Supplementary Materials: Design and modelling of metallic bipolar plates for a fuel cell range extender

Uwe Reimer [1,*], Ekaterina Nikitsina [1], Holger Janßen [1], Martin Müller [1], Dieter Froning [1], Steven B. Beale [1,2] and Werner Lehnert [1,3]

Contents

S1. Computational fluid dynamics simulations

Table S1 summarizes the effective properties of the porous transport layer as adapted by the model 'porous zone' in the software OpenFOAM.

Table S1: Properties of the porous transport layer [59,60].

Parameter	Value
in-plane permeability (x)	$10^{-12}\,m^2$
in-plane permeability (y)	$10^{-12}\,m^2$
through-plane permeabilty (z)	$10^{-13}\,m^2$
overall thickness	$3 \cdot 10^{-4}\,m$

Table S2 summarizes the molar fluxes at anode and cathode as references for the mass flow, which is used as a boundary condition in the CFD simulations.

Table S2: Molar fluxes at the cathode side for the average current density of $0.44\,A\,cm^{-2}$.

	Inlet $\dot{n}$ / $mol\,s^{-1}$	Outlet $\dot{n}$ / $mol\,s^{-1}$
O_2	$2.28 \cdot 10^{-4}$	$1.14 \cdot 10^{-4}$
N_2	$8.58 \cdot 10^{-4}$	$8.58 \cdot 10^{-4}$
H_2O	-	$2.28 \cdot 10^{-4}$
total		$12.00 \cdot 10^{-4}$

S2. Mesh dependency

For the mesh dependency study a four channel parallel serpentine flow field was chosen with an active area of 10 cm^2, as shown in Figure S1. This arrangement was chosen because it reflects the interaction of fluid flow between the porous transport layer and the channels.

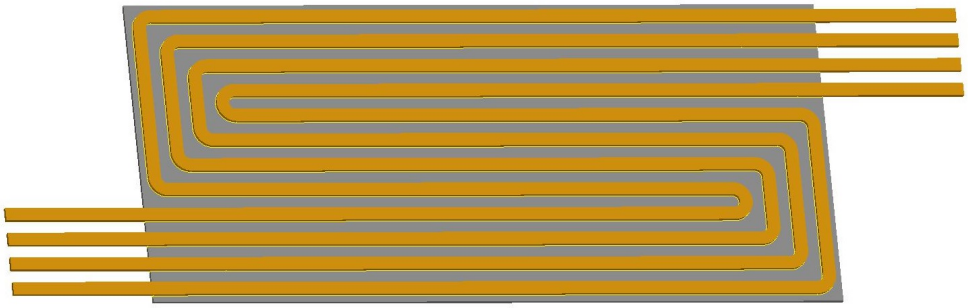

Figure S1. Four channel parallel serpentines.

The mesh variants are summarized in Table 1. The meshes named 'E' possess an even spacing of cells over the channel cross section, while the meshes named 'R' have refined cells near the walls. Figure S2 to Figure S4 illustrate the meshing procedure for the serpentine channels.

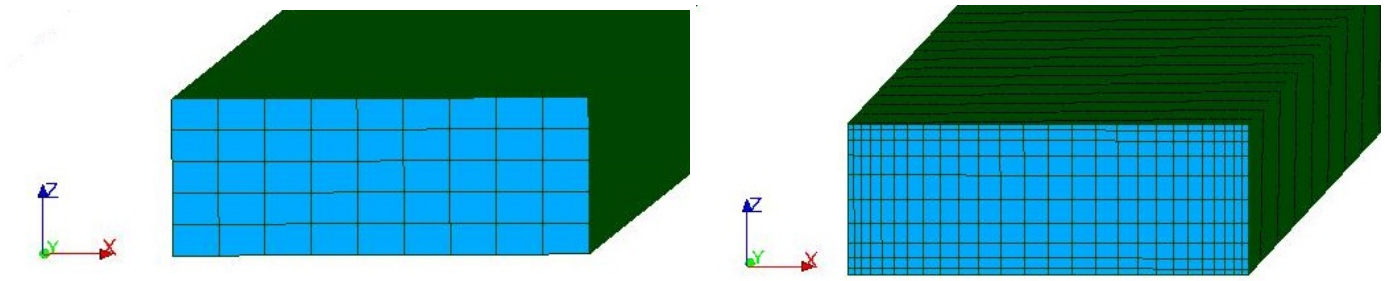

Figure S2. Channel cross section for mesh type E9 (left) and mesh type R16 (right).

