al., 2019][4]. It is possible to use a different ordering. In this case, the keywords `*Ordering` (e.g. `TrianglesP2Ordering`) are used to give the positions of the degrees of freedom: see the documentation of the `libMeshb`[5]. Bézier indices are used to locate each vertex of the element. The localization of these points is supposed to be evenly spaced such that each Bézier index is an integer defined by the multiplication of a barycenter with the degree of the element. By way of example, we describe how to reorder a P^2-Triangle.

Default P^2-Triangle If a P^2-Triangle is considered, the default numbering is the following:

[4] see Section 11.1 of [George et al., 2019]
[5] see Sections 4.1 and 4.3 of https://github.com/LoicMarechal/libMeshb/blob/master/Documentation/libMeshb7.pdf

```
3
6 5
1 4 2
```

Figure 19 illustrates this numbering.

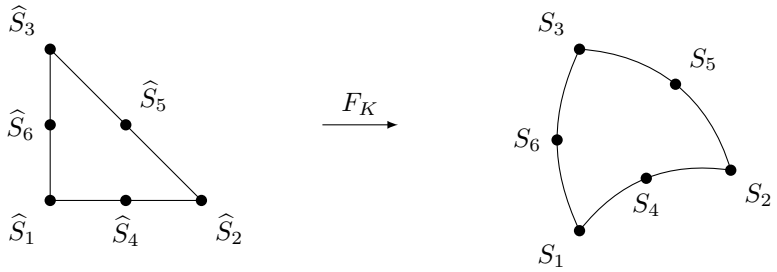


Figure 19: Default numbering of a P^2-Triangle. F_K is the geometrical transformation between a reference element (left) and the mesh element (right)

Following the documentation of the `libMeshb`, for each of the 6 vertices, a set of three Bezier indices is defined. The 3 barycentric coordinates are noted (α_1, α_2, α_3) and are summarized in Table 5. The 3 Bézier indices, noted (b_1, b_2, b_3) are deduced from these barycentric coordinates and verify, in this degree 2 case, $b_1 + b_2 + b_3 = 2$.

	α_1	α_2	α_3	b_1	b_2	b_3
V_1	1	0	0	2	0	0
V_2	0	1	0	0	2	0
V_3	0	0	1	0	0	2
V_4	0.5	0.5	0	1	1	0
V_5	0	0.5	0.5	0	1	1
V_6	0.5	0	0.5	1	0	1

Table 5: Default TrianglesP2Ordering

This results in the mesh file in

```
TrianglesP2Ordering
6
2 0 0
0 2 0
0 0 2
1 1 0
0 1 1
1 0 1
```

Hence, the file `TriangleP2_default.mesh` given in Listing 1 produces the same mesh than `TriangleP2.mesh`.

```
MeshVersionFormatted 3
Dimension 2

Vertices
6
 0.0   0.0   1
 1.0   0.15  2
 0.0   1.0   3
 0.5   0.2   4
 0.55  0.65  5
-0.1   0.5   6

TrianglesP2Ordering
6
2 0 0
0 2 0
0 0 2
1 1 0
0 1 1
1 0 1

TrianglesP2
1
1 2 3 4 5 6 1

End
```

Listing 1: TriangleP2_default.mesh

Reordering P^2-Triangle Let now consider a different ordering for this P^2-triangle, for instance:

```
5
6 4
1 2 3
```

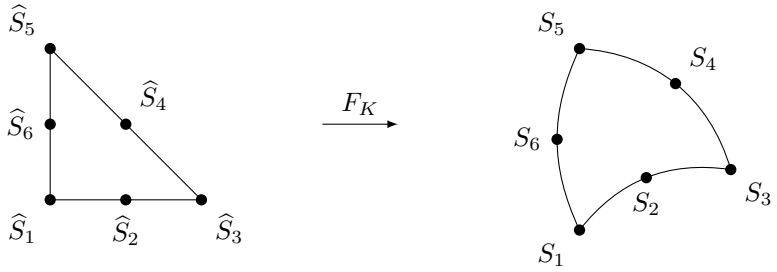


Figure 20: Renumbering of a P^2-Triangle. F_K is the geometrical transformation between a reference element (left) and the mesh element (right)

	α_1	α_2	α_3	b_1	b_2	b_3
V_1	1	0	0	2	0	0
V_2	0.5	0.5	0	1	1	0
V_3	0	1	0	0	2	0
V_4	0	0.5	0.5	0	1	1
V_5	0	0	1	0	0	2
V_6	0.5	0	0.5	1	0	1

Table 6: Another TrianglesP2Ordering

Figure 20 illustrates this different numbering. Then, the barycentric cordinates and the Bézier indices are different and are given in Table 6. This results in the mesh file in the introduction of the keyword `TrianglesP2Ordering` as below

```
TrianglesP2Ordering
6
2 0 0
1 1 0
0 2 0
0 1 1
0 0 2
1 0 1
```

Hence, the file `TriangleP2_renum.mesh` given in Listing 2 produces the same mesh than `TriangleP2.mesh` and `TriangleP2_default.mesh`.

```
MeshVersionFormatted 3
Dimension 2

Vertices
6
 0.0   0.0   1
 0.5   0.2   4
 1.0   0.15  2
 0.55  0.65  5
 0.0   1.0   3
-0.1   0.5   6

TrianglesP2Ordering
6
2 0 0
1 1 0
0 2 0
0 1 1
0 0 2
1 0 1

TrianglesP2
1
1 2 3 4 5 6 1

End
```

Listing 2: TriangleP2_renum.mesh

6 Solution files

ViZiR 4 is able to display high-order solutions until degree 10. In this section, a list of keywords on solutions is given. It follows the libMeshb format. Some examples of solutions are given in Section 9.2 and in the Tutorial directory.

6.1 Structure of a solution file

In the manner of the mesh writing, write a solution follows the libMeshb format and is a two-step scheme:

1. create an empty mesh file with the right version and dimension defined when writing a mesh file.
2. write every field (vertices, elements...).

There are three different types of solutions: solutions at vertices, low-order solutions at elements and high-order solutions at elements. In the next subsections, a list of keywords for these types of solutions are given. Some examples are given in Section 9.
When solutions are defined, the type of each solution field must be set in the following way:

1. for a scalar,
2. for a vector,
3. for a symmetric matrix,
4. for a full matrix.

Solutions at vertices. The keyword SolAtVertices is used to define a solution at vertices. The keyword SolAtVertices is followed by the number of vertices. Then, in the next line, the number of fields is given and for each field its type: 1 for a scalar, 2 for a vector, 3 for symmetric matrix and 4 for a full matrix. Finally, for each vertex (one line corresponds to one vertex) the values of the solution are given.

Low-order solutions at elements. The corresponding keyword is followed by the number of elements. Next, the number of fields and their types are given. Finally, the values are set (one line by element). Note that in the case of low-order solutions at elements, it is in fact constant solution by element which is considered.

High-order solutions at elements. The difference with low-order solutions at elements is that there is one more line giving the degree of the element and the number of degrees of freedom associated. Thus, after the two lines which gives the number of elements and the number of fields and their types, there is a new line with two integers (degree and number of d.o.f.). Finally, the degrees of freedom are set (one line by element).

6.2 Solutions at vertices

Considering solutions at vertices (P^1 or Q^1 solution), the keyword is:

- SolAtVertices

6.3 Low-order solutions at elements

Considering low-order solutions at elements (P^1 or Q^1 solution), the keywords are:

- SolAtEdges
- SolAtTriangles
- SolAtQuadrilaterals
- SolAtTetrahedra
- SolAtPyramids
- SolAtPrisms
- SolAtHexahedra

6.4 High-order solutions at elements

Considering high-order solutions at elements, the keywords are:

- `HOSolAtEdgesP1` → `HOSolAtEdgesP4`
- `HOSolAtTrianglesP1` → `HOSolAtTrianglesP4`
- `HOSolAtQuadrilateralsQ1` → `HOSolAtQuadrilateralsQ4`
- `HOSolAtTetrahedraP1` → `HOSolAtTetrahedraP4`
- `HOSolAtPyramidsP1` → `HOSolAtPyramidsP4`
- `HOSolAtPrismsP1` → `HOSolAtPrismsP4`
- `HOSolAtHexahedraQ1` → `HOSolAtHexahedraQ4`

6.5 Types of solutions fields

As explained previously, 4 types of solutions fields exist:

1. for a scalar,
2. for a vector,
3. for a symmetric matrix,
4. for a full matrix.

Remark. *Note that at the moment,* `ViZiR 4` *is able to display only scalar fields. Thus, when a vector is considered, its norm is plotted and when a matrix is considered, its determinant is displayed.*

The sizes of the types according to the dimension are summarized in Table 7.

Type	Name of the type	Size in dimension 2	Size in dimension 3
1	scalar	1	1
2	vector	2	3
3	symmetric matrix	3	6
4	full matrix	4	9

Table 7: Size of the different types according to the dimension

`ViZiR 4` reads matrices by column.
If a symmetric matrix is considered such as

$$\mathcal{M} = \begin{pmatrix} \mathcal{M}_{00} & \mathcal{M}_{01} \\ \text{sym} & \mathcal{M}_{11} \end{pmatrix} \quad \text{or} \quad \mathcal{M} = \begin{pmatrix} \mathcal{M}_{00} & \mathcal{M}_{01} & \mathcal{M}_{02} \\ \text{sym} & \mathcal{M}_{11} & \mathcal{M}_{12} \\ \text{sym} & \text{sym} & \mathcal{M}_{22} \end{pmatrix},$$

the solution field should be written

- in dimension 2: $\mathcal{M}_{00}$ $\mathcal{M}_{01}$ $\mathcal{M}_{11}$,
- in dimension 3: $\mathcal{M}_{00}$ $\mathcal{M}_{01}$ $\mathcal{M}_{11}$ $\mathcal{M}_{02}$ $\mathcal{M}_{12}$ $\mathcal{M}_{22}$.

When considering a full matrix such as

$$\mathcal{M} = \begin{pmatrix} \mathcal{M}_{00} & \mathcal{M}_{01} \\ \mathcal{M}_{10} & \mathcal{M}_{11} \end{pmatrix} \quad \text{or} \quad \mathcal{M} = \begin{pmatrix} \mathcal{M}_{00} & \mathcal{M}_{01} & \mathcal{M}_{02} \\ \mathcal{M}_{10} & \mathcal{M}_{11} & \mathcal{M}_{12} \\ \mathcal{M}_{20} & \mathcal{M}_{21} & \mathcal{M}_{22} \end{pmatrix}$$

the solution field should be written

- in dimension 2: $\mathcal{M}_{00}$ $\mathcal{M}_{10}$ $\mathcal{M}_{01}$ $\mathcal{M}_{11}$,
- in dimension 3: $\mathcal{M}_{00}$ $\mathcal{M}_{10}$ $\mathcal{M}_{20}$ $\mathcal{M}_{01}$ $\mathcal{M}_{11}$ $\mathcal{M}_{12}$ $\mathcal{M}_{02}$ $\mathcal{M}_{12}$ $\mathcal{M}_{22}$.

6.5.1 Rendering of vector fields

Vector arrows can be displayed in the Tools > Solution Visualization menu. Arrows are by default in black but can be colored as the field. There are 3 modes: all unit, scaled by elements norms or scaled by elements sized. Figure 21 shows an example.

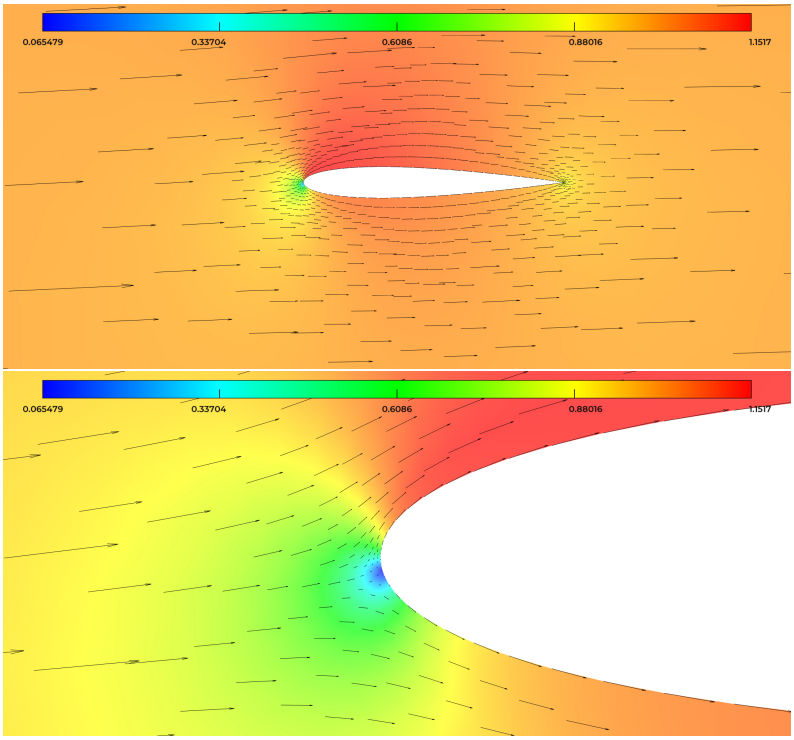


Figure 21: Rendering of a vector field

6.5.2 Rendering of metric fields

An example of rendering of metric field is shown in Figure 22.

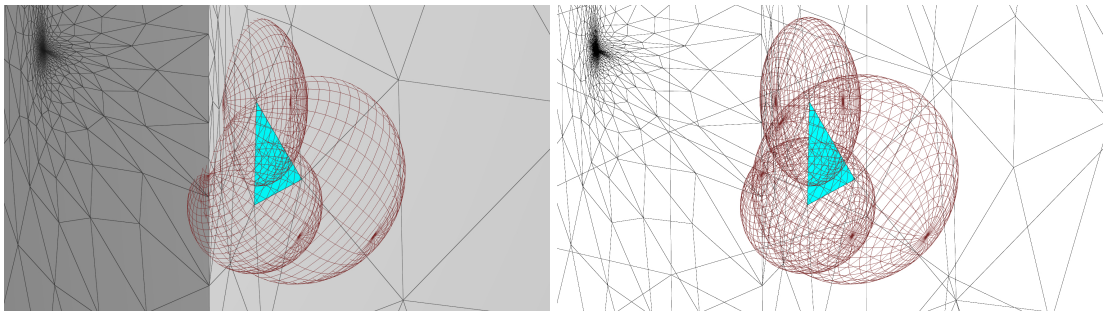


Figure 22: Rendering of a metric field

6.6 Case of several fields

When there are several fields, all values are written node after node for a solution at vertices and element after element for a solution at elements. For instance, let consider this piece of a solution file:

```
MeshVersionFormatted 3

Dimension 2

SolAtVertices
5000
3 1 2 1
1.16120015122728 201.598239372078 -3.15392848724367e-05 267499.97
1.16120020930559 201.598188004068 -3.43853328933686e-05 267499.98
1.16120017860897 201.598073350552 -2.03890366946454e-05 267499.95
1.16120013398399 201.598172471341 0.000125709082451808 267499.96
1.16119986392418 201.598280050529 6.06555114922069e-05 267499.90
...
...
```

There are 3 solutions fields, a scalar, a vector (of dimension 2) and a scalar. For the first vertex, the value of the first field is 1.16120015122728, the value of the vector is (201.598239372078 -3.15392848724367e-05) and the last field 267499.97.
In the case of high-order solutions, for each element, the degrees of freedom are written nodes after nodes. Thus, the format is:
for $i = 1$, number of nodes
  for $iFld = 1$, number of solution fields
    for $iCmp = 1$, size of the field $iFld$
      write the value of the component $iCmp$ of the field $iFld$ at the node i.
An example is the following:

```
HOSolAtQuadrilateralsQ1
2771
3 1 2 1
5 36
1.23 342.77 11.80 303560.53  1.24 342.76 11.78 303580.91 ... ...
... ...
... ...
```

6.7 Reordering the solutions nodes

By default, the nodes of the high order solutions are supposed to be evenly spaced and again follow the numbering defined by George et al [George et al., 2019]. It is possible to use its own set of nodes. In this case, the user specifies the coordinates of the nodes following the format of the `libMeshb`[6] with the keywords `*NodesPositions`.
The nature of the data given under the keyword `*NodesPositions` depends on the type of element considered. It can be barycentric coordinates (whose sum is by definition equal to 1) or the interpolating coordinates in the reference element. We summarized this in Table 8. Note that in this table, [deg] denotes the degree of the geometrical element, which can be equal to 1, 2, 3 or 4. For instance, `HOSolAtEdgesP[deg]NodesPositions` denotes `HOSolAtEdgesP1NodesPositions`, `HOSolAtEdgesP2NodesPositions`, `HOSolAtEdgesP3NodesPositions` or `HOSolAtEdgesP4NodesPositions`.

