Scripting as an approach to automated CFD simulation for packed bed catalytic reactor modelling

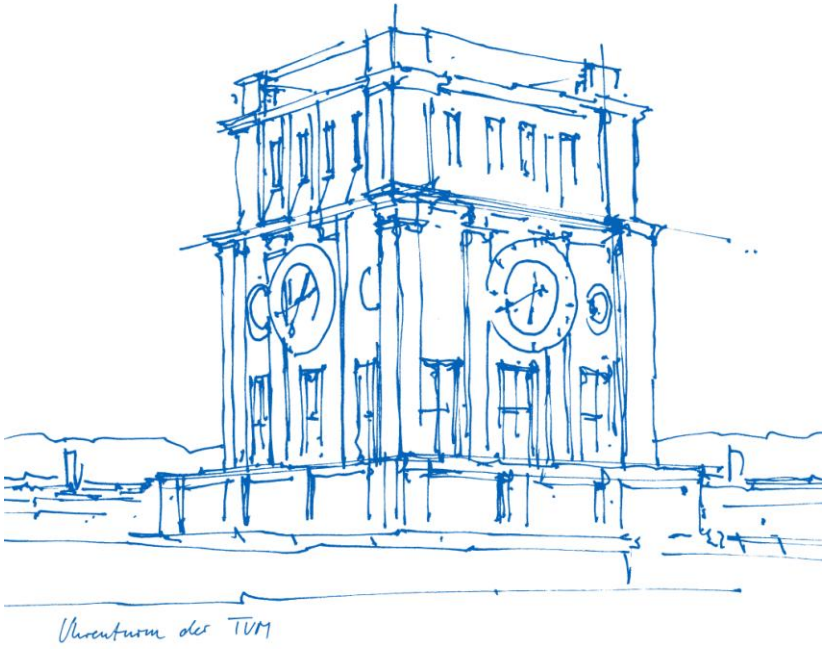
Johanna Fernengel | Florian Habla | Olaf Hinrichsen

Technical University Munich

Department of Chemistry

Chair I of Technical Chemistry

Guimarães, June 2016



Introduction

Fixed Bed Reactors

- Widely used in chemical industry – reaction, separation, purification
- Packed bed with pellets




- Focus is set on packed beds with low to medium cylinder-to-pellet diameter ratios
  - used for highly endothermic and exothermic reactions in process industries

CFD Simulation - Work Flow and Softwares

Pre-processing
- case set-up
  - geometry
  - mesh settings
  - flow and reaction conditions
- mesh generation

↓

Simulation

↓

Post-processing
- data sampling and visualisation

Software packages

OpenFOAM®
- blockMesh
- snappyHexMesh
- simpleFoam
- scalarTransportFoam
- customized applications

blender

ParaView

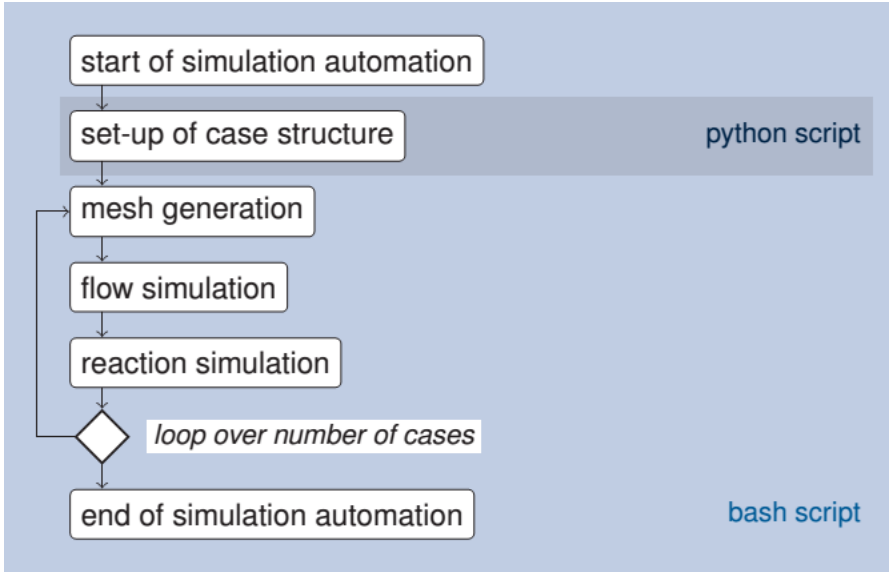
Standard workflow for many simulations with little variation of the case settings

Use of scripts may be an efficient alternative to manual stepwise simulation execution

Automated Simulation Routine

Automated Simulation Routine

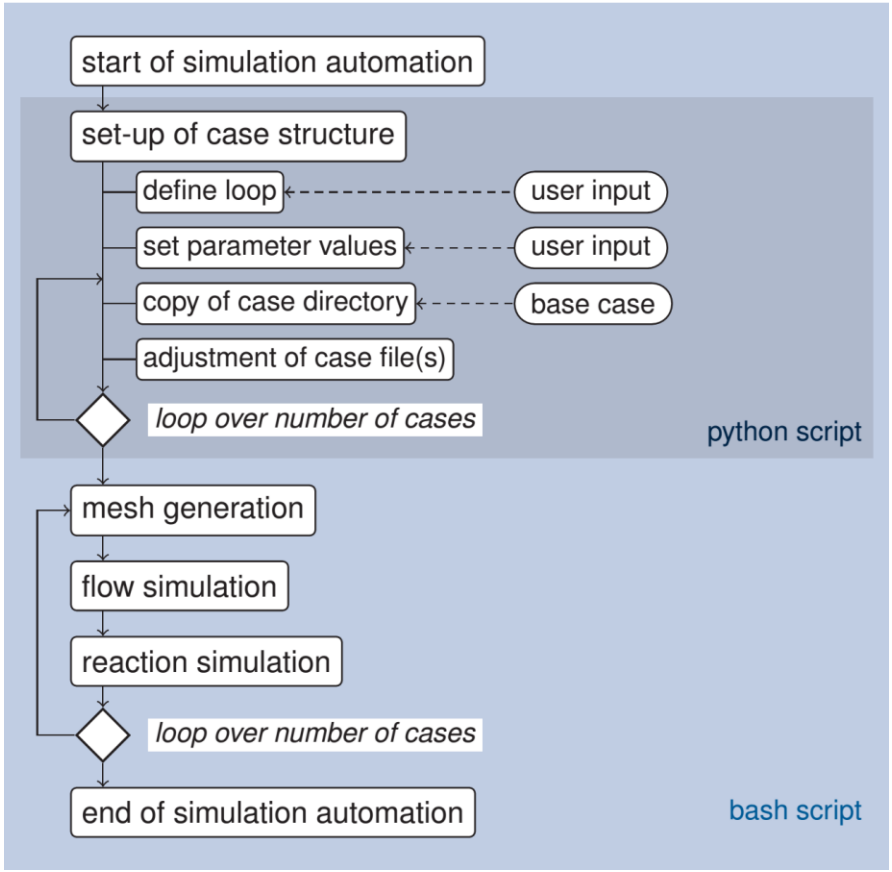
- Aim
  - automation strategy covering multiple entire CFD simulation runs
- Approach
  - use of python and bash scripting for software execution and file manipulation
  - integration of softwares via scripting interface



Automated Simulation Routine – Case Set-up

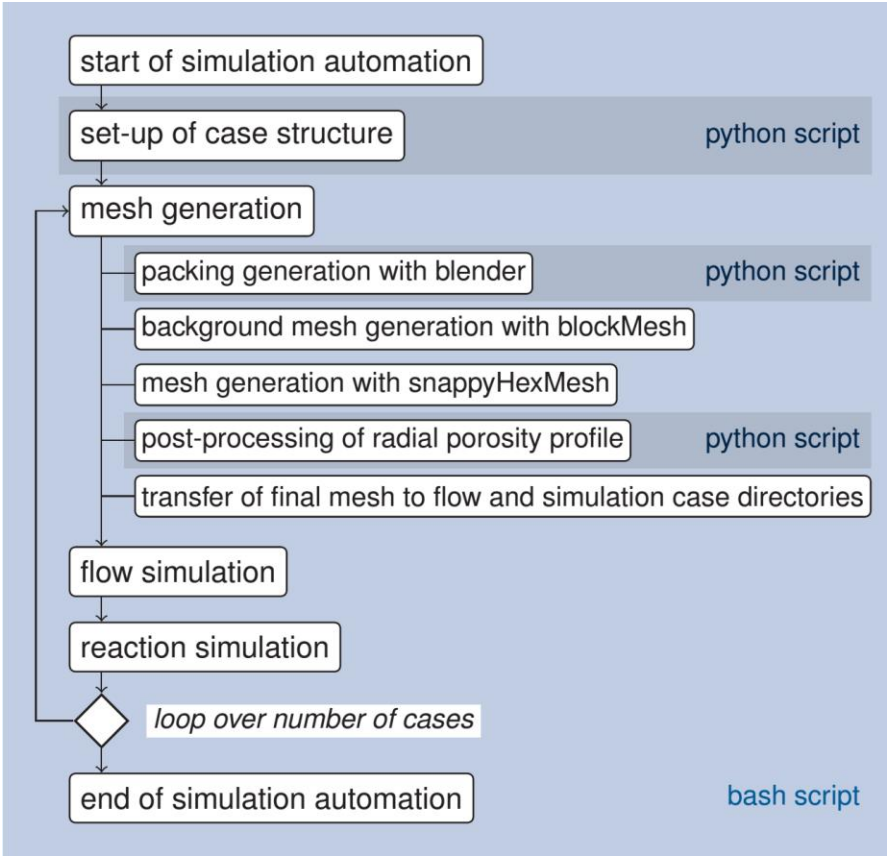
- Utilisation of „base case“ containing common file and directory structure
- High parameterisation of „base case“ files
  - Python functionality allows to change all parameter values according to user input

```
q = dict()
q['L '] = bedLength  #calculated value

# … read blockMeshDict file …

for key in q.keys():
  for line in lines:
    if line.startswith(key):
      lineNumber = lines.index(line)
      lines[number] = str(key) + \t +
                  str(q[key]) + ';\r\n'

# … write blockMeshDict file …
```



