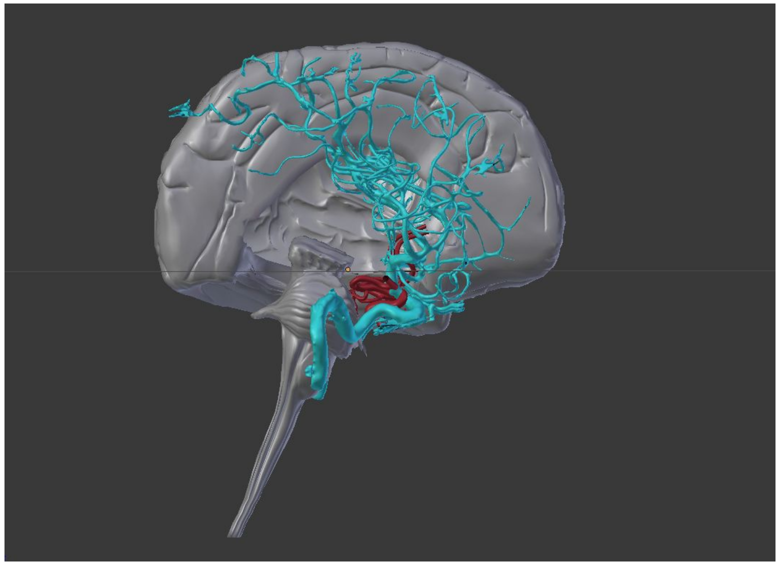
Application: Simple magnetic drug targeting simulations with HemeLB



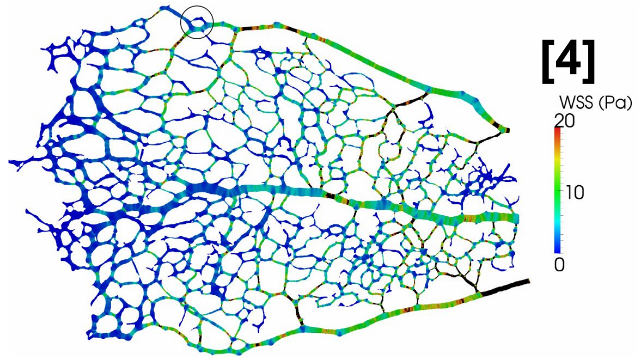
Robin Richardson
UCL

This project has received funding from the European Union's Horizon 2020 research and innovation programme.

What is HemeLB?

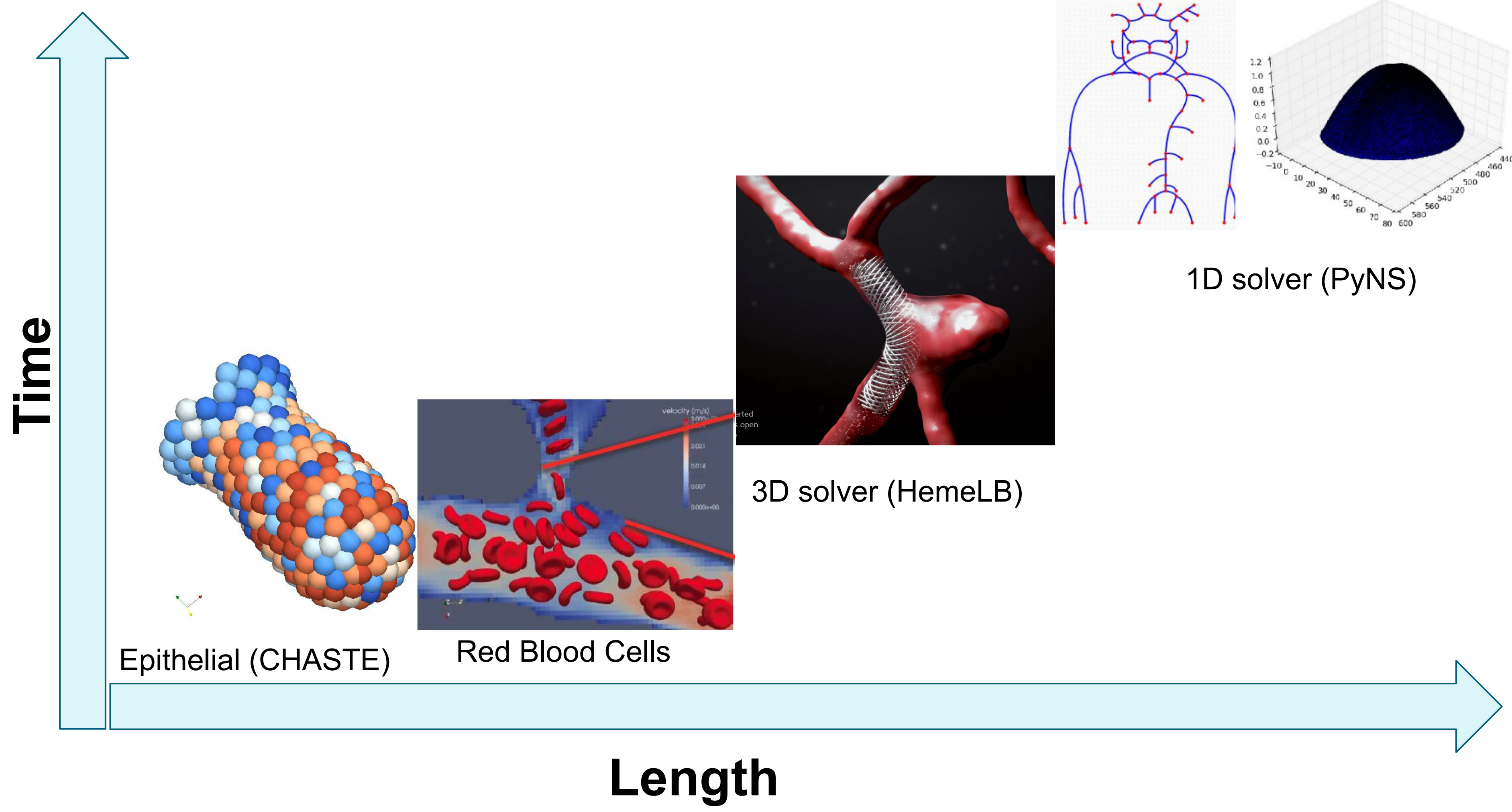
Blood flow through the Circle of Willis
connecting arteries to the brain
SCIENCE MUSEUM

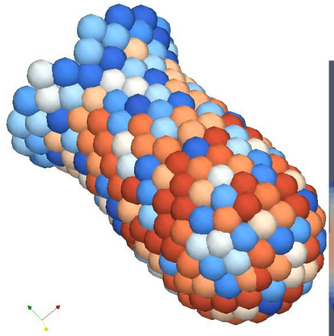
- Flow solver based on the lattice-Boltzmann method.
- Optimised for sparse, patient-specific geometries[1].
- Supports a range of collision kernels and boundary conditions[2].
- Easy compilation, execution, analysis using FabHemeLB[3].

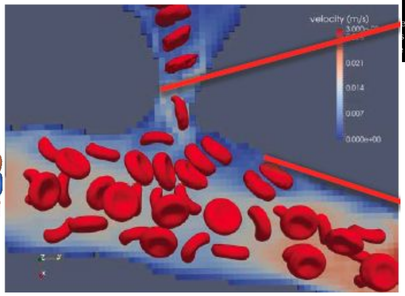


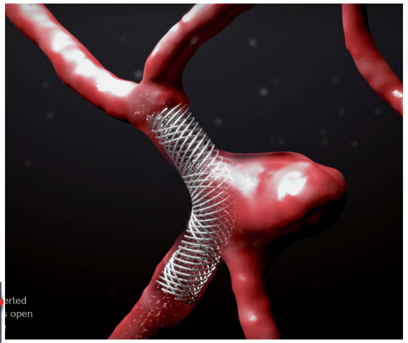
[1] Groen et al., JoCS 4(5), 2013.
[2] Nash et al., Phys Rev E 89, 023033, 2014.
[3] Groen et al., arXiv:1512.02194
[4] Bernabeu et al., J. R. Soc. Interface, 11(99), 2014.

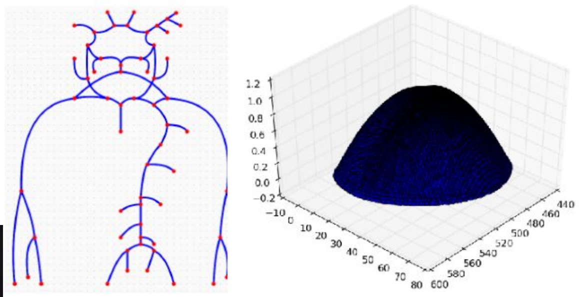
Many scales




Epithelial (CHASTE)

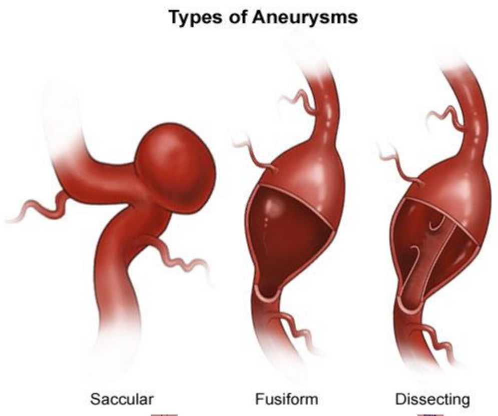

Red Blood Cells

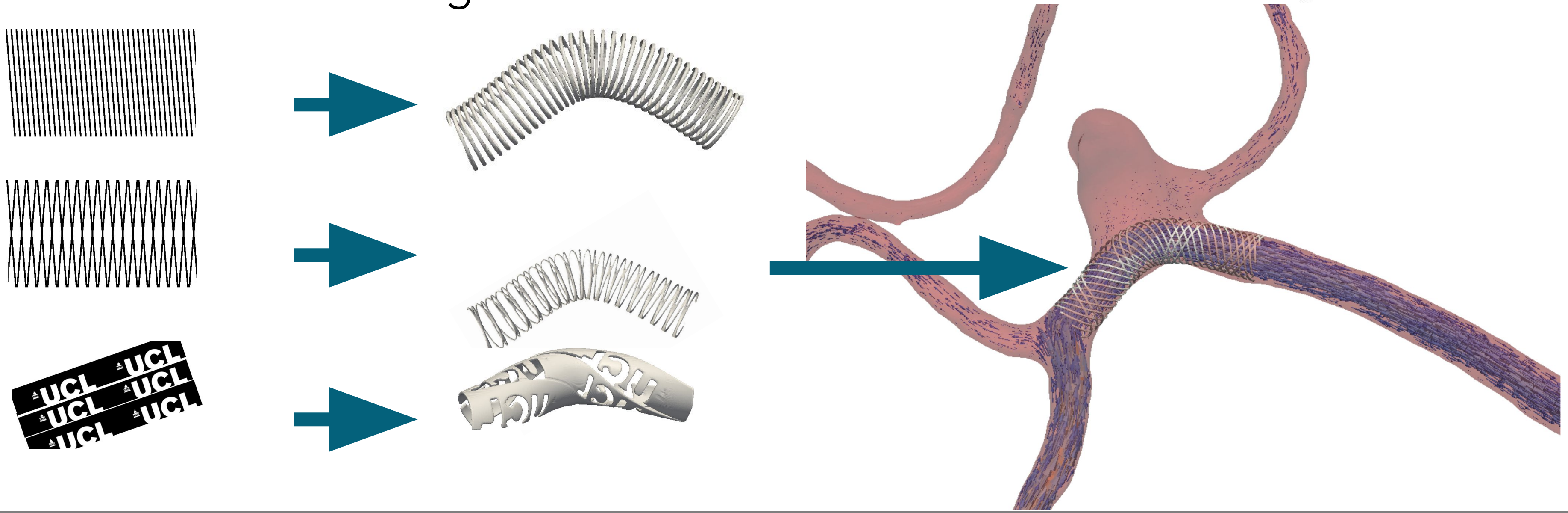

3D solver (HemeLB)


1D solver (PyNS)

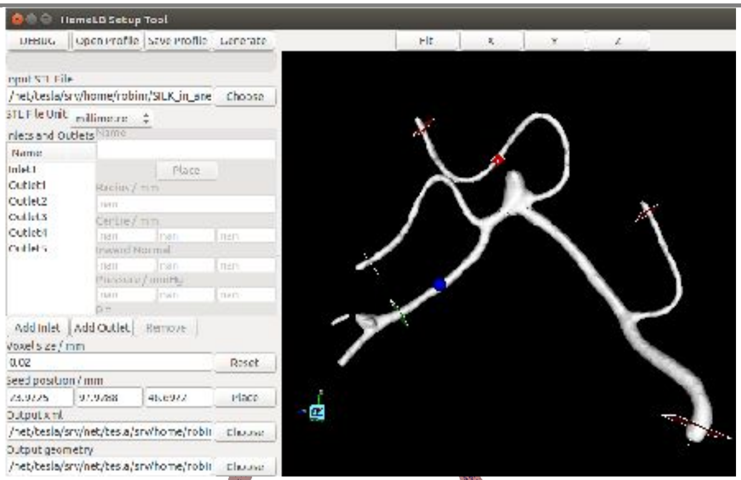
Aneurysm treatment with stents

- Prediction of stresses arising from treatment of aneurysms with flow diverting stents
  - Patient specific models and inflow conditions
  - Exploring the effects of different stent designs





Why simulation?


HemeLB Setup Tool

DEBUG | Open Profile | Save Profile | Generate

Input STL File: /net/tesla/srv/home/robin/SILK_in_ane — Choose

STL File Unit: millimetre

Inlets and Outlets

Name
Inlet1
Outlet1
Outlet2
Outlet3
Outlet4
Outlet5

Name — Place

Radius / mm: nan

Centre / mm: nan nan nan

Inward Normal: nan nan nan

Pressure / mmHg: nan nan nan

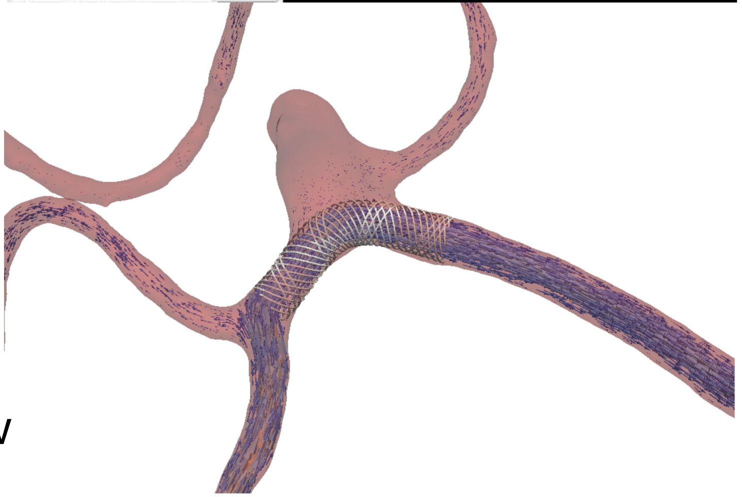
Add Inlet | Add Outlet | Remove

Voxel size / mm: 0.02 — Reset

Seed position / mm: 24.9225 97.9288 46.e977 — Place

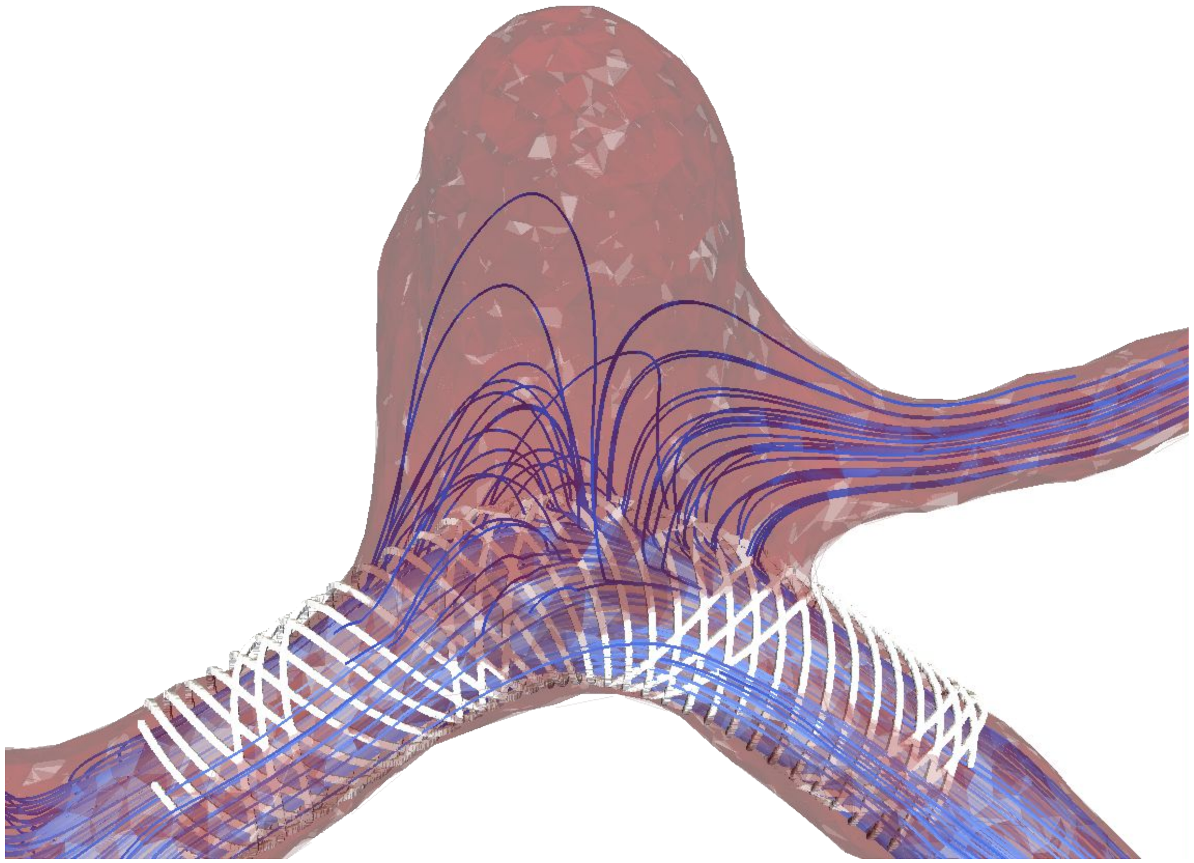
Output xml: /net/tesla/srv/net/tesla/srv/home/robi — Choose