6.7.1 An example of reordering of HO solution (Gauss Lobatto) nodes in a quadrilateral

If we want to use Gauss Lobatto points of degree 3 in a quadrilateral (of degree 1) as in Figure 23, where 16 degrees of freedom are introduced, the underlying `HOSolAtQuadrilateralsQ1NodesPositions` are given as following:

[6] see Section 4.4 of https://github.com/LoicMarechal/libMeshb/blob/master/Documentation/libMeshb7.pdf

Keyword	Data	Description
HOSolAtEdgesP[deg]NodesPositions	2 reals	set of **barycentric** coordinates for each of the high order solution nodes
HOSolAtHexahedraQ[deg]NodesPositions	3 reals	set of **interpolating** coordinates for each of the high order solution nodes
HOSolAtPrismsP[deg]NodesPositions	4 reals	set of 3 **barycentric** coordinates and an **interpolating** coefficient for each of the high order solution nodes
HOSolAtPyramidsP[deg]NodesPositions	3 reals	set of **interpolating** coordinates for each of the high order solution nodes
HOSolAtQuadrilateralsQ[deg]NodesPositions	2 reals	set of **interpolating** coordinates for each of the high order solution nodes
HOSolAtTetrahedraP[deg]NodesPositions	4 reals	set of **barycentric** coordinates for each of the high order solution nodes
HOSolAtTrianglesP[deg]NodesPositions	3 reals	set of **barycentric** coordinates for each of the high order solution nodes

Table 8: List of solution keywords

```
HOSolAtQuadrilateralsQ1NodesPositions
16
0.000000 0.000000
0.276393 0.000000
0.723607 0.000000
1.000000 0.000000
0.000000 0.276393
0.276393 0.276393
0.723607 0.276393
1.000000 0.276393
0.000000 0.723607
0.276393 0.723607
0.723607 0.723607
1.000000 0.723607
0.000000 1.000000
0.276393 1.000000
0.723607 1.000000
1.000000 1.000000
```

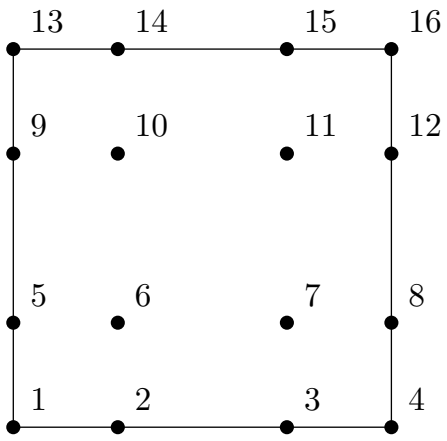


Figure 23: Example of Gauss Lobatto points used in a Quadrilateral

Following this, the mesh and solution files are defined in Listings 3 and 4. The rendering of this Gauss-Lobatto form function is shown in Figure 24.

```
MeshVersionFormatted 3

Dimension 2

Vertices
4
0 0 1
1 0 1
1 1 1
0 1 1

Quadrilaterals
1
1 2 3 4 1

End
```

Listing 3: `GaussLobatto.mesh`

```
MeshVersionFormatted 3

Dimension 2

HOSolAtQuadrilateralsQ1NodesPositions
16
0.000000 0.000000
0.276393 0.000000
0.723607 0.000000
1.000000 0.000000
0.000000 0.276393
0.276393 0.276393
0.723607 0.276393
1.000000 0.276393
0.000000 0.723607
0.276393 0.723607
0.723607 0.723607
1.000000 0.723607
0.000000 1.000000
0.276393 1.000000
0.723607 1.000000
1.000000 1.000000

HOSolAtQuadrilateralsQ1
1
1 1
3 16
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0

End
```

Listing 4: `GaussLobatto.sol`

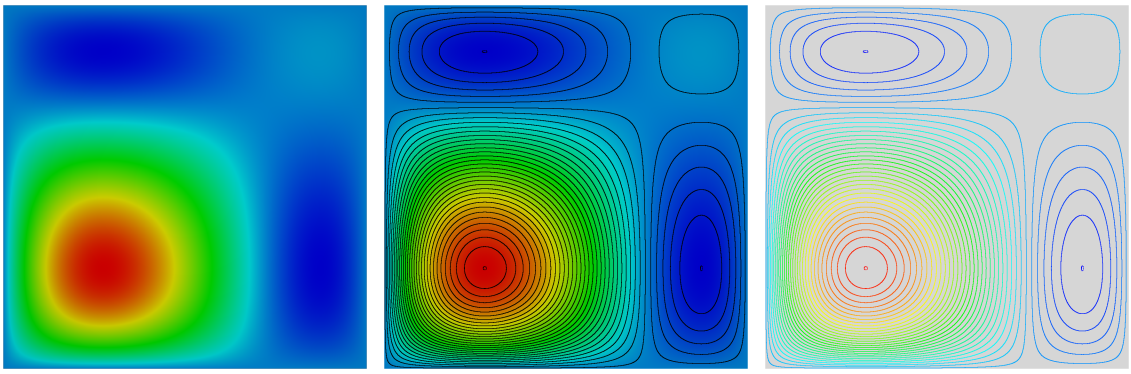


Figure 24: A Gauss Lobatto basis function

6.7.2 An example of reordering of HO solution nodes in a triangle

Let now show an example of reordering in a triangle of degree 1. The keyword used is then `HOSolAtTrianglesP1NodesPositions`. In this example, the degree of the solution is 3 and there are 10 degrees of freedom and then 10 nodes needs to be defined. These nodes are defined in Listing 6 as barycentric coordinates (3 values between 0 and 1) as explained in Table 8. Figure 25 shows the rendering of this basis function.

```
MeshVersionFormatted 3

Dimension
2

Vertices
3
0.0   0.0   1
1.0   0.0   1
0.0   1.0   1

Triangles
1
1 2 3 1

End
```

Listing 5: `reordTriSolP3.mesh`

```
MeshVersionFormatted 3

Dimension 2

HOSolAtTrianglesP1NodesPositions
10
1.0 0.0 0.0
0.0 1.0 0.0
0.0 0.0 1.0
0.788675 0.211325 0.0
0.211325 0.788675 0.0
0.0 0.788675 0.211325
0.0 0.211325 0.788675
0.211325 0.0 0.788675
0.788675 0.0 0.211325
0.333333 0.333333 0.333333

HOSolAtTrianglesP1
1
1 1
3 10
0 0 0 0 0 1 0 0 0 0

End
```

Listing 6: `reordTriSolP3.sol`

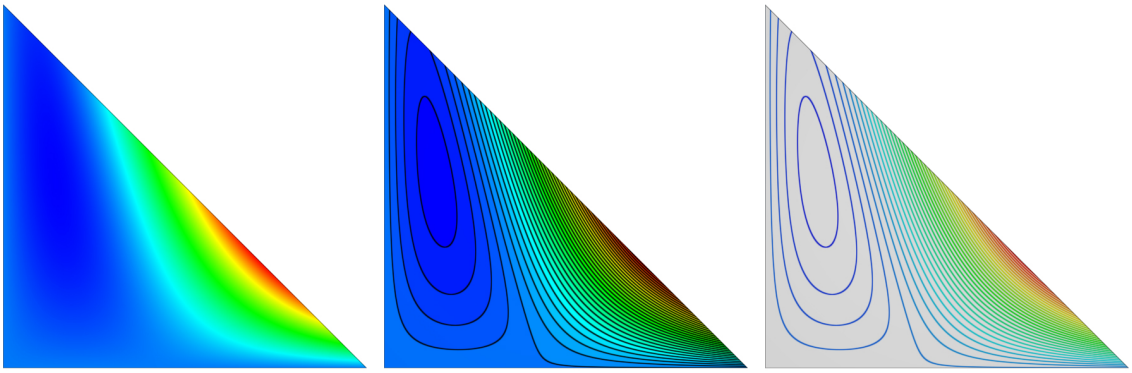


Figure 25: A degree 3 solution with an ordering defined by Listing 6

6.8 Solutions names

Using keyword `ReferenceStrings`, it is possible to attach a solution name to each solution field. Each field is defined by 3 parameters: an integer corresponding to the solution keyword (example 62 for `GmfSolAtVertices`, 107 for `GmfHOSolAtTrianglesP1`...), an integer for the index of the field (starting at 1) and a string for the name.
Here is an example for a `GmfSolAtVertices` (integer 62):

```
ReferenceStrings
3
62  1  Pressure
62  2  Velocity
62  3  Temperature
```

Here is an example of use of `libMeshb`[7] code to define it:

```
GmfSetKwd(MshIdx, GmfReferenceStrings, 3);
GmfSetLin(MshIdx, GmfReferenceStrings, GmfSolAtVertices, 1, "Pressure");
GmfSetLin(MshIdx, GmfReferenceStrings, GmfSolAtVertices, 2, "Velocity");
GmfSetLin(MshIdx, GmfReferenceStrings, GmfSolAtVertices, 3, "Temperature");
```

[7] https://github.com/LoicMarechal/libMeshb

7 Customizing ViZiR 4 with default.vizir file

To define rendering options in ViZiR 4, a `default.vizir` or `DEFAULT.vizir` file can to be created in the directory. Keywords are used to activate these options. Note that these keywords are not case sensitive.

7.1 Palette

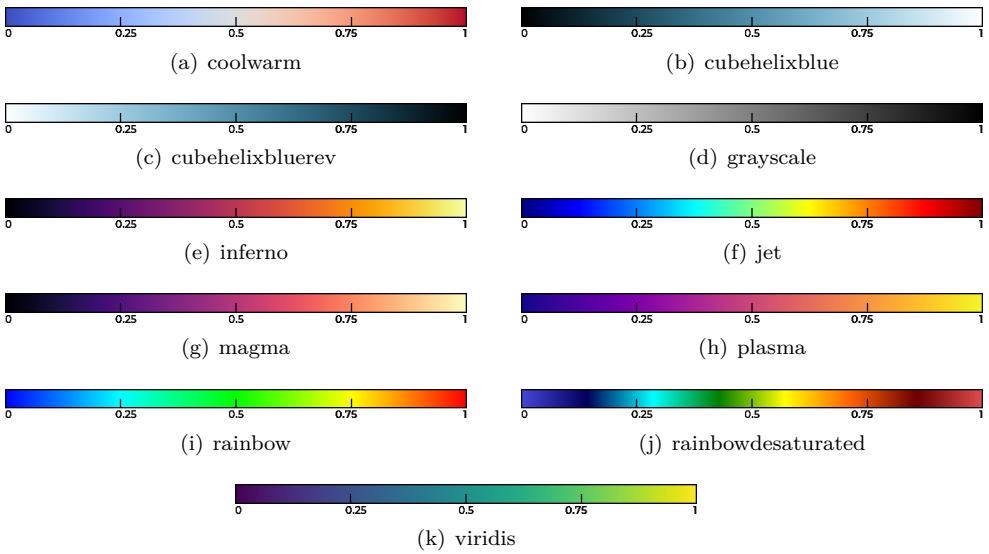
It is possible to give (i.e. force) the range of the palette with the keyword `Palette`. It is defined from 5 doubles and can be not linear. Here is an example:

```
Palette
-1   -0.5   0   0.5   1
```

7.2 Colormap

The colormap can be changed with `Colormap`. Figure 26 sums up the ones available in ViZiR 4. Again, the names of the colormaps are not case sensitive. Here is an example:

```
Colormap
CoolWarm
```



(a) coolwarm (b) cubehelixblue

(c) cubehelixbluerev (d) grayscale

(e) inferno (f) jet

(g) magma (h) plasma

(i) rainbow (j) rainbowdesaturated

(k) viridis

Figure 26: Colormaps available in ViZiR 4

7.3 User Cut Planes

The user can give cut plane equations that will be displayed when ViZiR 4 is launched. Note that the mesh is also displayed. To hide parts of the mesh (by reference number), pick an element and press **s**. Two keywords exist to define cut planes and are defined hereafter. The keyword `CutPlaneNormal` allows to define cut planes from a point and a normal while the coefficients of the plane are directly given with the keyword `CutPlaneEquation`.

7.3.1 CutPlaneEquation

After the keyword `CutPlaneEquation`, an integer defines the number of planes. Then, for each plane, the 4 coefficients a, b, c, d are given where the plane equation is $ax + by + cz + d = 0$. An example is the following:

```
CutPlaneEquation
1
1.0 0. 0. 0.5
```

7.3.2 CutPlaneNormal

The format for the keyword `CutPlaneNormal` is the following.

- The first line is the keyword `CutPlaneNormal`.
- The second line is the number of planes.
- Then, for each plane, there are two lines:
  - The first line is composed of 3 reals corresponding to the coordinates (x, y, z) of a point of the plane.
  - The second line is composed of 3 reals corresponding to the normal of the plane.

  Then, it goes to the next plane.

An example of parameters if the following:

```
CutPlaneNormal
3
0. 0.5 0.
0. 1. 0.
0.5 0. 0.
1. 0. 0.
0. 0. 0.5
0. 0. 1.
```

and is shown in Figure 27. In this example, there are 3 planes. The first plane is defined with the point (0. 0.5 0.) and its normal is directed in y-direction. The second plane is defined with the point (0.5 0. 0.) and its normal is directed in x-direction. Finally, the last plane is defined with the point (0. 0. 0.5) and its normal is directed in z-direction.

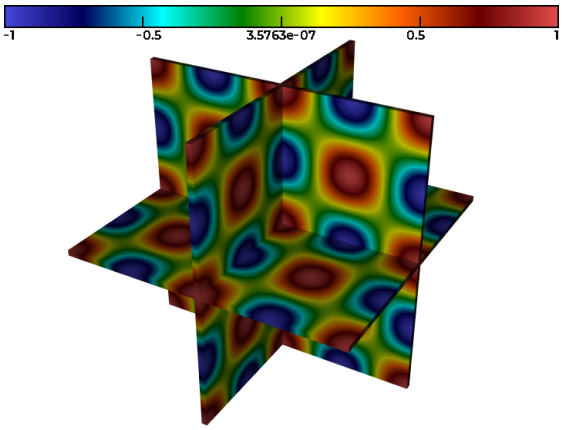


Figure 27: Example of cutplane option

Note that both keywords `CutPlaneEquation` and `CutPlaneNormal` can be used in the same time and in this case the number of planes is the addition given by each keyword.

7.4 User Capping Planes

Similarly to user cut planes, user capping planes can be defined and works in the same way with two keywords: `CapPlaneNormal` and `CapPlaneEquation`.

