Perform moving mesh simulations based on 3D RTE as explained in Gruenwald et al. 2021

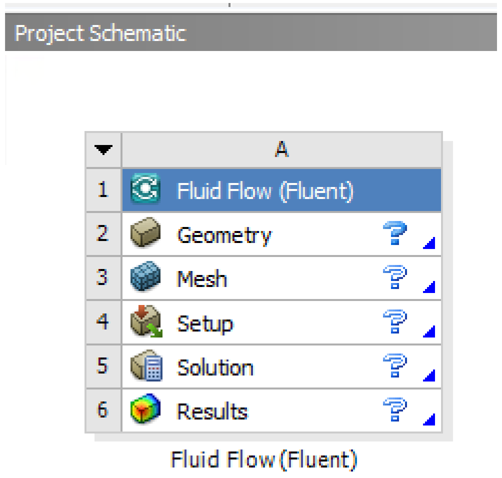
CVE AME | RWTH AACHEN UNIVERSITY

Prerequisites

- Visual Studio (https://visualstudio.microsoft.com/en/downloads/) full version with all packages
- Python and any IDE like Spyder or Sublime Text 3
  - Install: pip, numpy wheel, matplotlib, scipy
- UCD files (export of Tomtec ImageArena) for ventricular cavities at all time steps
- Ansys Workbench, SpaceClaim, Meshing, FLUENT, CFD-Post (Version 2021 R2)

Step 0: Workbench

- Unzip example project files from repository
- Load project file into Workbench

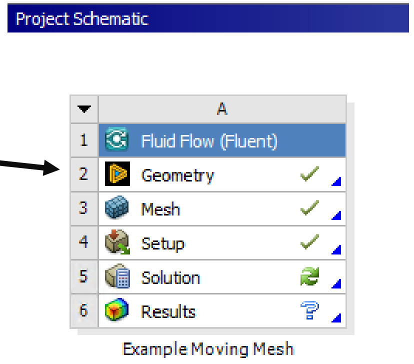


Step 1: Prepare STL files

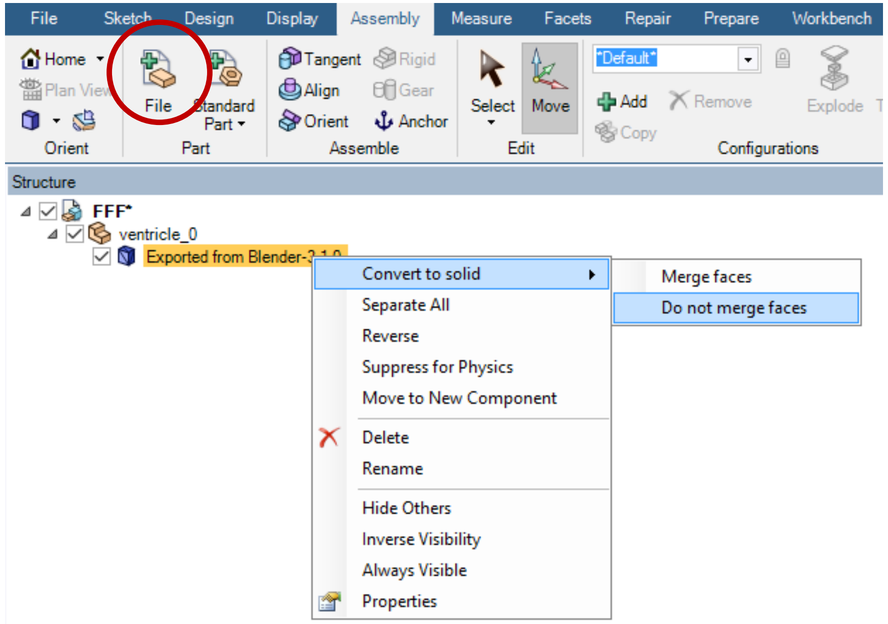
- Execute ucd2stl MATLAB code to convert UCD to STL files
- Copy and rename existing STL files (of all time steps) into the folder STL
  - "ventricle_n" with n starting from 0 (End-systolic volume)
  - Folder STL is in the same location as code ps_detNPts.py (Fluent folder in project files)