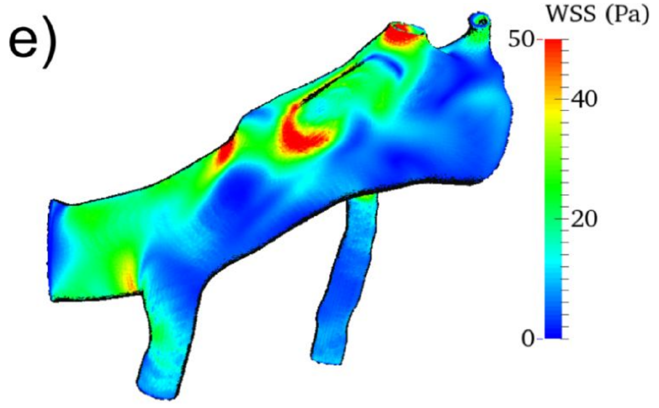
Output geometry: /net/tesla/srv/net/tesla/srv/home/robi — Choose

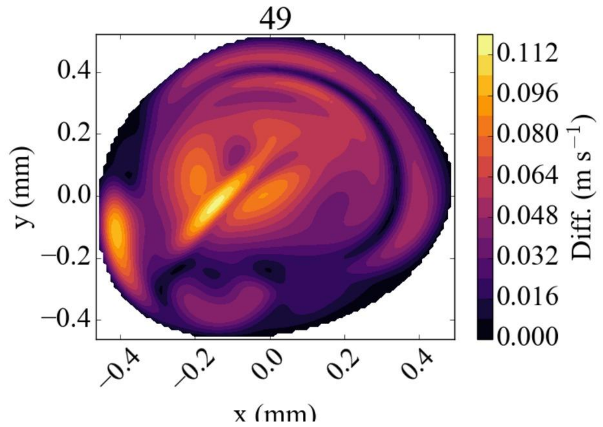


- Give clinicians extra info using data which is already collected
  - Non-invasive
  - How will introduction of stent affect flow and stresses in the system?
  - Wall shear stress, oscillatory shear stress, etc.

Flow, stresses, velocity comparisons, etc.

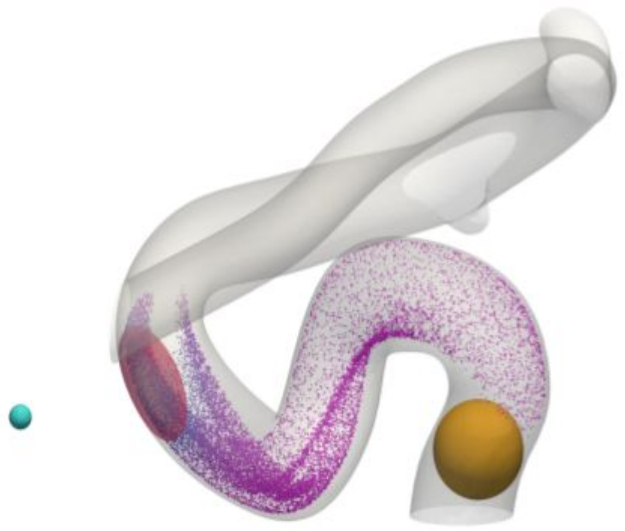




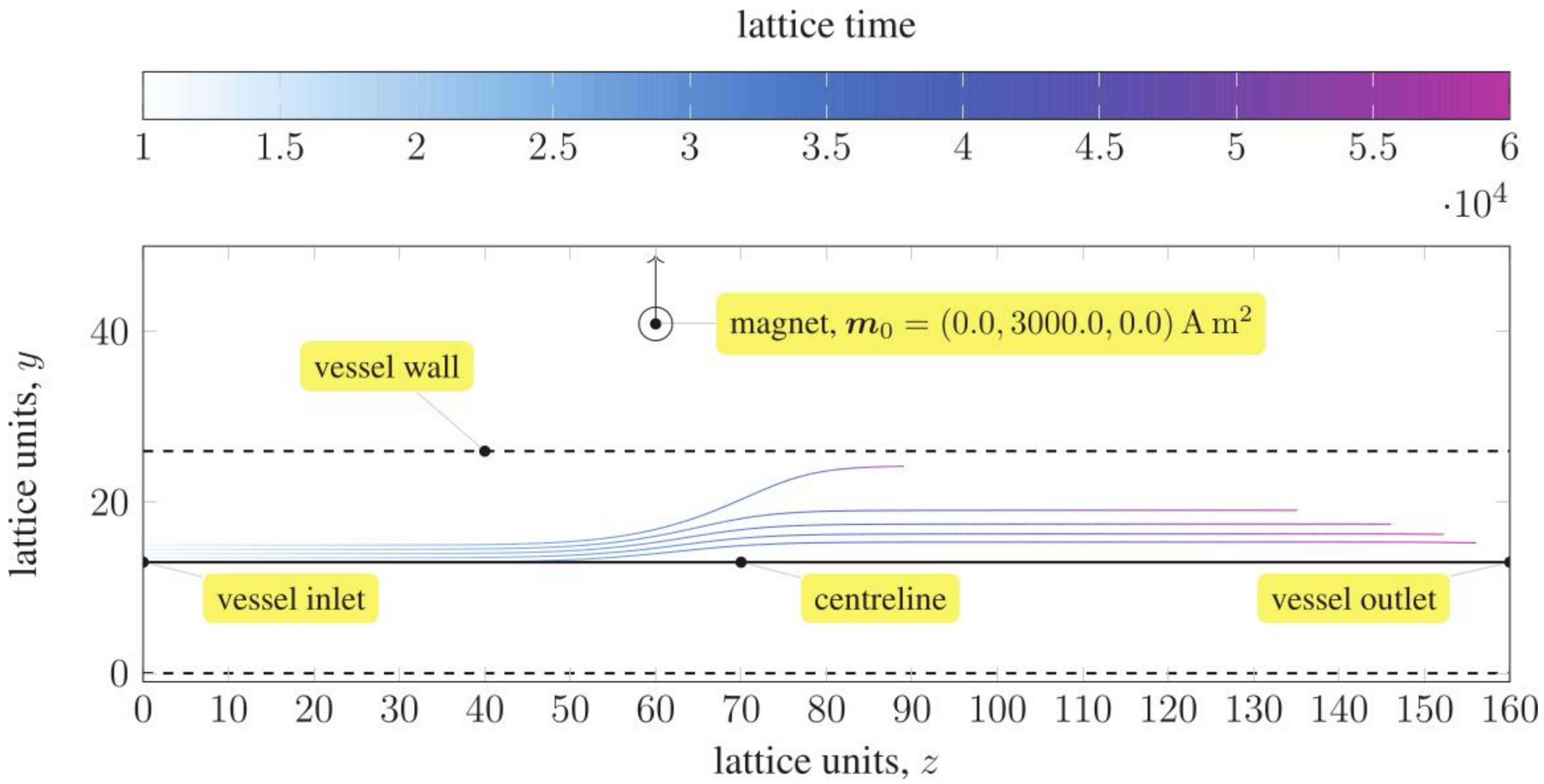


Magnetic Drug Targeting

- Super Paramagnetic Iron Oxide particles
- SPIONs can be coated with polymers to produce various colloidal interactions, or loaded with drugs
- Magnetic field can guide SPIONs to target site e.g. tumour
- Rapidly varying magnetic field causes induction heating, applying heat to the tumour or causing release of drugs
- We study the distribution and interactions of these colloids using CFD



Magnetic Drug Targeting



Step 1: Compile HemeLB

Get and build HemeLB

- *Get it from the shared folder on Marenostrum:*

```
cp -r /gpfs/projects/nct00/nct00004/hemelb-pure_public/  ~/
```

- *Unload/load necessary modules:*

```
module unload intel/2017.4
module unload impi/2017.4
module unload mkl/2017.4
module load gcc/7.1.0
module load openmpi/1.10.7
```

- *Compilation:*

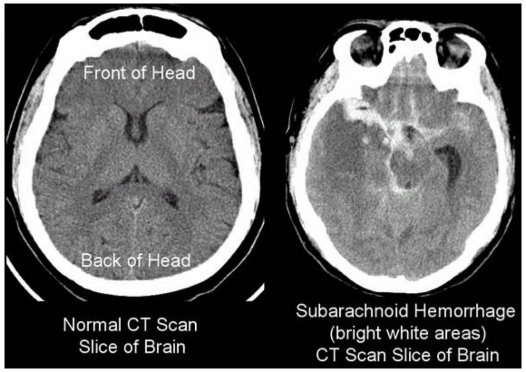
```
mkdir ~/hemelb-pure_public/src/build/
cd ~/hemelb-pure_public/src/build/
cmake ..
make
```

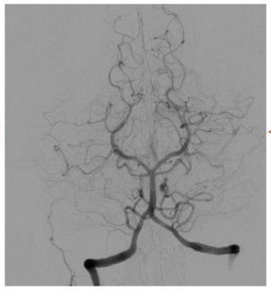
- Should take a couple of minutes to compile

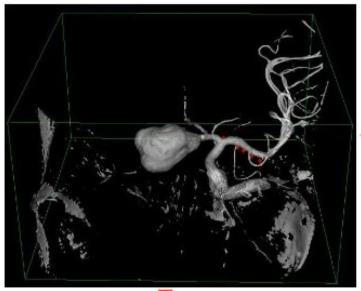
(The above instructions are available in hemelb-pure_public/README.md)

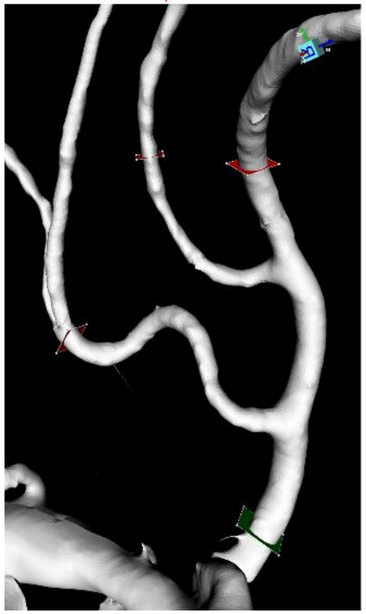
Step 2: Create input files

Segmentation of Medical Images



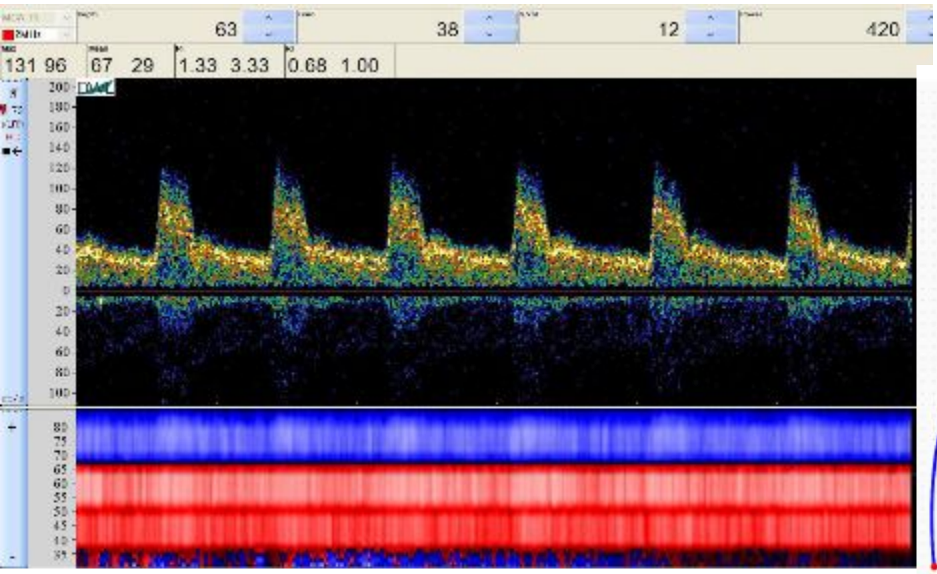


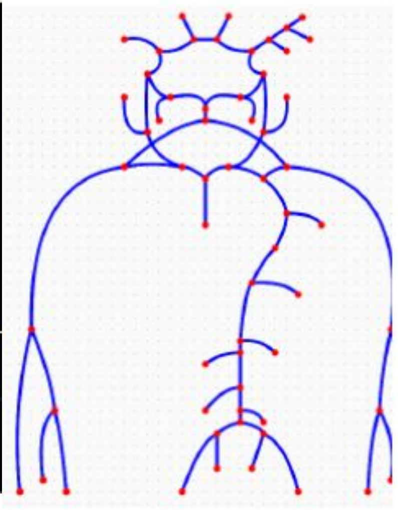


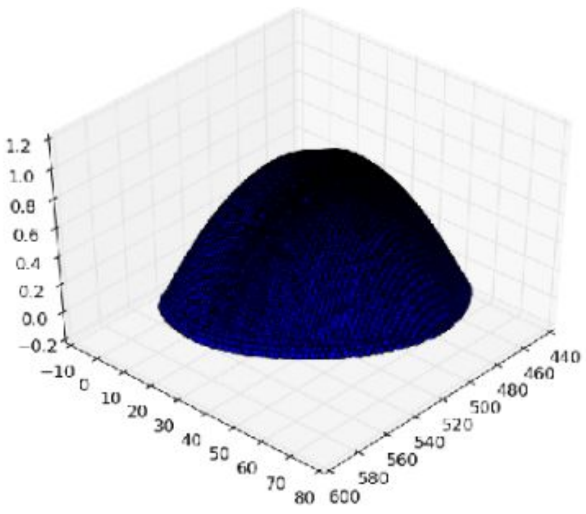


- Rotational Angiography or x-ray CT is used to take x-ray images at various angles around the patient's head
  - Can reconstruct the 3D volume revealed by the contrast agent.
  - Typically, image quality is better from the CT scan due to the faster acquisition time.
- Segmentation and post-processing (filtering and smoothing) can be carried out with software such as VMTKLab (http://vmtklab.orobix.com).

Predicting Velocity Data







- We need to specify the time-velocity profile at the inlets to the area of interest.
- Non-invasive measurement techniques include Transcranial Doppler (TCD)
- Due to the skull, technique can only be used at very limited number of locations around head.

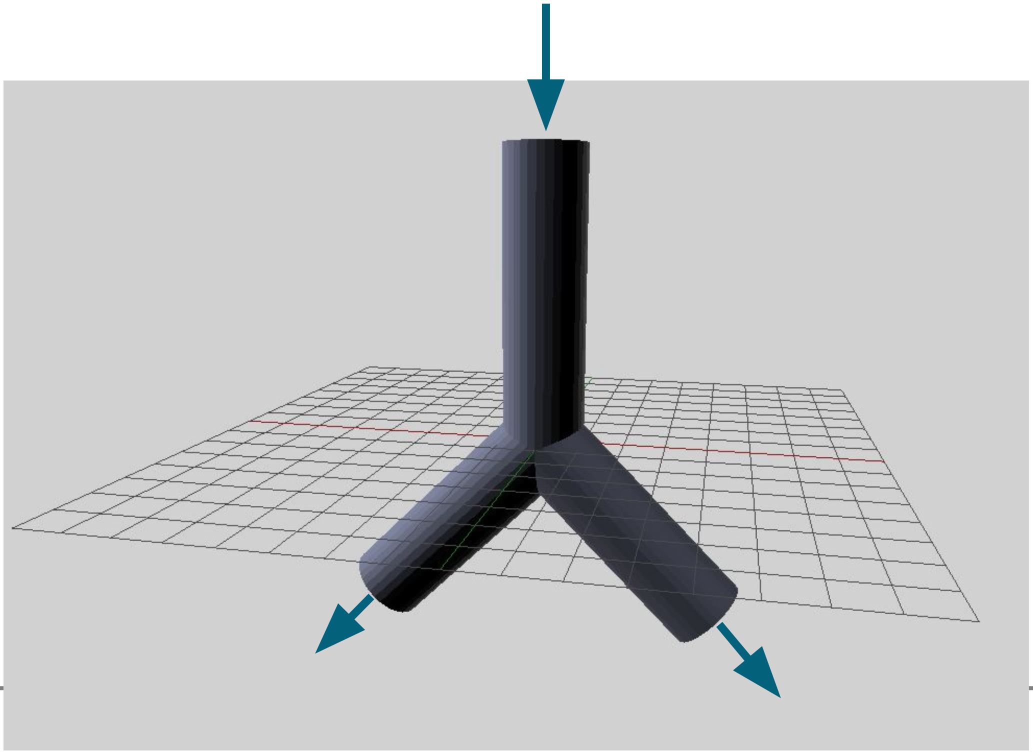
- Alternatively, couple 3D CFD solver to a 1-dimensional solver
- PyNS (A 1D Navier-Stokes solver running with Python).
- Can aid in predicting velocity profiles for areas of the brain which are not accessible to TCD.