7.4.1 CapPlaneEquation

After the keyword `CapPlaneEquation`, an integer defines the number of planes. Then, for each plane, the 4 coefficients a, b, c, d are given where the plane equation is $ax + by + cz + d = 0$. An example is the following:

```
CapPlaneEquation
1
1.0 0. 0. 0.5
```

7.4.2 CapPlaneNormal

The format for the keyword `CapPlaneNormal` is the following.

- The first line is the keyword `CapPlaneNormal`.
- The second line is the number of planes.
- Then, for each plane, there are two lines:
  - The first line is composed of 3 reals corresponding to the coordinates (x, y, z) of a point of the plane.
  - The second line is composed of 3 reals corresponding to the normal of the plane.

  Then, it goes to the next plane.

An example of parameters if the following:

```
CapPlaneNormal
3
0. 0.5 0.
0. 1. 0.
0.5 0. 0.
1. 0. 0.
0. 0. 0.5
0. 0. 1.
```

Note that both keywords `CapPlaneEquation` and `CapPlaneNormal` can be used in the same time and in this case the number of planes is the addition given by each keyword.

7.5 Isolines

By default, when isolines are rendered with the button o, these isolines are distributed regularly and their number is 35. It is possible to change this number of isolines or to give a list of specific isolines.

7.5.1 Regular isolines (by default)

The keyword `numberofsubisolines` is used to change the number of isolines. Note that the colorbar is subdivided in 4 parts and this number gives the subdivision of each part. More precisely, if we choose as `numberofsubisolines` a number n, the number of isolines rendered will be $4(n+1)-5$. By default, $n = 9$ and 35 isolines are rendered. The format is the following:

```
numberofsubisolines
3
```

Remark. *$n = 9$ and 35 isolines are the maximum number possible. If the user choose a number bigger than $n = 9$, the number of isolines will be 35.*

7.5.2 List of isolines

It is possible to give a list of isolines to be represented. The keyword is `isolines`. First, we give the number of isolines (an integer) and then the list of isolines. An example is the following:

```
isolines
5
-0.005
0.005
-0.0035
0.0035
0
```

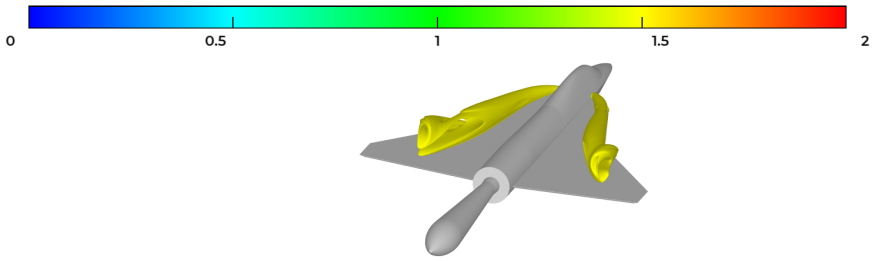
Note that in this case, the keyword `numberofsubisolines` is not taken into account even if it is read.

Remark. *The maximum number of isolines possible is 9. If more isolines are given in the list, only the first 9 values will be read.*

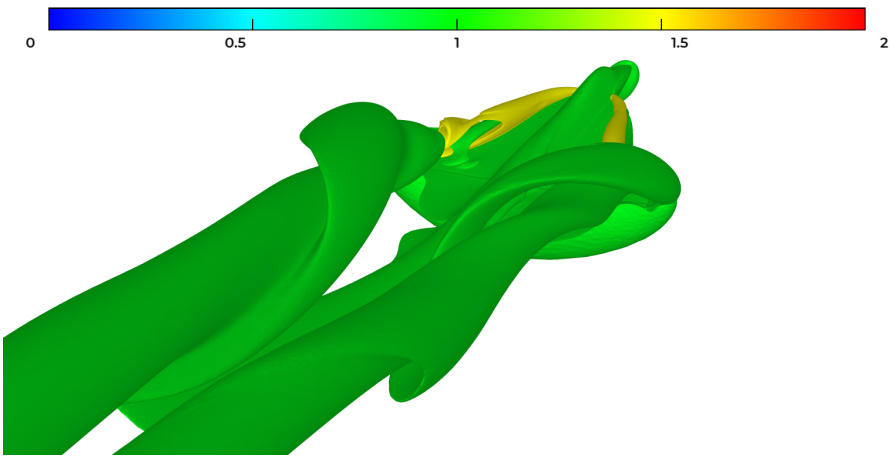
7.6 Isosurfaces

A list of isosurfaces can be given for any volume elements of degree 1 with the keyword `isosurfaces`. First, we give the number of isosurfaces (an integer) and then the list of isosurfaces.
An example is the following:

```
isosurfaces
2
0.95
1.5
```



(a) 1 isosurface



(b) 2 isosurfaces

Figure 28: Example of isosurfaces

7.7 IsosurfacesField

The field used to create the isosurface can be specified with the keyword `isosurfacesfield`. Note that the first field start is 1. Here is an example:

```
isosurfacesfield
2
```

7.8 Displayfield

By default, the first field is displayed. The keyword `displayfield` allows to change the initial solution field displayed. An integer is given as following :

```
displayfield
2
```

The first field can be displayed with `displayfield` equal to 1. Note that if this integer is wrong (for instance greater than the number of solutions fields), the first field is displayed. The button > (resp. <) shows the next (resp. previous) solution field if it exists.

7.9 WarpVec

The keyword `warpvec` allows to warp the mesh. Mesh vertices are deformed following a vector field. There are two parameters, an integer to define the vector field to use and a factor scale (real):

```
WarpVec
1
10.
```

Note that the 1 denotes the first field.

7.10 Checksurf

By default, the checksurf reconstruction mode is not activated. However, it is possible to force to use it with the keyword `checksurf` and an integer 0 or 1. To force it, use :

```
checksurf
1
```

7.11 Scrolling

By default, the scrolling is activated. Note that it can be set active by default in preferences menu. To be able to zoom in or out with the mouse wheel, the keyword `scrolling` must be set to 1:

```
scrolling
1
```

7.12 Boundingbox

It is possible to change the values of the bounding box. It is for instance useful to have a desired view. The keyword `boundingbox` is followed by 6 reals $x_{\min}$, $x_{\max}$, $y_{\min}$, $y_{\max}$, $z_{\min}$ and $z_{\max}$.

```
boundingbox
-2.5 2.5 -5 5 0 3
```

7.13 RefColor

By default, several colors are used to display the elements in color in the reference color. The rendering mode can be modified with `f`. The keyword `refcolor` can modify these reference colors. It is followed by the number of colors (an integer), and for each color three integers between 0 and 255 to define the rgb (red green blue). The color is defined by a modulo with the number of colors.

```
refcolor
8
255 255 0
128 128 0
0 0 128
255 0 0
128 0 128
0 128 0
64 64 64
128 0 0
```

7.14 Coordpalette

By default, the palette is set on the top of the window but its coordinates can be modified with the keyword `coordpalette` followed by the coordinates.
Here is an example for a palette on the top:

```
coordpalette
-0.9 0.95
-0.9 0.85
0.9 0.85
0.9 0.95
```

On the bottom:

```
coordpalette
-0.8 -0.9
-0.8 -0.8
0.8 -0.8
0.8 -0.9
```

On the left:

```
coordpalette
-0.95 -0.8
-0.85 -0.8
-0.85 0.8
-0.95 0.8
```

On the right:

```
coordpalette
0.95 0.8
0.85 0.8
0.85 -0.8
0.95 -0.8
```

7.15 Example of a complete file

In Listing 7, a complete example of file `DEFAULT.vizir` is given.

```
Palette
-1    -0.5    0    0.5    1

Colormap
Jet

CutPlane
3
0.5 0.5 0.5
0. 1. 0.
0.5 0.5 0.5
1. 0. 0.
0.5 0.5 0.5
0. 0. 1.
```

Listing 7: Example of a DEFAULT.vizir

8 Customizing ViZiR 4 with state files

It is possible to save rendering options and load it when ViZiR 4 is launched. For this purpose, in ViZiR 4, by pressing the button w, a file vizir.state is generated. Otherwise, the dedicated menu can be used to save a state file. It contains information on the view (center of the view, rotation, translation...), the plane equation, the solution (on/off), the isolines (on/off), the level of tessellation for high-order elements, the lines thickness...
The Listing 9 gives the list of state keywords.
Admissible Colormap names: rainbow, grayscale, coolwarm, magma, inferno, plasma, viridis, jet, cubehelixblue, cubehelixbluerev, rainbowdesaturated, cfmap_tecplot, nato1, nato2, gammawhite.

Keyword	Data	Description
WindowSize	2 integers	size of the window
Center	3 reals	center of the view in the real physical world
Rotation	4 reals	rotation of the view
Translation	3 reals	translation of the view in the window world
Fovy	1 real	zoom
CutPlaneEquation	4 reals	a, b, c, d from equation $ax+by+cz+d = 0$
UseCutPlane	boolean	activate cut/capping plane
ShowCut	boolean	display the edges of the cut plane
Capping	boolean	activate capping
LineOn	boolean	display the mesh edges
TessOn	boolean	display the tessellation
TessLevel	1 integer	tessellation level for high-order meshes
EdgeOn	boolean	display the edges
SolOn	1 integer	display solution: 0 off, 1 exact, 2 filled
IsoOn	1 integer	display isolines: 0 off, 1 colors, 2 black
PalOn	boolean	display the palette
UsePalette	boolean	use the given palette
Palette	5 reals	values of the palette (used if `UsePalette` is true)
CurSol	1 integer	solution type: 0 no sol., 1 sol. at vertices, 2 sol. at elements
ithSol	1 integer	index of the field displayed (1 is the first)
IsoSiz	1 real	size of the isolines
VecComp	1 integer	vector component: 0 norm, 1 x, 2 y, 3 z
TextOn	boolean	display the text
LineSurfOn	boolean	show the mesh lines on surfaces
TransOn	boolean	activate transparency
ShaMod	1 integer	change the shading rendering: 0 usual, 1 Phong, 2 toon, 3 no
RenMod	1 integer	change the rendering: off, back, ref, grey
RenEdgMod	1 integer	change the rendering of edges: back, ref, grey
WireSiz	1 real	size of the lines
Colormap	1 string	set the colormap
UseBoundingBox	boolean	use the given bounding box
BoundingBox	6 reals	xmin, xmax, ymin, ymax, zmin, zmax (used if `UseBoundingBox` is true)
HideSurfOn	boolean	hide all surfaces
HideSolSurfOn	boolean	hide solutions on surfaces
HideSolVolOn	boolean	hide solutions on volumes
CutoffBelowOn	boolean	use the given cutoff below value
CutoffBelowVal	1 real	filter values below (used if `CutoffBelowOn` is true)
CutoffAboveOn	boolean	use the given cutoff above value
CutoffAboveVal	1 real	filter values above (used if `CutoffAboveOn` is true)
NumberOfHiddenRefObjects	1 integer + list	number of hidden surface + list of types and references

Table 9: List of state keywords

9 Examples of rendering

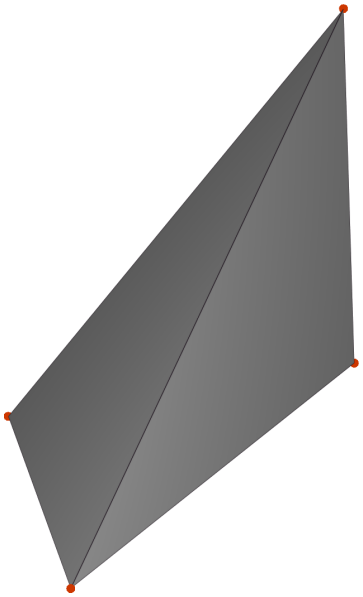
Many examples and samples can be found in our website https://pyamg.saclay.inria.fr/vizir4examples.html.

9.1 Meshes examples

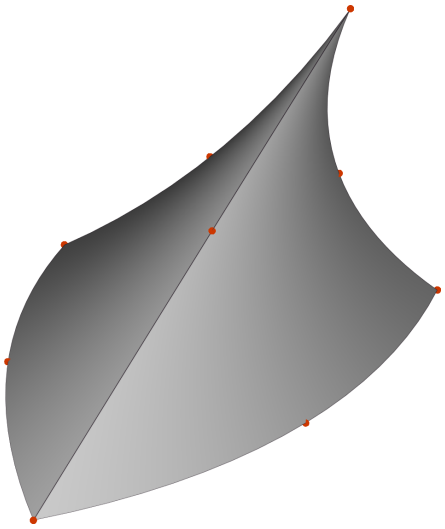
9.1.1 Single elements samples

Several single elements are described in the following.

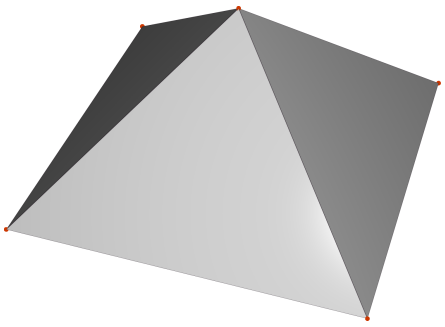
```
MeshVersionFormatted 3

Dimension
3

Vertices
4
0.0   0.0  0.0  1
1.0   0.0  0.0  1
0.0   1.0  0.0  1
0.0   1.0  1.0  1

Triangles
4
2  3  4  1
4  3  1  1
4  1  2  1
2  1  3  1

Tetrahedra
1
1  2  3  4  1

End
```



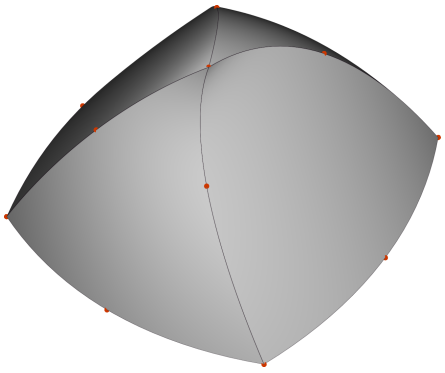
```
MeshVersionFormatted 3

Dimension
3

Vertices
10
0.0   0.0  0.0  1
1.0   0.0  0.0  1
0.0   1.0  0.0  1
0.5   0.5  1.0  1
0.5  -0.1  0.0  1
0.6   0.6  0.0  1
-0.1  0.5  0.0  1
0.3   0.3  0.5  1
0.6   0.3  0.5  1
0.3   0.6  0.5  1

TrianglesP2
4
2  3  4  6  10  9  1
4  3  1  10  7  8  1
4  1  2  8  5  9  1
2  1  3  5  7  6  1

TetrahedraP2
1
1  2  3  4  5  6  7  8  9  10  1

End
```



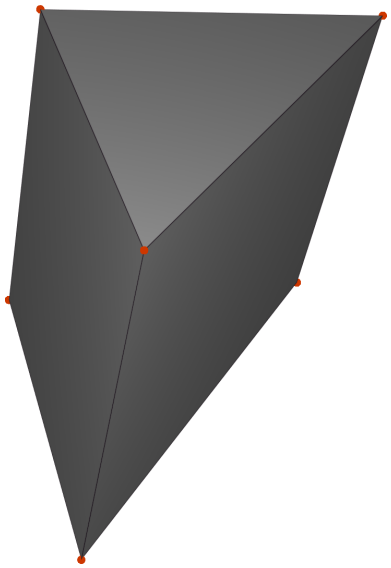
```
MeshVersionFormatted 3

Dimension
3

Vertices
5
0.0   0.0  0.0  1
1.0   0.0  0.0  1
1.0   1.0  0.0  1
0.0   1.0  0.0  1
0.5   0.5  0.5  1

Triangles
4
1  2  5  1
2  3  5  1
3  4  5  1
1  5  4  1

Quadrilaterals
1
1  4  3  2  1

Pyramids
1
1  2  3  4  5  1

End
```