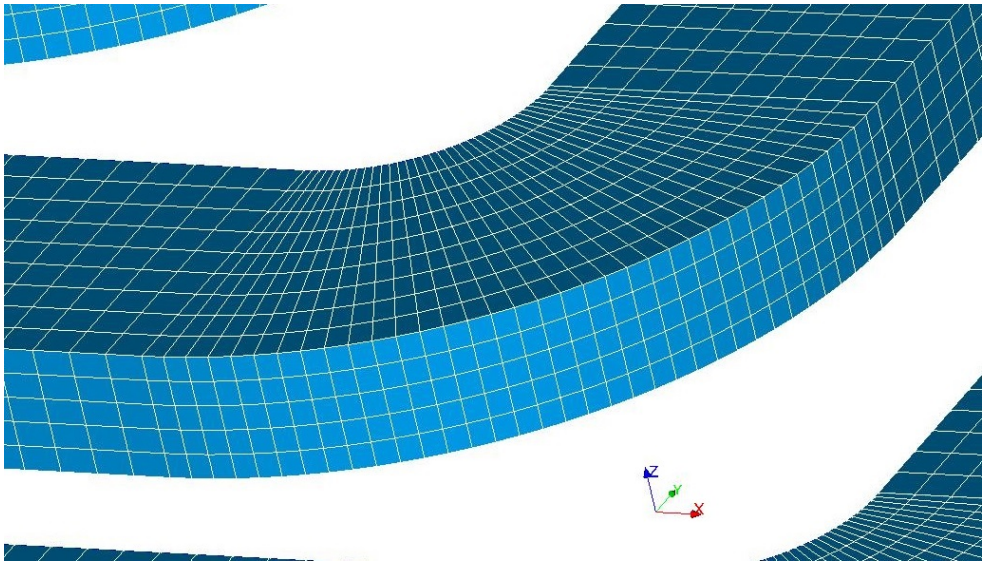

Figure S3. Section of a serpentine bend for mesh type E9.

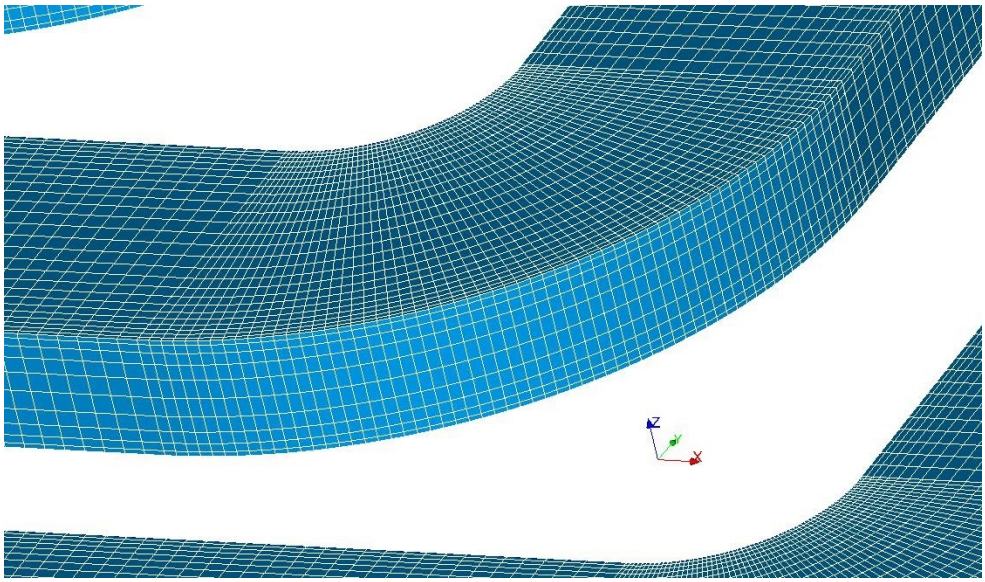

Figure S4. Section of a serpentine bend for mesh type R16.

S3. Mass flow in the coolant flow field

The resulting mass flow is calculated for specific positions within the flow field (see Figure 18). In the following Figures S5 to Figure S7 these values are shown for the operating condition MAX, FULL and MIN. The values for the mass flow are normalized, i. e. a value of unity corresponds to the arithmetic average – which is marked by a dotted line in the plots.