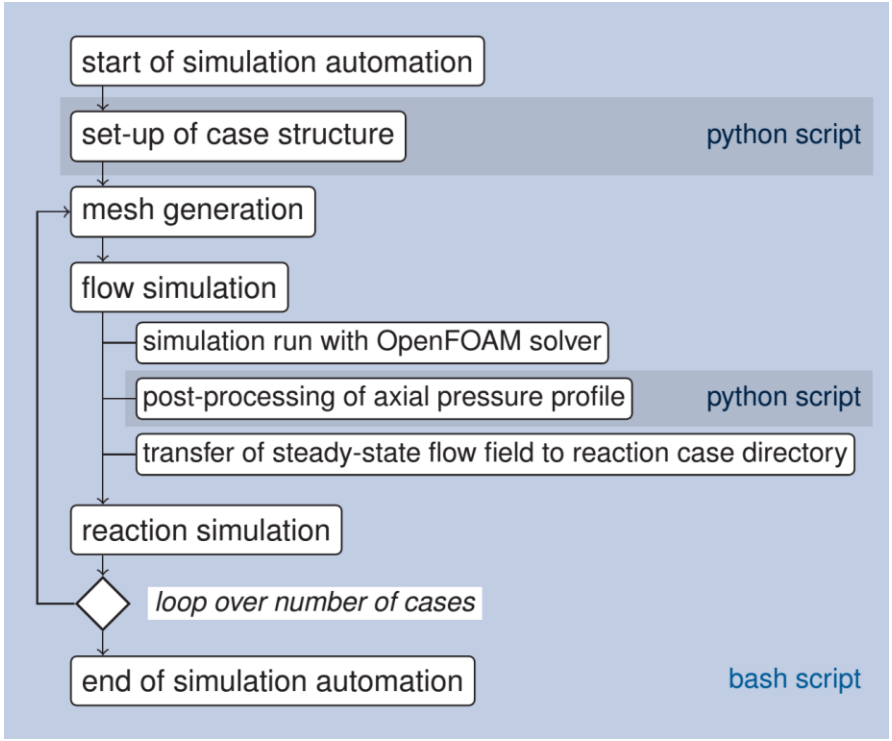
Automated Simulation Routine - Mesh Generation

- Packing generation
  - identify pellet positions by DEM
  - Blender software
  - Python interface with highly parameterised base script
- Meshing
  - cylindrical background mesh with blockMesh
  - snappyHexMesh
- Post-processing of axial and radial porosity profiles
  - ParaView via Python interface



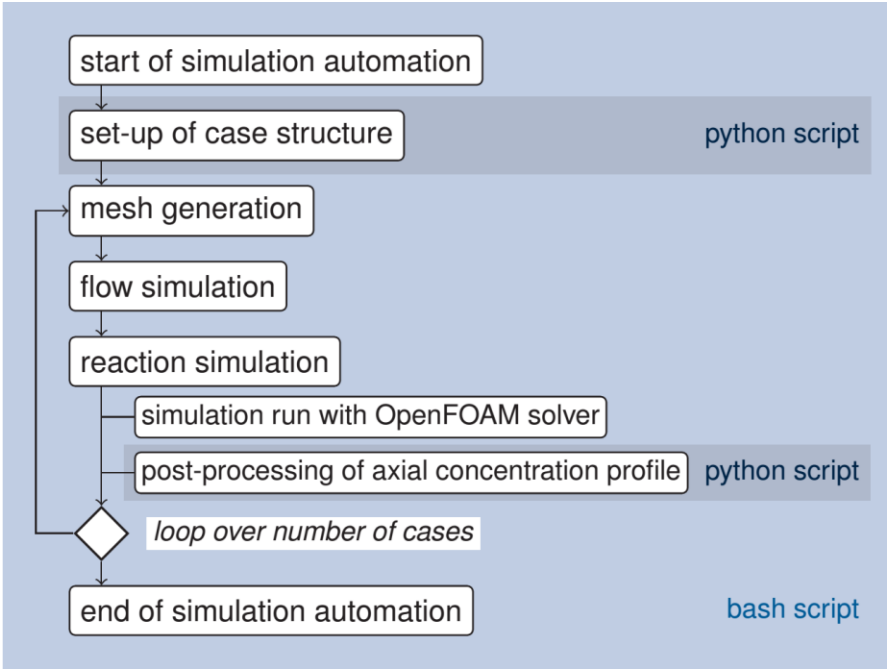
Automated Simulation Routine – Flow Simulation

- Flow simulation based on standard OpenFOAM solvers
  - assumption that flow is independent of reaction
  - cold-flow simulation with simpleFoam
  - evaluation of residence time distribution with scalarTransportFoam
- Post-processing of axial and radial profiles of pressure and velocity
  - ParaView via Python interface
- Post-processing of residence time distribution
  - modified patchAverage tool



Automated Simulation Routine – Reaction

- A simple test reaction considered in this work
  - $A \rightarrow B$
  - 1st-order, no volume contraction/expansion
  - solution of convection-diffusion equation
  - reaction is implemented as boundary condition on pellet surface
  - diffusive transport of reactant = depletion of reactant on surface
  - depletion of reactant = product formation
- Simulation of species concentration with customized scalarTransportFoam
- Post-processing of concentration profiles
  - ParaView via Python interface



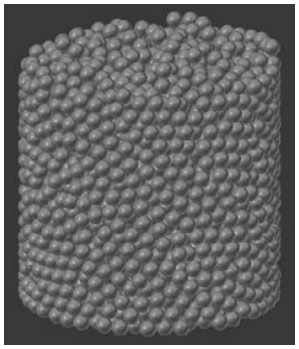
Details
Mesh Generation

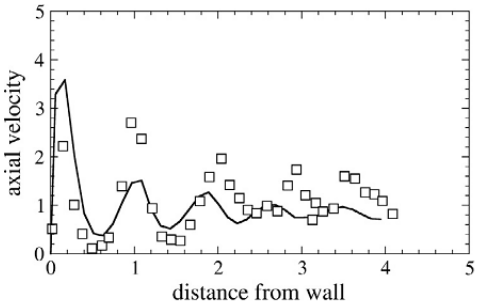
Packed Bed Generation with blender

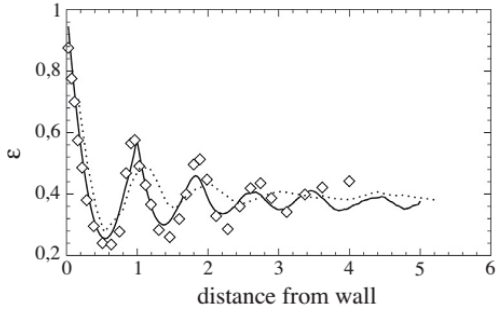
- Blender
  - free and open source 3D animation suite published by the Blender Foundation, www.blender.org
  - allows advanced physics simulation like rigid-body physics by integrating the Bullet Physics Library, www.bulletphysics.org
  - Python interface offers wide range of possibilities to vary the packed bed by varying e.g. pellet shape, size distribution, …

- Boccardo et al.
  - use of Blender for packed bed generation for CFD simulations
  - good agreement with real packings