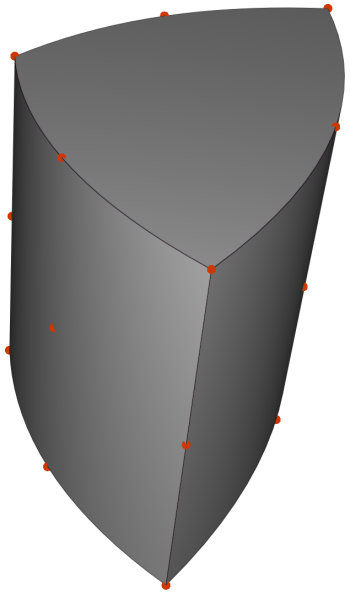
```
MeshVersionFormatted 3

Dimension
3

Vertices
14
0.0   0.0  0.0  1
1.0   0.0  0.0  1
1.0   1.0  0.0  1
0.0   1.0  0.0  1
0.5   0.5  0.5  1
0.5  -0.1  0.0  1
1.1   0.5  0.0  1
0.5   1.1  0.0  1
-0.1  0.5  0.0  1
0.2   0.3  0.4  1
0.8   0.3  0.4  1
0.8   0.7  0.4  1
0.2   0.7  0.4  1
0.5   0.5  0.0  1

TrianglesP2
4
1  2  5  6  11  10  1
2  3  5  7  12  11  1
3  4  5  8  13  12  1
1  5  4  10  13  9  1

QuadrilateralsQ2
1
1  4  3  2  9  8  7  6  14  1

PyramidsP2
1
1  2  3  4  5  6  7  8  9  10  11  12  13  14  1

End
```



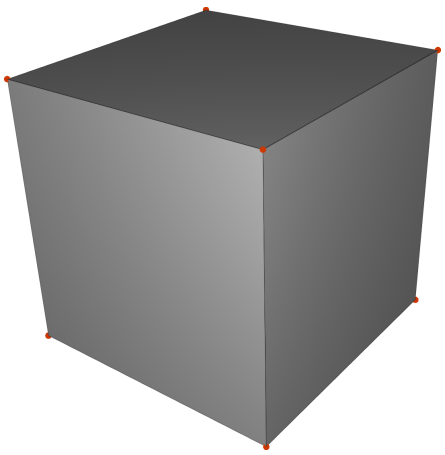
```
MeshVersionFormatted 3

Dimension
3

Vertices
6
0.0   0.0  0.0  1
1.0   0.0  0.0  1
0.0   1.0  0.0  1
0.0   0.0  1.0  1
1.0   0.0  1.0  1
0.0   1.0  1.0  1

Triangles
2
1  3  2  1
4  5  6  1


Quadrilaterals
3
1  2  5  4  1
2  3  6  5  1
1  4  6  3  1


Prisms
1
1  2  3  4  5  6  1

End
```



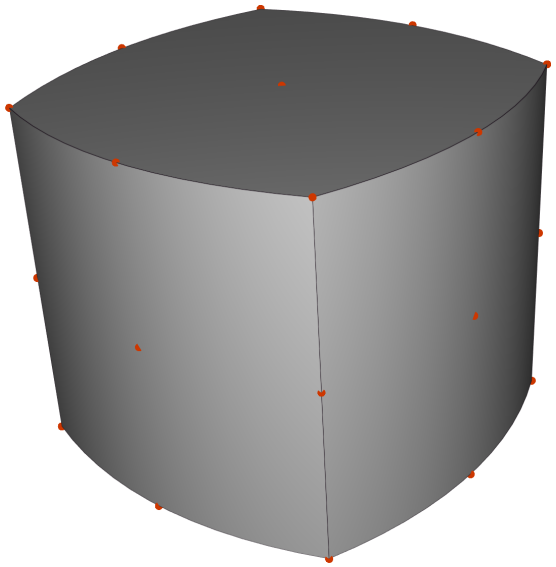
```
MeshVersionFormatted 3

Dimension
3

Vertices
18
0.0   0.0  0.0  1
1.0   0.0  0.0  1
0.0   1.0  0.0  1
0.0   0.0  1.0  1
1.0   0.0  1.0  1
0.0   1.0  1.0  1
0.5  -0.1  0.0  1
0.6   0.6  0.0  1
-0.1  0.5  0.0  1
0.5  -0.1  1.0  1
0.6   0.6  1.0  1
-0.1  0.5  1.0  1
0.0   0.0  0.5  1
1.0   0.0  0.5  1
0.0   1.0  0.5  1
0.5  -0.1  0.5  1
0.6   0.6  0.5  1
-0.1  0.5  0.5  1

TrianglesP2
2
1  3  2  9  8  7  1
4  5  6  10  11  12  1

QuadrilateralsQ2
3
1  2  5  4  7  14  10  13  16  1
2  3  6  5  8  15  11  14  17  1
1  4  6  3  13  12  15  9  18  1

PrismsP2
1
1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  1

End
```



```
MeshVersionFormatted 3

Dimension
3

Vertices
8
0.0   0.0  0.0  1
1.0   0.0  0.0  1
1.0   1.0  0.0  1
0.0   1.0  0.0  1
0.0   0.0  1.0  1
1.0   0.0  1.0  1
1.0   1.0  1.0  1
0.0   1.0  1.0  1

Quadrilaterals
6
1  4  3  2  1
1  2  6  5  1
1  5  8  4  1
7  3  4  8  1
7  6  2  3  1
7  8  5  6  1

Hexahedra
1
1  2  3  4  5  6  7  8  1

End
```



```
MeshVersionFormatted 3

Dimension
3

Vertices
27
0.0   0.0  0.0  1
1.0   0.0  0.0  1
1.0   1.0  0.0  1
0.0   1.0  0.0  1
0.0   0.0  1.0  1
1.0   0.0  1.0  1
1.0   1.0  1.0  1
0.0   1.0  1.0  1
0.5  -0.1  0.0  1
1.1   0.5  0.0  1
0.5   1.1  0.0  1
-0.1  0.5  0.0  1
0.5  -0.1  1.0  1
1.1   0.5  1.0  1
0.5   1.1  1.0  1
-0.1  0.5  1.0  1
0.0   0.0  0.5  1
1.0   0.0  0.5  1
1.0   1.0  0.5  1
0.0   1.0  0.5  1
0.5   0.5  0.0  1
0.5   0.5  1.0  1
0.5  -0.1  0.5  1
1.1   0.5  0.5  1
0.5   1.1  0.5  1
-0.1  0.5  0.5  1
0.5   0.5  0.5  1

QuadrilateralsQ2
6
1  4  3  2  12  11  10  9  21  1
1  2  6  5  9  18  13  17  23  1
1  5  8  4  17  16  20  12  26  1
7  3  4  8  19  11  20  15  25  1
7  6  2  3  14  18  10  19  24  1
7  8  5  6  15  16  13  14  22  1

HexahedraQ2
1
1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  1

End
```



9.1.2 A simple mesh of quadrilaterals and hexahedra

A first simple mesh of quadrilaterals and hexahedra is given in Listing 8 (file `cube_struct.mesh`). It contains a list of vertices and the connectivity of the elements. To display this mesh:

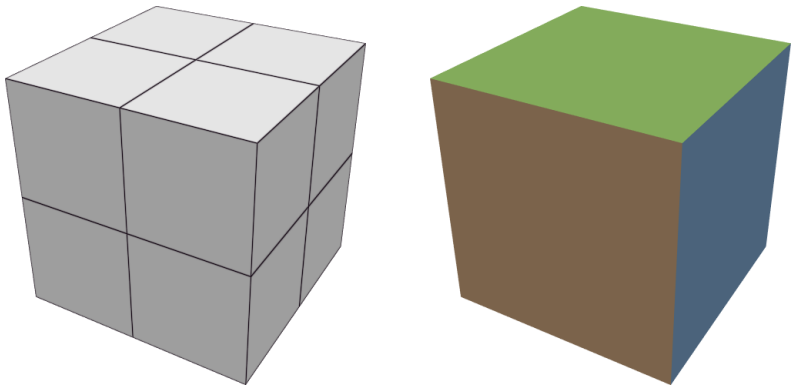
```
vizir4 -in cube_struct.mesh
```

```
MeshVersionFormatted 3
Dimension
3
Vertices
27
0       0       0       1
1       0       0       2
0       1       0       3
1       1       0       4
1       1       1       5
1       0       1       6
0       1       1       7
0       0       1       8
0       1       0.5     1
0.5     1       1       2
1       1       0.5     3
0.5     1       0       4
0       0.5     0       5
1       0.5     0       6
1       0       0.5     7
0.5     0       1       8
0       0       0.5     9
0.5     0       0       10
0       0.5     1       11
1       0.5     1       12
0.5     0       0.5     14
0.5     0.5     0       16
0.5     1       0.5     18
0.5     0.5     1       20
1       0.5     0.5     22
0       0.5     0.5     24
0.5     0.5     0.5     26

Quadrilaterals
24
 2 15 21 18 14
18 21 17  1 14
15  6 16 21 14
21 16  8 17 14
 2 14 22 18 16
18 22 13  1 16
14  4 12 22 16
22 12  3 13 16
 5 11 23 10 18
10 23  9  7 18
11  4 12 23 18
23 12  3  9 18
 6 20 24 16 20
16 24 19  8 20
20  5 10 24 20
24 10  7 19 20
```

```
 2 15 25 14 22
14 25 11  4 22
15  6 20 25 22
25 20  5 11 22
 8 17 26 19 24
19 26  9  7 24
17  1 13 26 24
26 13  3  9 24

 Hexahedra
 8
21 15  2 18 27 25 14 22 26
27 25 14 22 23 11  4 12 26
17 21 18  1 26 27 22 13 26
26 27 22 13  9 23 12  3 26
16  6 15 21 24 20 25 27 26
24 20 25 27 10  5 11 23 26
 8 16 21 17 19 24 27 26 26
19 24 27 26  7 10 23  9 26
 End
```

Listing 8: File: cube_struct.mesh

The keyword `Vertices` is followed by the number of vertices, and for each vertex, x, y, z and a reference are given. In the case of a 2D problem, there are only 3 information: x, y and a reference.
The keyword `Quadrilaterals` (resp. `Hexahedra`) is followed by the number of quadrilaterals (resp. hexahedra). Then, for each element, 4 (resp. 8) integers give the connectivity followed by the reference.
The rendering of this mesh of cube is shown in Figure 29.



(a) A mesh of 8 hexahedra

(b) Each face has a given reference number

Figure 29: Example of a mesh: cube_struct.mesh

9.2 Solutions examples

We present how to render different solutions on a single triangle. The mesh file is given in Listing 9 and is made of the unit triangle. On the same triangle, 3 solutions are defined hereafter:

- A P^0 solution (i.e. constant) with the keyword `SolAtTriangles`.
- A P^1 solution (i.e. affine) with the keyword `SolAtVertices`.
- A P^3 solution (i.e. of degree 3) with the keyword `HOSolAtTrianglesP1` (where `P1` in `HOSolAtTrianglesP1` refers to the geometry which is P^1).

```
MeshVersionFormatted 3

Dimension 2

Vertices
3
0 0 1
0 1 1
1 0 1

Triangles
1
1 2 3 1

End
```

Listing 9: Mesh file of this single triangle

9.2.1 A P^0 solution on a single triangle

First, we display a constant solution. The mesh file `P0SolOnTriangle.mesh` is the one previously given in Listing 9. The solution is defined in `P0SolOnTriangle.sol` and is given in Listing 10. The keyword `SolAtTriangles` is used and follows the `libMeshb` format.
After, the keyword `SolAtTriangles`:

- The first line gives the number of elements considered: here 1.
- The second line gives the number of solutions and the type of solutions: here 1 for one solution and 1 for scalar.
- The other lines (one for each element) give the local value: here 2.0.

```
MeshVersionFormatted 3

Dimension 2

SolAtTriangles
1
1 1
2.0

End
```

Listing 10: `P0SolOnTriangle.sol`

To display the solution, use :

```
vizir4 -in P0SolOnTriangle
```

9.2.2 A P^1 solution on a single triangle

Let display an affine solution. For this purpose, the keyword `SolAtVertices` is used where the solution is defined at each vertex of the mesh. The solution file `P1SolOnTriangle.sol` is given in Listing 10.
The keyword `SolAtVertices` is followed by the relevant information:

- The first line gives the number of vertices here 3.
- The second line gives the number of solutions and the type of solutions: here 1 for one solution and 1 for scalar.
- The other lines (one for each vertex) give the local value: here 2.0 for the first vertex, 6.0 for the second and 4.0 for the last.

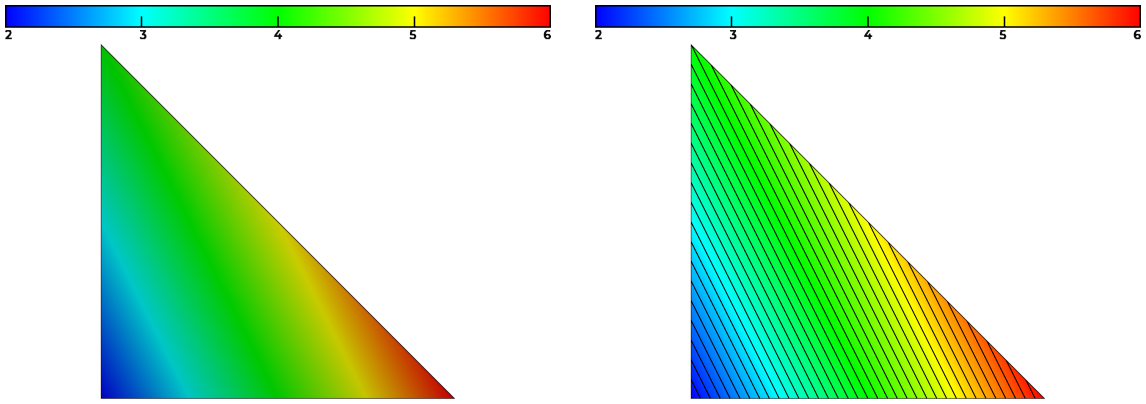
```
MeshVersionFormatted 3

Dimension
2

SolAtVertices
3
1 1
2.0
6.0
4.0

End
```

Listing 11: P1SolOnTriangle.sol

To display the solution, use :

```
vizir4 -in P1SolOnTriangle
```

The rendering obtained is shown in Figure 30. In ViZiR 4, press the key m to display or hide the solution and press the key o to display the isolines.