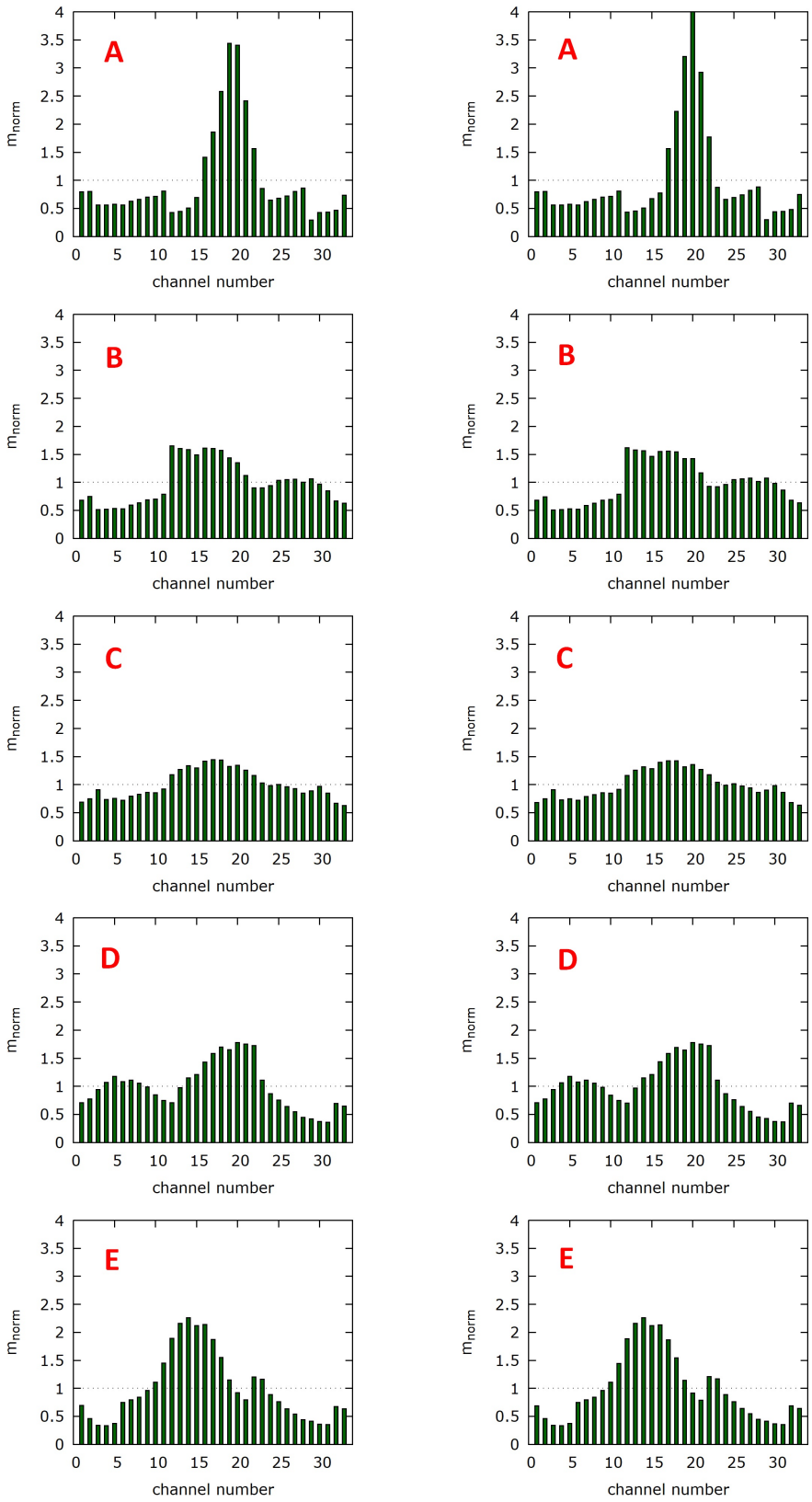

Figure S5. Normalized mass flow at positions A to E for condition MAX. (**Left**) Normal flow at the inlet channels and (**Right**) partial blocking of channel #1.