Input files for simple case

- Go to the sample input file directory:

```
cd ~/hemelb-pure_public/cases/bifurcation/
```

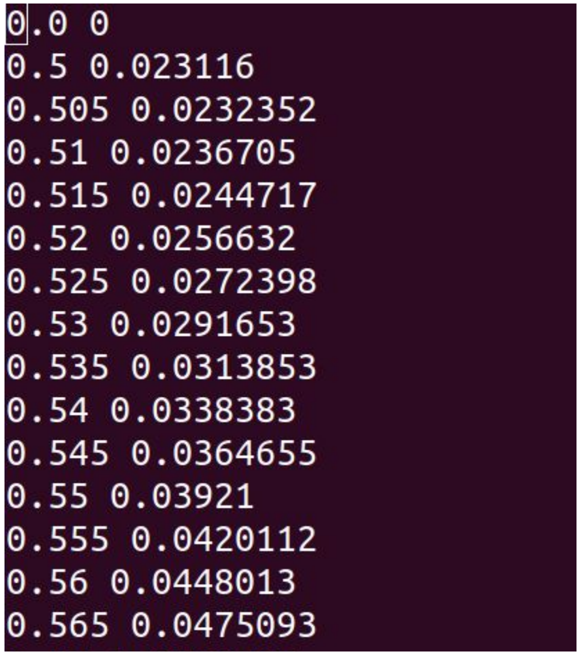
- Let's look through the input files
- We'll use a simple bifurcation geometry to start with



bifurcation.stl

Velocity input files

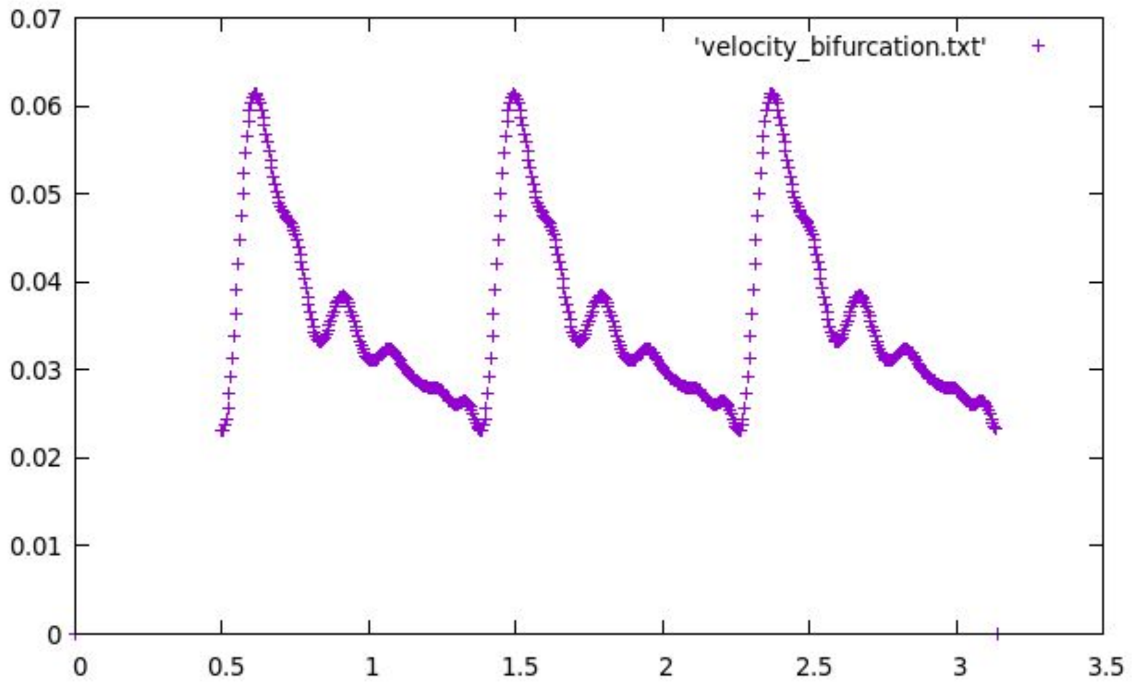
- A peak velocity time-profile for use at the (one) inlet



```
0.0 0
0.5 0.023116
0.505 0.0232352
0.51 0.0236705
0.515 0.0244717
0.52 0.0256632
0.525 0.0272398
0.53 0.0291653
0.535 0.0313853
0.54 0.0338383
0.545 0.0364655
0.55 0.03921
0.555 0.0420112
0.56 0.0448013
0.565 0.0475093
```

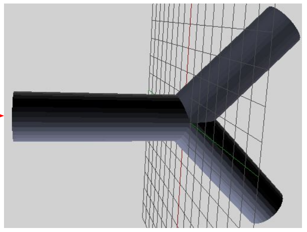
.
.
.

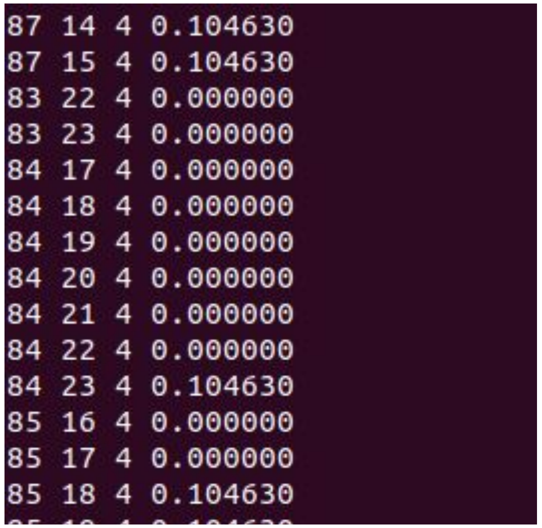
velocity_bifurcation.txt



Velocity input files

- What is velocity_bifurcation.txt.weights.txt?
  - Sets the velocity distribution across the inlet
  - e.g. parabolic profile

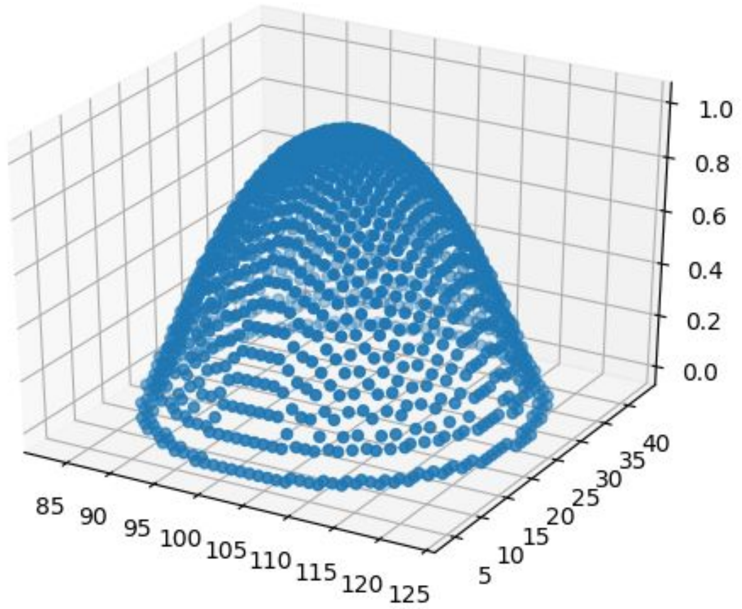




```
87 14 4 0.104630
87 15 4 0.104630
83 22 4 0.000000
83 23 4 0.000000
84 17 4 0.000000
84 18 4 0.000000
84 19 4 0.000000
84 20 4 0.000000
84 21 4 0.000000
84 22 4 0.000000
84 23 4 0.104630
85 16 4 0.000000
85 17 4 0.000000
85 18 4 0.104630
```

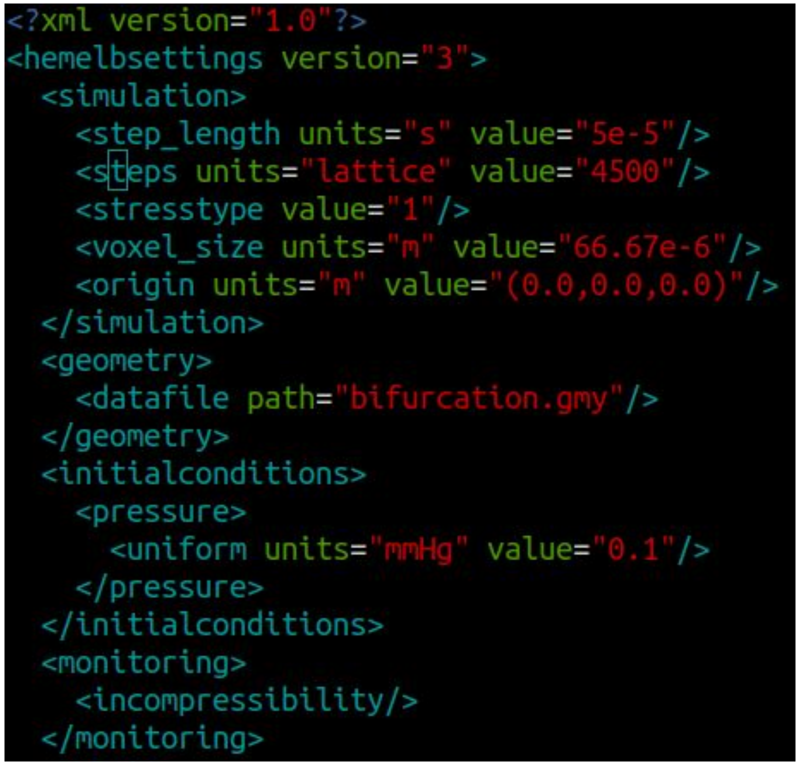
.
.
.

velocity_bifurcation.txt.weights.txt



input.xml

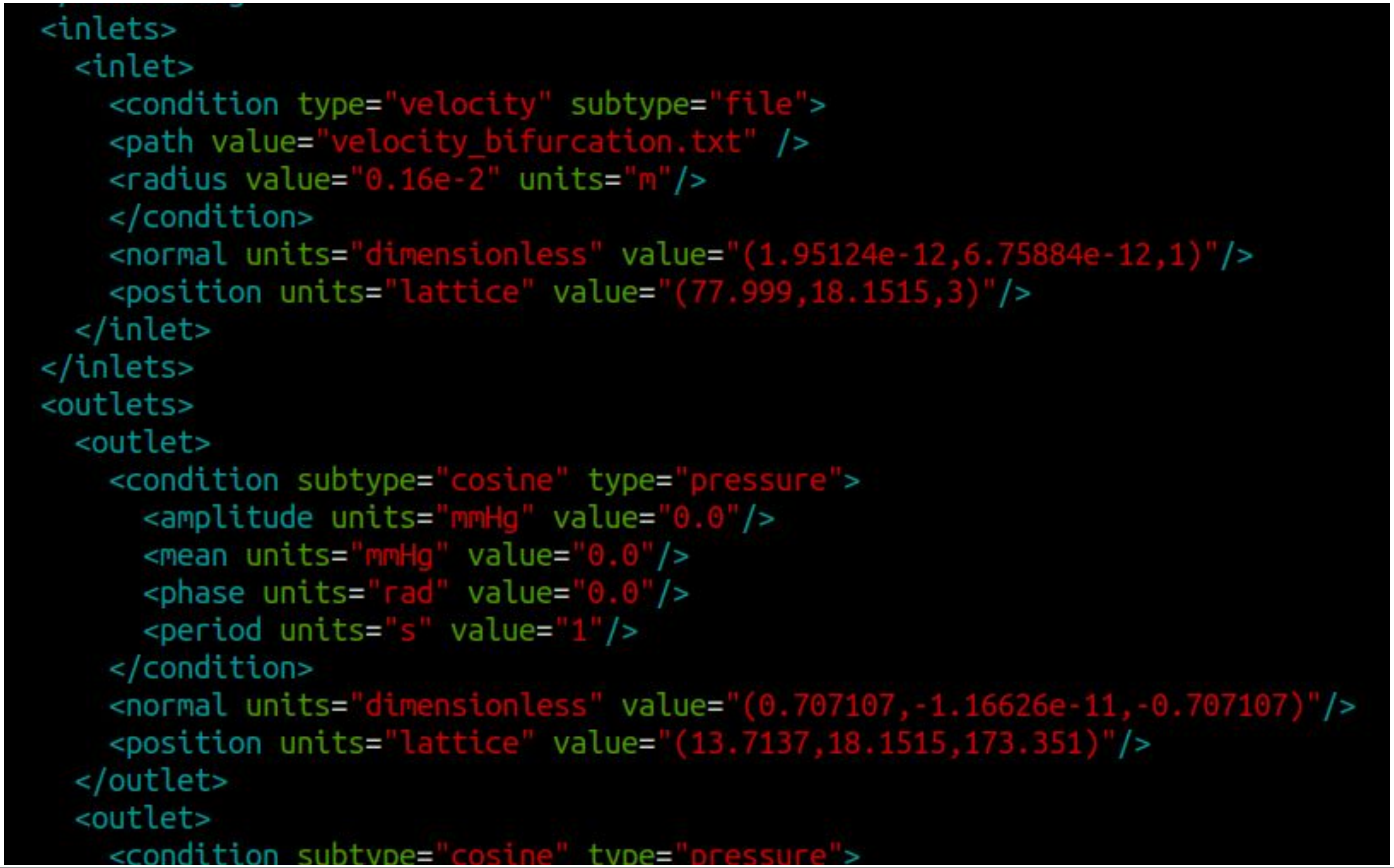
- A single file that tells HemeLB where to find any data it needs
- Normally a graphical user interface, or automated pipeline would generate most of this
- First we have parameters related to the lattice-Boltzmann algorithm:



```
<?xml version="1.0"?>
<hemelbsettings version="3">
  <simulation>
    <step_length units="s" value="5e-5"/>
    <steps units="lattice" value="4500"/>
    <stresstype value="1"/>
    <voxel_size units="m" value="66.67e-6"/>
    <origin units="m" value="(0.0,0.0,0.0)"/>
  </simulation>
  <geometry>
    <datafile path="bifurcation.gmy"/>
  </geometry>
  <initialconditions>
    <pressure>
      <uniform units="mmHg" value="0.1"/>
    </pressure>
  </initialconditions>
  <monitoring>
    <incompressibility/>
  </monitoring>
```

input.xml

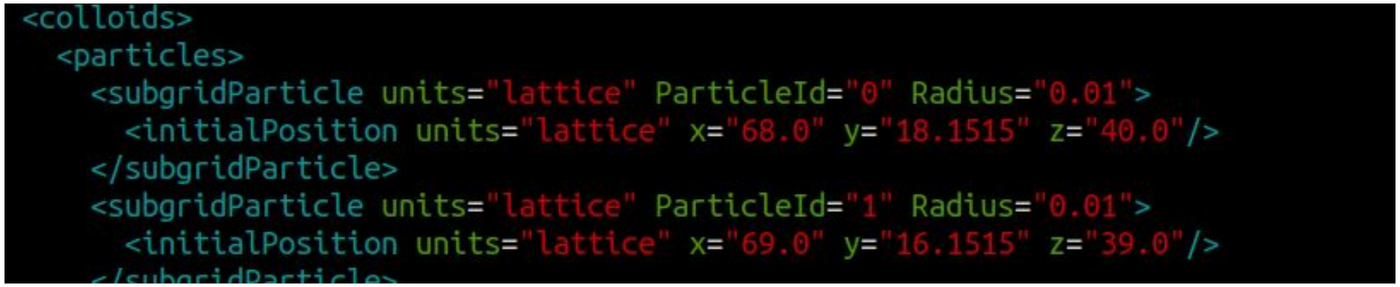
- Information about the location and type of inlets/outlets to the geometry



```
<inlets>
  <inlet>
    <condition type="velocity" subtype="file">
    <path value="velocity_bifurcation.txt" />
    <radius value="0.16e-2" units="m"/>
    </condition>
    <normal units="dimensionless" value="(1.95124e-12,6.75884e-12,1)"/>
    <position units="lattice" value="(77.999,18.1515,3)"/>
  </inlet>
</inlets>
<outlets>
  <outlet>
    <condition subtype="cosine" type="pressure">
      <amplitude units="mmHg" value="0.0"/>
      <mean units="mmHg" value="0.0"/>
      <phase units="rad" value="0.0"/>
      <period units="s" value="1"/>
    </condition>
    <normal units="dimensionless" value="(0.707107,-1.16626e-11,-0.707107)"/>
    <position units="lattice" value="(13.7137,18.1515,173.351)"/>
  </outlet>
  <outlet>
    <condition subtype="cosine" type="pressure">
```