(a) Rendering (b) Isolines

Figure 30: P^1 solution on a triangle

9.2.3 A P^3 solution on a single triangle

We present now a P^3 polynomial function on a single triangle. The mesh file P3SolOnTriangle.mesh is given in Listing 9 and is composed of the unit triangle. The solution is a polynomial function of degree 3 defined in P3SolOnTriangle.sol and given in Listing 12.

```
MeshVersionFormatted 3

Dimension
2

HOSolAtTrianglesP1
1
1 1
3 10
-1 1 1 1 0 -1 0 1 -1 1
```

Listing 12: P3SolOnTriangle.sol

The keyword HOSolAtTrianglesP1 is followed by these information:

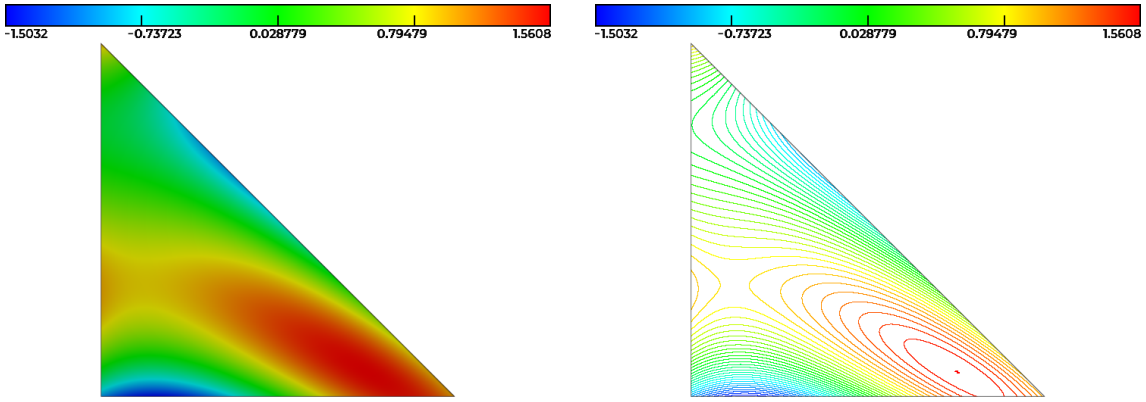
- The first line gives the number of elements considered: here 1.
- The second line gives the number of solutions and the type of solutions: here 1 for one solution and 1 for scalar.
- The third line gives the order considered, here 3, and the number of degrees of freedom for each element, here 10 for each triangle.

- The other lines (one for each element) give the local degrees of freedom value: here -1 1 1 1 0 -1 0 1 -1 1.

To display the solution, use :

```
vizir4 -in P3SolOnTriangle
```

The rendering obtained is shown in Figure 31.



(a) Rendering (b) Isolines

Figure 31: P^3 solution on a triangle

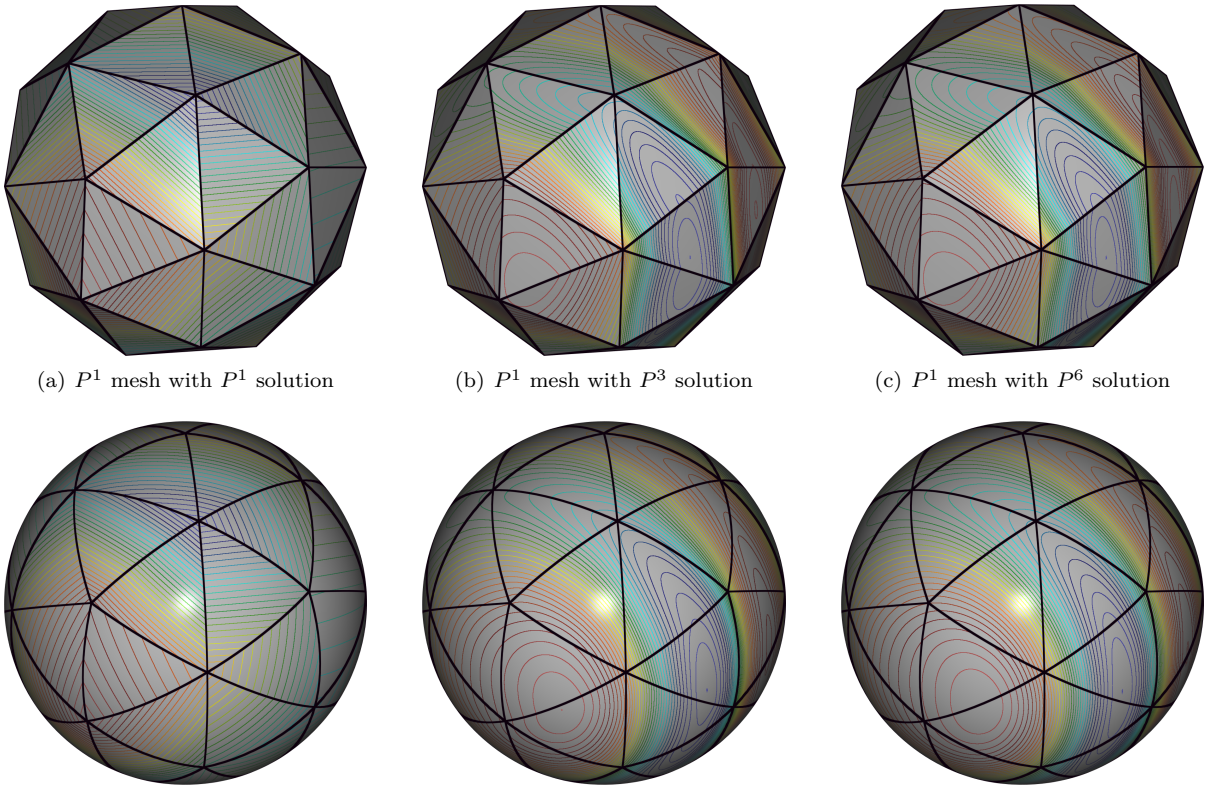
Remark. *The numbering used in this case in the default one given in* ViZiR 4. *It follows the numbering given by George et al [George et al., 2019] and the interpolation points are supposed to be evenly spread. It is possible to have a different numbering as explained in Section 5.8.*

10 Gallery

The first example is the rendering of an analytic function in a sphere with different meshes and solutions (see Figure 32). The function

$$f(x,y,z) = \cos(2\pi x)e^{(x-0.5)^2+(y-0.5)^2+z^2}$$

is defined in a sphere centered at origin and whose radius is one. Two meshes are defined, a P^1 mesh (top of Figure 32) and a P^3 mesh (bottom of Figure 32). P^1 solutions (i.e. affine) are defined is these meshes (Figures 32 (a) and (d)). Then, P^3 solutions are defined in Figures 32 (b) and (e) and finally P^6 solutions in Figures 32 (c) and (f).



(a) P^1 mesh with P^1 solution (b) P^1 mesh with P^3 solution (c) P^1 mesh with P^6 solution

(d) P^3 mesh with P^1 solution (e) P^3 mesh with P^3 solution (f) P^3 mesh with P^6 solution

Figure 32: Isolines rendering of the approximation of an analytic function in a sphere for different low/high order solutions on low/high order meshes.

Let now consider a Q^6 solution of a wave propagation problem for different times. We thank Sébastien Imperiale (Inria) for the data. More details can be found in [Baffet et al., 2019]. The mesh is a simple structured grid of 8000 hexahedra shown in Figure 34.a. The Q^6 solution is displayed for different times in Figure 33. The solution is rich as shown in the zoom in Figure 34.b.

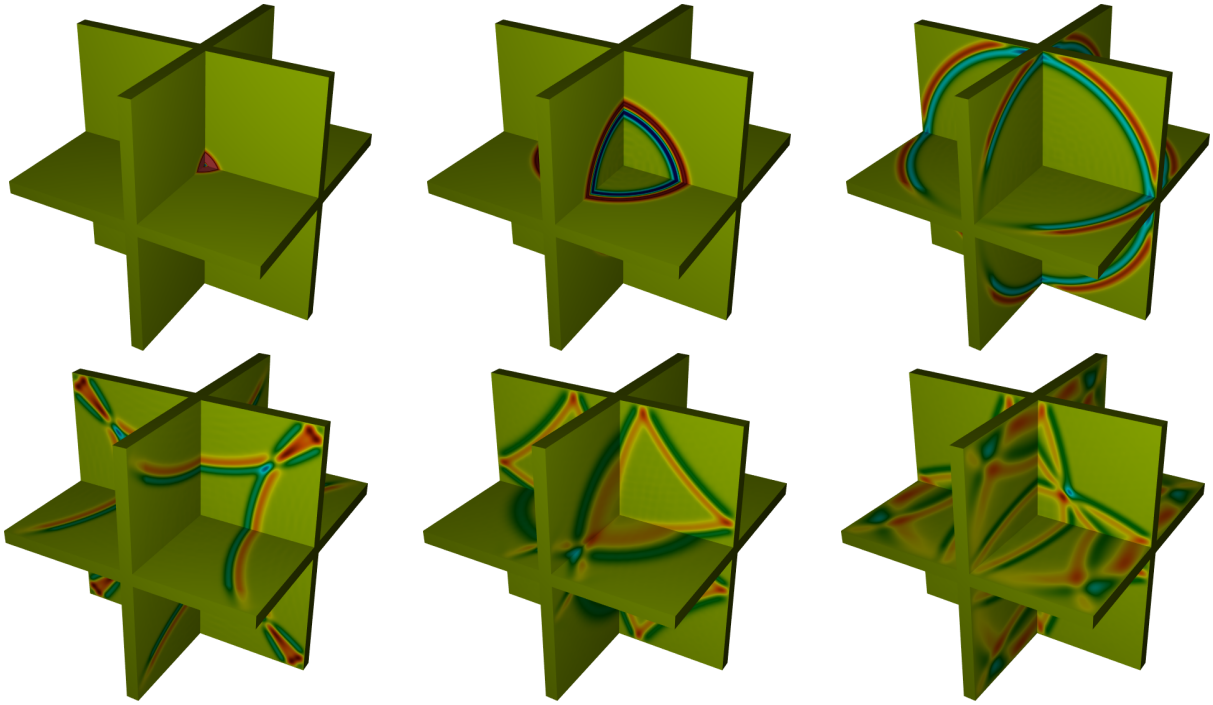
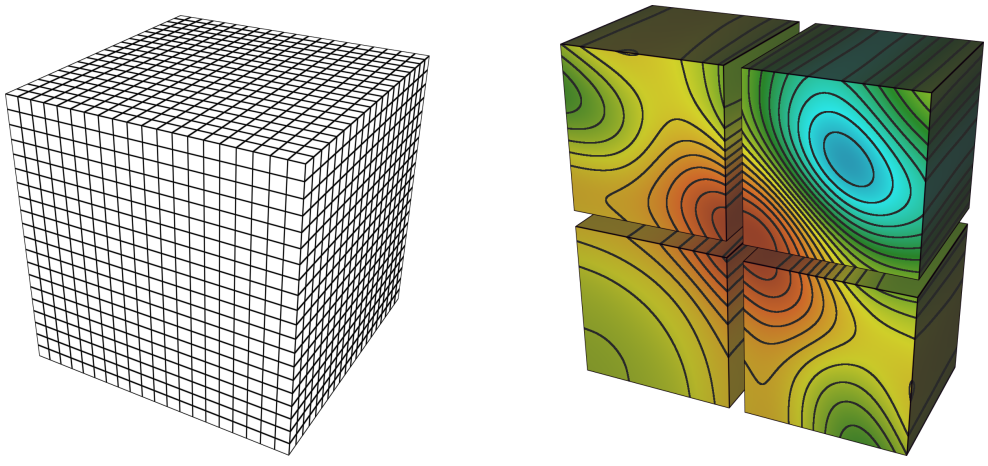


Figure 33: Q^6 solution of a wave propagation problem for different times. Courtesy of Sébastien Imperiale [Baffet et al., 2019].



(a) Structured mesh (b) Zoom on 4 hexahedra

Figure 34: Figure a: Mesh of 8000 hexahedra. Figure b: Zoom (isoline rendering) on 4 hexahedra in the last case of Figure 33. Solution of degree 6 for each cube.

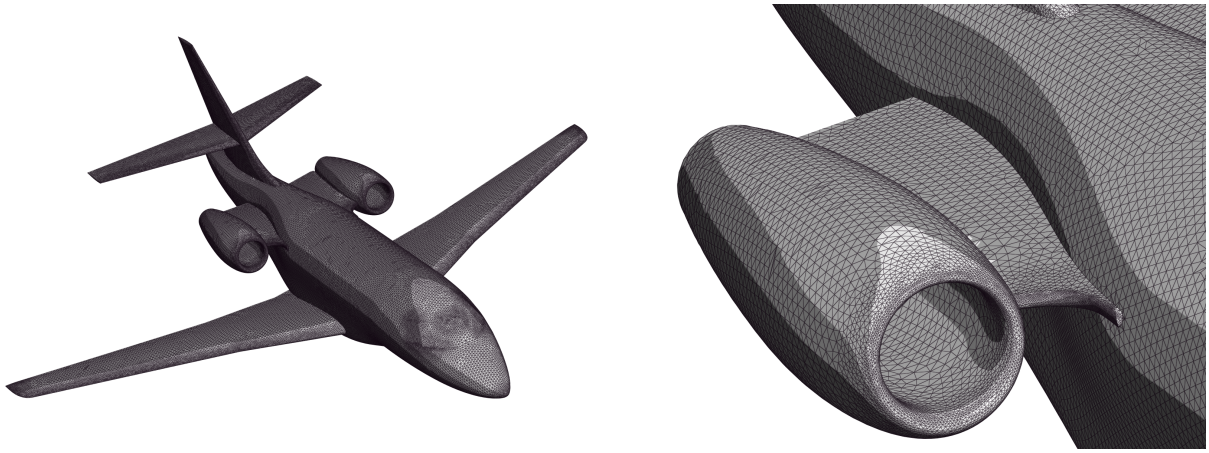


Figure 35: Rendering of a P^3-mesh of a Falcon.

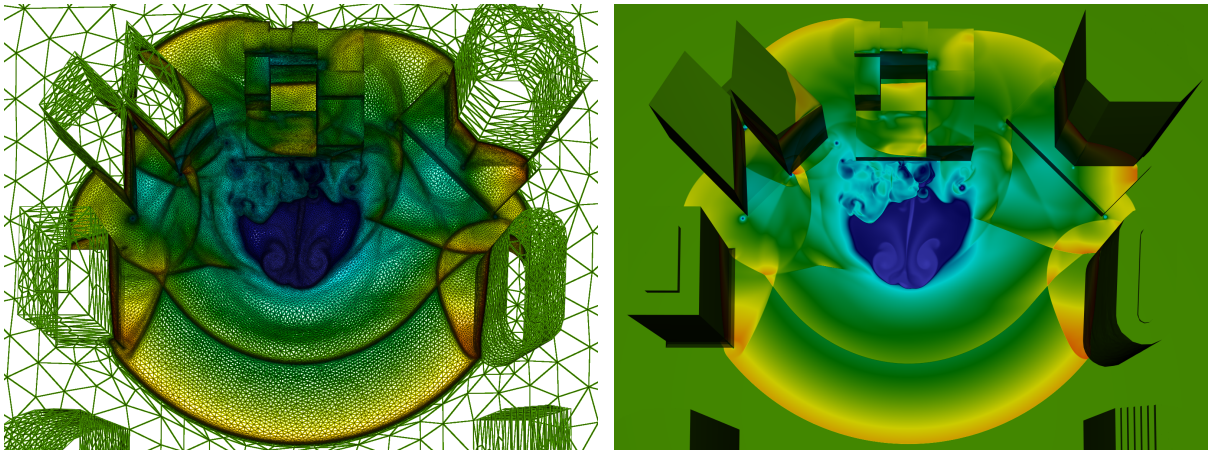


Figure 36: City blast propagation. The mesh and the solution are both P^1. The mesh has been adapted to the solution computed with `Wolf` (see http://pages.saclay.inria.fr/frederic.alauzet).