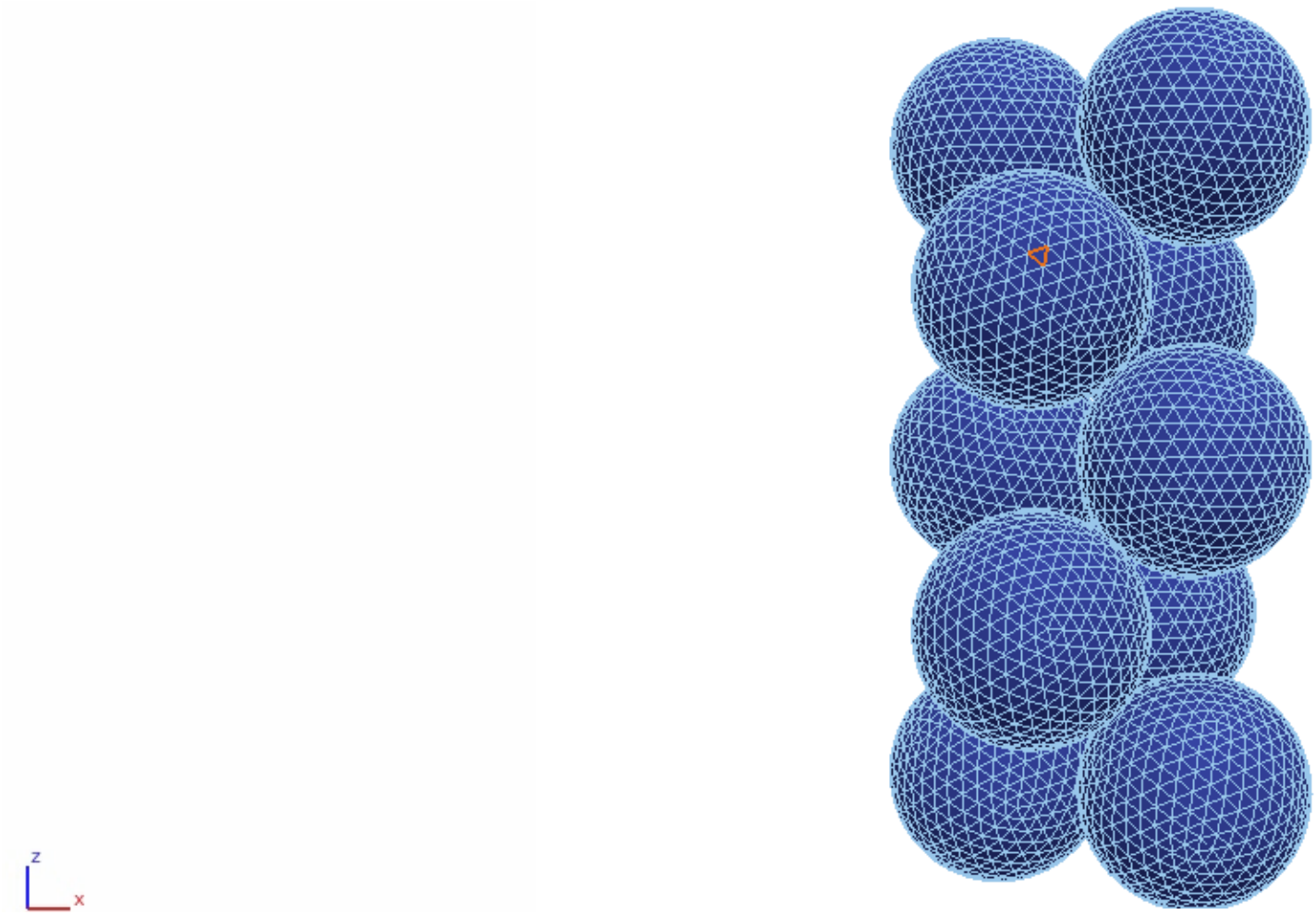
[Boccardo et al., Chem Eng J 279 (2015) 809-820]







Packed Bed Generation with blender™ - Result



Packed Bed Generation with blender™ - Script I

```
#!/usr/bin/python
import bpy
…
### parameters ###
particleNumber = 12
particle Diameter = 1
cylinderDiameter = 2
…
particleFriction = 0
particleRestitution = 0
particleLinearDamping = 0.999
particleAngularDamping = 0.999
…
simulationFrameStart = 1
simulationFrameEnd = 2000
…
```

- Python script declaration
- Load blender

- High parameterisation for easy value accessibility

- Parameters include
  - number of pellets
  - pellet and cylinder geometry
  - rigid body settings like friction
  - rigid body simulation settings

Packed Bed Generation with blender - Script II

```
# particle
while n < particleNumber:
bpy.ops.mesh.primitive_ico_sphere_add(
 subdividions = … , size = … , location = …)
bpy.ops.object.origin_set(…)
bpy.ops.rigidbody.objects_add(type=ACTIVE)
obj = bpy.context.object.rigid_body
obj.enabled = True
obj.mass = particleMass
obj.collision_shape = SPHERE
obj.friction = particleFriction
obj.restitution = particleRestitution
obj.linear_damping = particleLinearDamping
obj.angular_damping = particleAngularDamping
n += 1
```

- Loop over number of pellets
- Add pellet (here: sphere)
- Initial location

- Declare as active object

- Set particle mass
- Set collision shape

- Set rigid body properties

Packed Bed Generation with blender™ - Script III

```
# cylinder
bpy.ops.mesh.primitive_cylinder_add(
 end_fill_type = … , vertices = … ,
 radius = … , depth = … , location = …)


bpy.ops.rigidbody.objects_add(type=PASSIVE)
bpy.ops.object.modifier_add(type=SOLIDIFY)
bpy.context.object.modifiers[Solidify].thickness
  = cylinderThickness
bpy.ops.object.modifier_apply(…)

obj = bpy.context.object.rigid_body
obj.collision_shape = MESH
obj.mesh_source = BASE
obj.friction = …
```

- Add cylinder
- Cylinder location
- Declare as passive object
- Solidify cylinder, i.e. add thickness to the object
- Set collision shape
- Set rigid body properties

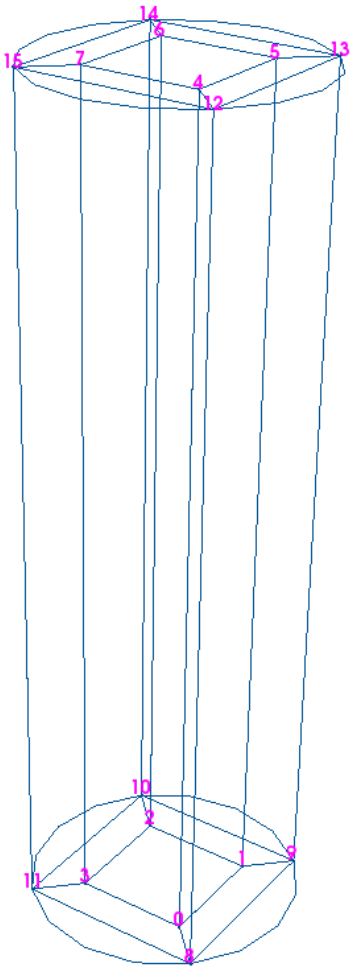
Packed Bed Generation with blender - Script IV

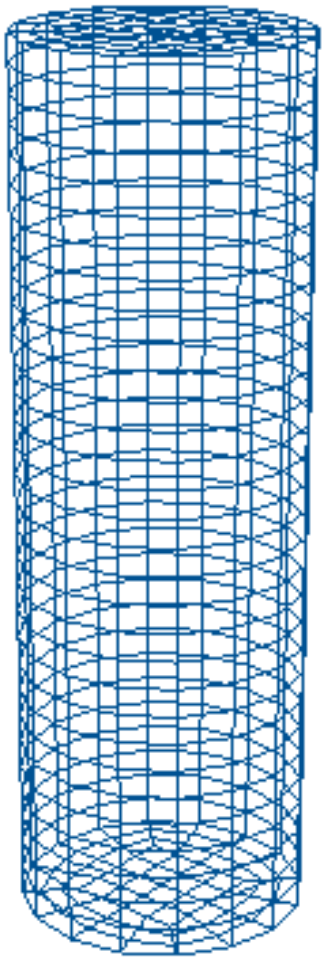
```
# animation settings
…

# run animation
…
while simulationFrameCurrent <=
simulationFrameEnd:
  bpy.context.scene.frame_set(frame =
    simulationFrameCurrent)
  simulationFrameCurrent += 1

# export stl
bpy.ops.object.select_all(action=SELECT)
bpy.ops.export_mesh.stl(filepath= … )

# safe file
bpy.ops.wm.save_as_mainfile(filepath= … )
```

- Update frame while simulation has not reached the final frame
- Export pellets as stl-file
- Save Blender file