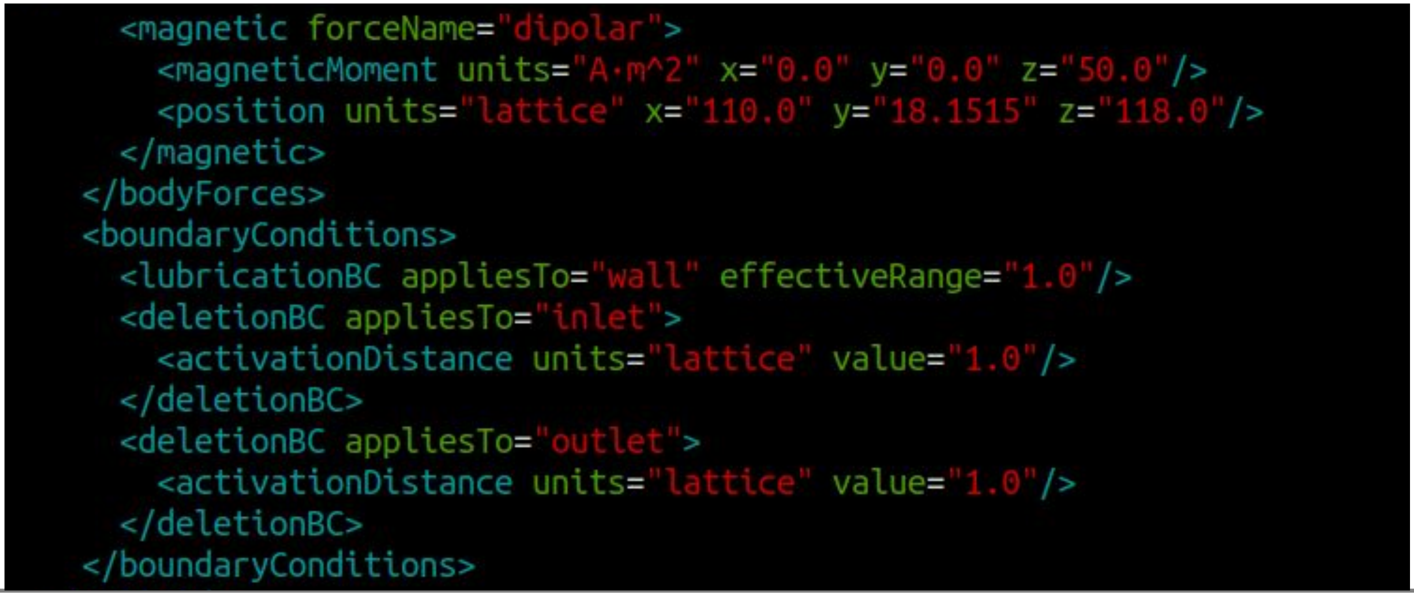
input.xml

- The position of the paramagnetic particles



```
<colloids>
  <particles>
    <subgridParticle units="lattice" ParticleId="0" Radius="0.01">
      <initialPosition units="lattice" x="68.0" y="18.1515" z="40.0"/>
    </subgridParticle>
    <subgridParticle units="lattice" ParticleId="1" Radius="0.01">
      <initialPosition units="lattice" x="69.0" y="16.1515" z="39.0"/>
    </subgridParticle>
```

- Magnet location, strength, wall lubrication interaction, etc.



```
      <magnetic forceName="dipolar">
        <magneticMoment units="A·m^2" x="0.0" y="0.0" z="50.0"/>
        <position units="lattice" x="110.0" y="18.1515" z="118.0"/>
      </magnetic>
    </bodyForces>
    <boundaryConditions>
      <lubricationBC appliesTo="wall" effectiveRange="1.0"/>
      <deletionBC appliesTo="inlet">
        <activationDistance units="lattice" value="1.0"/>
      </deletionBC>
      <deletionBC appliesTo="outlet">
        <activationDistance units="lattice" value="1.0"/>
      </deletionBC>
    </boundaryConditions>
```

- And so on…

Step 3: Run HemeLB

Running HemeLB

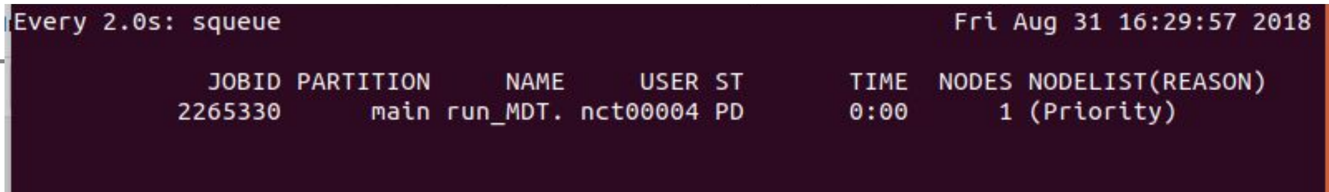
```
#!/bin/bash
#SBATCH --ntasks=8
#SBATCH -o %J.out
#SBATCH -e %J.err
#SBATCH -t 010:00
#SBATCH --reservation=VPHSUMMER18

rm -rf results
srun ~/hemelb-pure_public/src/build/hemelb -i input.xml
```

Submit the above SLURM job script using:

```
sbatch run_bifurc_reservation.sh
```

Check progress with:

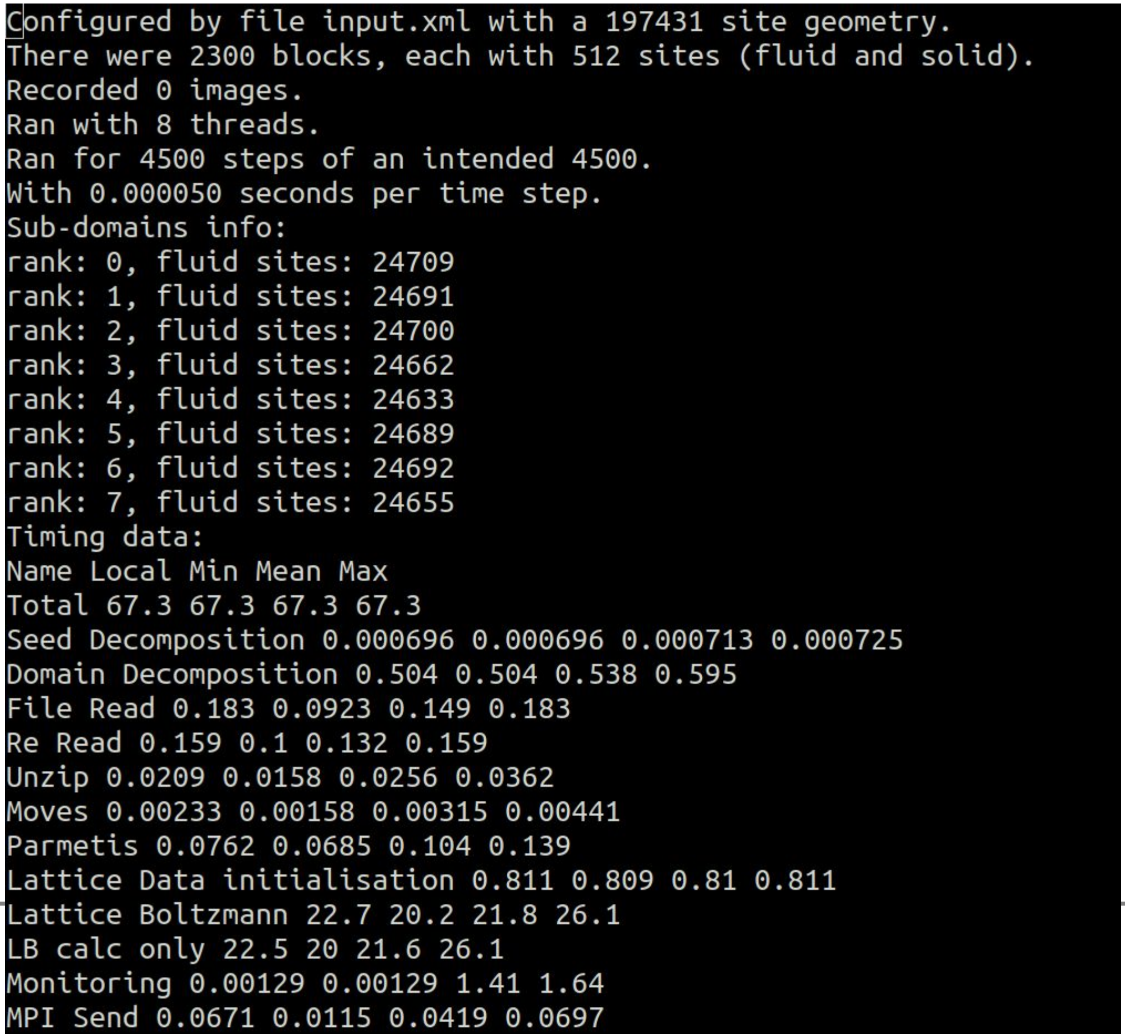
```
watch squeue
```



```
Every 2.0s: squeue                                        Fri Aug 31 16:29:57 2018

          JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
        2265330      main run_MDT. nct00004 PD       0:00      1 (Priority)
```

Check reports.txt

- A new directory should have appeared, called `results/`
- If you're waiting, have a look at `report.txt`



```
Configured by file input.xml with a 197431 site geometry.
There were 2300 blocks, each with 512 sites (fluid and solid).
Recorded 0 images.
Ran with 8 threads.
Ran for 4500 steps of an intended 4500.
With 0.000050 seconds per time step.
Sub-domains info:
rank: 0, fluid sites: 24709
rank: 1, fluid sites: 24691
rank: 2, fluid sites: 24700
rank: 3, fluid sites: 24662
rank: 4, fluid sites: 24633
rank: 5, fluid sites: 24689
rank: 6, fluid sites: 24692
rank: 7, fluid sites: 24655
Timing data:
Name Local Min Mean Max
Total 67.3 67.3 67.3 67.3
Seed Decomposition 0.000696 0.000696 0.000713 0.000725
Domain Decomposition 0.504 0.504 0.538 0.595
File Read 0.183 0.0923 0.149 0.183
Re Read 0.159 0.1 0.132 0.159
Unzip 0.0209 0.0158 0.0256 0.0362
Moves 0.00233 0.00158 0.00315 0.00441
Parmetis 0.0762 0.0685 0.104 0.139
Lattice Data initialisation 0.811 0.809 0.81 0.811
Lattice Boltzmann 22.7 20.2 21.8 26.1
LB calc only 22.5 20 21.6 26.1
Monitoring 0.00129 0.00129 1.41 1.64
MPI Send 0.0671 0.0115 0.0419 0.0697
```

Step 4: Analyse results

Extract data from results/

- We want to visualise what has been output to the results/ directory

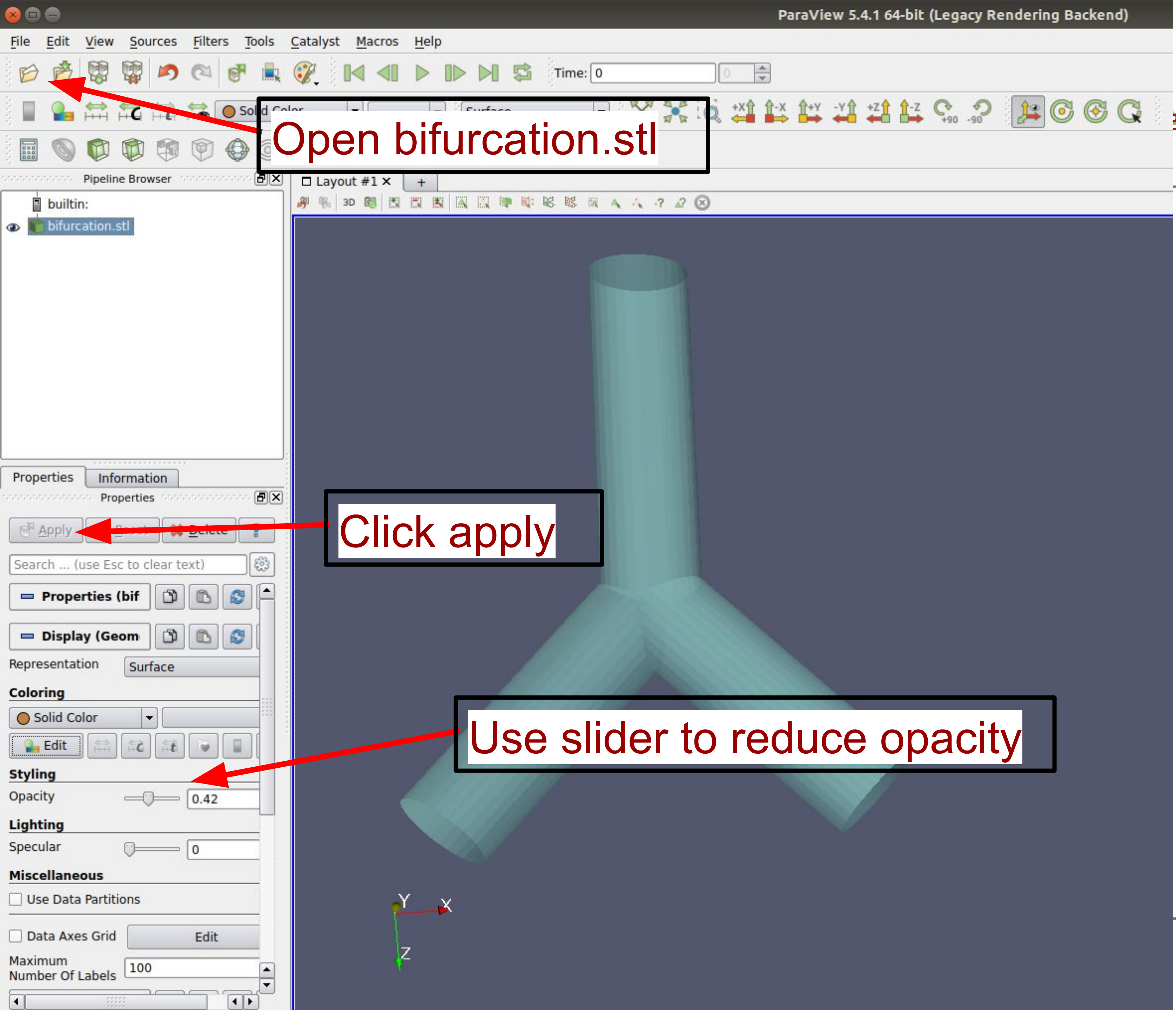
- The following command will extract the information needed for visualization:

```
./hemeXtract -X results/ColloidOutput.xdr -o colloids.csv
```

- Copy this file to your laptop (or wherever you have paraview installed)
- Copy bifurcation.stl to your laptop too