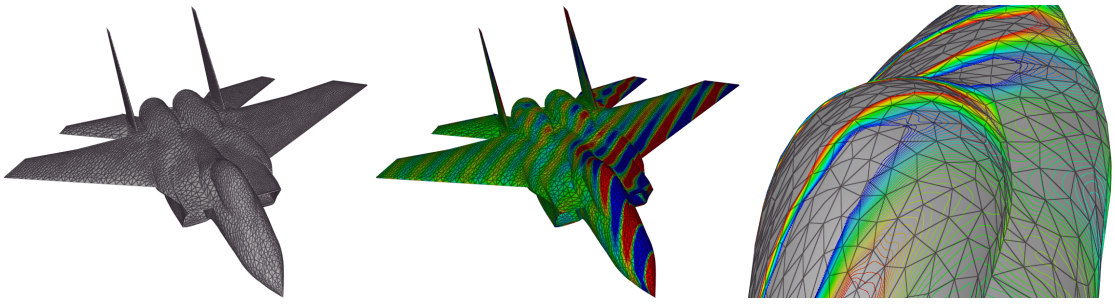


Figure 37: High-order (P^3) BEM resolution of the scattering of plane waves on an aircraft. Left: mesh. Middle: mesh with solution. Right: zoom on isolines around the cockpit. Courtesy of Stéphanie Chaillat (ENSTA) [Chaillat et al., 2018].

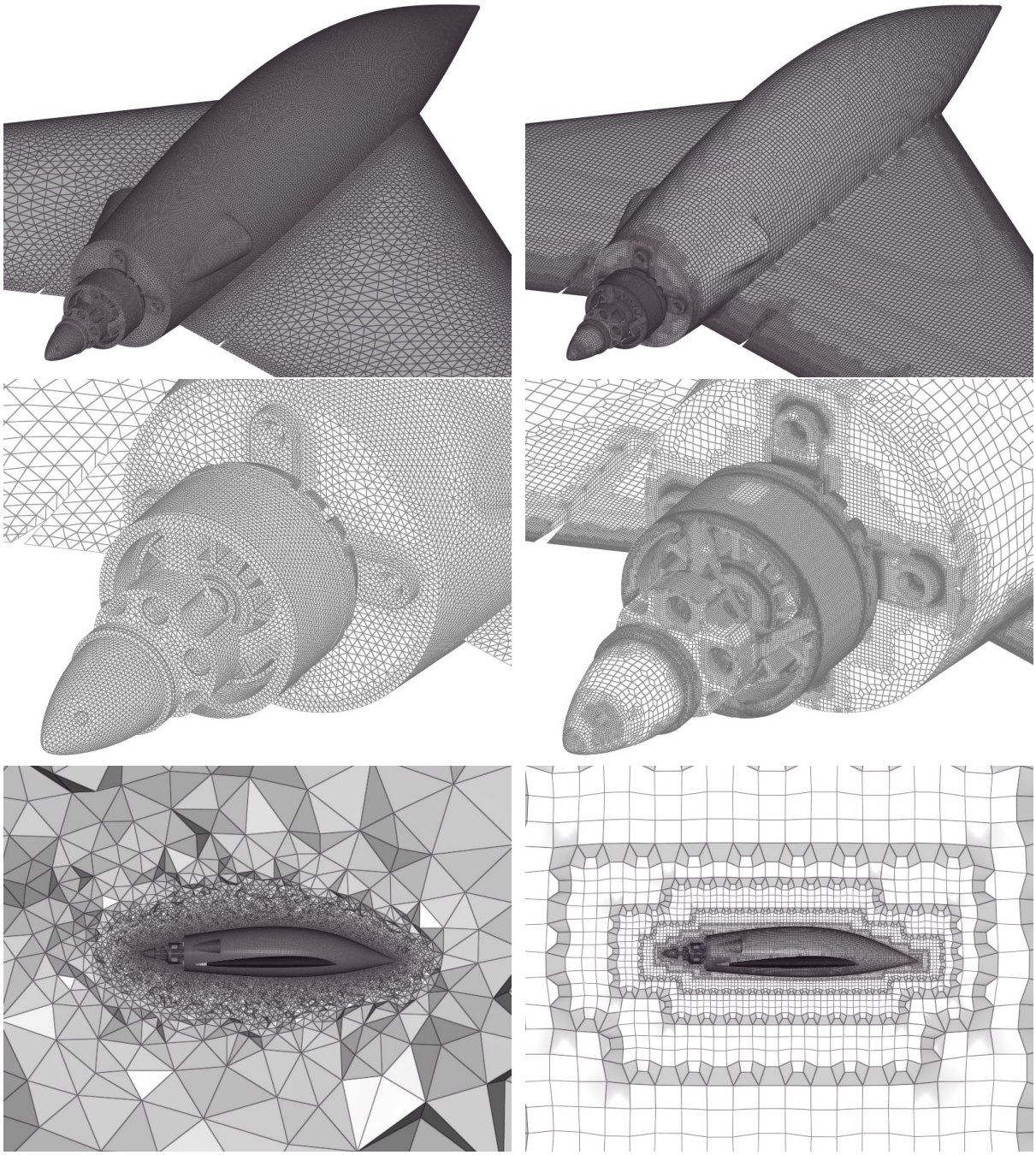


Figure 38: Tetrahedra (left) and hexahedra (right) meshes.

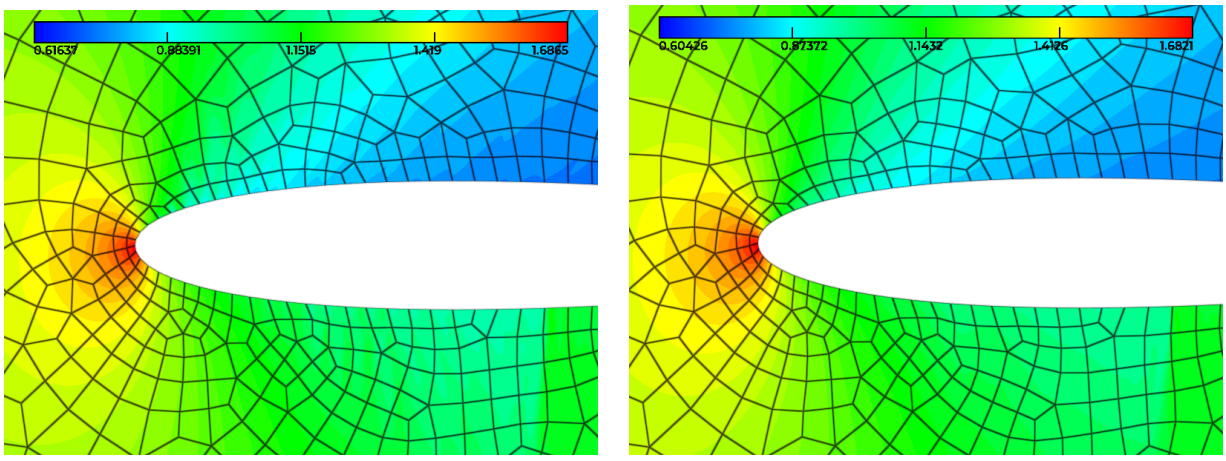


Figure 39: Degree 5 resolution of the Euler equations on a NACA0012 airfoil with a Q^1-mesh (left) and with a Q^2-mesh (right). Mesh edges are shown in black and filled isolines are displayed and show the inaccuracy in the linear case on the top of the airfoil. Courtesy of Christophe Peyret (ONERA).

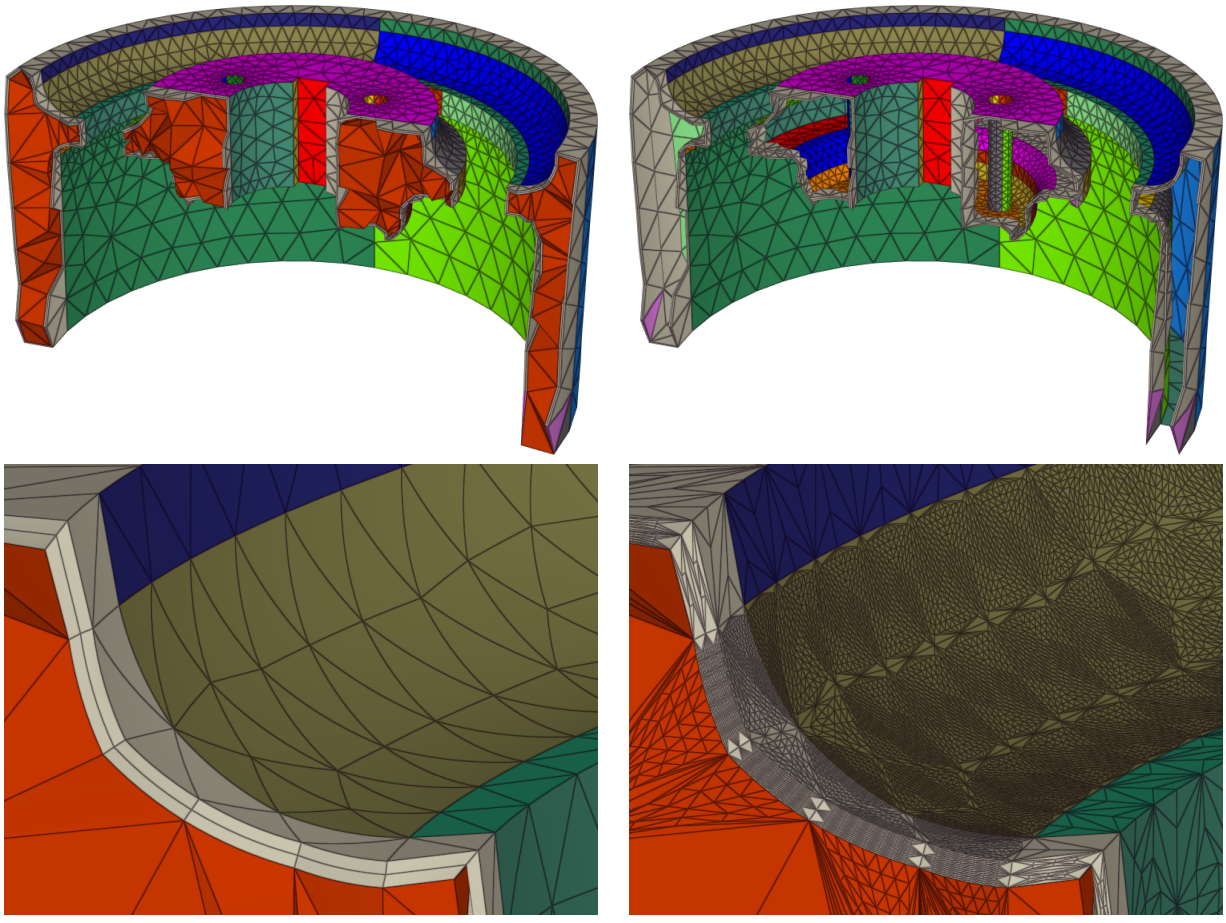


Figure 40: Rendering of a mesh composed of hybrids P^2-elements with clipping. References (patch ids) are shown in color. Top Left: clipping is activated (only a part of the mesh is shown). Top Right: The volume (in red in this figure) is hidden. Bottom Left: Zoom on some curved elements. Bottom Right: The tessellation of these elements is displayed. Courtesy of Loïc Maréchal (Inria).

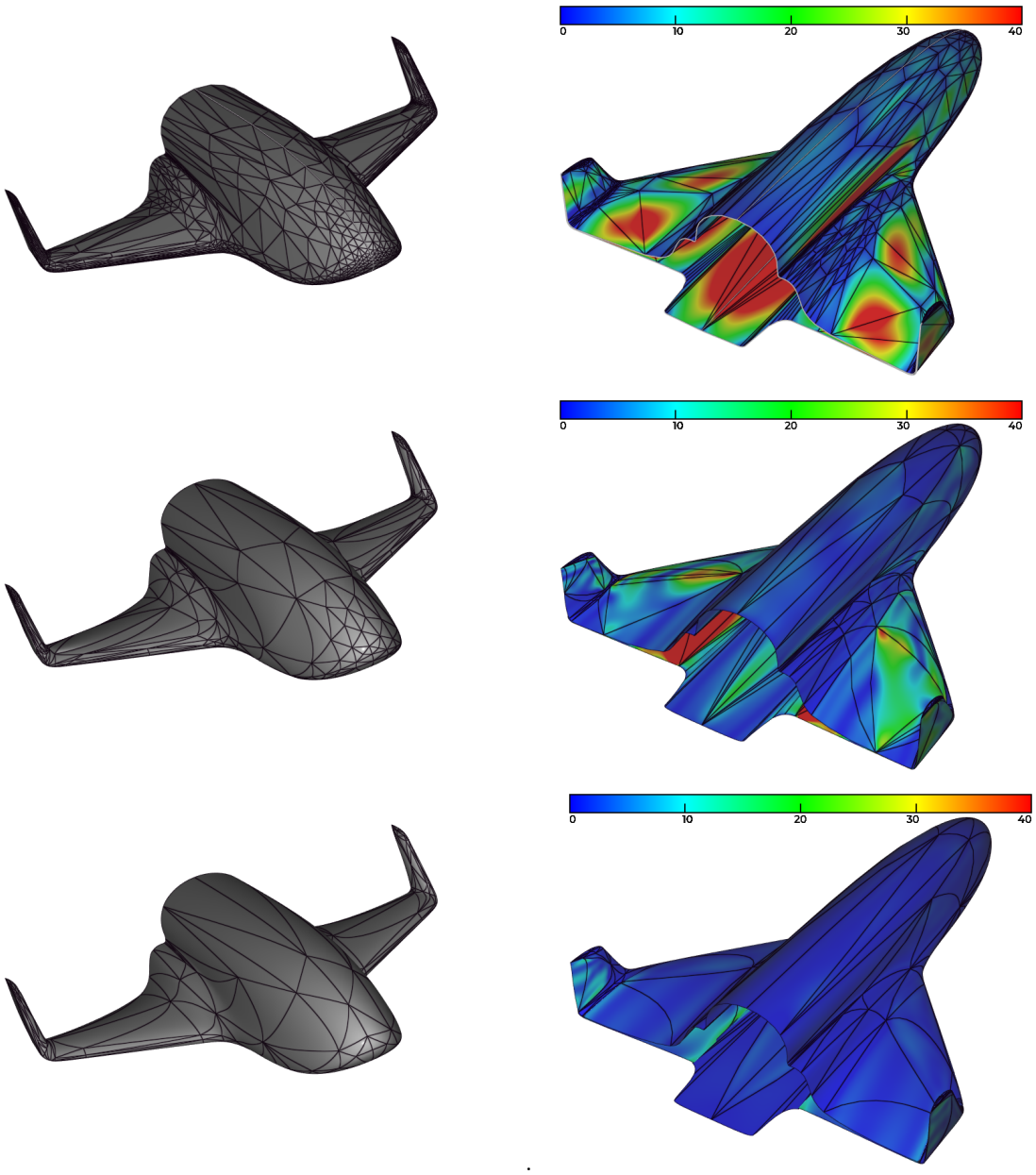


Figure 41: P^1 (top), P^2 (middle) and P^3 (bottom) meshes and point-wise distance to the shuttle geometry [Feuillet et al., 2019].