Background Mesh with blockMesh

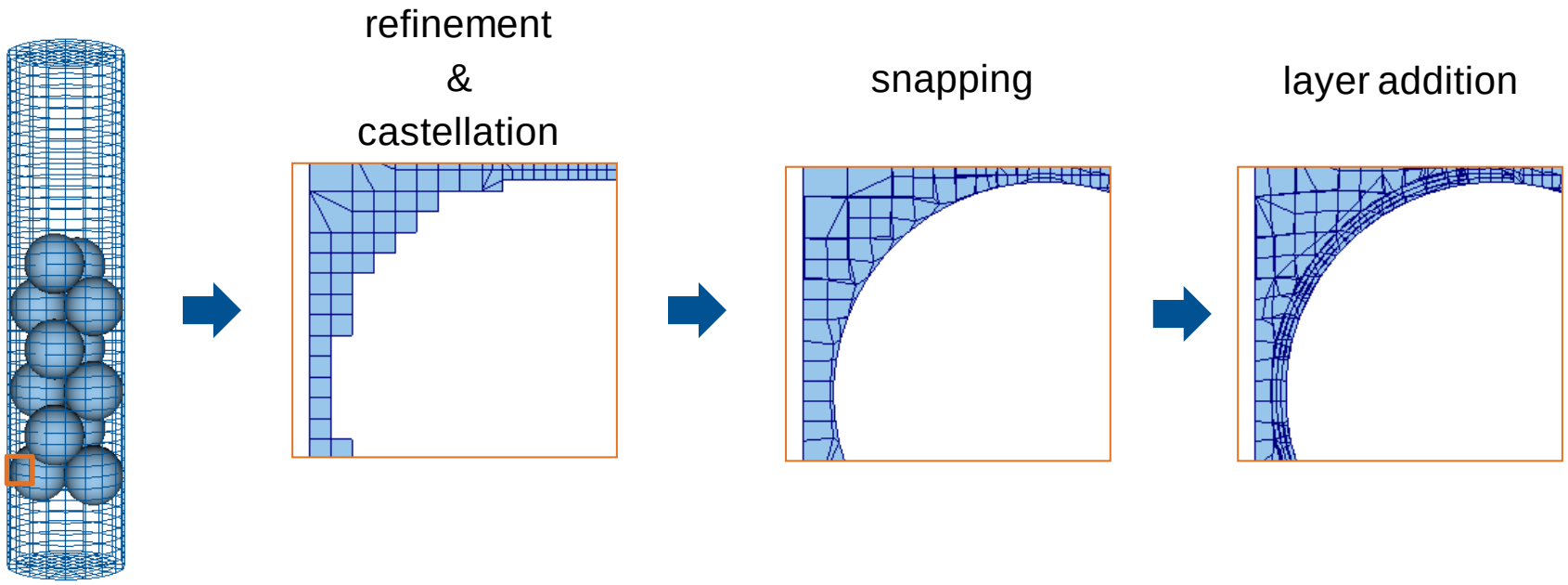
```
// parameters
D 7.35;
L 5.085134;
deltaR 0.2;
deltaZ $deltaR;
point 0.3;

x0 #calc "-0.5*$D";
x1 #calc "-$point*$D";
x2 0;
x3 #calc "$point*$D";
x4 #calc "0.5*$D";

vertices
(
    ($x2 $y3 $z0) // 0
     …
);
```





Mesh Generation with snappyHexMesh

- Mesh generation from hexahedral background mesh (cylinder) and surface geometry in stereolithography format (pellets stl-file)



Details
Python based Post-Processing with ParaView

Post-Processing of Radial Porosity Profile

- Porosity = void fraction
  - $\varepsilon = \frac{V_{void}}{V}$
  - measure of voidage in a specified volume

- Radial porosity profile evaluation based on the generated mesh
  - reduction of generated mesh to region with packed bed
  - generation of isosurfaces at various radii using ParaView's contour filter
  - evaluation of surface area in relation to total area of corresponding open cylinder

Post-Processing of Radial Porosity Profile I

```
#!/opt/paraviewopenfoam410/bin/pvpython

from paraview.simple import *
import math // import os // import vtk // …

lastVTKNumber = sorted( … , key=int)[-1]
vtkInputFile = [os.path.join(os.curdir,'VTK',f)
  for f in … if lastVTKNumber in f]
stlInputFile = [os.path.join(os.curdir, … , f)
  for f in fnmatch.filter( … , '*.stl')]

vtkReader =
  LegacyVTKReader(FileNames=vtkInputFile)
stlReader = STLReader(FileNames=stlInputFile)
Show()

(xmin,xmax,ymin,ymax,zmin,zmax) =
  stlReader.GetDataInformation().GetBounds()
```

- Declaration as ParaView Python application
- Import required modules
- Identify latest vtk time step, i.e. final mesh
- Locate latest vtk file as well as stl file
- Create reader to access vtk and stl file
- Get packing bounds from stl file

Post-Processing of Radial Porosity Profile II

```
upperZ = zmax - 1.5*particleDiameter
lowerZ = zmin + 0.5*particleDiameter
packingHeight = upperZ - lowerZ

reducedCylinder = Clip(Input = vtkReader,
  ClipType = 'Plane')
reducedCylinder.ClipType.Normal = [0, 0, 1]
reducedCylinder.ClipType.Origin = [0, 0, lowerZ]
packing = Clip(Input = reducedCylinder,
  ClipType = 'Plane')
packing.ClipType.Normal = [0, 0, -1]
packing.ClipType.Origin = [0, 0, upperZ]

calc = Calculator(Input = packing,
  ResultArrayName = 'radius',
  Function = 'sqrt(coordsX^2 + coordsY^2)')
calcData = servermanager.Fetch(calc)
r = calcData.GetPointData().GetArray('radius')
  .GetRange()
```

- Calculate packing height and also eliminate influence of incomplete pellet layers
- Cut cylinder mesh to packing bounds with ParaView‘s Clip filter
- Define radius with ParaView‘s calculator tool
- Get min and max radius

Post-Processing of Radial Porosity Profile III

```
contour = Contour(
  Input = calc, ContourBy = ['POINTS','radius'],
  Isosurfaces = [(r[1]-r[0])/2])

while i < imax-2:
 radius=r[1]-float(i+1)/float(imax+1)*(r[1]-r[0])
 cylinderArea = 2*radius*float(math.pi)*
    packingHeight
 contour.Isosurfaces = [radius]
 I = IntegrateVariables(contour)
 data = paraview.servermanager.Fetch(I)
 contourArea = data.GetCellData()
    .GetArray('Area').GetValue(0)
 outputLine = str(radius) + '\t' + … +
    str(contourArea/cylinderArea) + '\n'
 resultsFile.write(outputLine)
 i += 1

resultsFile.close()
```

- Initialise ParaView's Contour filter
- Loop over range of radii
- Calculate radius
- Calculate area of a cylinder with given height and radius
- Reset contour filter to current radius
- Obtain contour area by integration
- Write value like radius, contour area, etc. to results file

Results

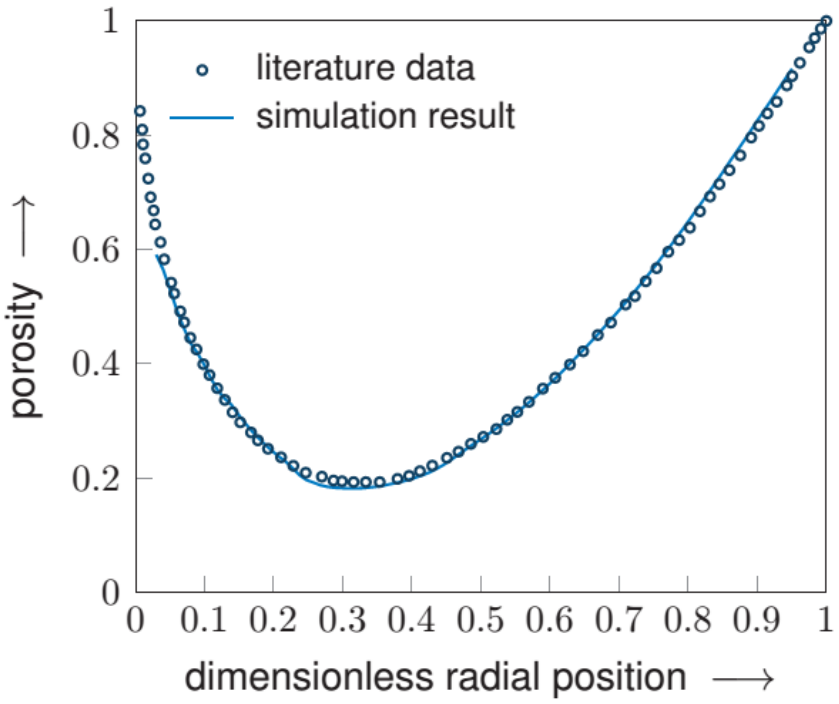
Case Study

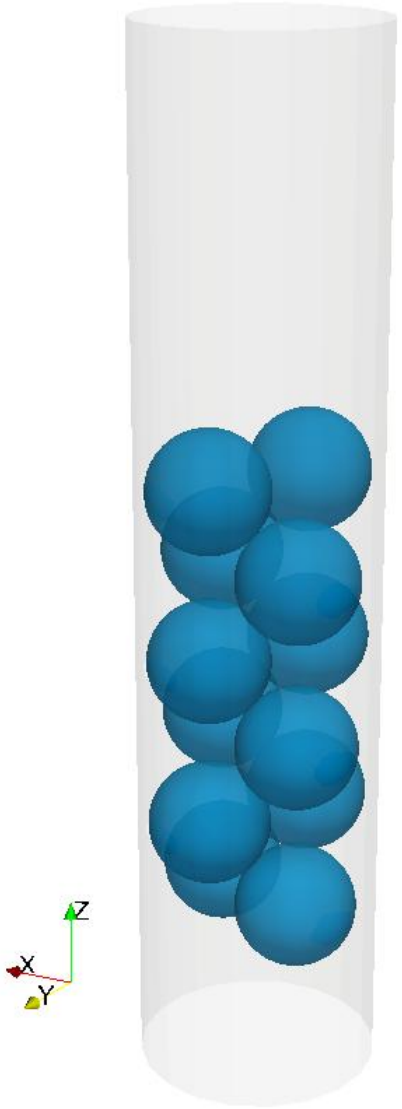
- Spherical pellets with uniform size

- Single phase gas flow
- Bed Reynolds number = 1 (laminar flow)
- Irreversible 1st-order surface reaction

- Results presented for
  - D/d_P = 2 / 7.35 / 8.41
  - Radial porosity profile
  - Axial pressure profile
  - Flow visualization
  - Residence time distribution
  - Concentration on pellet surface
  - Conversion