Step 2: SpaceClaim

- Open SpaceClaim through Workbench



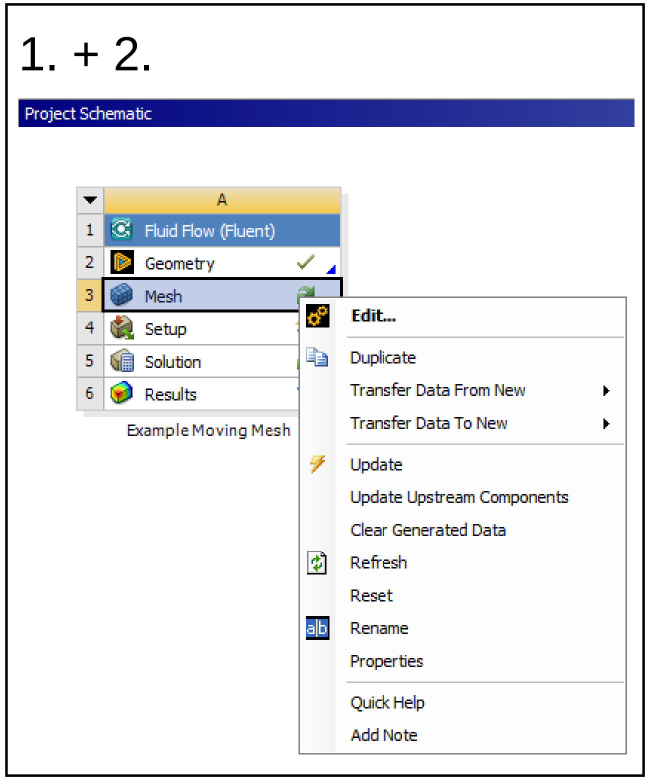
- Import files:
  - Assembly → File → Choose only initial timeframe (‘ventricle_0.stl‘)
  - Options → File Options → STL → Set Import units (not automatic)
- Convert to solid → Do not merge faces
- Close SpaceClaim

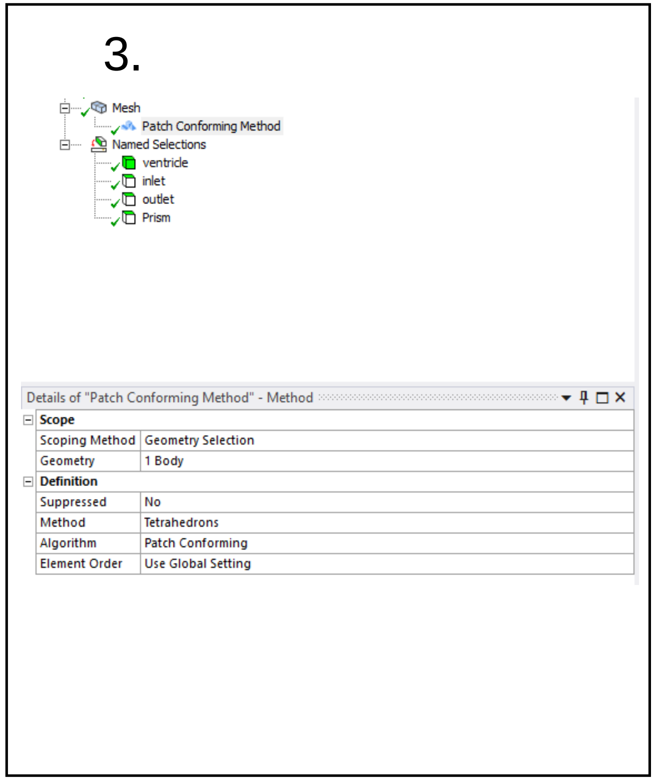


Step 3: Meshing I

1. Update upstream Components
2. Edit Mesh
3. Select Body for meshing method

    Customize settings of meshing method
4. Assign named selections
    1. Ventricle (Whole ventricle body)
    2. Inlet/MV (Faces of MV)
    3. Outlet/AV (Faces of AV)
    4. Prism layers (All faces except MV/AV)
5. Generate Mesh (Check mesh afterwards)
6. Close meshing
7. Update upstream components of Setup