- Once you have both files, on your laptop open paraview

```
paraview
```



Open bifurcation.stl

Click apply

Use slider to reduce opacity

Representation: Surface

Coloring: Solid Color

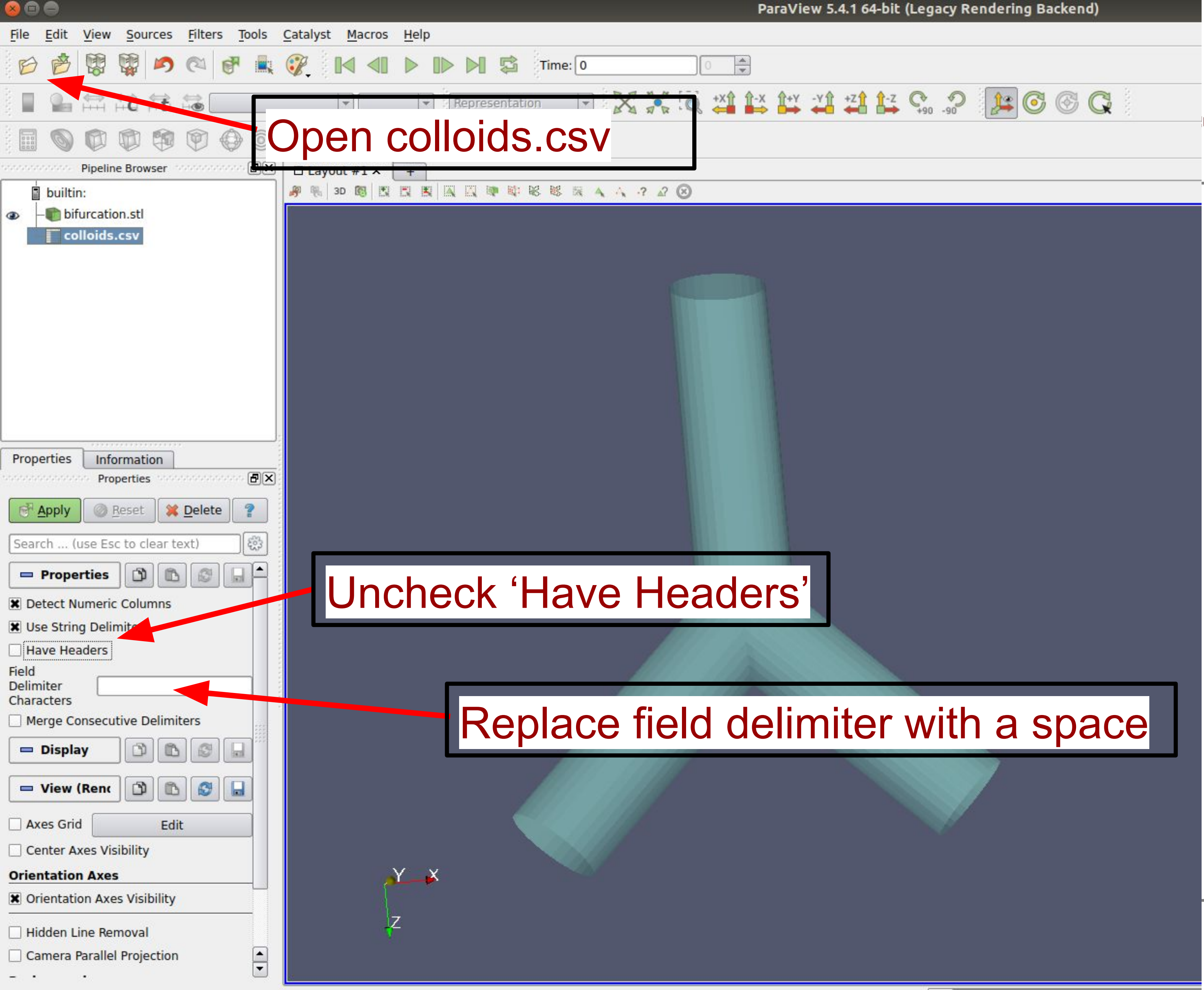
Opacity: 0.42

Specular: 0

☐ Use Data Partitions

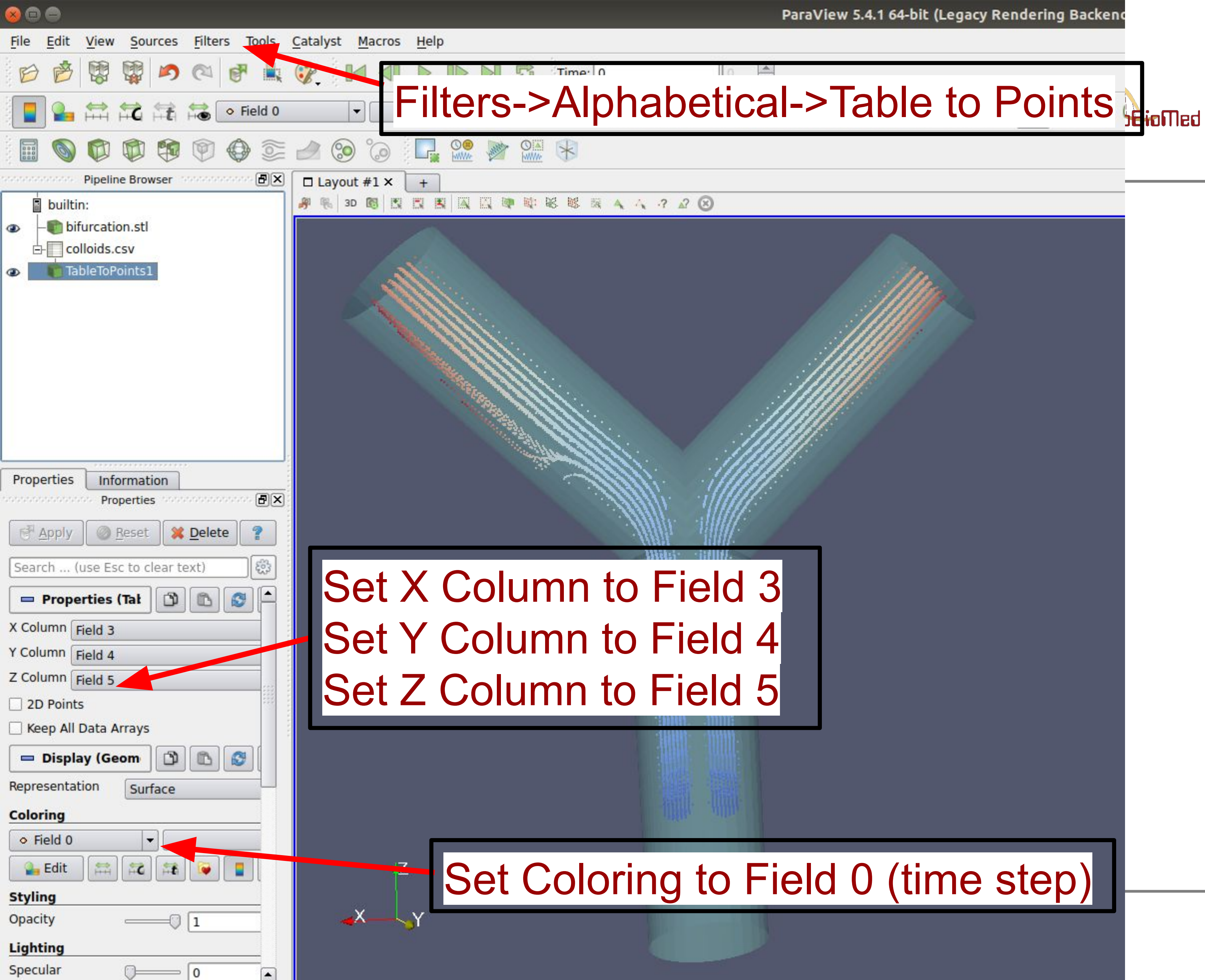
☐ Data Axes Grid — Edit

Maximum Number Of Labels: 100


Open colloids.csv

Uncheck ‘Have Headers’

Replace field delimiter with a space



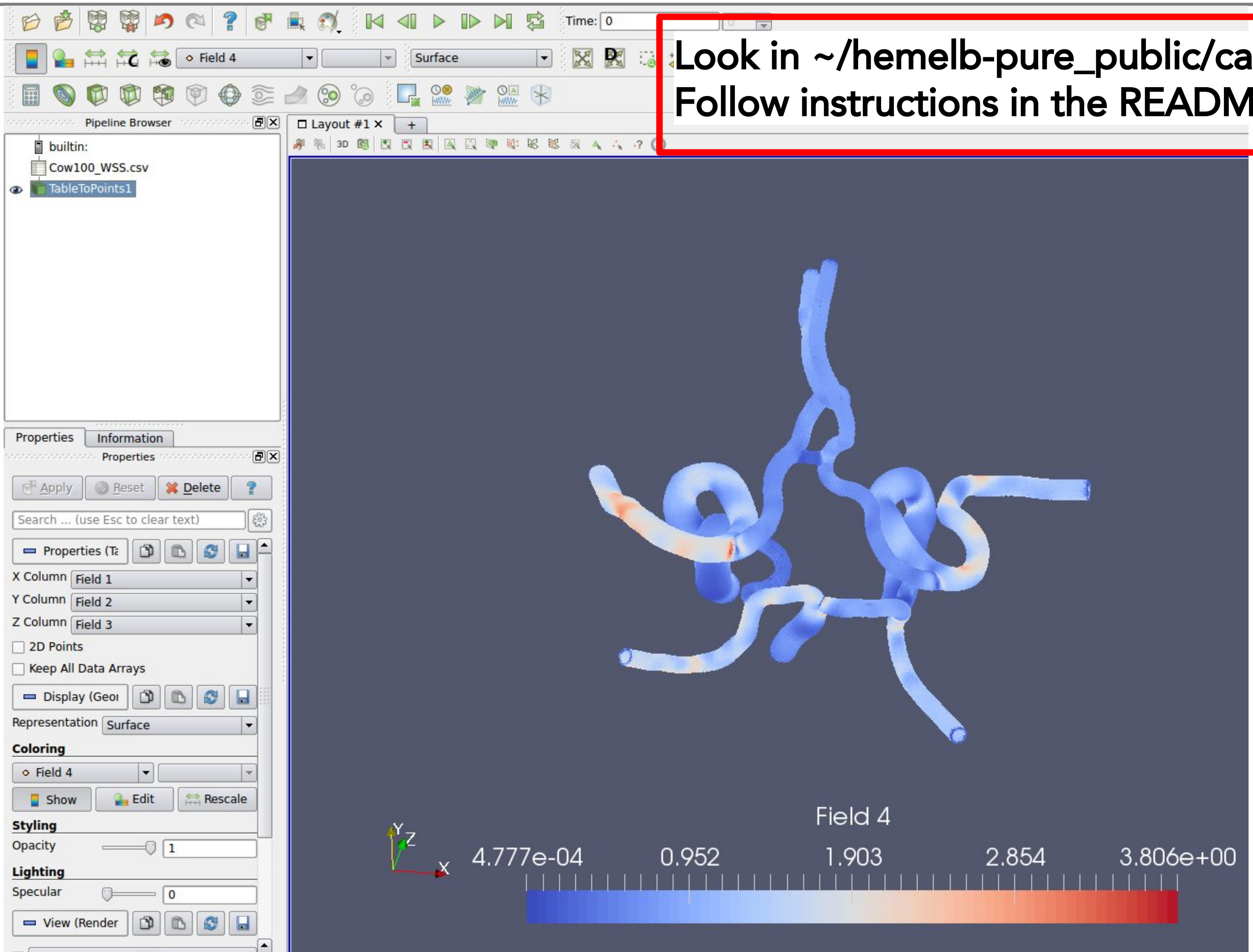
Filters->Alphabetical->Table to Points

X Column: Field 3
Y Column: Field 4
Z Column: Field 5
2D Points
Keep All Data Arrays
Coloring: Field 0
Opacity: 1
Specular: 0

Set X Column to Field 3
Set Y Column to Field 4
Set Z Column to Field 5

Set Coloring to Field 0 (time step)

If there's still time at the end...



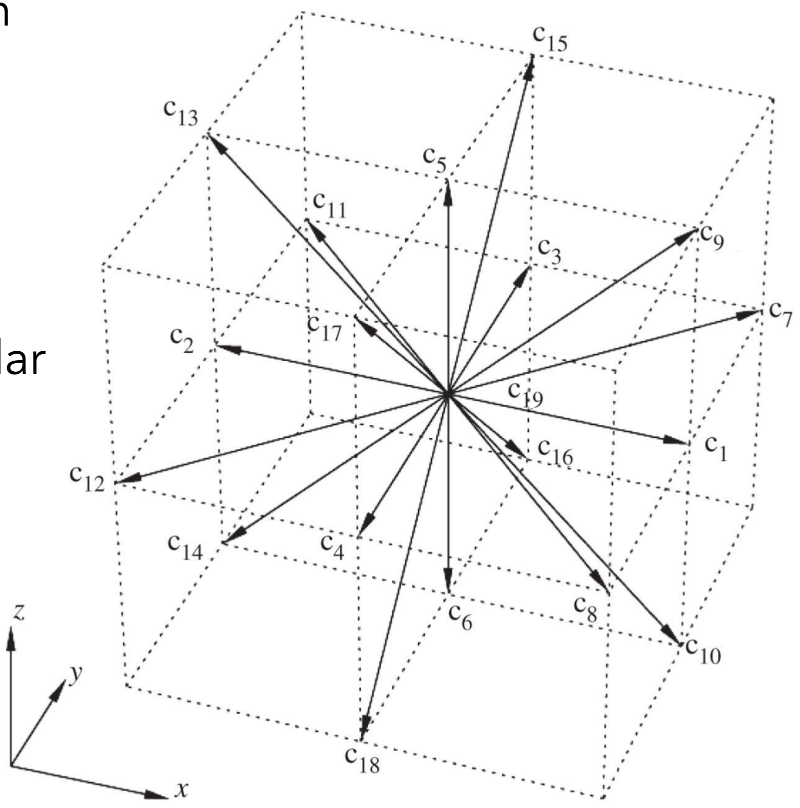
X Column: Field 1
Y Column: Field 2
Z Column: Field 3
2D Points
Keep All Data Arrays
Representation: Surface
Coloring: Field 4
Styling
Opacity: 1
Lighting
Specular: 0

Look in ~/hemelb-pure_public/cases/CoW100
Follow instructions in the README file there

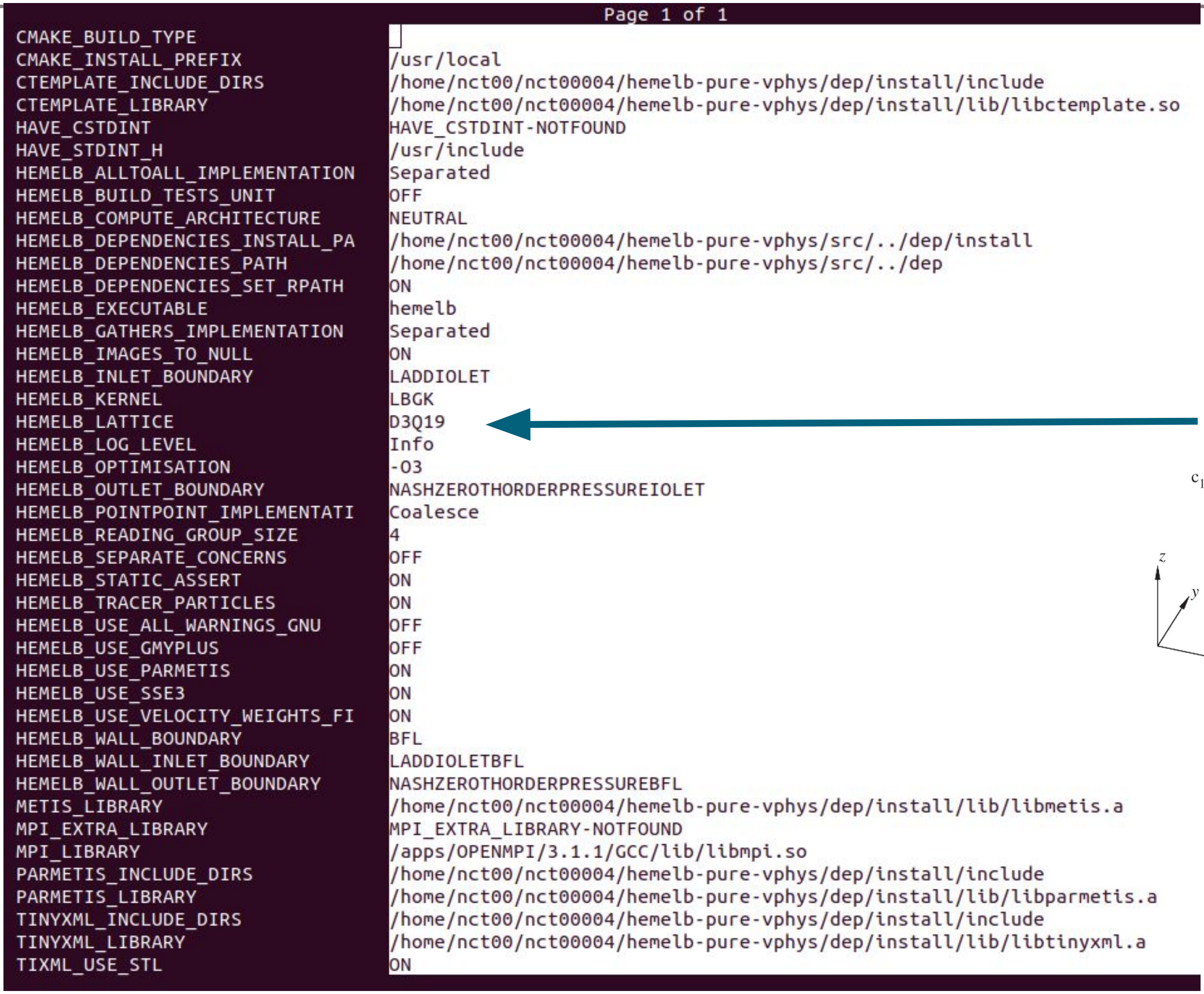
What algorithm is HemeLB running?