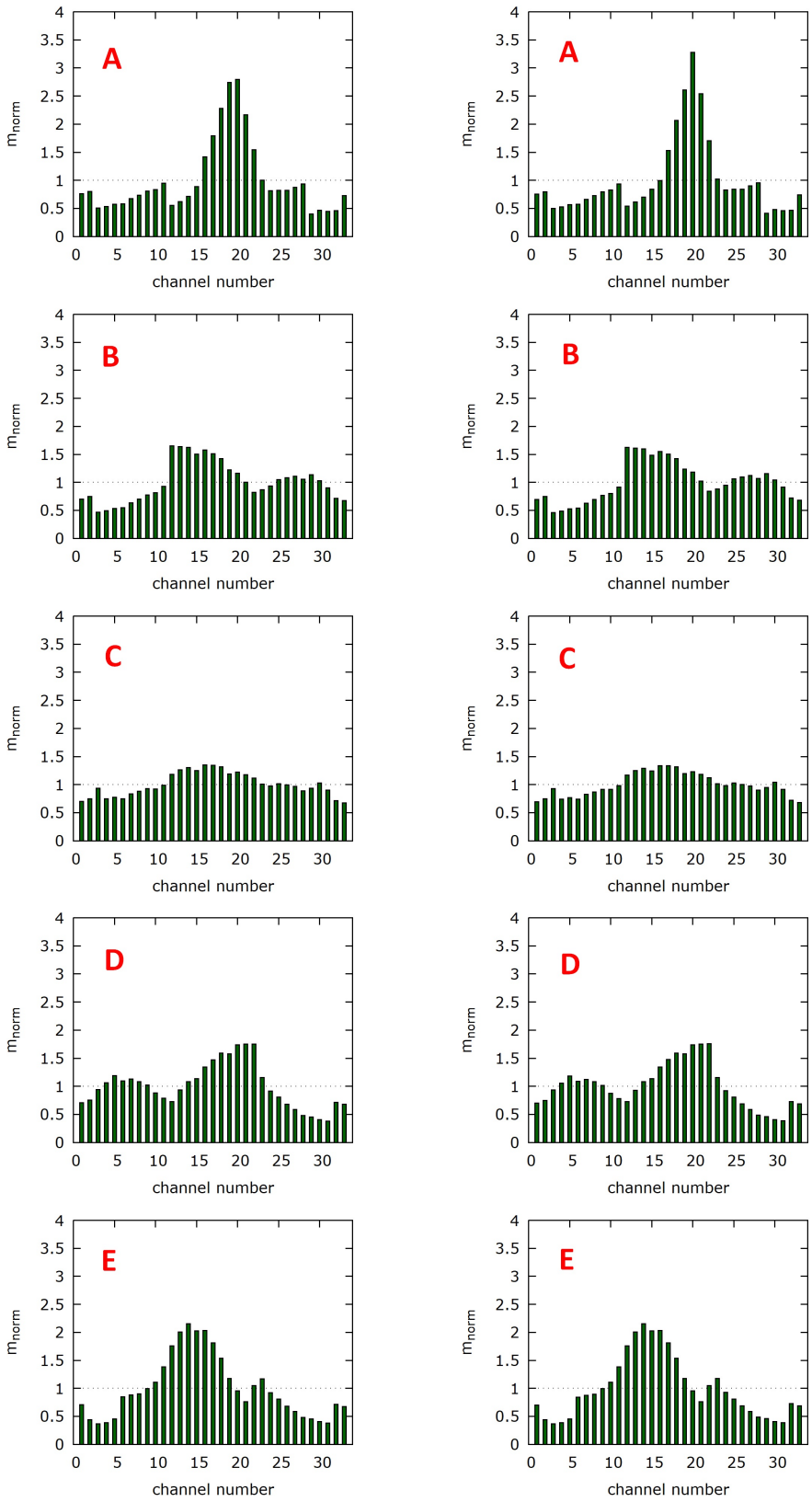

Figure S6. Normalized mass flow at positions A to E for condition FULL. (**Left**) Normal flow at the inlet channels and (**Right**) partial blocking of channel #1.