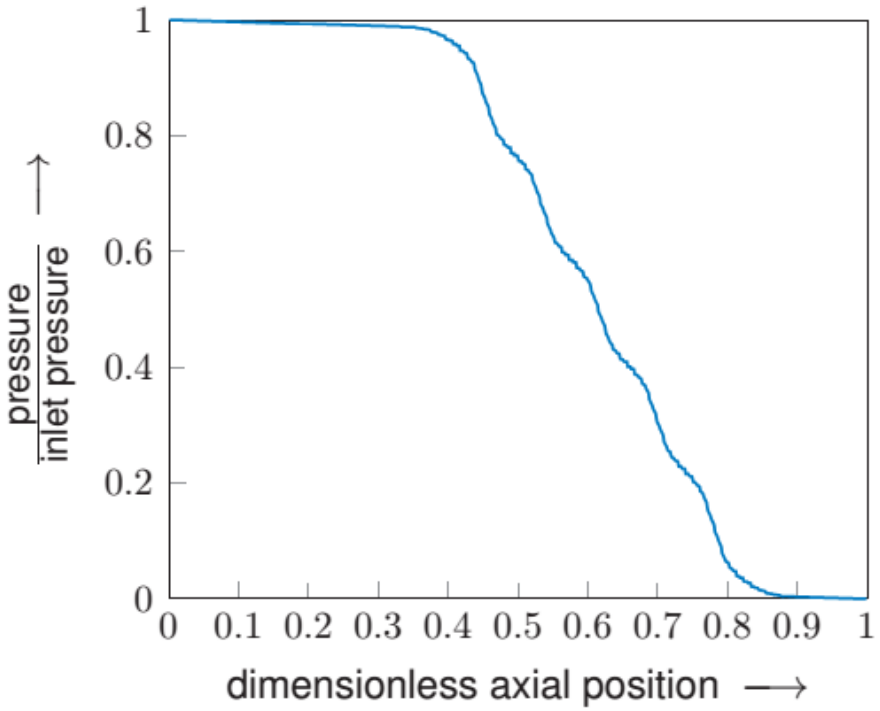
Results for $D/d_P = 2$

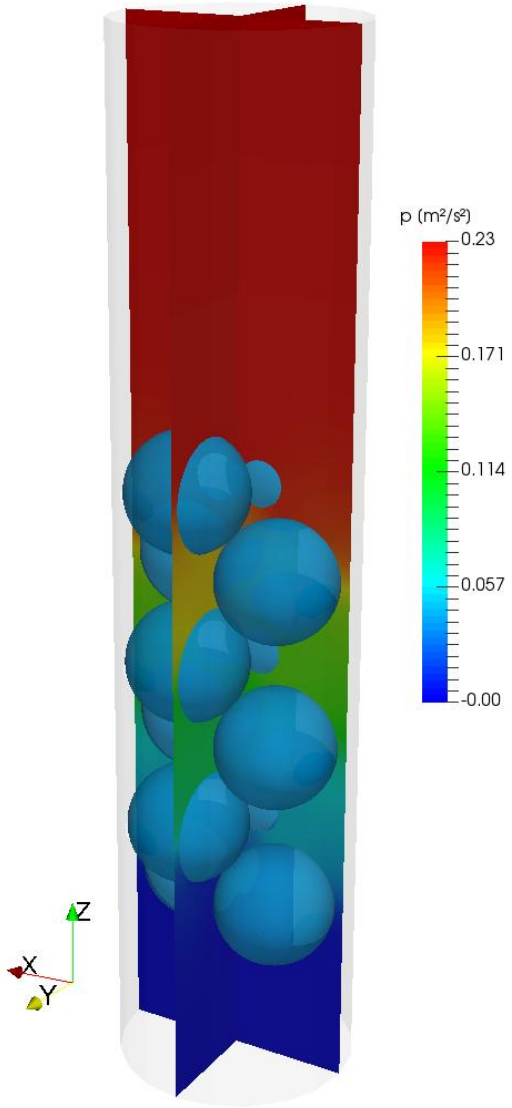




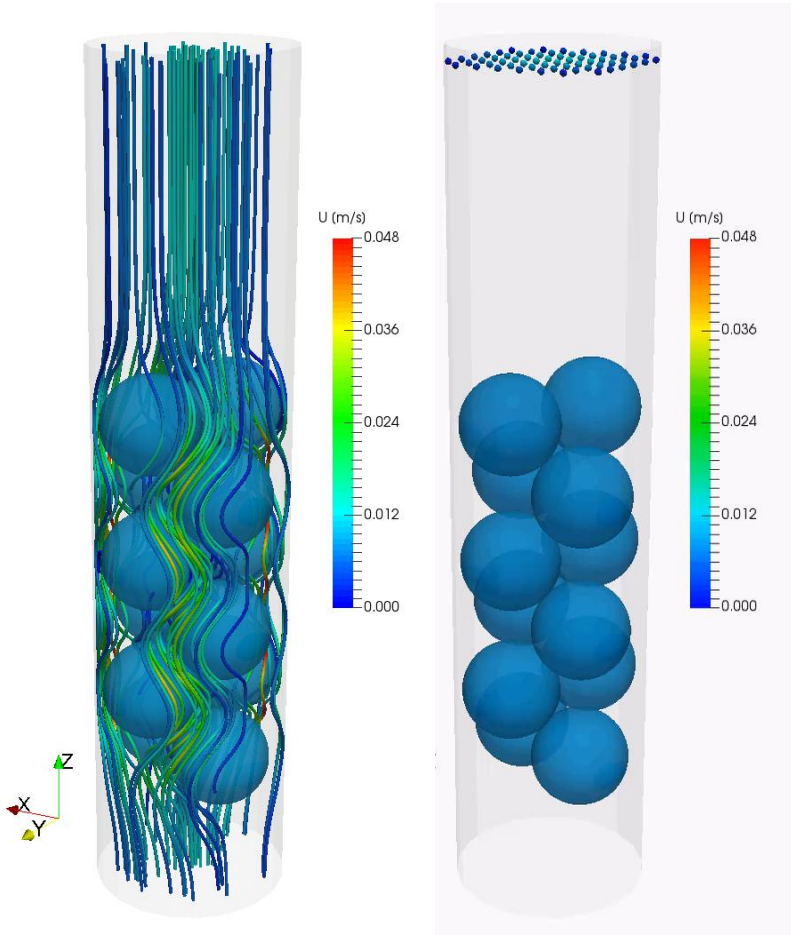
- literature values from Govindarao et al.
  [Govindarao et al., Chem. Eng. Sci. 47 (1992) 2105-2109]