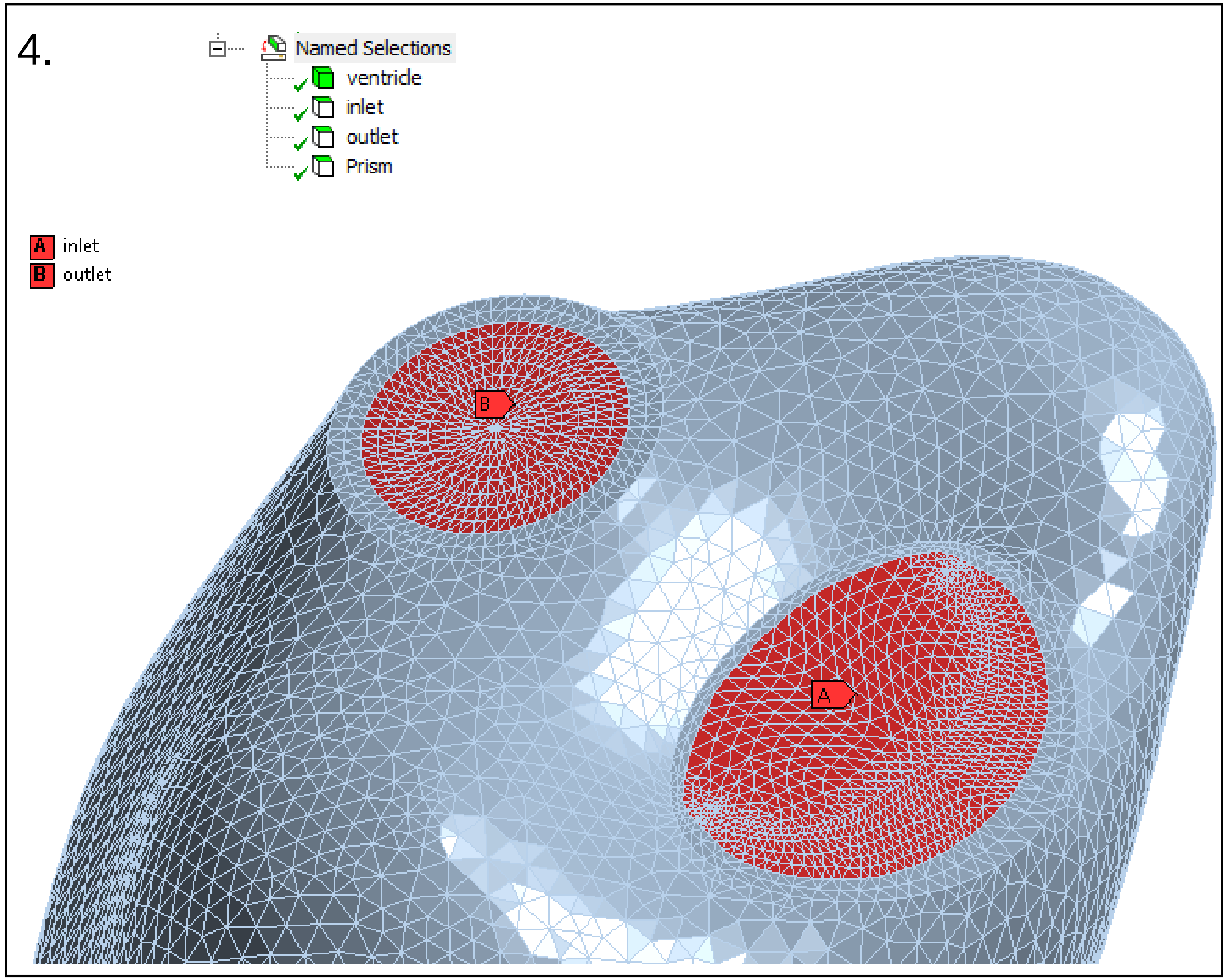




Step 3: Meshing II

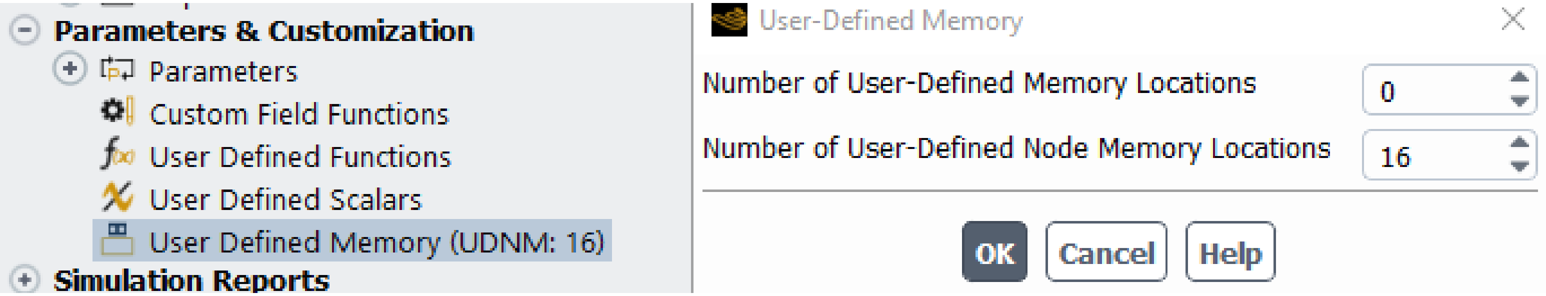
1. Update upstream Components
2. Edit Mesh
3. Select Body for meshing method