Lattice Boltzmann Method

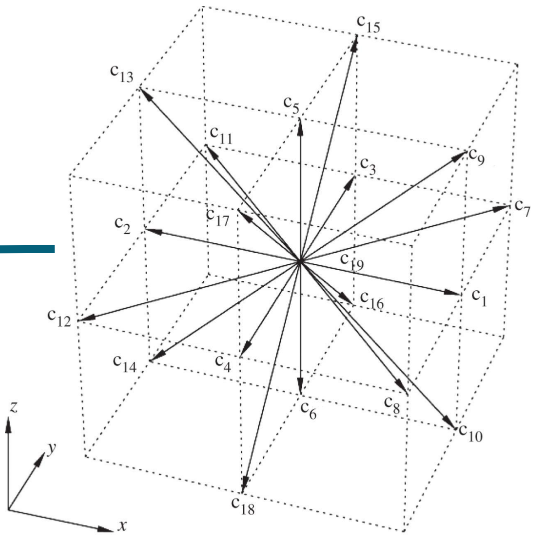
- A discrete way of solving Boltzmann equation
    - Can be shown to satisfy incompressible Navier-Stokes equation
    - Advantages for multiphase, and non-Newtonian flows.
- Why use LB?
    - Divide up simulation domain into a regular grid of lattice sites
        - Fluid sites
        - Wall sites
    - 2 steps: Streaming and Collision
    - Only nearest-neighbour interactions (typically)
        - Extremely scalable
        - Works well with sparse geometries



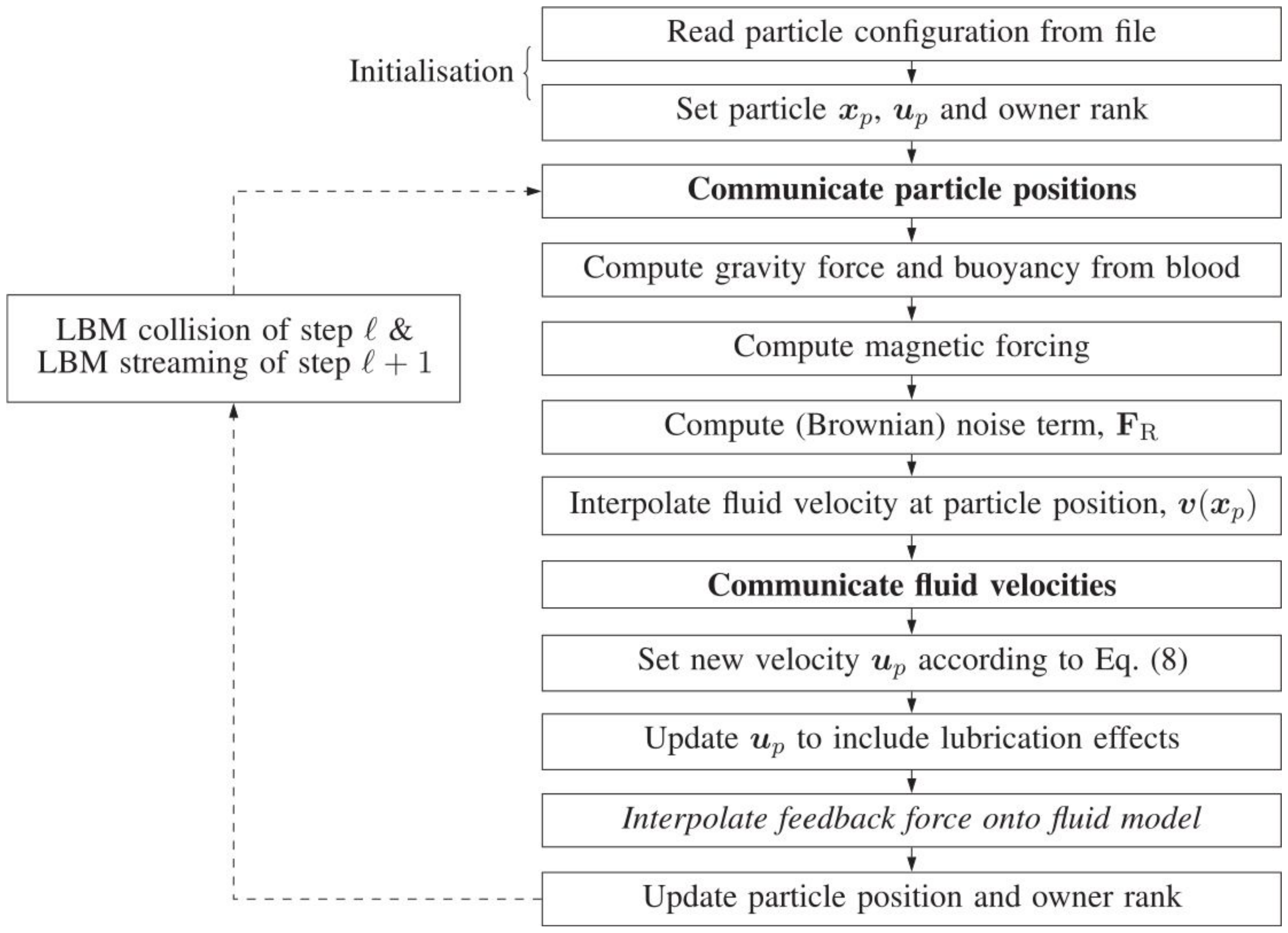
A note on compilation flags



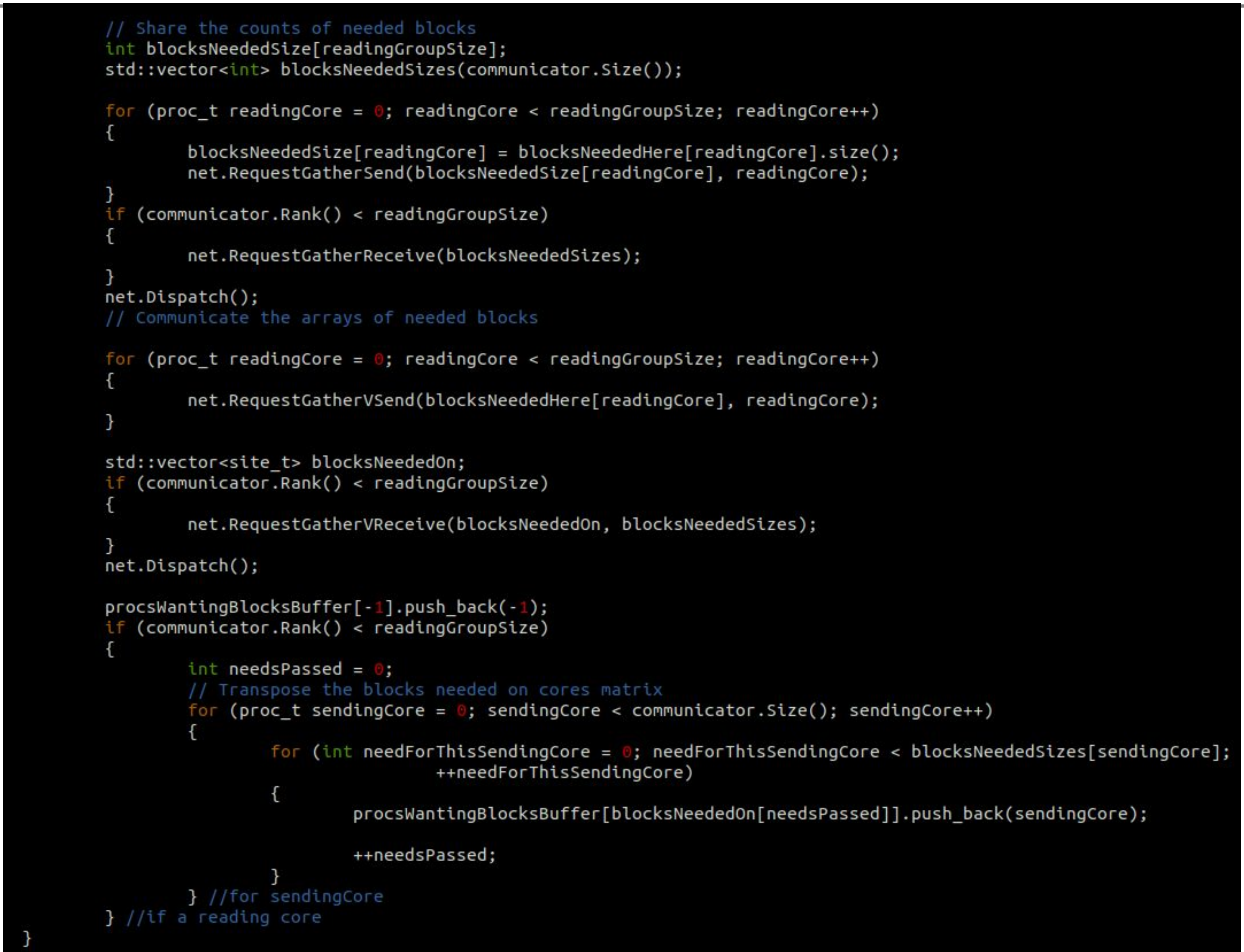
```
                              Page 1 of 1
CMAKE_BUILD_TYPE                
CMAKE_INSTALL_PREFIX            /usr/local
CTEMPLATE_INCLUDE_DIRS          /home/nct00/nct00004/hemelb-pure-vphys/dep/install/include
CTEMPLATE_LIBRARY               /home/nct00/nct00004/hemelb-pure-vphys/dep/install/lib/libctemplate.so
HAVE_CSTDINT                    HAVE_CSTDINT-NOTFOUND
HAVE_STDINT_H                   /usr/include
HEMELB_ALLTOALL_IMPLEMENTATION  Separated
HEMELB_BUILD_TESTS_UNIT         OFF
HEMELB_COMPUTE_ARCHITECTURE     NEUTRAL
HEMELB_DEPENDENCIES_INSTALL_PA  /home/nct00/nct00004/hemelb-pure-vphys/src/../dep/install
HEMELB_DEPENDENCIES_PATH        /home/nct00/nct00004/hemelb-pure-vphys/src/../dep
HEMELB_DEPENDENCIES_SET_RPATH   ON
HEMELB_EXECUTABLE               hemelb
HEMELB_GATHERS_IMPLEMENTATION   Separated
HEMELB_IMAGES_TO_NULL           ON
HEMELB_INLET_BOUNDARY           LADDIOLET
HEMELB_KERNEL                   LBGK
HEMELB_LATTICE                  D3Q19
HEMELB_LOG_LEVEL                Info
HEMELB_OPTIMISATION             -O3
HEMELB_OUTLET_BOUNDARY          NASHZEROTHORDERPRESSUREIOLET
HEMELB_POINTPOINT_IMPLEMENTATI  Coalesce
HEMELB_READING_GROUP_SIZE       4
HEMELB_SEPARATE_CONCERNS        OFF
HEMELB_STATIC_ASSERT            ON
HEMELB_TRACER_PARTICLES         ON
HEMELB_USE_ALL_WARNINGS_GNU     OFF
HEMELB_USE_GMYPLUS              OFF
HEMELB_USE_PARMETIS             ON
HEMELB_USE_SSE3                 ON
HEMELB_USE_VELOCITY_WEIGHTS_FI  ON
HEMELB_WALL_BOUNDARY            BFL
HEMELB_WALL_INLET_BOUNDARY      LADDIOLETBFL
HEMELB_WALL_OUTLET_BOUNDARY     NASHZEROTHORDERPRESSUREBFL
METIS_LIBRARY                   /home/nct00/nct00004/hemelb-pure-vphys/dep/install/lib/libmetis.a
MPI_EXTRA_LIBRARY               MPI_EXTRA_LIBRARY-NOTFOUND
MPI_LIBRARY                     /apps/OPENMPI/3.1.1/GCC/lib/libmpi.so
PARMETIS_INCLUDE_DIRS           /home/nct00/nct00004/hemelb-pure-vphys/dep/install/include
PARMETIS_LIBRARY                /home/nct00/nct00004/hemelb-pure-vphys/dep/install/lib/libparmetis.a
TINYXML_INCLUDE_DIRS            /home/nct00/nct00004/hemelb-pure-vphys/dep/install/include
TINYXML_LIBRARY                 /home/nct00/nct00004/hemelb-pure-vphys/dep/install/lib/libtinyxml.a
TIXML_USE_STL                   ON
```



MDT Algorithm


Read particle configuration from file
Initialisation
Set particle $\boldsymbol{x}_p$, $\boldsymbol{u}_p$ and owner rank
Communicate particle positions
Compute gravity force and buoyancy from blood
LBM collision of step ℓ & LBM streaming of step $\ell + 1$
Compute magnetic forcing
Compute (Brownian) noise term, $\mathbf{F}_\mathrm{R}$
Interpolate fluid velocity at particle position, $\boldsymbol{v}(\boldsymbol{x}_p)$
Communicate fluid velocities
Set new velocity $\boldsymbol{u}_p$ according to Eq. (8)
Update $\boldsymbol{u}_p$ to include lubrication effects
Interpolate feedback force onto fluid model
Update particle position and owner rank

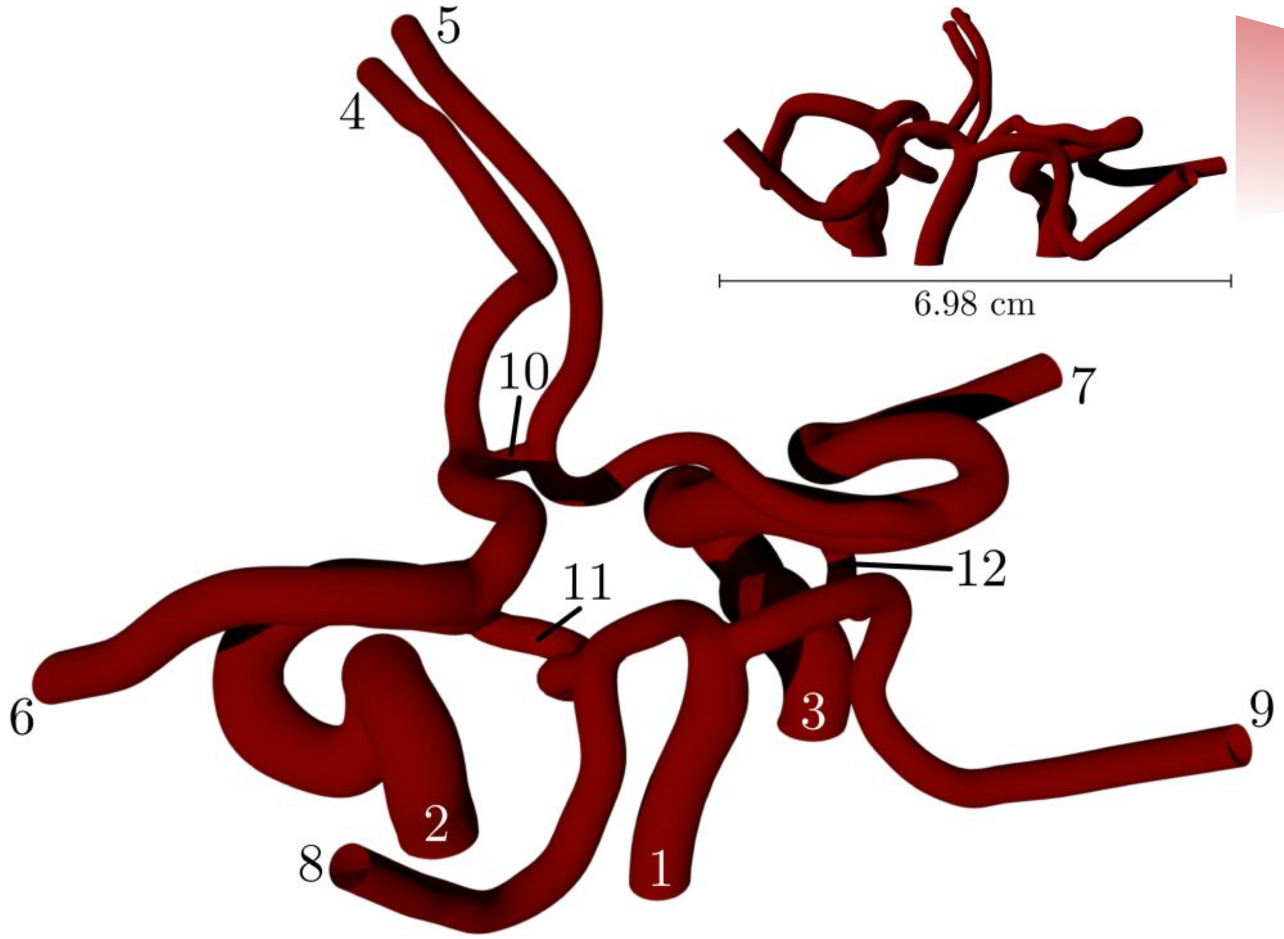
Example Code Fragment

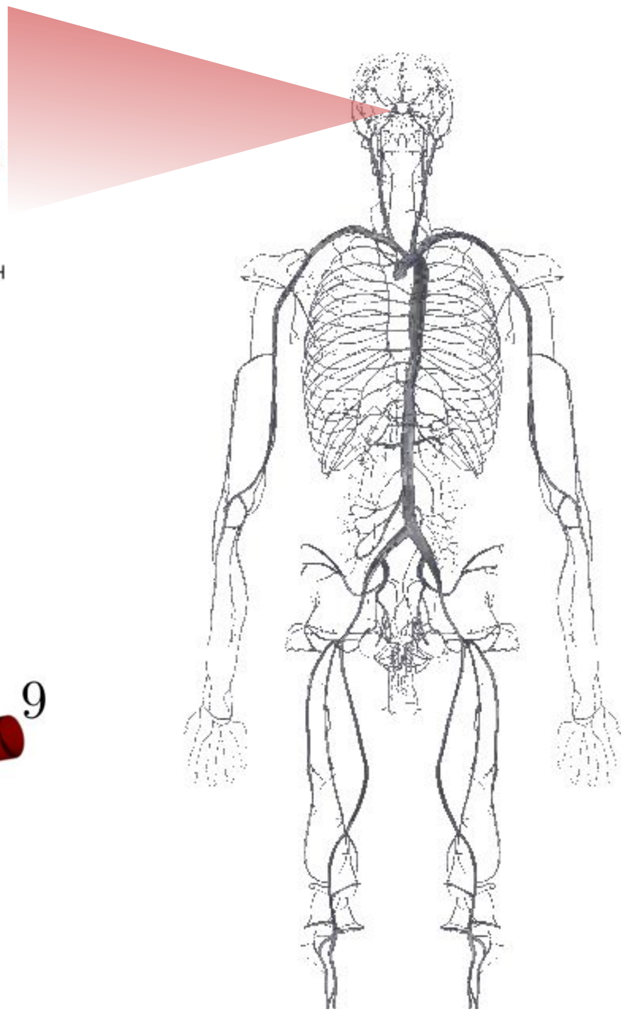


```
        // Share the counts of needed blocks
        int blocksNeededSize[readingGroupSize];
        std::vector<int> blocksNeededSizes(communicator.Size());

        for (proc_t readingCore = 0; readingCore < readingGroupSize; readingCore++)
        {
                blocksNeededSize[readingCore] = blocksNeededHere[readingCore].size();
                net.RequestGatherSend(blocksNeededSize[readingCore], readingCore);
        }
        if (communicator.Rank() < readingGroupSize)
        {
                net.RequestGatherReceive(blocksNeededSizes);
        }
        net.Dispatch();
        // Communicate the arrays of needed blocks

        for (proc_t readingCore = 0; readingCore < readingGroupSize; readingCore++)
        {
                net.RequestGatherVSend(blocksNeededHere[readingCore], readingCore);
        }

        std::vector<site_t> blocksNeededOn;
        if (communicator.Rank() < readingGroupSize)
        {
                net.RequestGatherVReceive(blocksNeededOn, blocksNeededSizes);
        }
        net.Dispatch();

        procsWantingBlocksBuffer[-1].push_back(-1);
        if (communicator.Rank() < readingGroupSize)
        {
                int needsPassed = 0;
                // Transpose the blocks needed on cores matrix
                for (proc_t sendingCore = 0; sendingCore < communicator.Size(); sendingCore++)
                {
                        for (int needForThisSendingCore = 0; needForThisSendingCore < blocksNeededSizes[sendingCore];
                                        ++needForThisSendingCore)
                        {
                                procsWantingBlocksBuffer[blocksNeededOn[needsPassed]].push_back(sendingCore);

                                ++needsPassed;
                        }
                } //for sendingCore
        } //if a reading core
}
```

What’s the point?