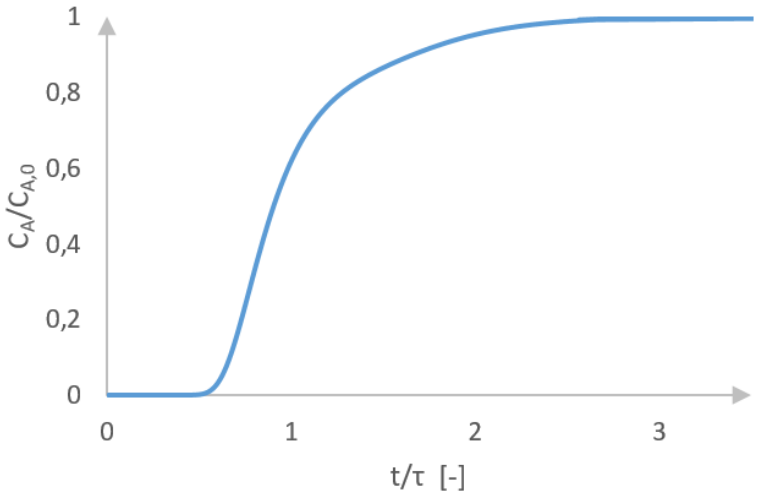
Results for $D/d_P = 2$

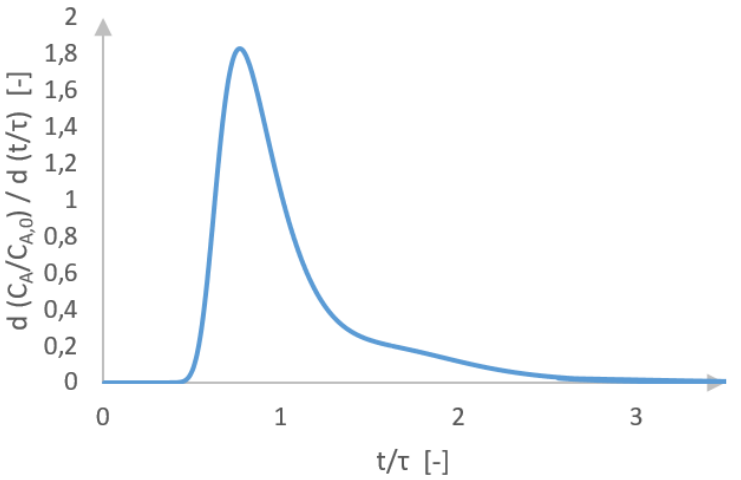




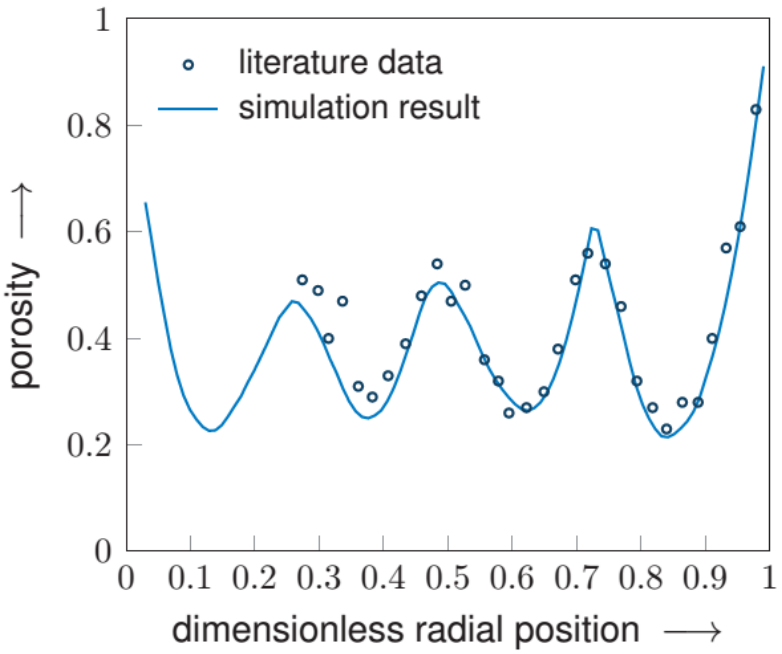
Results for $D/d_P = 2$





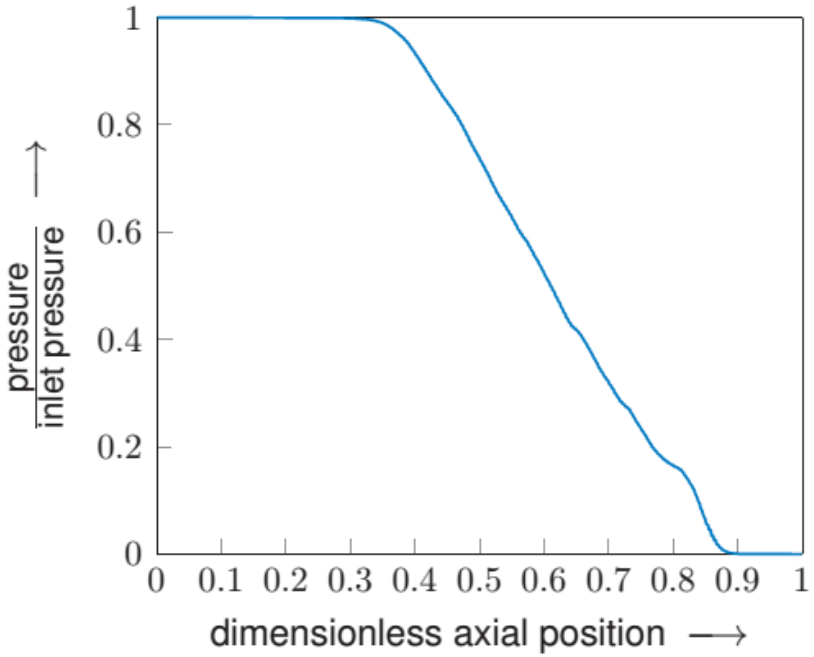


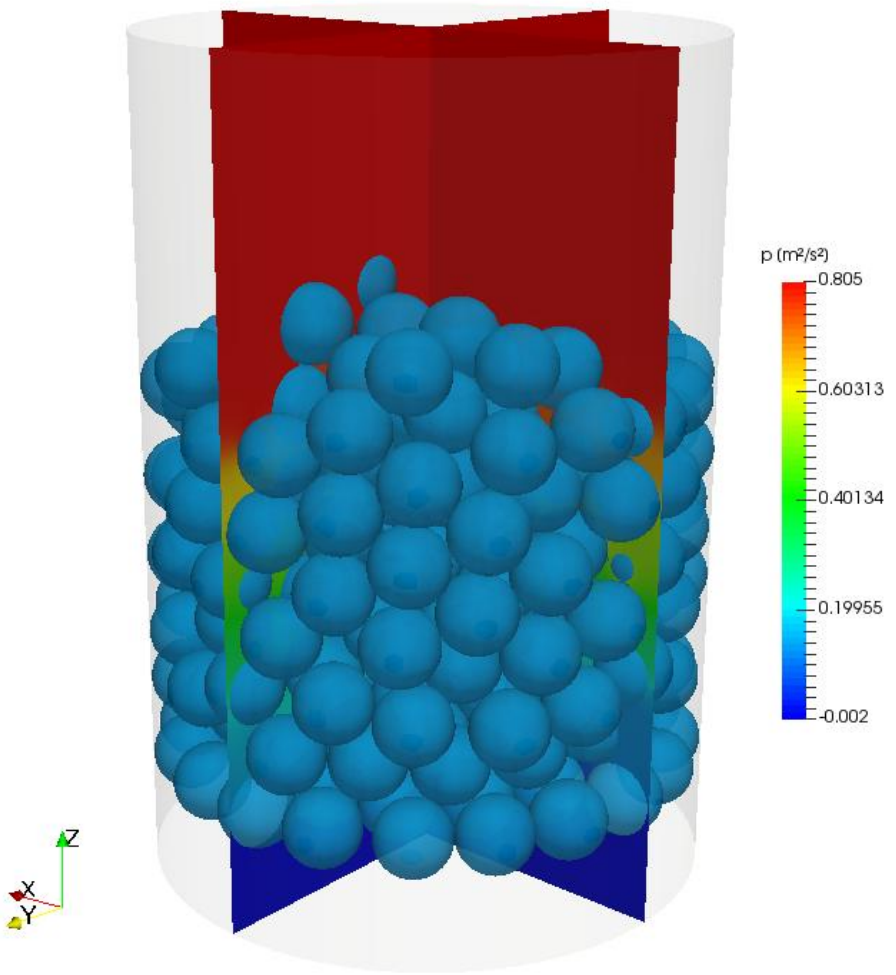
Results for $D/d_P = 7.35$



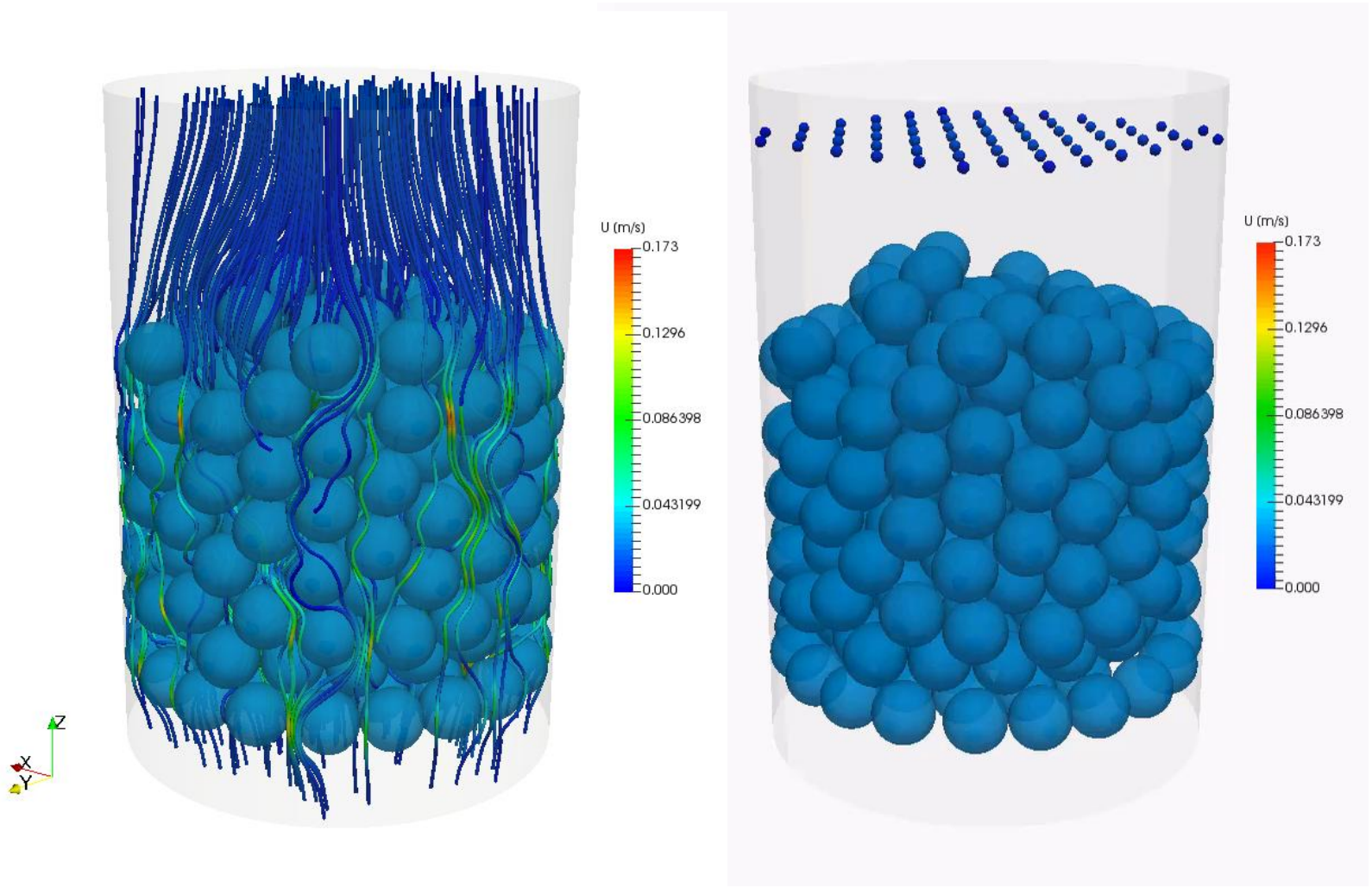
- literature values from Goodling et al.
  [Goodling et al., Powder Technology 35 (1983) 23-29]