   Customize settings of meshing method
4. Assign named selections
   1. Ventricle (Whole ventricle body)
   2. Inlet/MV (Faces of MV)
   3. Outlet/AV (Faces of AV)
   4. Prism layers (All faces except MV/AV)
5. Generate Mesh (Check mesh afterwards)
6. Close meshing
7. Update upstream components of Setup



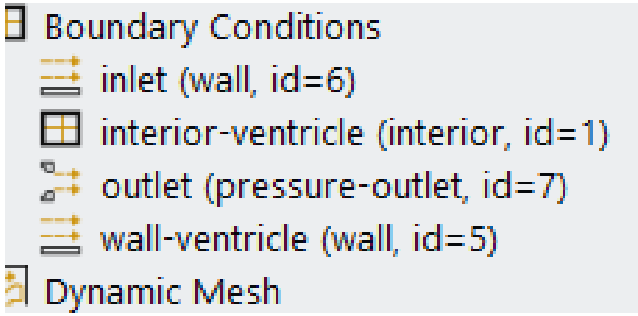
Step 4: FLUENT Setup solver nodes

- Start FLUENT with double precision and desired Solver processes (#Nodes)
- Change Solver processes (Parameters & Customization → User-defined memory)



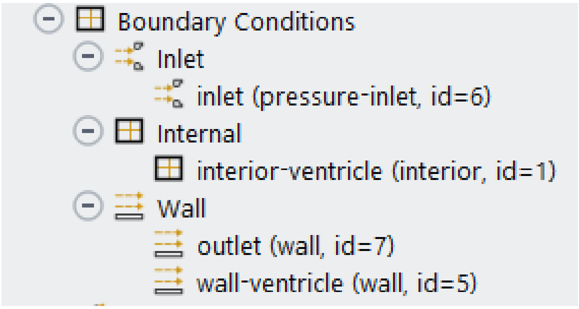
- Check and adjust the zone IDs of the outer walls (here: Inlet, outlet and wall-ventricle) in the UDF (udf_exPts.c)

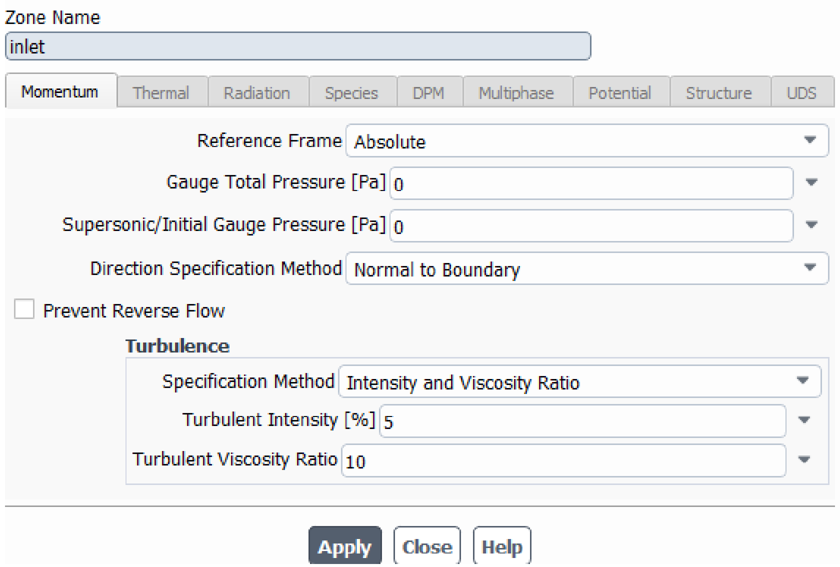
```
/* define zoneIDs of ventricle, inlet and outlet */
#define nFZones 3
#define ventricle 5
#define inlet 6
#define outlet 7
```