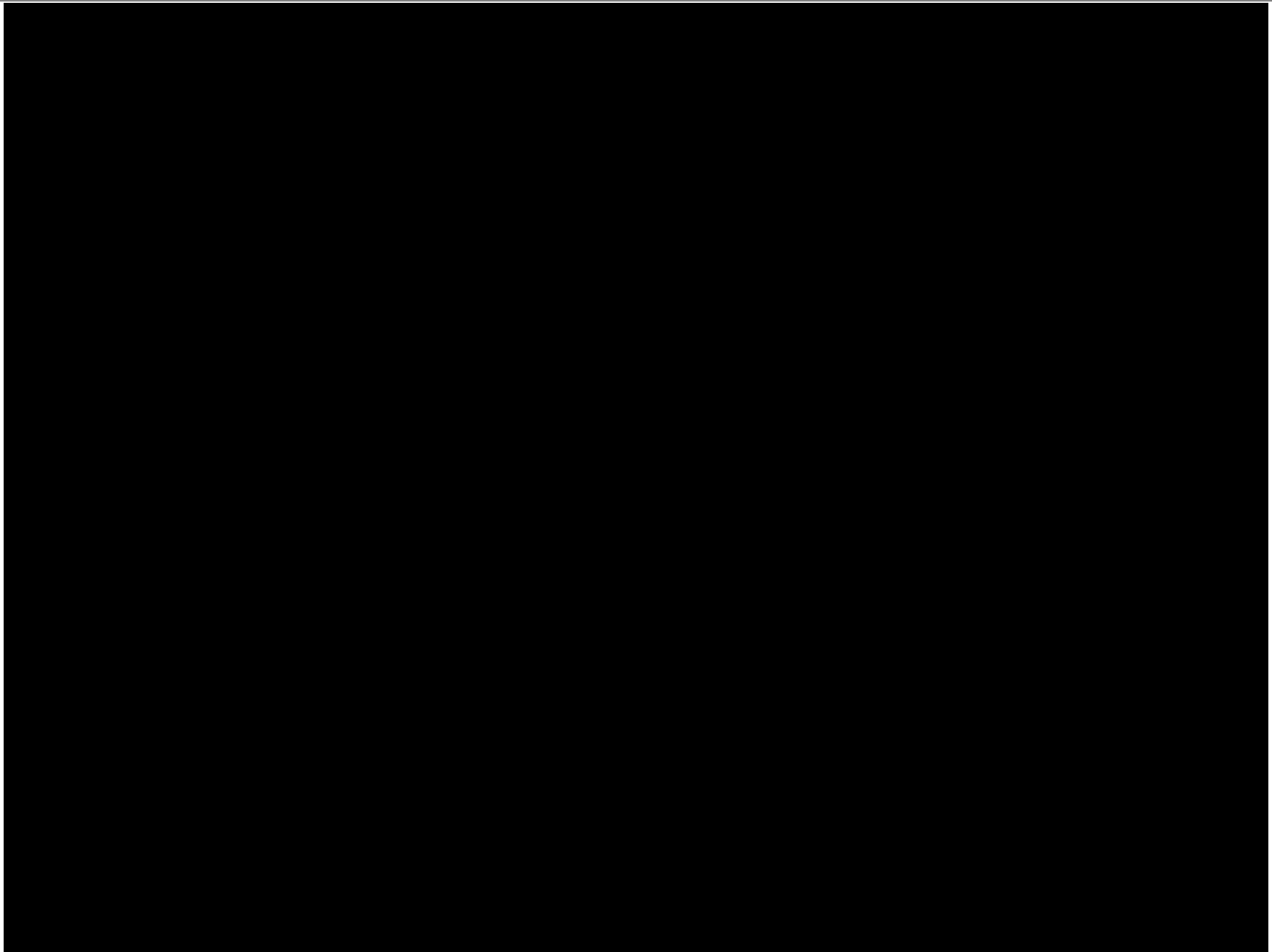
Circle of Willis



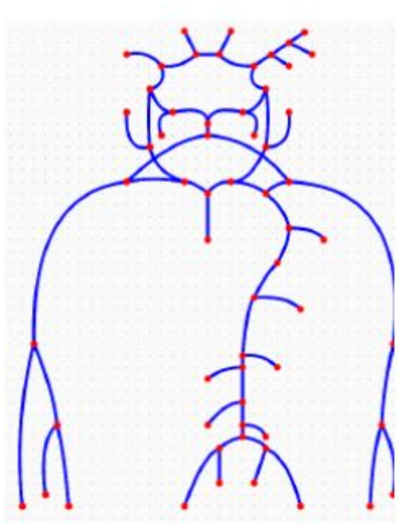


Patronis et al., Front. Physiol. 9:331 (2018)

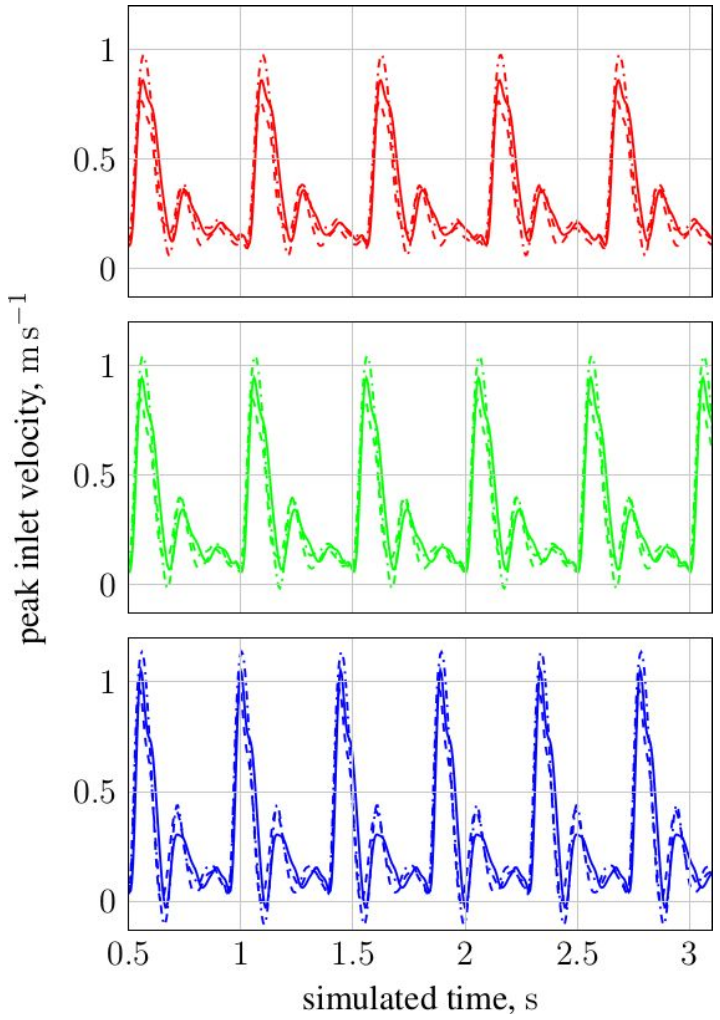
Tracer particle flow in CoW



Velocity input files

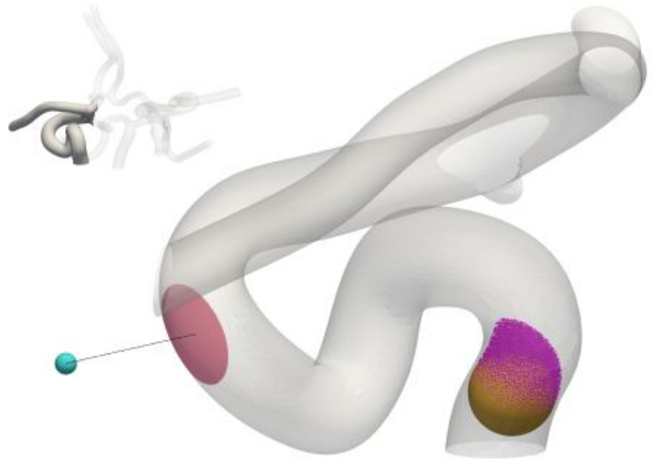


- Inputs generated from 1D model

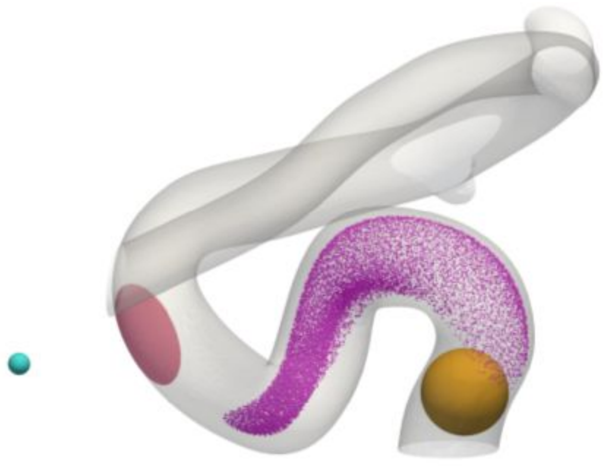


Video of particles in CoW

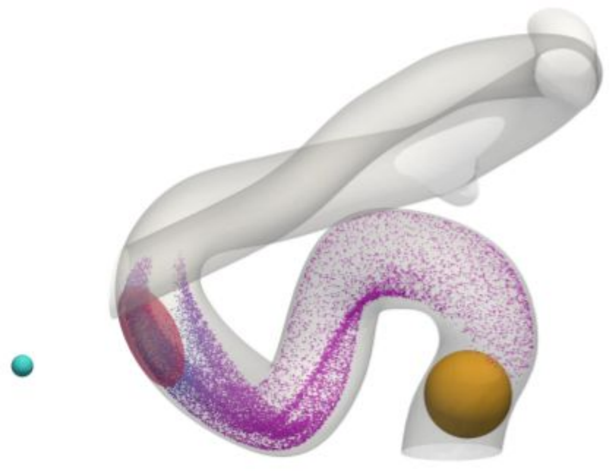
Particles at target site



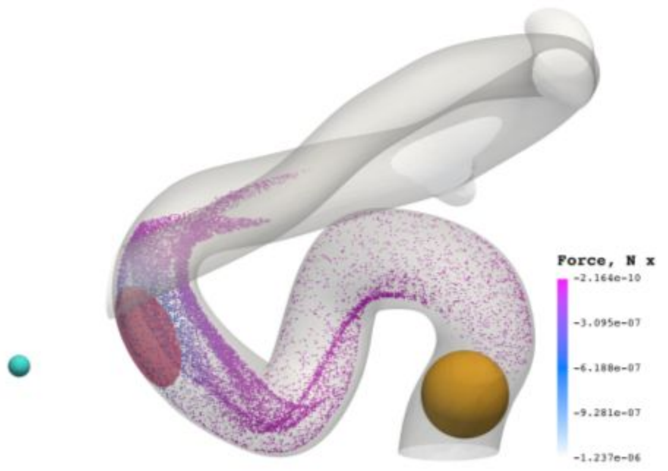
(**7a**) Particle positions at 0.078 s.



(**7b**) Particle positions at 0.273 s.



(**7c**) Particle positions at 0.351 s.

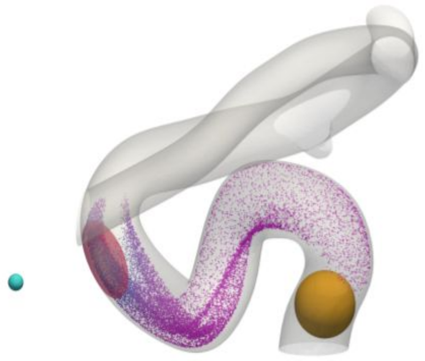


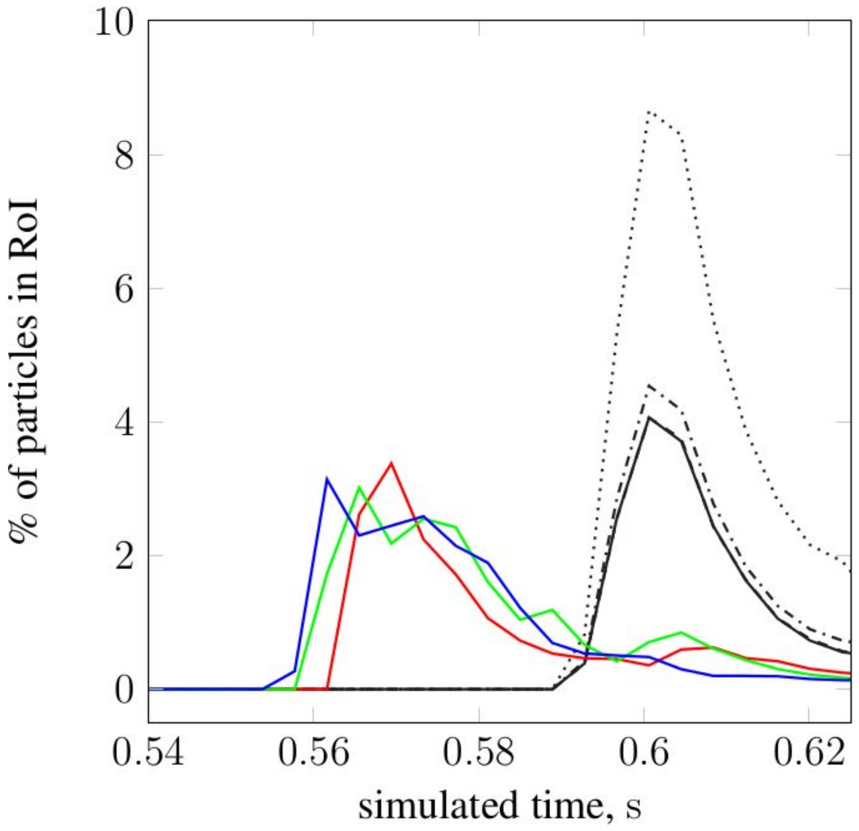
(**7d**) Particle positions at 0.39 s.

Patronis et al., Front. Physiol. 9:331 (2018)

Particles at target site

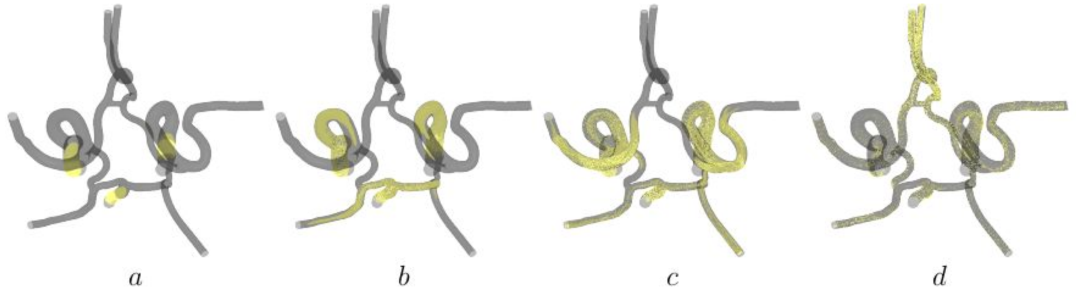
- Superparamagnetic Iron Oxide Nanoparticles with drug coating
- Aim: Predict required dose given patient specific geometry and physiological state, magnet configuration etc.

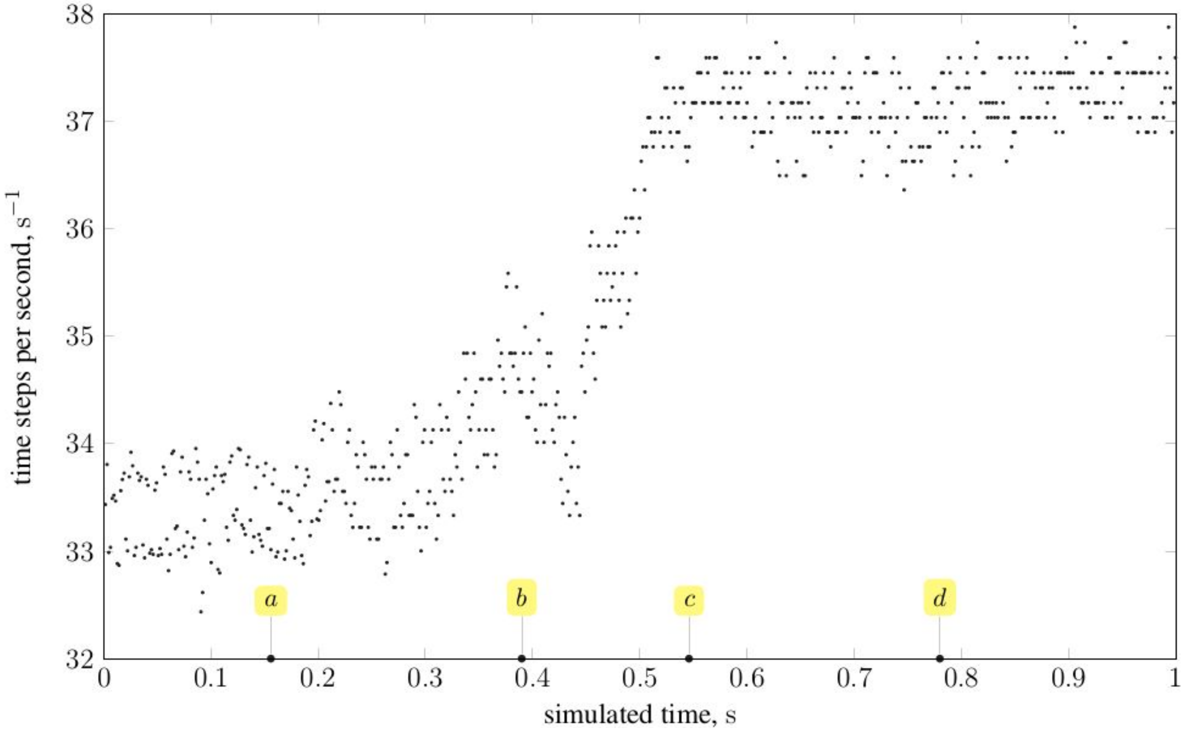

(7c) Particle positions at 0.351 s.