Results for $D/d_P = 7.35$

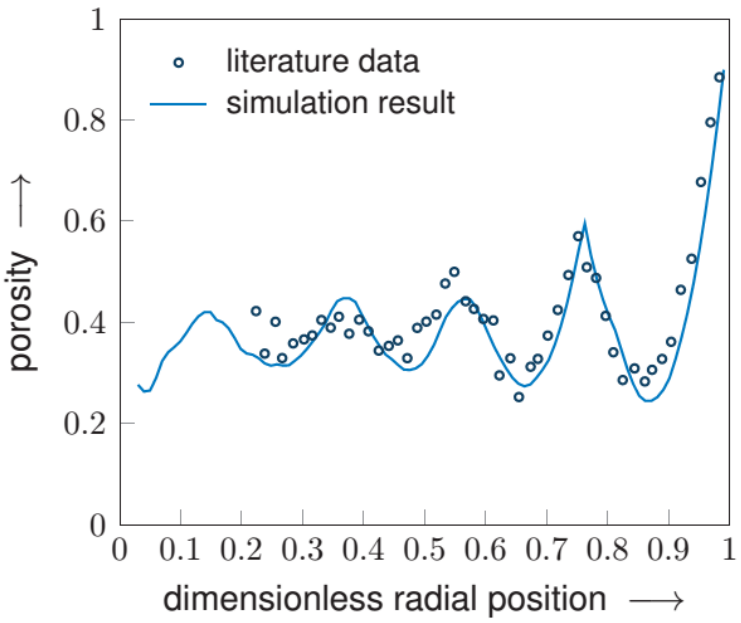


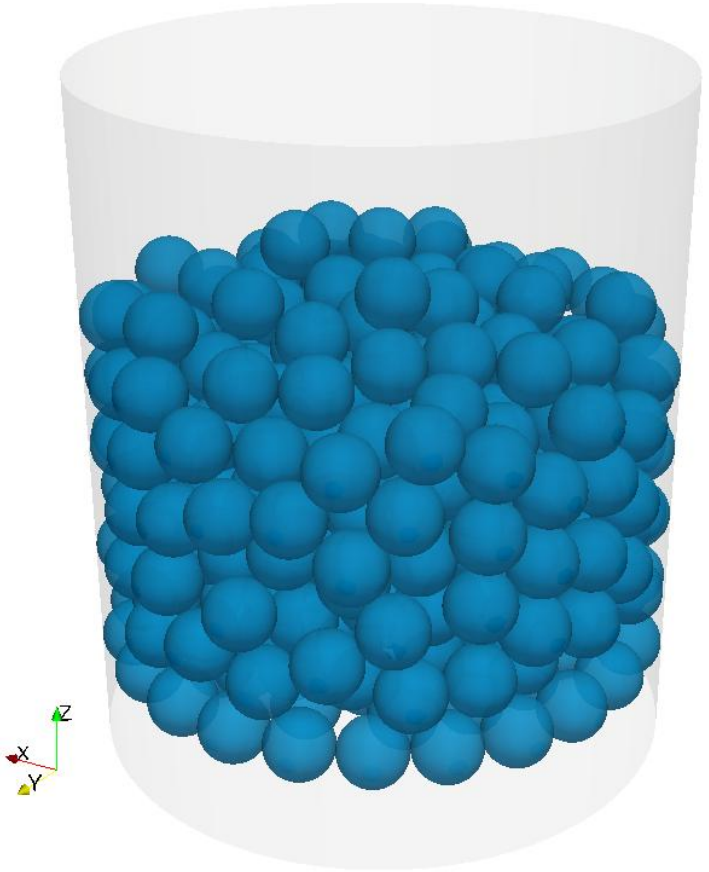


Results for $D/d_P = 7.35$



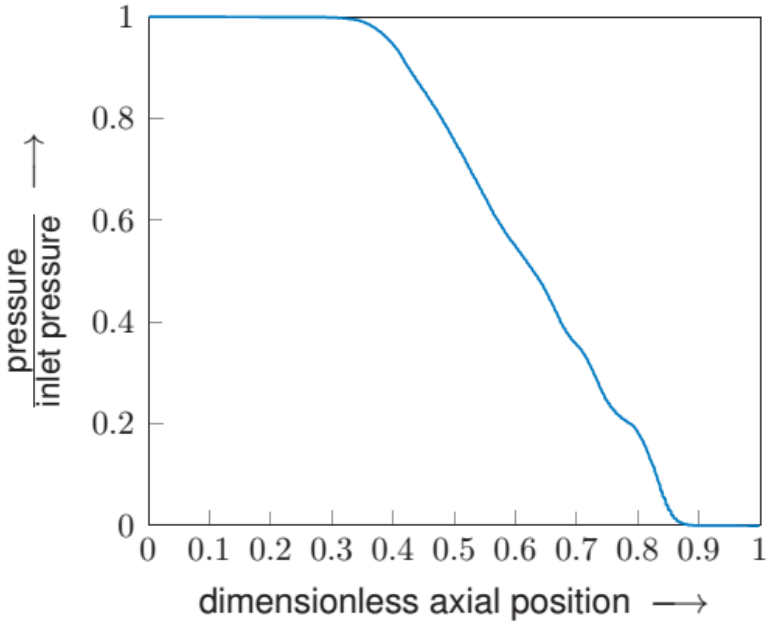
Results for $D/d_P = 8.41$

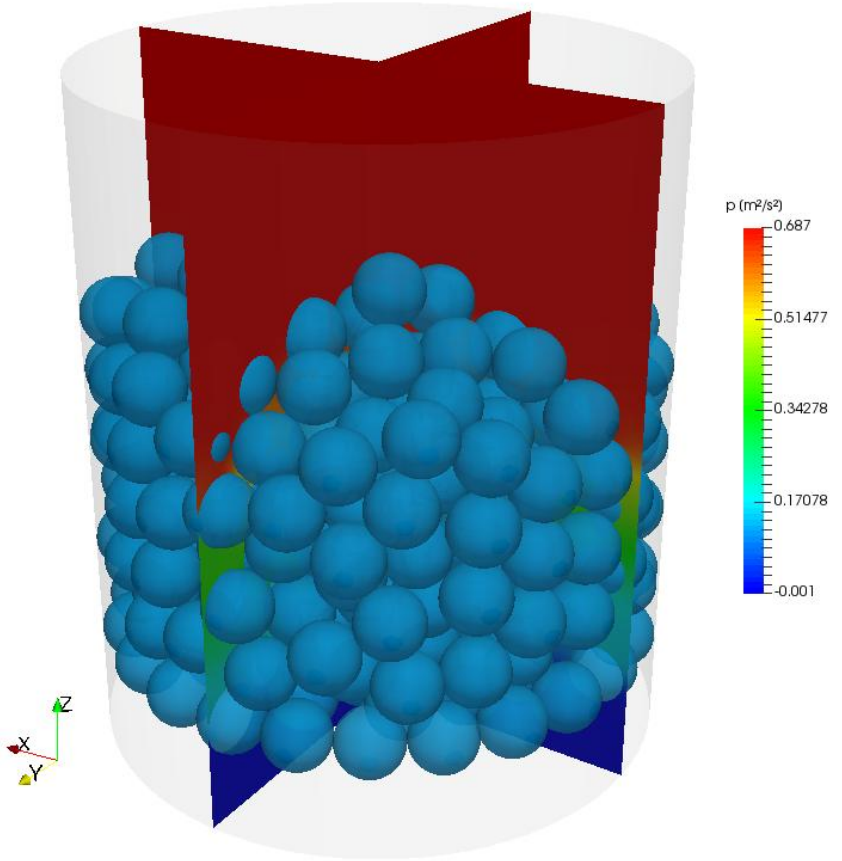




- literature values from Goodling et al.
  [Goodling et al., Powder Technology 35 (1983) 23-29]