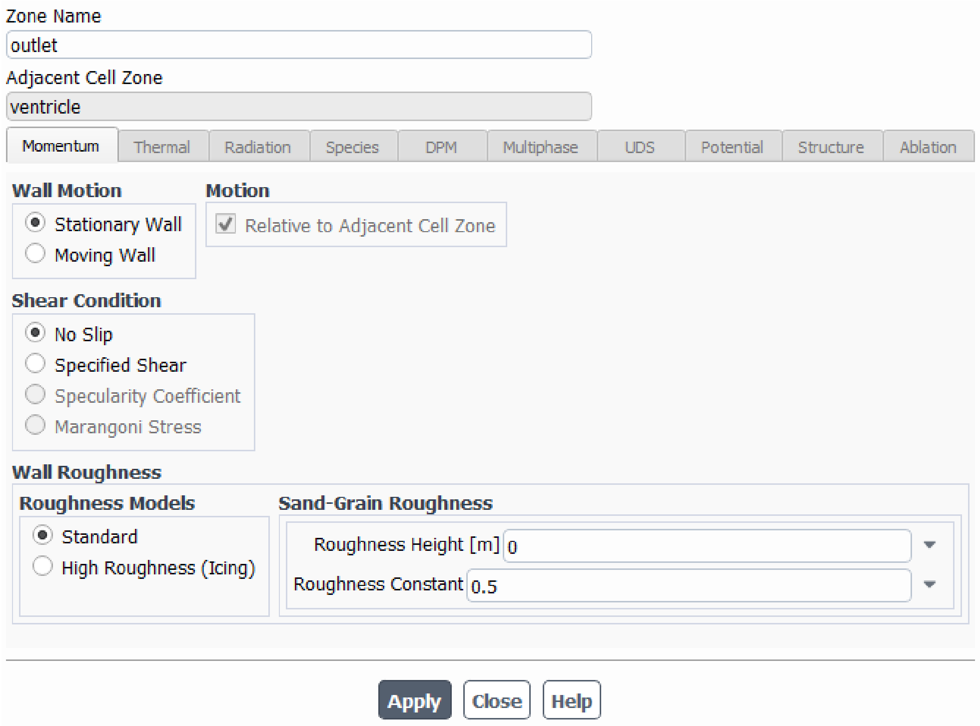


Step 5: FLUENT Setup up initial Boundaries

- Check Boundaries in Fluent
  - Initial state (End-systole)
    - Inlet abs. pressure (0 pa)
    - Outlet wall (closed)

