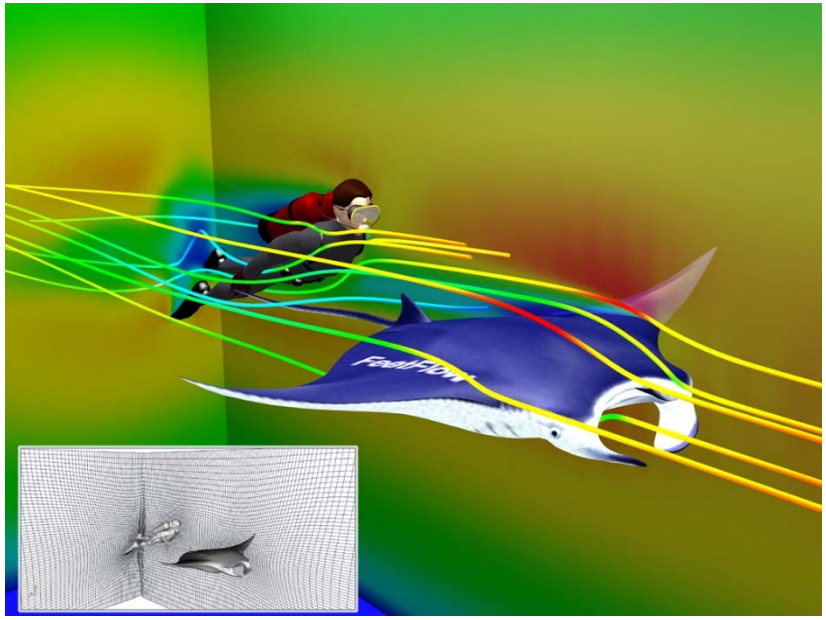
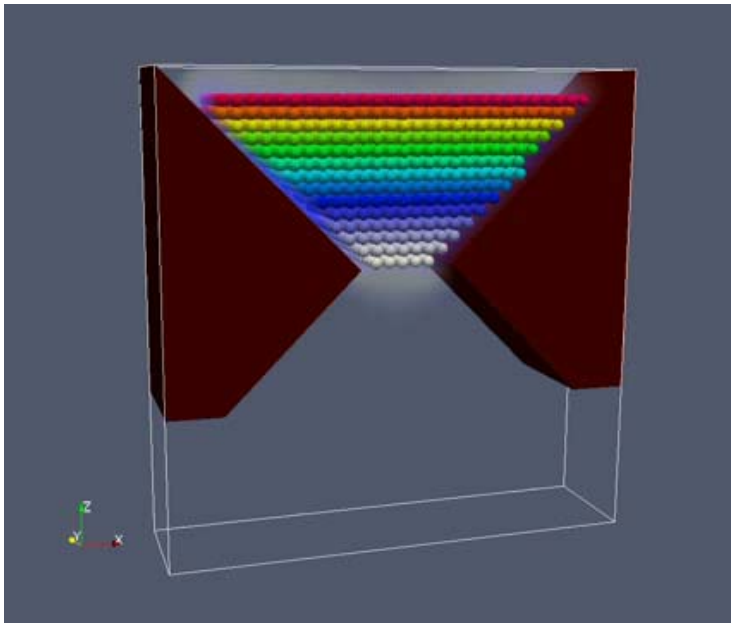
Finite Element-Fictitious Boundary Methods for the Numerical Simulation of Particulate Flows





Stefan Turek, Raphael Münster, Otto Mierka
Institut für Angewandte Mathematik, TU Dortmund
http://www.mathematik.tu-dortmund.de/LS3
http://www.featflow.de

Basic CFD Tool: FeatFlow

technische universität dortmund

HPC features:
- Moderately parallel
- GPU computing
- Open source

Hardware -oriented Numerics

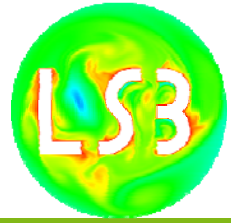
Numerical features:
- Higher order (Q2P1) FEM in space & (semi-) Implicit FD/FEM in time
- Semi-(un)structured meshes with dynamic adaptive grid deformation
- **Fictitious Boundary (FBM) methods**
- Newton-Multigrid-type solvers

Non-Newtonian flow module:
- generalized Newtonian model (Power-law, Carreau,...)
- viscoelastic model (Giesekus, FENE, Oldroyd,...)