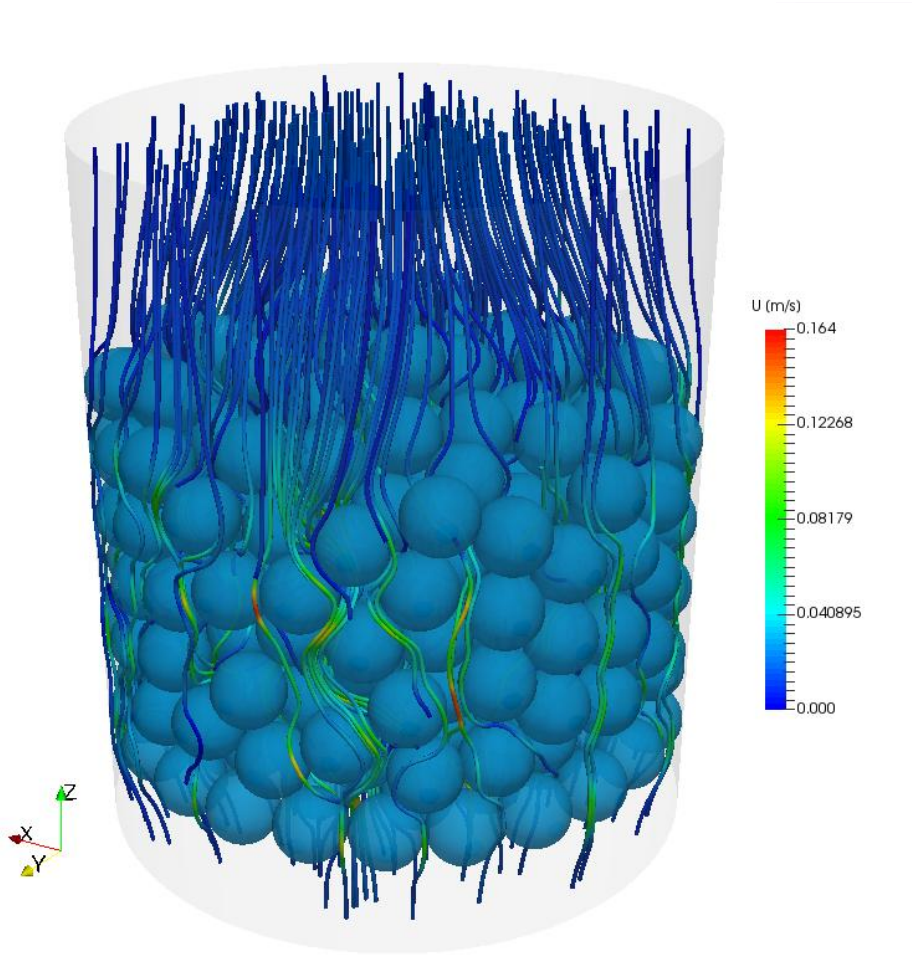
Results for $D/d_P = 8.41$

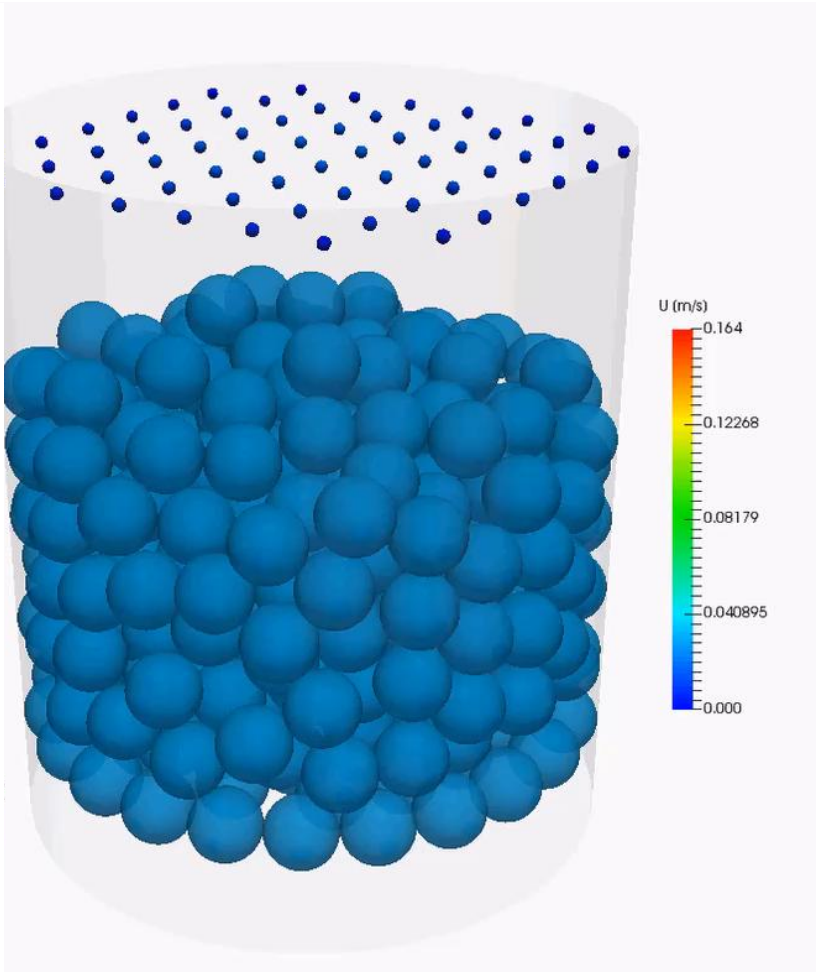




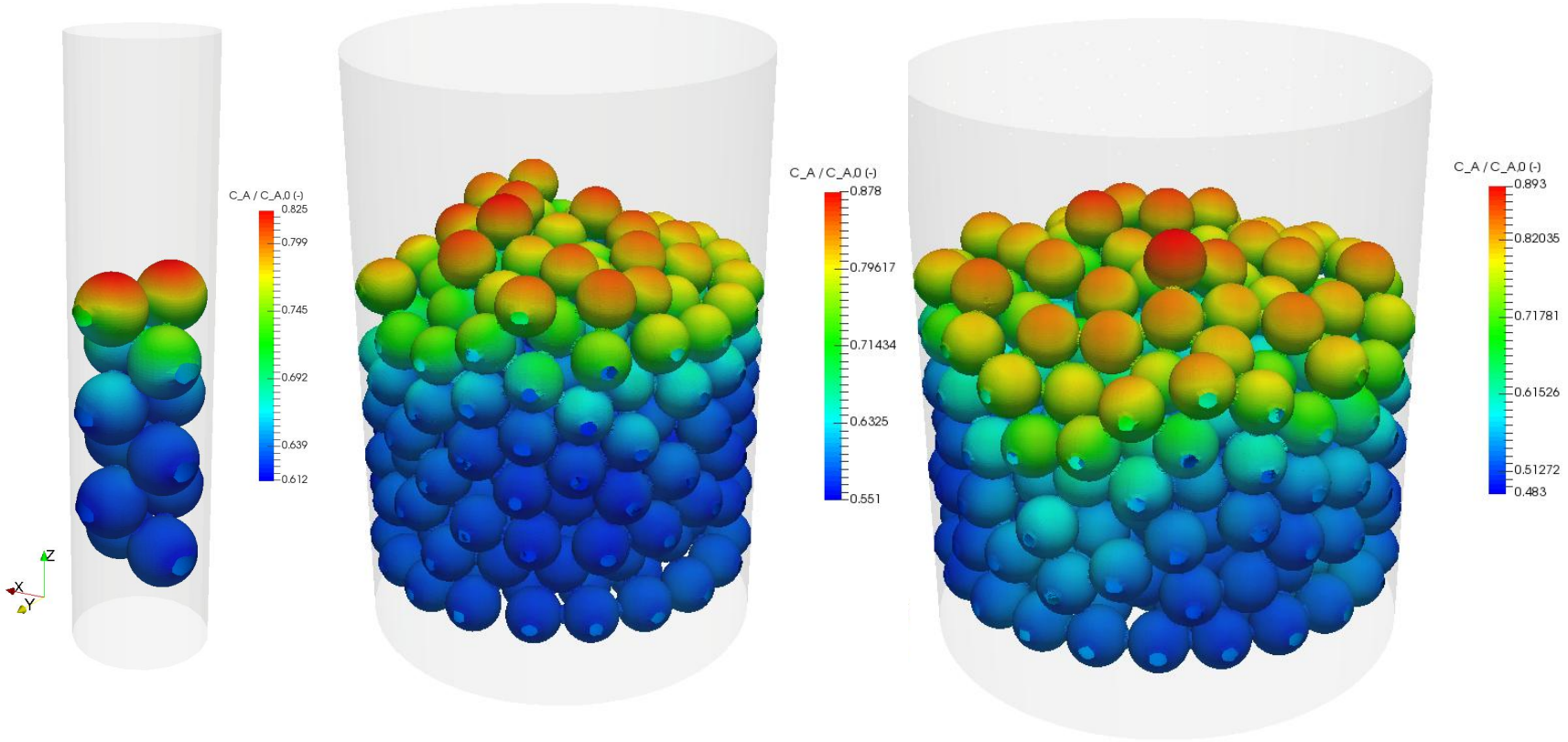
- literature values from Goodling et al.
  [Goodling et al., Powder Technology 35 (1983) 23-29]

Results for $D/d_P = 8.41$





Results for surface reaction



Results for surface reaction

Comparison of conversion $X = \frac{C_{A,0} - C_A}{C_{A,0}}$ to ideal reactors:

D/d	2	7.35	8.41
Simulation	0.354	0.398	0.442
PFR	0.869	0.903	0.877
CSTR	0.670	0.700	0.677

$$X = 1 - e^{-k\tau}$$

$$X = \frac{k\tau}{1+k\tau}$$

Current & Future Work

Current & Future Work

- Implementation of energy balance
- Coupling with DAKOTA® for parameter optimisation

Thank you for your attention!

florian.habla@ch.tum.de