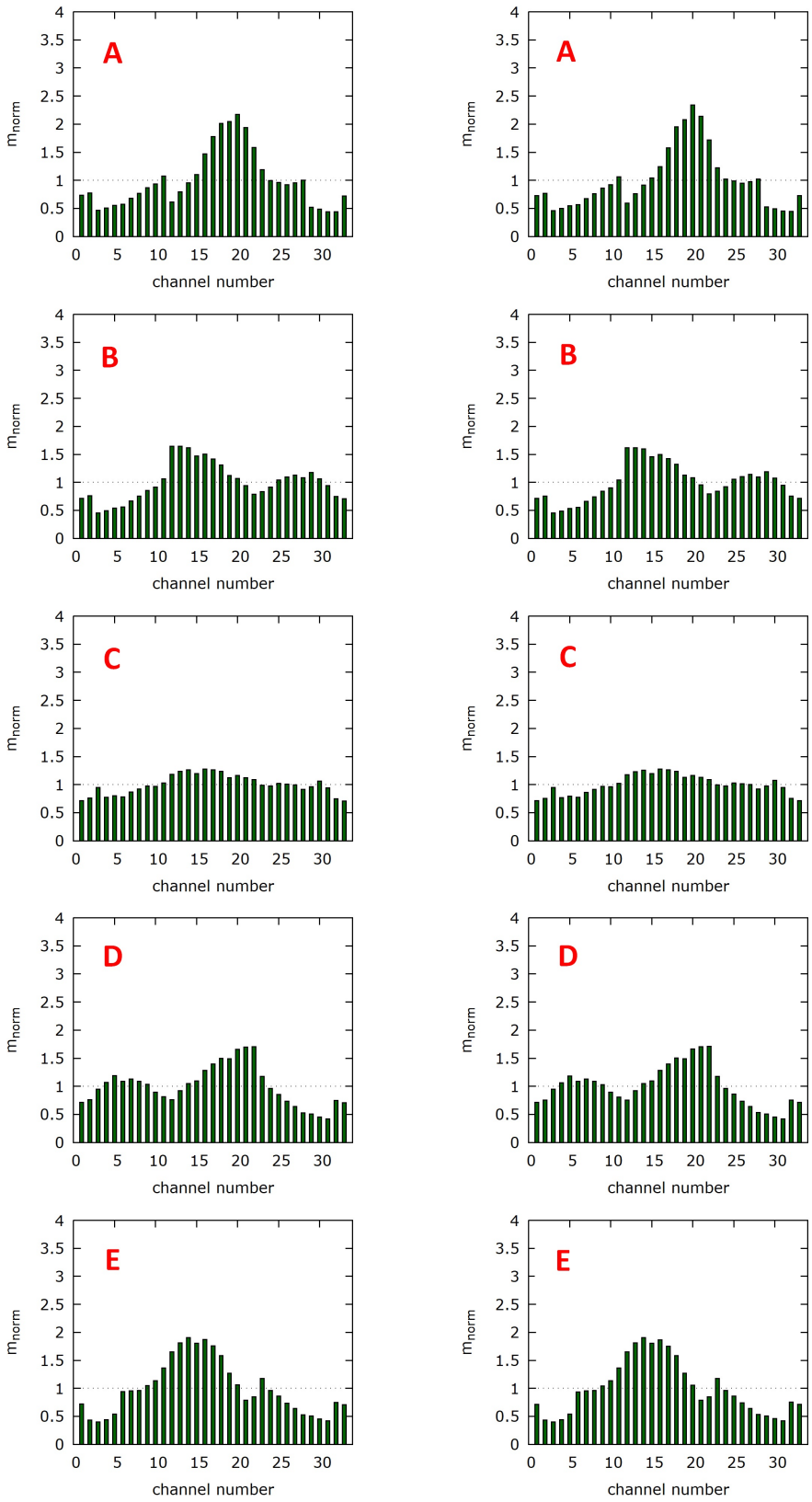

Figure S7. Normalized mass flow at positions A to E for condition MIN. (**Left**) Normal flow at the inlet channels and (**Right**) partial blocking of channel #1.

S4. Mesh generation with SALOME

S4.1. Building blocks

Sometimes it is possible to break down the full geometry into a small number of similiar pieces. These pieces can be generated more easily. The full geometry is obtained by the combination of these pieces as shown in Figure S8 and described in the paper of Reimer et al. [61].

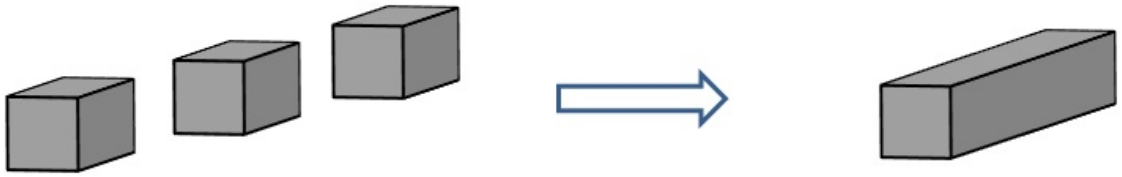

Figure S8. Concept of buiding blocks.

Small blocks are easier to mesh than big, complex geometries. For programmers these blocks are 'objects'. These objects are coded in the programming language PYTHON, which is called from inside the software SALOME. Each object has the following four functions.

- Definition of points (this is needed later for geometry, faces and mesh)
- Creation of the geometry
- Definition of faces and volumes
- Creation of the mesh

S4.2. Example scripts

The example scripts are contained in the following directories.

basic-blocks scripts of the basic blocks and example figures

straight-channel demo flow field with straight parallel channels

par-serpentine demo flow field with parallel serpentine channels

sep-serpentine demo flow field with separate serpentine channels

coolant the coolant flow field from the paper

S4.3. Basic blocks

There are four basic shapes which are needed to create serpentine-like flow fields, which are show in Figure S9.

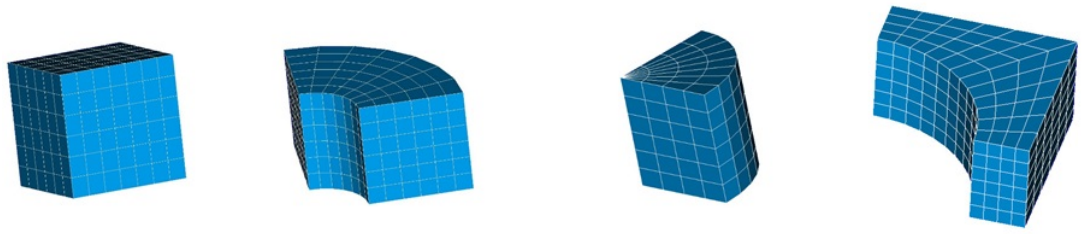

Figure S9. The four basic building blocks with example meshes.

The blocks are located in the subdirectory */shapes*. They are called by wrapper-scripts from the directory *basic-blocks*, which provide all necessary input information and can serve as a template. (These sripts need the information of the full path information on your computer. Please update the path inside the script.) The following subdirectories are available.

fig-developer Example figures that show the numbering of internal points and egdes.

fig-docu Example figures that show the overall shape and structure of the mesh.

shapes Scripts for the basic blocks.