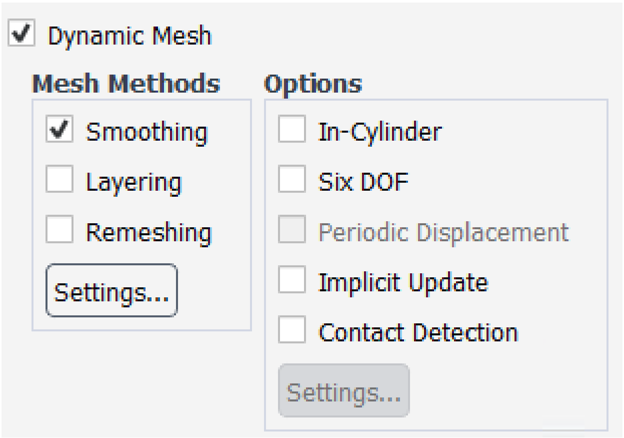


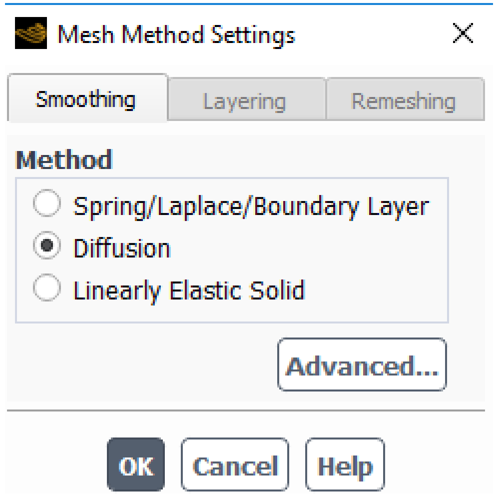


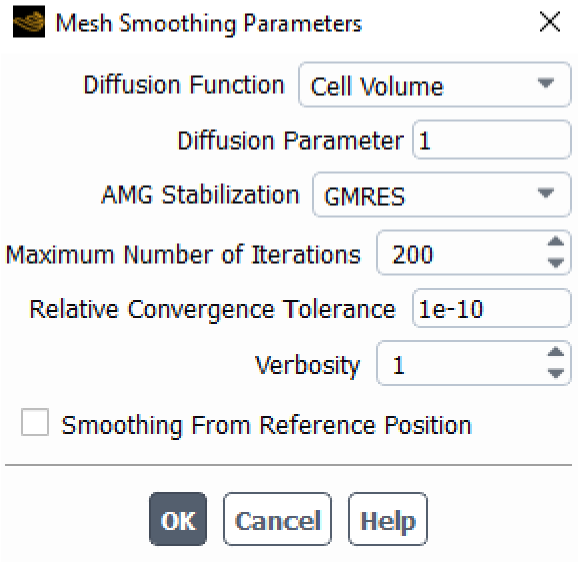


Step 6: FLUENT Setup up dynamic mesh settings

- Setup → Dynamic Mesh → Settings







Step 7: FLUENT Setup up dynamic Boundaries

- Setup dynamic mesh event settings
  - Setup → Dynamic Mesh → Events
  - 3 events to switch between diastole and systole (here diastole to systole)
    - Inlet/Outlet to wall
    - Outlet/Inlet to pressure
    - Setup pressure profile

Number of Events 15

On	Name	At Time [s]	
✓	inlet-to-wall	0.31892	Define...
✓	wall-to-pressure	0.31892	Define...
✓	pressure-outl	0.31892	Define...
✓	outlet-to-wall	0.557	Define...
✓	wall-to-pressure	0.557	Define...
✓	pressure-intl	0.557	Define...

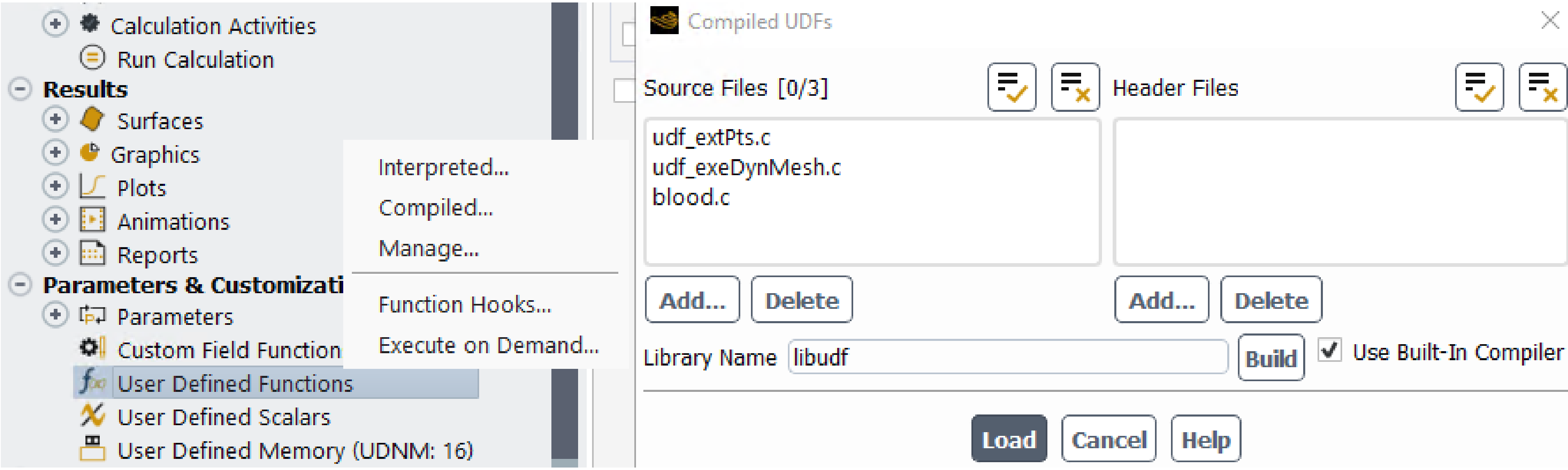
In this example: Time diastole: 0,31892s   RR-duration: 0,557s

Command: define boundary-conditions pressure-outlet outlet y y n outpres_ao_3 pres_ao n y n n y 5 10 y n n n

- At end of cycle boundaries are not resetted to initial state
  - Inlet to pressure-inlet, Outlet to wall
  - Event – time adjusted to MV closing of next cycle

Step 8: FLUENT Compile UDF library

- Parameters & Customization → User Defined Functions → Compiled
- Add... (blood.c; udf_exeDynMesh.c; edf_exPts.c)
- Use Built-In Compiler
- Build
- Load