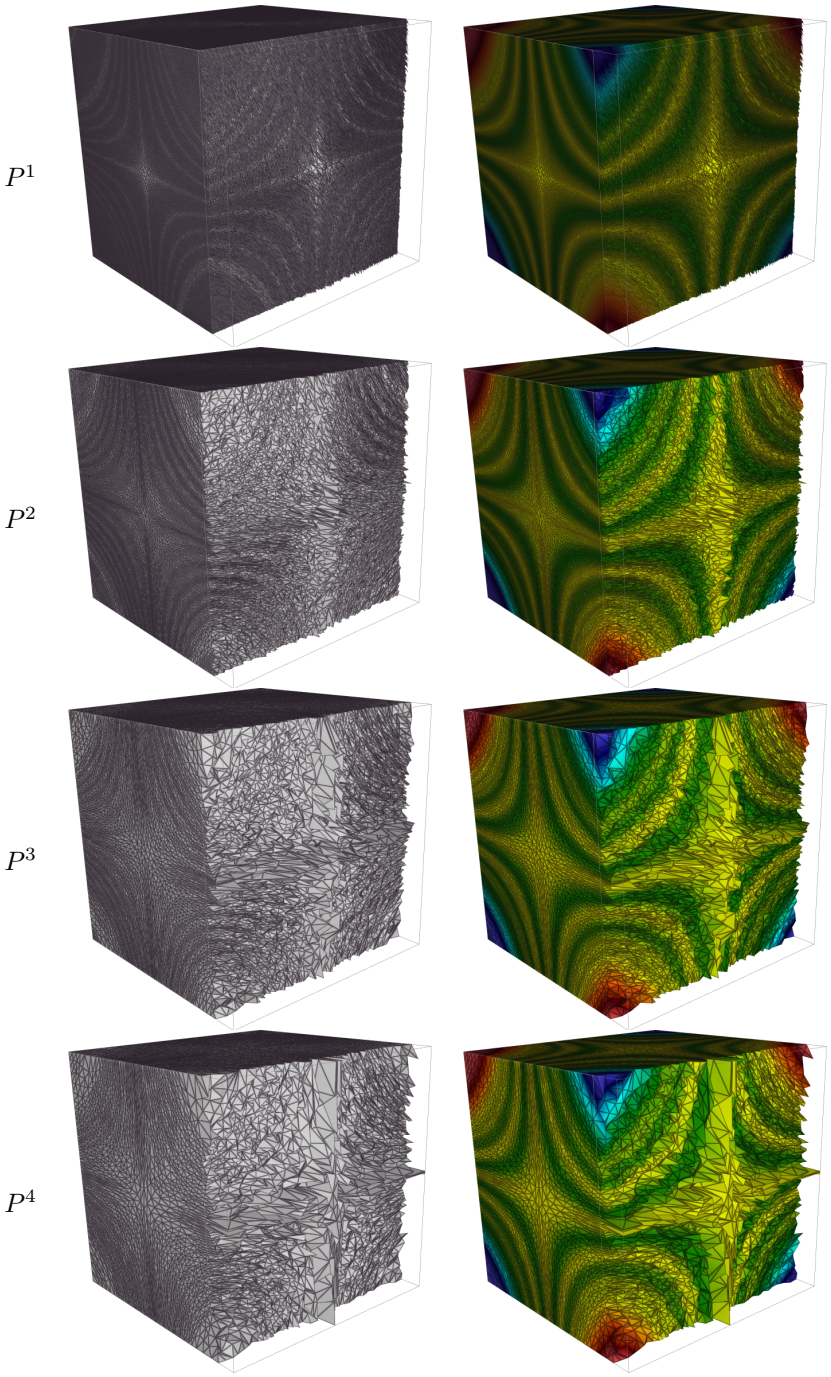


Figure 42: From top to bottom, P^1 to P^4 adapted meshes (left) with the corresponding solution on it (right) [Coulaud and Loseille, 2016].

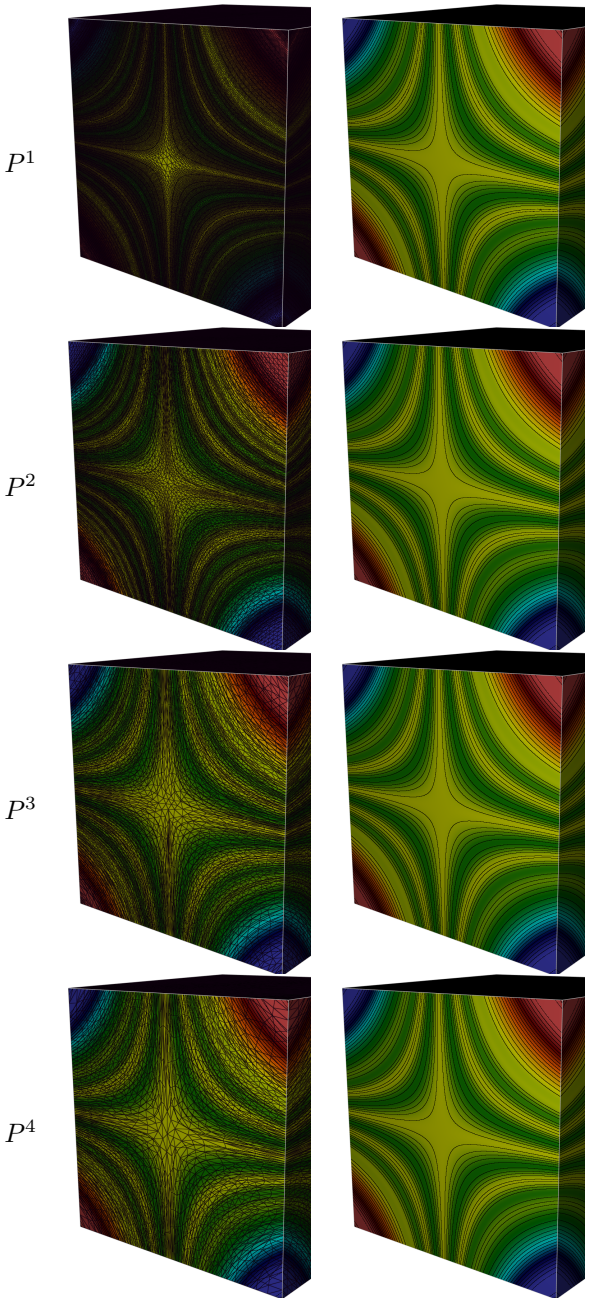


Figure 43: From top to bottom, P^1 to P^4 adapted meshes with the solution (left) and isovalues (right) [Coulaud and Loseille, 2016].

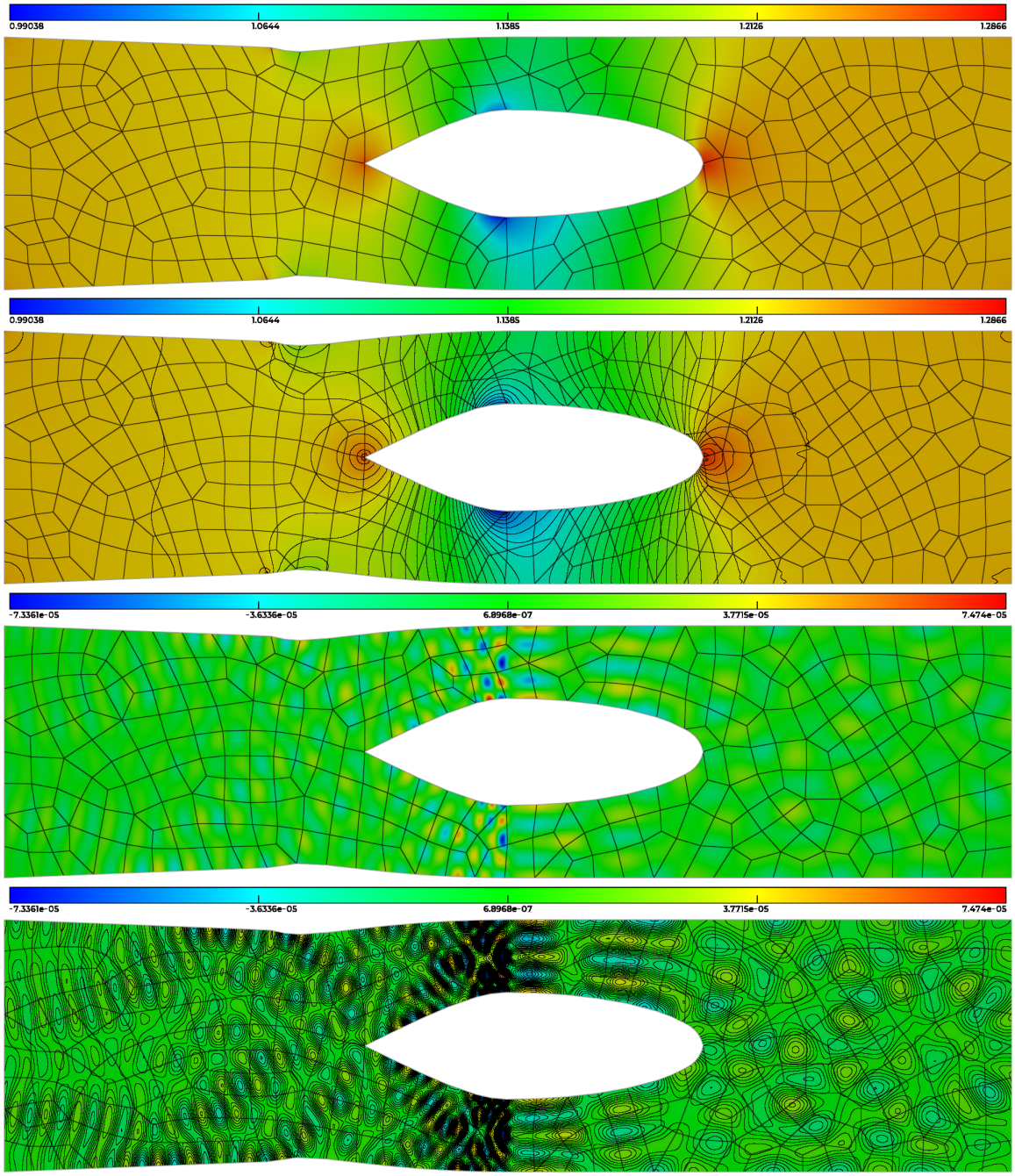
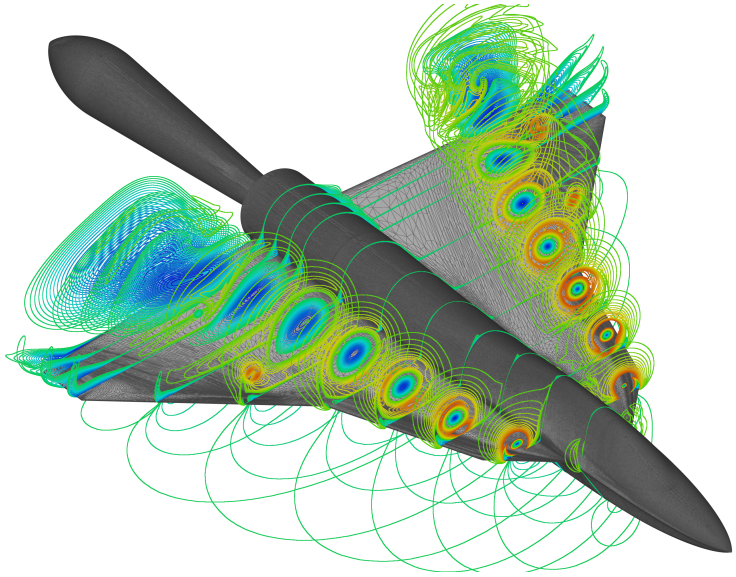


Figure 44: Top (two first pictures), mean flow around an engine using a Q^2 mesh with Q^{10} solution. Bottom (two last pictures), perturbation over the mean flow using a Q^8 solution. Courtesy of Christophe Peyret (ONERA) [Peyret, 2017].



11 Additional modes

11.1 -help or -h: Help

ViZiR 4 help can be launched with -help or -h:

```
vizir4 -help
```

```
vizir4 -h
```

11.2 -helpmodes: Help for all modes

The help for all modes can be launched with -helpmodes:

```
vizir4 -helpmodes
```

11.3 -checksurf: Surface reconstruction mode

The checksurf mode reconstructs the missing surfaces. In case volume elements, this mode checks if some surfaces are missing in the mesh file and create a new mesh file containing the complete mesh with all the volume and surface elements. Furthermore, if there is a solution, it adds in a new solution file the missing solution at surface elements. The format is

```
vizir4 -checksurf -in xxx.mesh[b] (-sol xxx.sol[b] -out xxx -ascii)
```

where

- -in is required to define the input mesh files
- -sol is optional to define the input solution files
- -out is optional to define the output mesh/sol files
- -ascii is optional to set ascii output (binary by default otherwise)

The argument after -in is as usual the input mesh file name. The argument after -sol is as usual the input solution file name. It is optional. If there is no -sol, ViZiR 4 will use the mesh name to try to find a solution file. The argument after -out is the output file name. It is optional and if it is not given, a new file will be created whose name is based on the input mesh file name.
To launch the help of this mode:

```
vizir4 -checksurf
```

An example. Let consider a trivial example involving a single tetrahedron (see Listing 13) where a HO solution of degree is defined (see Listing 14). The surfaces are missing.

```
MeshVersionFormatted 3

Dimension
3

Vertices
4
0.0  0.0  0.0  1
1.0  0.0  0.0  1
0.0  1.0  0.0  1
0.0  1.0  1.0  1

Tetrahedra
1
1 2 3 4 1

End
```

Listing 13: TetraP1_SolP2.mesh

```
MeshVersionFormatted 3

Dimension 3

HOSolAtTetrahedraP1
1
1 1
2 10

1 2 3 4 5 6 7 8 9 10

End
```

Listing 14: TetraP1_SolP2.sol

Then, using

```
vizir4 -checksurf -in TetraP1_SolP2.mesh
```

gives

```
 Launching the surface reconstruction mode

%% TetraP1_SolP2.mesh OPENED
Dimension 3
Number Of Vertices                      4
Number Of Tetrahedra                    1
x:[0 1] y:[0 1] z:[0 1]

%% TetraP1_SolP2.sol OPENED
Degree of HOSolAtTetrahedraP1           2

4 missing triangles faces are added

%% Mesh written in TetraP1_SolP2.surfOK.meshb
Dimension 3
Number Of Vertices                      4
Number Of Triangles                     4
Number Of Tetrahedra                    1
x:[0 1] y:[0 1] z:[0 1]

4 missing Solutions at Triangles are added

%% Solution written in TetraP1_SolP2.surfOK.sol
Degree of HOSolAtTrianglesP1            2
Degree of HOSolAtTetrahedraP1           2

Thank you for using the surface reconstruction mode of ViZiR 4
```

and two new files `TetraP1_SolP2.surfOK.mesh` and `TetraP1_SolP2.surfOK.sol` are created (see Listings 15 and 16).

```
MeshVersionFormatted 3

Dimension 3

Vertices
4
0 0 0 1
1 0 0 1
0 1 0 1
0 1 1 1

Triangles
4
2 3 4 1
4 3 1 1
4 1 2 1
2 1 3 1

TrianglesP1Ordering
3
1 0 0
0 1 0
0 0 1

Tetrahedra
1
1 2 3 4 1

TetrahedraP1Ordering
4
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1

End
```

Listing 15: `TetraP1_SolP2.surfOK.mesh`

```
MeshVersionFormatted 3

Dimension 3

HOSolAtTrianglesP1
4
1 1
2 6
2 3 4 6 10 9
4 3 1 10 7 8
4 1 2 8 5 9
2 1 3 5 7 6

HOSolAtTetrahedraP1
1
1 1
2 10
1 2 3 4 5 6 7 8 9 10

End
```

Listing 16: `TetraP1_SolP2.surfOK.sol`

11.4 -seq: State sequences mode

The state sequences mode automatically loads several state files and generates the images. It can be launched from the menu or with

```
vizir4 -seq -in xxx.mesh[b] (-sol xxx.sol[b] -nameseq xxx)
```

where

- -in is required to define the input mesh files,
- -sol is optional to define the input solution files,

- -nameseq is optional to define the state sequences files

To launch the help of this mode:

```
vizir4 -seq
```

A .seq file is required to give the name of all the state files. For each line of this file, the name of a state file is given and optionally a name of a png file. If only a state file is given, the png file will take the name of the state file with the extension png.
Here is an example of a sequence file:

```
view1.state      top.png
view2.state      bottom.png
view3.state      zoom.png
```

Listing 17: File `vizir.seq`

11.5 `-movie`: Movie generator mode

When several meshes and/or solutions are involved, it is possible to automatically generate images which are screenshots. This is particularly interesting when a time dependent problem is considered. For this purpose, a file named `vizir.movie` needs to be created and contains the name of meshes and solutions files. An example of file is given in Listing 18 where 6 solutions are defined in 6 different meshes. Images in format `jpg` are created in the current directory.

```
mesh1.meshb      sol1.solb
mesh2.meshb      sol2.solb
mesh3.meshb      sol3.solb
mesh4.meshb      sol4.solb
mesh5.meshb      sol5.solb
mesh6.meshb      sol6.solb
```

Listing 18: File `vizir.movie`

One can specify the name of the images like in the following:

```
adap_640K/file.meshb     adap_640K/file.Cf.solb     mesh_00640K.jpg
adap_1280K/file.meshb    adap_1280K/file.Cf.solb    mesh_01280K.jpg
adap_2560K/file.meshb    adap_2560K/file.Cf.solb    mesh_02560K.jpg
adap_5120K/file.meshb    adap_5120K/file.Cf.solb    mesh_05120K.jpg
adap_10240K/file.meshb   adap_10240K/file.Cf.solb   mesh_10240K.jpg
adap_20480K/file.meshb   adap_20480K/file.Cf.solb   mesh_20480K.jpg
```

Listing 19: Another file `vizir.movie`

This movie frames generator is launched with `-movie` instead of `-in`:

```
vizir4 -movie
```

Note that in this case, there is nothing after `-movie` as the names of the meshes/solutions files are given in `vizir.movie`.
For each solution file, a screenshot is generated. For instance, 6 images in format `png` are created as shown in Figure 45. The mesh and the solution are displayed. We thank Christophe Peyret (ONERA) for the state.