The building blocks possess more than one function for mesh generation. The choice of the function depends on the overall concept for building the final geometry, where the single blocks are glued together. This operation is performed

by the SALOME function 'Concatenate'. This function requires the specification of the tolerance, for which mesh points should be merged. The correct value for the parameter 'tolerance' needs to be adapted to lie below the smallest distance within the chosen meshing pattern. Figure S10 shows two examples of meshes with concentrated points near the walls. This effect is realized by defining a mesh gradient along edges. It requires that each side of the shape is broken into two edges.

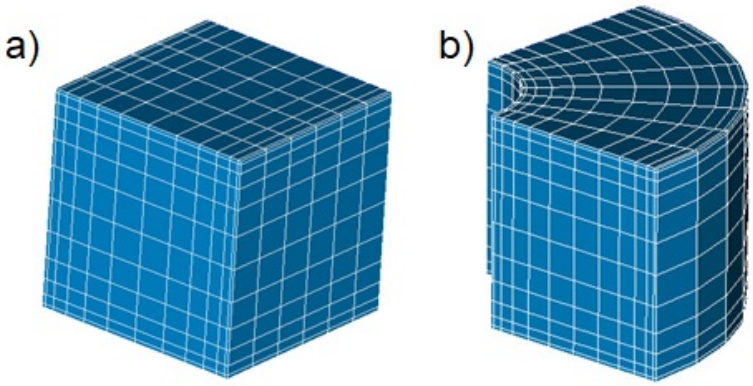

Figure S10. Refined mesh near walls. (**a**) script 'demo-BoxGrad.py' and (**b**) script 'demo-BendGrad.py'.

It is also possible to use tetrahedrons and wedges for the mesh construction, as shown in Figure S11. There is some random component in the distribution of the triangle faces within the present versions of the SALOME software. Therefore, it is problematic to merge two meshes like in Figure S11 b), because the mesh points at the faces may not match.

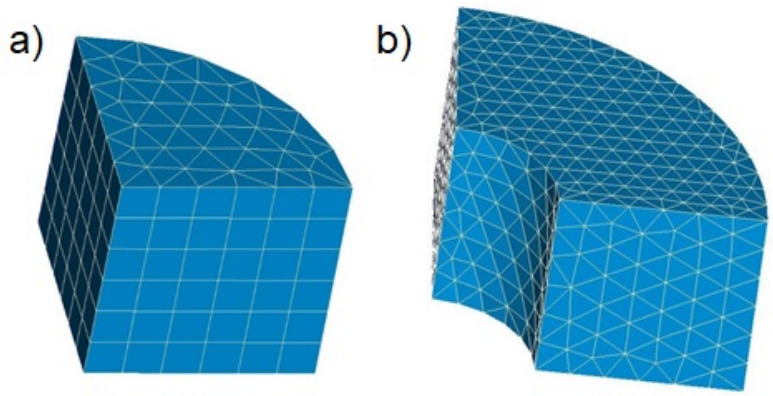

Figure S11. Mesh with wedges and tetrahedrons. (**a**) script 'demo-PieTri.py' and (**b**) script 'demo-BendTri.py'.

S4.4. Creation of flow fields

The flow fields of the paper are constructed for each half cell, i.e. anode side and cathode side separately, as shown in Figure S12. It consists of two layers. Layer 1 defines the flow channels. These channels must be open towards layer 2 and are enclosed by boundaries otherwise. The following boundary conditions are applied within this layer: inlet (velocity), outlet (pressure), wall (no slip condition). Layer 2 resembles the porous transport layer and has the following boundary conditions: pwall (slip) and inlet/outlet (velocity). The mesh of layer 2 needs to be given a name in order to be recognized as a cell zone by OpenFOAM. The key challenge is that the faces of the mesh of each building block must fit exactly to the neighbouring block.

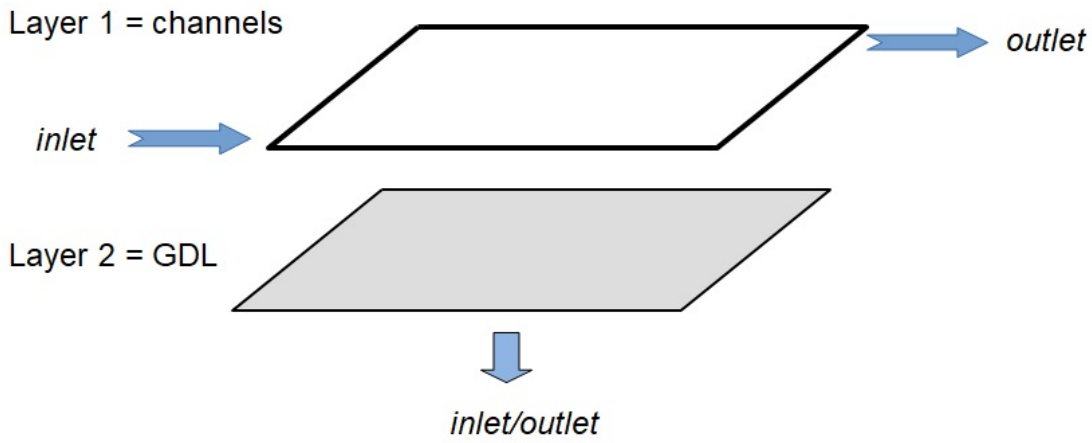

Figure S12. Sketch of a flow field for half cell simulations.