Step 9: FLUENT Execute udf_extPts.c

- Parameters & Customization → User Defined Functions → Execute on Demand...
  - First_AssignID::libudf
  - File „surface“ is created in the simulation folder
  - Number of nodes for each core are shown in the console (Will be necessary for udf_exeDynMesh)

surface file

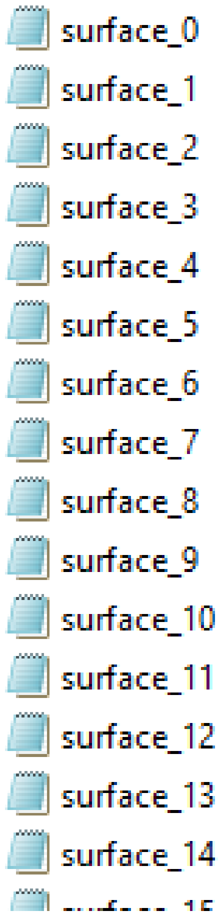
```
0 6.485890 16.681522 28.130762
1 6.862917 16.265652 28.036011
2 6.569711 15.943861 27.625748
3 7.079221 15.724083 27.819656
4 7.156017 16.587324 28.446129
5 7.513730 15.881123 28.214535
6 7.481522 16.395054 28.535452
7 7.545926 15.367387 27.893742
8 7.887307 16.074398 28.607815
```

Step 10: FLUENT Execute ps_detNPts.py ➔ STL to PTS I

- With python e.g. in Sublime Text 3 or Spyder
- If first execution: Answer Do you need to generate correlation file? Yes: 1. No: 0. with “1”
- The reference frame ID is the number of the STL-file in the folder STL, that has been loaded and meshed with ANSYS (see step 1).
- Output:
  - Number of Faces
  - allPtsNum

Step 11: FLUENT Execute ps_detNPts.py → STL to PTS II

- First timepoint of cycle as Start FrameID (commonly 0)
- Last used timepoint as End FrameID
- Code creates "surface_" files in a folder PTS



Step 12: FLUENT Execute ps_intNPts.py ➔ PTS to UDFPTS

- Execute the ps_intNPts code in (same location as PTS folder / fluent folder)
- Set number of stl files per cycle as „Total number of frames in a cycle"
- Set number of interpolation steps between each stl „Number of intermediate frames"

  → This will result in the number of timesteps each cycle

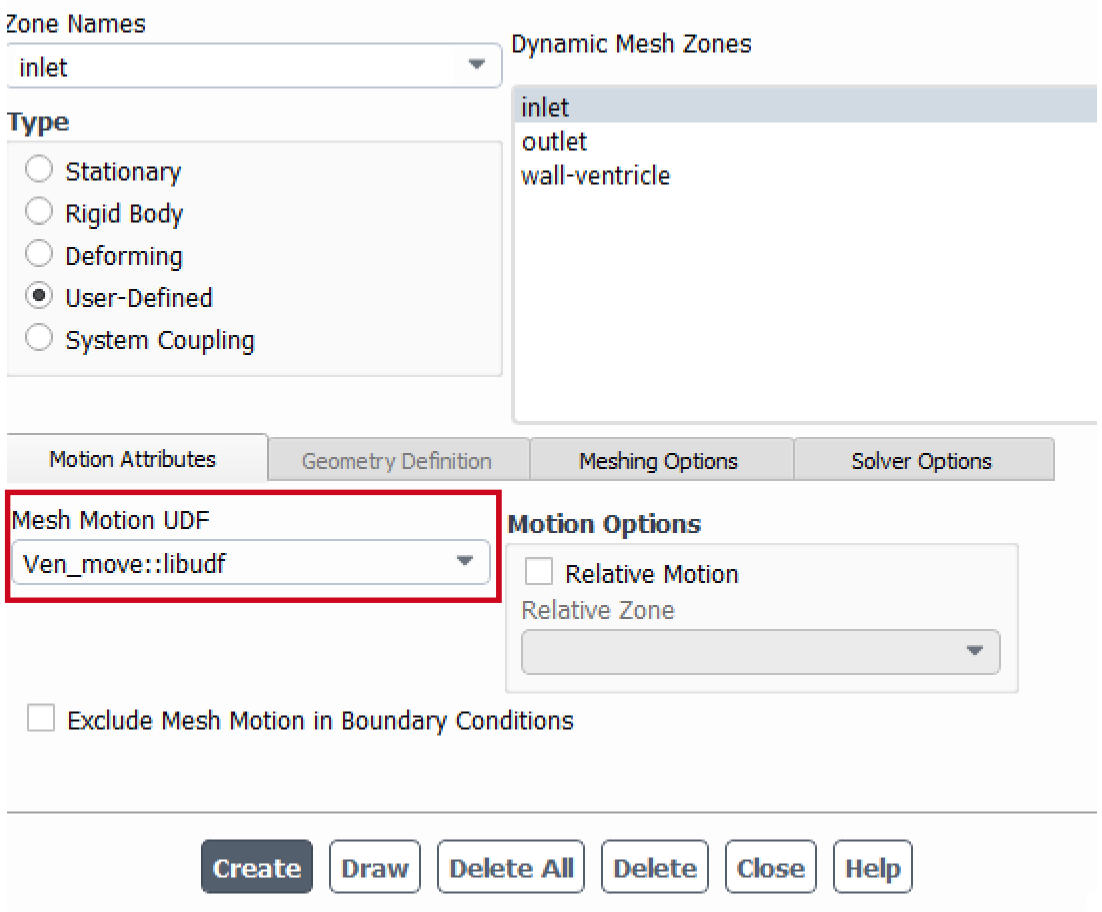
  (Number of intermediate frames x number of stl files)
- Choose a temporal interpolation method between the frames
- Output of Total Number of Frames after interpolation: # of timesteps for one cycle
- Code creates udfsurface.asc files in folder UDFPTS → necessary for mesh movement