- 80 mmHg, 4.8 l min−1, 68 bpm
- 112 mmHg, 10.7 l min−1, 113 bpm
- 116 mmHg, 11.9 l min−1, 120 bpm
- 122 mmHg, 13.2 l min−1, 134 bpm

Load Balancing

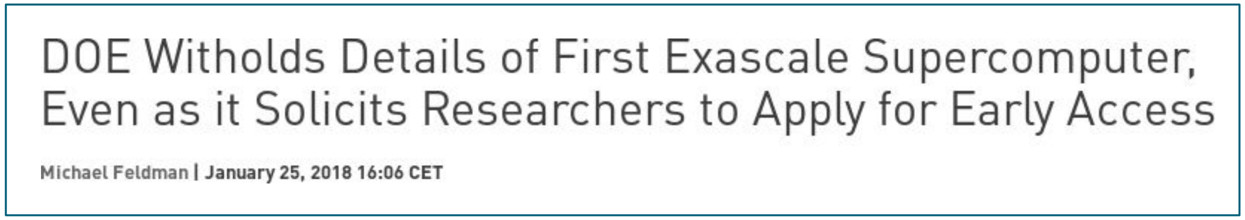




Massive supercomputers

What about the Exascale?

- “Soon”



DOE Witholds Details of First Exascale Supercomputer, Even as it Solicits Researchers to Apply for Early Access

Michael Feldman | January 25, 2018 16:06 CET

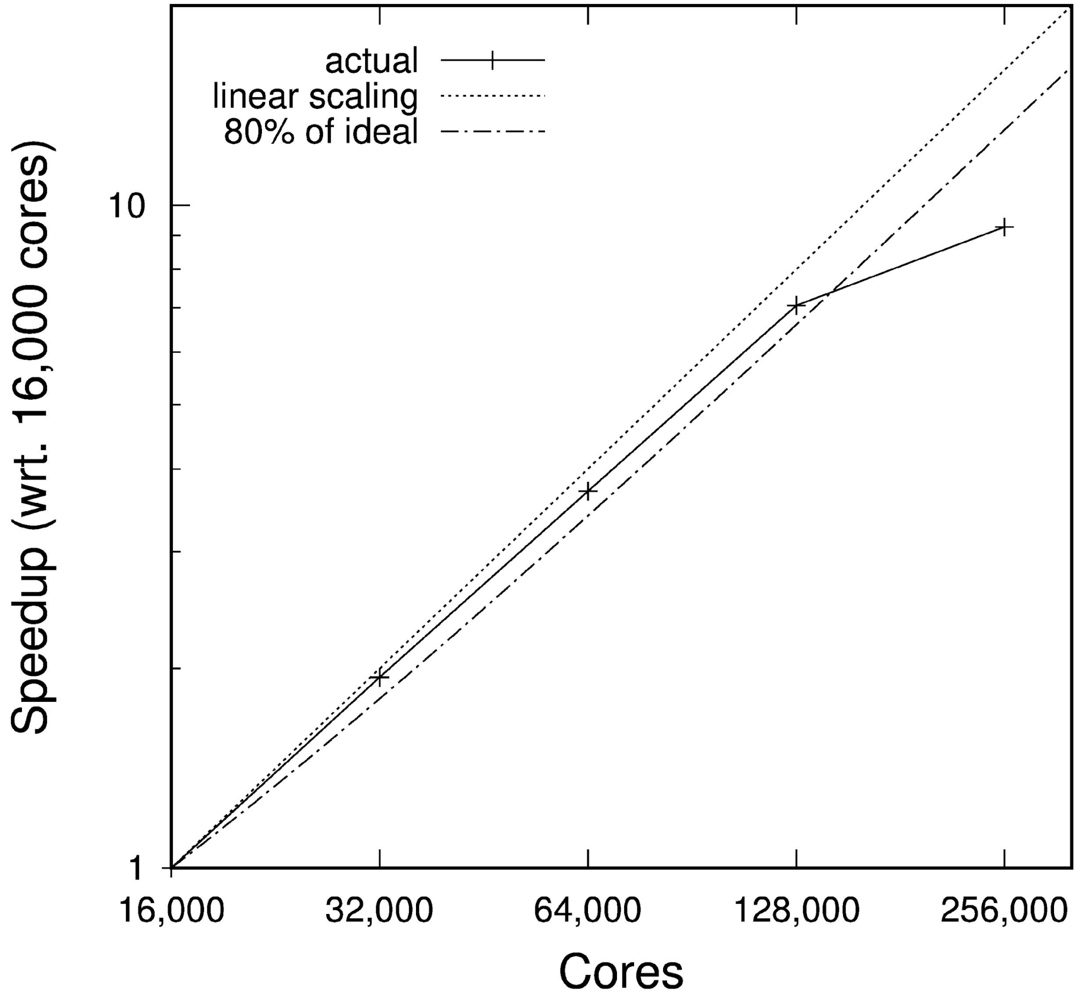
https://www.top500.org

- US Department of Energy, expects Aurora exascale machine by 2021
  - But won’t give details
- Mixtures of GPU and CPU nodes
- “real challenge here is to keep the power draw to something in the neighborhood of 400 to 600 watts per node”

Scaling up

- Speedup = t_1/t_N
- t_1 = time on one core
- t_N = time on N cores

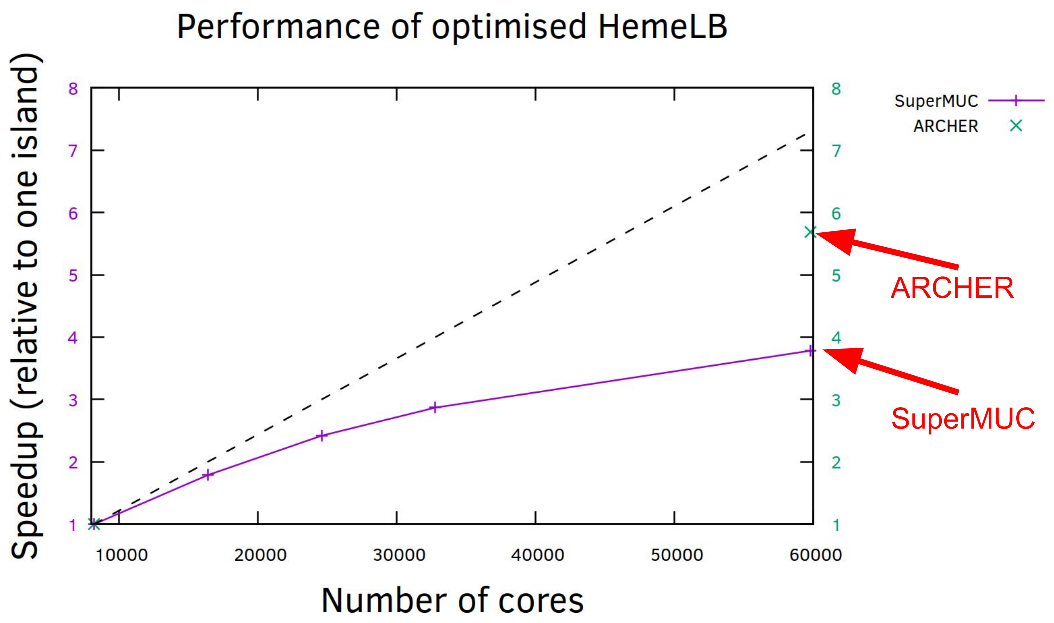
- Ideal would be N times speedup on N cores



Blue Waters, ~5 billion fluid sites

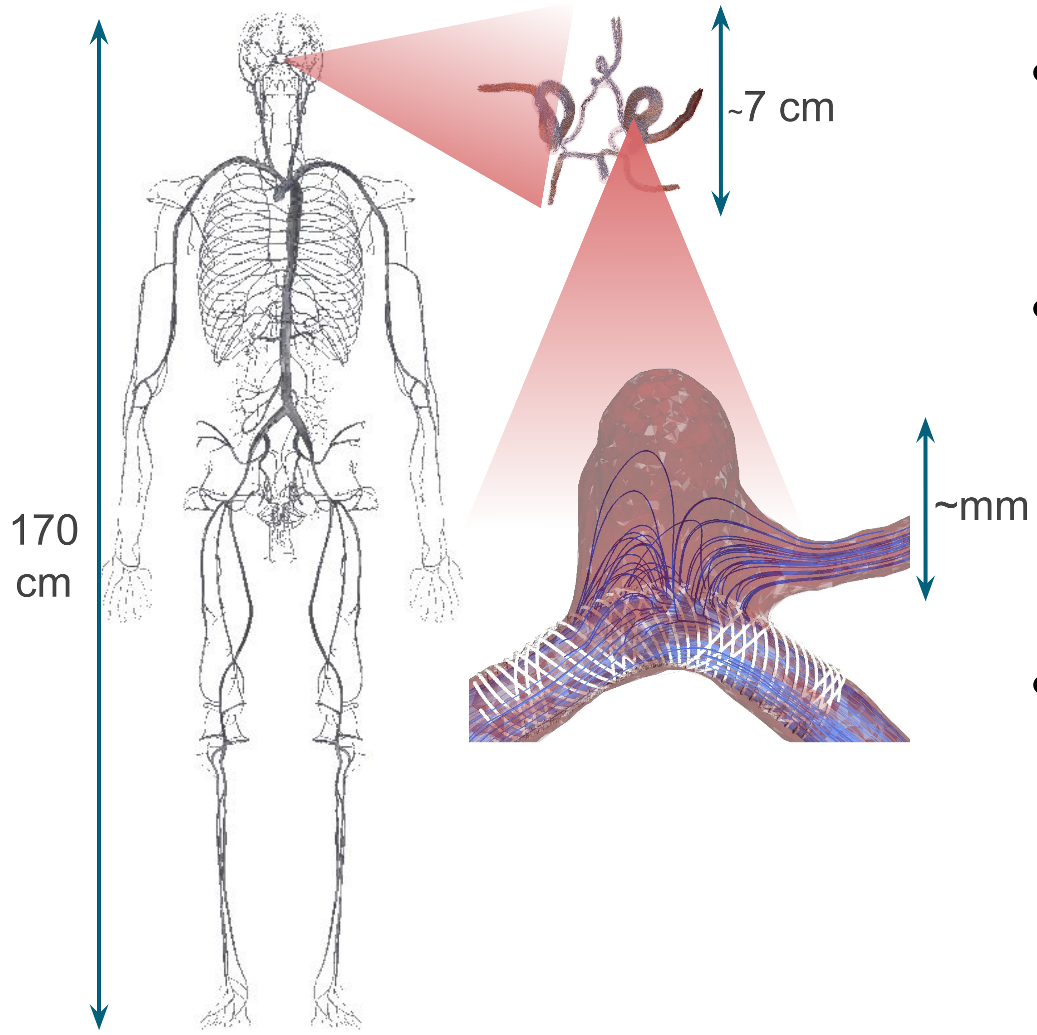
Load balance and topology issues

- SuperMUC island topology is more challenging
  - Requires tuning of load decomposition
- Challenge due to high sparsity of vascular systems (CoW has << 1% volume fluid sites)



Scaling relative to 1 SuperMUC island (8192 cores) for a 20um circle of Willis geometry (around 360 million fluid sites)

Full Human Arterial Tree



- Typically use CT-scan data (Angiogram)
  - Segmentation of files ~9.8 GB
  - Full human arterial tree obtained through MRI scan
- Voxelization and geometry building takes ~10+ hours on 100s-1000s of cores for the ***largest*** cases
  - Requiring ~300G memory for voxelization
  - Outputting ~5 Terabytes fluid site data (uncompressed)
  - Final geometry file is ~10GB
- Lattice-Boltzmann simulation (HemeLB)
  - Large cases require 30k cores+ for 20 hours+

Acknowledgements

UCL

Alex Patronis
Derek Groen
Sebastian Schmieschek
Glen Anderson
Ulf Schiller
Rupert Nash
James Hetherington
Miguel Bernabeu

Hoskote Chandrashekar
Fergus Robertson

Peter Coveney

CompBioMed

MAPPER

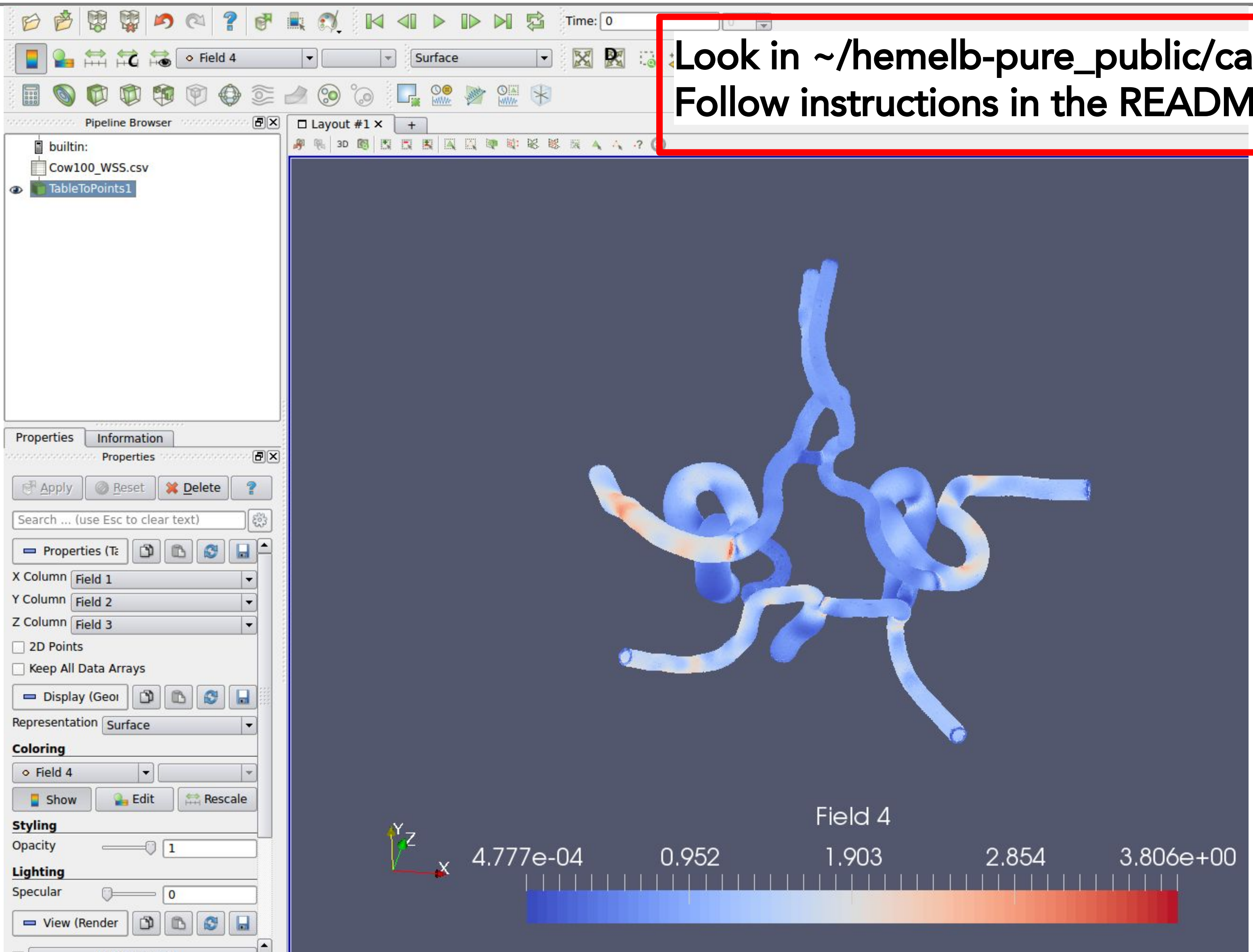
COMPAT
Computing Patterns for High Performance Multiscale Computing

UKCOMES

الصندوق القطري لرعاية البحث العلمي
Qatar National Research Fund

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 671564.

Still time…?



X Column: Field 1
Y Column: Field 2
Z Column: Field 3
2D Points
Keep All Data Arrays
Representation: Surface
Coloring: Field 4
Styling
Opacity: 1
Lighting
Specular: 0

Look in ~/hemelb-pure_public/cases/CoW100
Follow instructions in the README file there