The flow field is actually created in four steps.

1. Creation of the channels for the first layer.
2. The same method is used to create the space of the 'channels' within the porous layer (second layer). Here, a different height and different boundaries are used.
3. Filling the void space in layer 2. This corresponds to the structure of the ribs in a flow field.
4. The three submeshes from 1) - 3) are merged. The three submeshes can be viewed separately in the object browser within the software SALOME.

S4.5. Straight parallel channels

Figure S13 shows an example of five straight parallel channels from the script 'straight/straight-chan.py'.

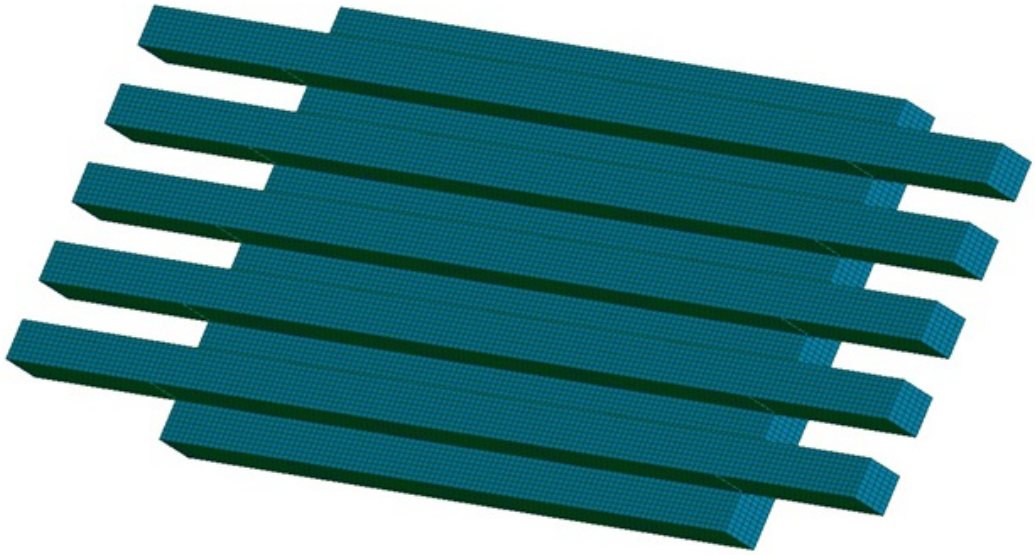

Figure S13. Flow field with 5 straight parallel channels.

The following parameters can be changed in the header section of the script 'straight/straight-chan.py'.

- number of channels
- channel width and height
- channel/rib ratio
- height of GDL
- flow field length
- number of mesh cells per channel cross section
- number of mesh cells for channel height
- name of boundary patches, i.e. 'wall', 'membrane', 'inlet', 'outlet', 'pwall'
- name for cell zones, i.e. 'porous', 'chan'

S4.6. Parallel serpentine channels

Figure S14 shows an example flow field with three parallel serpentine channels from the script 'serpentine/serpentine-FF.py'.

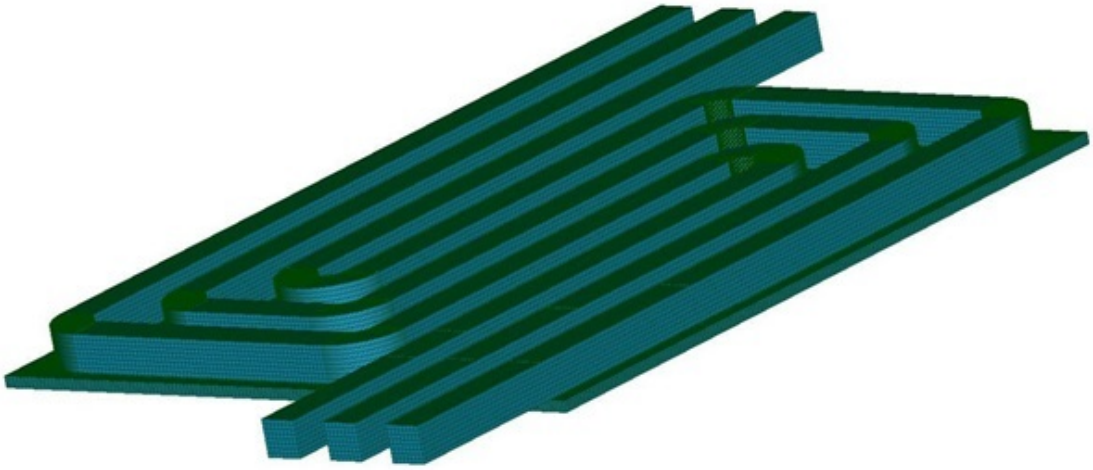

Figure S14. Flow field with 3 parallel serpentine channels.