Step 13: FLUENT Adjust and execute udf_exeDynMesh.c

- Unload libudf
- Delete libudf-folder
- Adjust udf exeDynMesh
  - nTimeSteps: Number of timesteps= maximum index of udf_surface (output of ps_intNPts.py)
  - allPtsNum: Number of all nodes, output of ps_detNPts.py
  - PtsNum[]: number of points per core (saved as array); Console output after execution of UDF First_Assign_ID
- Compile UDF (see slide 12)
- Parameters & Customization → User Defined Functions → Execute on Demand...
  - loadMesh::libudf

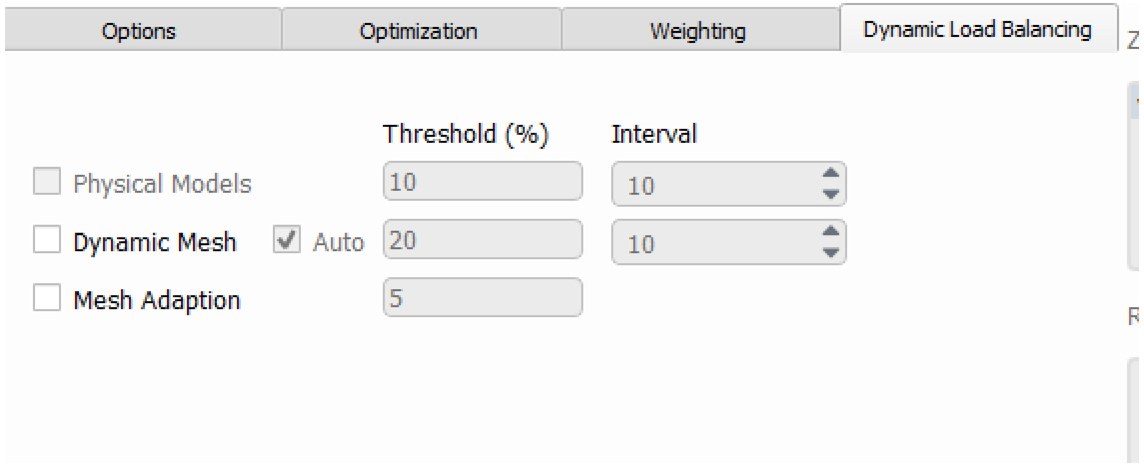
Step 14: Setup use of udf_exeDynMesh.c

- Dynamic Mesh → Dynamic Mesh zones → create/edit → user defined function for inlet, outlet, wall-ventricle erstellen



Step 15: Prohibit repartitioning

- Parallel → Partition/Load Balance → Check off Dynamic Mesh



Step 16: FLUENT Initialize

- Initialization → Initialize with Hybrid initialization (standard initialization also possible)
- If not the first timestep was put out in console after loadMesh::libudf

```
 UDFPTS/udfsurface_1.asc

 0
Finish Loading
```

  - Repeat execution of loadMesh::libudf (see slide 18)
  - Initialization → Initialize

Step 17: Run Simulation

- Optinally before simulation start → preview mesh motion
  - Preview Mesh Motion
- If mesh movement doesn‘t work (e.g. negative cell volumes):
  - Check if mesh node-connectivity is consistent
  - Recompute mesh nodes (steps 2-12)
  - (Settings: Dynamic Mesh → Mesh Methods → Remeshing with default settings)
- Run simulation (Solution → Run Calculation → Calculate)

Contact

Michael Neidlin

neidlin@ame.rwth-aachen.de or

michael.neidlin@gmail.com