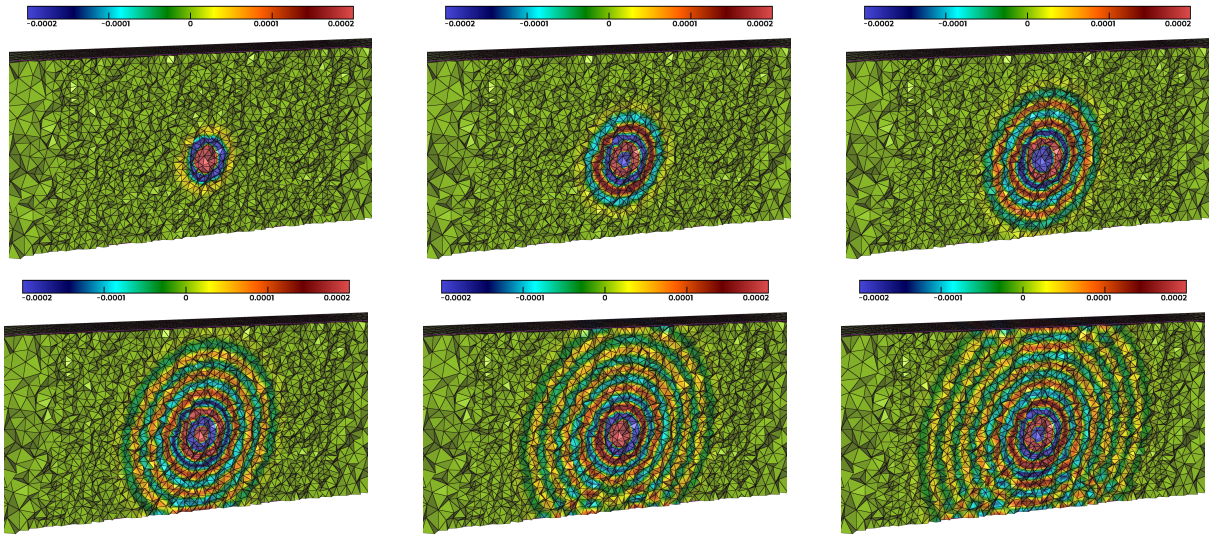


Figure 45: Example of `-movie`

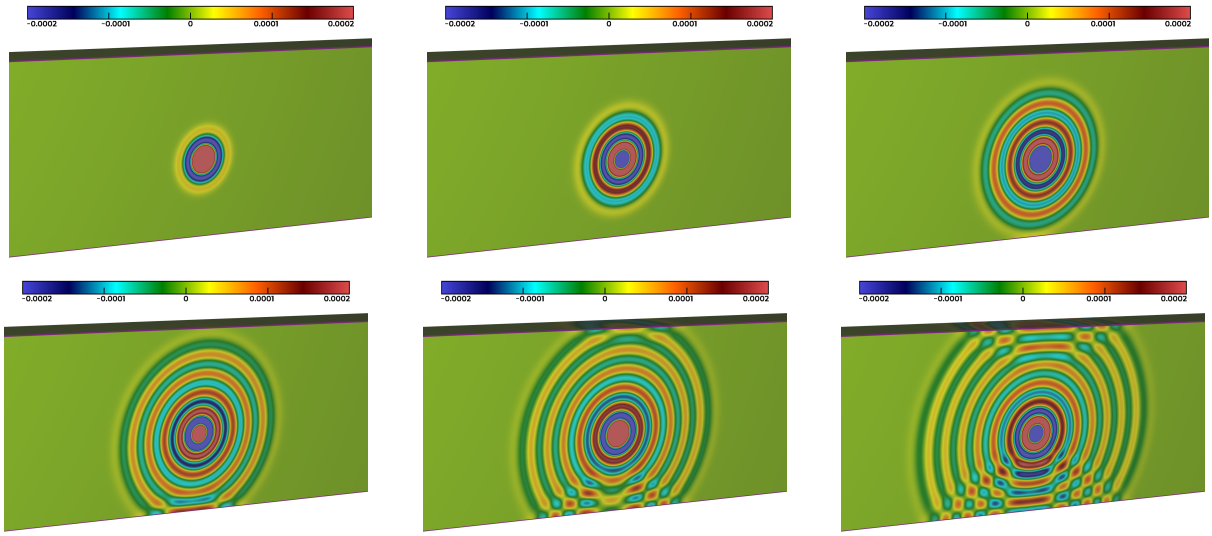


Figure 46: Example of `-movie` with capping activated

A `.state` file can be specified to have a specific view and rendering options simply by using `-state`:

```
vizir4 -movie -state xxx.state
```

Otherwise, `ViZiR 4` tries to find if `vizir.state` file exists. If it is the case, the parameters will be loaded and used to generate the images.

11.6 `-images`: Images generator mode

A more generic images generator has been added. The idea is to give a list of mesh / solution / state / image. Between two cases, the mesh is opened only if it is different than the previous one. Same for the solution and the state files. This list is defined in a file.

```
final1.meshb   file1.Cf.solb   view1_mesh.state    file1_mesh_cf_cfdview_01.jpg
final1.meshb   file1.Cf.solb   view2_mesh.state    file1_mesh_cf_cfdview_02.jpg
final1.meshb   file1.Cf.solb   view3_mesh.state    file1_mesh_cf_cfdview_03.jpg
final1.meshb   file1.Cf.solb   view1_exact.state   file1_cf_cfdview_01.jpg
final1.meshb   file1.Cf.solb   view2_exact.state   file1_cf_cfdview_02.jpg
final1.meshb   file1.Cf.solb   view3_exact.state   file1_cf_cfdview_03.jpg
final2.meshb   file2.Cf.solb   view1_mesh.state    file2_mesh_cf_cfdview_01.jpg
final2.meshb   file2.Cf.solb   view2_mesh.state    file2_mesh_cf_cfdview_02.jpg
final2.meshb   file2.Cf.solb   view3_mesh.state    file2_mesh_cf_cfdview_03.jpg
```

Listing 20: Example of `vizir.images` file

To launch this:

```
vizir4 -images xxx
```

where `xxx` is the name of the file, for example `vizir.images`.

11.7 `-addcorners`: Add corners (that belong to at least 3 surface ids)

The `addcorners` mode adds corners (that belong to at least 3 surface ids). The format is

```
vizir4 -addcorners -in xxx.mesh[b] (-out xxx -ascii)
```

where

- -in is required to define the input mesh files
- -out is optional to define the output file
- -ascii is optional to set ascii output (binary by default otherwise)

If no output is given, the output name will be based on the input name and end with .e.mesh[b].
References of edges are set by connected components.
To launch the help of this mode:

```
vizir4 -addcorners
```

11.8 `-addedges`: Add missing edges in a 2D mesh mode

The `addedges` mode adds missing edges for a 2D mesh. The format is

```
vizir4 -addedges -in xxx.mesh[b] (-out xxx -ascii)
```

where

- -in is required to define the input mesh files
- -out is optional to define the output file
- -ascii is optional to set ascii output (binary by default otherwise)

If no output is given, the output name will be based on the input name and end with .e.mesh[b].
References of edges are set by connected components.
To launch the help of this mode:

```
vizir4 -addedges
```

11.9 `-capping`: Output capping mesh/solution files mode

The `capping` mode allows to output the capped mesh and solution if any for a given cut plane equation.
The format is

```
vizir4 -capping -in xxx.mesh[b] -plane a b c d (-sol xxx.sol[b] -out xxx)
```

where

- -in is required to define the input mesh files
- -plane is required to define the parameters of the plane: a b c d

- -sol is optional to define the input solution files
- -out is optional to define the output file
- -ascii is optional to set ascii output (binary by default otherwise)
- -connected is optional to group the reference of elements as connected components

To launch the help of this mode:

```
vizir4 -capping
```

If no output is given, the output name will be based on the input name and end with .capping.mesh[b]. Otherwise, if a capping mesh is displayed, it is possible to save it with a right click and select "Save Capping Mesh".

11.10 -mergevertices: Merge duplicated vertices mode

The `mergevertices` mode merges duplicated vertices. It is based on `libOL`[8] library using `LolGetNearest` function. The format is

```
vizir4 -mergevertices -in xxx.mesh[b] (-dist d -out xxx -ascii)
```

where

- -in is required to define the input mesh files
- -dist is optional to define the distance (0 by default)
- -out is optional to define the output file
- -ascii is optional to set ascii output (binary by default otherwise)

Note that there is only one mesh file.
If no output is given, the output name will be based on the input name and end with .merge.mesh[b].
To launch the help of this mode:

```
vizir4 -mergevertices
```

11.11 -extr: Surface extraction (with a plane) mode

The `extr` mode adds missing edges for a 2D mesh. The goal is to extract the solutions obtained as intersection between surfaces (only those whose references are given) and several cut planes. The format is

```
vizir4 -extr -in xxx.mesh[b] -dat xxx.dat (-sol xxx.sol[b] -out xxx)
```

where

- -in is required to define the input mesh files
- -dat is required to define the parameters files
- -sol is optional to define the input solution files
- -out is optional to define the output file

In the parameters file, three keywords are used:

- cutplaneequation: followed by the number of planes and a, b, c, d of $ax + by + cz + d = 0$ (be careful, it is not $ax + by + cz = d$)
- reference : followed by the number of planes (which needs to be the same than given in cutplaneequation). Then, for each plane, the number of surfaces references, and the list of these references
- frominch : it is an optional keyword to convert values of cut planes from inches to meters

[8] https://github.com/LoicMarechal/libOL

A mesh of edges called `edges.mesh` is obtained. It contains all the vertices and edges created. The references are the surfaces references. You can check if some vertices belongs to less than 2 edges (i.e. probably holes) as they should be tagged as corner (button `g`, in red).
Plus, for each plane i, it created a `sol_i.dat` file with X Y Z Sol values (after reordering).
To launch the help of this mode:

```
vizir4 -extr
```

11.12 `-plotline`: Plot over line mode

This mode extracts the solution along a line. As output, a mesh of edges is generated as well as a dat file containing the values of the solutions on the created vertices. The format is

```
vizir4 -plotline -in xxx.mesh[b] -p1 x1 y1 z1 -p2 x1 y1 z1 (-sol xxx.sol[b]
-out xxx)
```

where

- -in is required to define the input mesh files
- -p1 is required to define the point 1 coordinates: x1 y1 z1
- -p2 is required to define the point 2 coordinates: x2 y2 z2
- -sol is optional to define the input solution files
- -out is optional to define the output file

To launch the help of this mode:

```
vizir4 -plotline
```

11.13 `-rad`: Radial distance mode

This mode output a mesh with solution accoding to a radial distance to an axis and center. The format is

```
vizir4 -rad -in xxx.mesh[b] -axis a b c -cent cx cy cz -dist val (-sol
xxx.sol[b] -out xxx)
```

where

- -in is required to define the input mesh files
- -axis is required to define the axis
- -cent is required to define the center
- -dist is required to define the distance to the axis
- -sol is optional to define the input solution files
- -out is optional to define the output file
- -ascii is optional to set ascii output (binary by default otherwise)

To launch the help of this mode:

```
vizir4 -rad
```

References

[Baffet et al., 2019] Baffet, D., Grote, M., Imperiale, S., and Kachanovska, M. (2019). Energy decay and stability of a perfectly matched layer for the wave equation. *Journal of Scientific Computing*.

[Chaillat et al., 2018] Chaillat, S., Groth, S., and Loseille, A. (2018). Metric-based anisotropic mesh adaptation for 3D acoustic boundary element methods . *Journal of Computational Physics*, 372:473–499.

[Coulaud and Loseille, 2016] Coulaud, O. and Loseille, A. (2016). Very High Order Anisotropic Metric-Based Mesh Adaptation in 3D. *Procedia Engineering*, 163:353–364.

[Feuillet, 2019] Feuillet, R. (2019). *Embedded and high-order meshes: two alternatives to linear body-fitted meshes*. PhD thesis, Université Paris-Saclay.

[Feuillet et al., 2019] Feuillet, R., Coulaud, O., and Loseille, A. (2019). Anisotropic Error Estimate for High-Order Parametric Surface Mesh Generation. In *28th International Meshing Roundtable*, pages 1 – 15.

[Feuillet et al., 2021] Feuillet, R., Maunoury, M., and Loseille, A. (2021). On pixel-exact rendering for high-order mesh and solution. *Journal of Computational Physics*, 424:109860.

[George et al., 2019] George, P., Borouchaki, H., Alauzet, F., Laug, P., Loseille, A., and Maréchal, L. (2019). *Meshing, Geometric Modeling and Numerical Simulation 2: Metrics, Meshes and Mesh Adaptation*. John Wiley & Sons.

[Loseille and Feuillet, 2018] Loseille, A. and Feuillet, R. (2018). Vizir: High-order mesh and solution visualization using OpenGL 4.0 graphic pipeline. In *2018 AIAA Aerospace Sciences Meeting*, page 1174.

[Maunoury et al., 2022] Maunoury, M., Feuillet, R., and Loseille, A. (2022). Using ViZiR 4 to analyze the 4th AIAA CFD High Lift Prediction Workshop Simulations. In *AIAA SCITECH 2022 Forum*, page 2246.

[Peyret, 2017] Peyret, C. (2017). A Full High Order Method for Computational AeroAcoustics. In *23rd AIAA/CEAS Aeroacoustics Conference*.

[Sellers et al., 2013] Sellers, G., Wright Jr, R. S., and Haemel, N. (2013). *OpenGL superBible: comprehensive tutorial and reference*. Addison-Wesley.

[Wolff, 2011] Wolff, D. (2011). *OpenGL 4.0 shading language cookbook*. Packt Publishing Ltd.