The following parameters can be changed in the header section of the script 'serpentine/serpentine-FF.py'.

- number of channels
- number of U-turns
- channel width and height
- channel/rib ratio
- height of GDL
- flow field length
- number of mesh cells per channel cross section
- number of mesh cells for channel height
- name of boundary patches, i.e. 'wall', 'membrane', 'inlet', 'outlet', 'pwall'
- name for cell zones, i.e. 'porous', 'chan'

The script 'serpentine/serpentine-FF-grad.py' creates the same geometry with a refined mesh near the walls. For the refinement a parameter 'seggrad' is used, which is hard coded (i.e. in '/shapes/BoxGrad.py'). It uses the fowlloing relations: 'segmin = seggrad * 0.2' and 'segmax = seggrad * 0.8'. Generally speaking, the paramater 'seggrad' is a reference length. The size of the smallest cells is 0.2 times this reference.

S4.7. Separate serpentine channels

Figure S15 shows an example flow field with three separate serpentine channels from the script 'snake/snake-FF.py'.

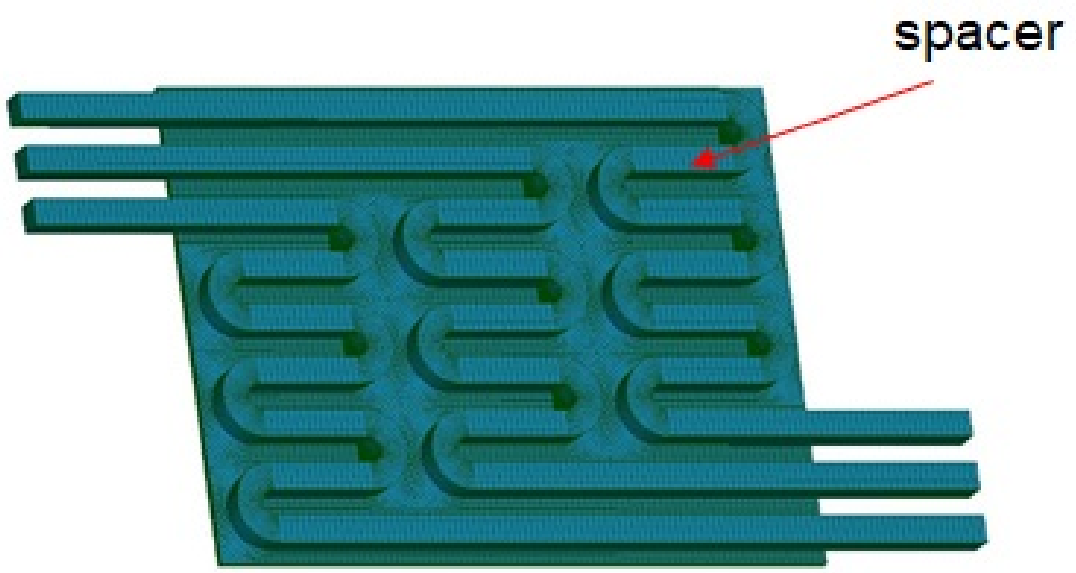

Figure S15. Flow field with 3 separate serpentine channels.

The following parameters can be changed in the header section of the script 'snake/snake-FF.py'.

- number of channels
- number of U-turns (even numbers only!)
- channel width and height
- channel/rib ratio
- height of GDL
- flow field length
- number of mesh cells per channel cross section
- number of mesh cells for channel height
- name of boundary patches, i.e. 'wall', 'membrane', 'inlet', 'outlet', 'pwall'
- name for cell zones, i.e. 'porous', 'chan'

S4.8. General work flow for mesh generation with SALOME

1. edit flow field script:
   - set correct path for sub-shapes folder
   - set number of channels/ mesh resolution/ ...
2. start SALOME
3. load script (CTRL + 'T' or File → Load script)
4. change to mesh-view interface
5. export mesh as *.unv file (File → Export or right mouse click on the mesh in the object browser)
6. shut down SALOME
7. import *.unv file into OpenFOAM (by using 'ideasUnvToFoam')

Other useful hints for using the mouse within SALOME are:

- CTRL + left click + move = zoom
- CTRL + right click + move = free rotation
- turn wheel = zoom (center at mouse pointer)
- CTRL + push wheel + move = placement, move object

See also the geometry builder online documentation at
https://docs.salome-platform.org/9/gui/GEOM/index.html and the mesh documentation at
https://docs.salome-platform.org/9/gui/SMESH/index.html.

Also check 'Python interface' for programming under
https://docs.salome-platform.org/9/gui/GEOM/geompy_page.html and
https://docs.salome-platform.org/9/gui/SMESH/smeshpy_interface.html.