Multiphase flow module (resolved interfaces):
- ℓ/ℓ – interface capturing (Level Set)
- s/ℓ – interface tracking (FBM)
- $s/\ell/\ell$ – combination of ℓ/ℓ and s/ℓ

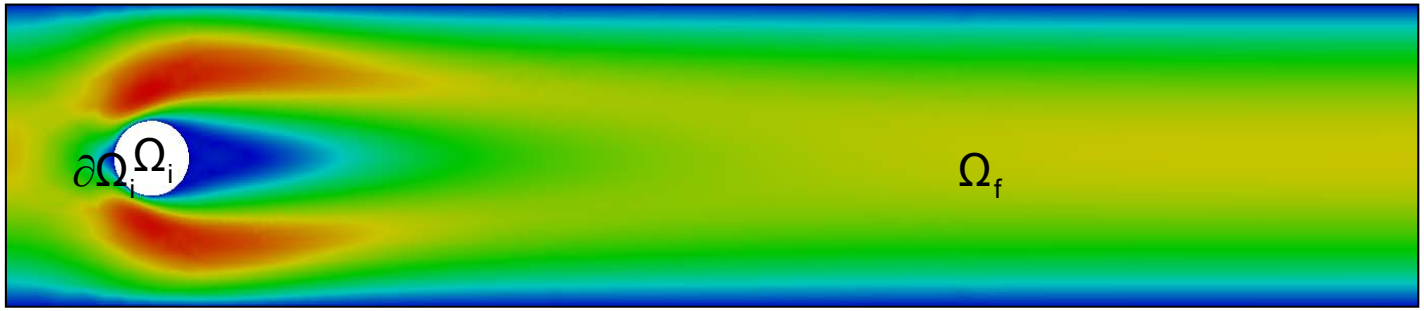
Engineering aspects:
- Geometrical design
- Modulation strategy
- Optimization

Here: FEM-based tools for the accurate simulation of multiphase flow problems, particularly with liquid-(rigid) solid interfaces



Liquid–(Rigid) Solid Interfaces

Consider the flow of N solid particles in a fluid with density ρ and viscosity μ. Denote by $\Omega_f(t)$ the domain occupied by the fluid at time t, by $\Omega_i(t)$ the domain occupied by the ith-particle at time t and let $\overline{\Omega} = \overline{\Omega}_f \cup \overline{\Omega}_i$.



The fluid flow is modelled by the **Navier-Stokes equations**:

$$\rho\left(\frac{\partial u}{\partial t} + u \cdot \nabla u\right) - \nabla \cdot \sigma = f, \quad \nabla \cdot u = 0$$

where σ is the total stress tensor of the fluid phase:

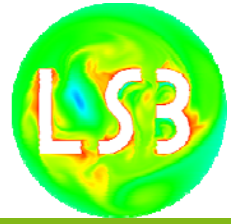
$$\sigma(X,t) = -pI + \mu[\nabla u + (\nabla u)^T]$$

Equations of Motion (I)

The motion of particles can be described by the **Newton-Euler equations**.
A particle moves with **a translational velocity** U_i and **angular velocity** ω_i which satisfiy:

$$M_i \frac{dU_i}{dt} = F_i + F_i' + (\Delta M_i) g, \qquad I_i \frac{d\omega_i}{dt} + \omega_i \times (I_i \omega_i) = T_i,$$

- M_i : mass of the i-th particle (i=1,...,N)
- I_i : moment of inertia tensor of the i-th particle
- ΔM_i : mass difference between M_i and the mass of the fluid
- F_i : hydrodynamic force acting on the i-th particle
- T_i : hydrodynamic torque acting on the i-th particle



Equations of Motion (II)

The position and orientation of the i-th particle are obtained by integrating the **kinematic equations**:

$$\frac{dX_i}{dt} = U_i, \quad \frac{d\theta_i}{dt} = \omega_i, \quad \frac{d\omega_i}{dt} = I_i^{-1} T_i$$

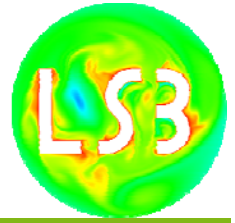
which can be done numerically by an explicit Euler scheme:

$$X_i^{n+1} = X_i^n + \Delta t U_i^n \qquad \omega_i^{n+1} = \omega_i^n + \Delta t \left(I_i^{-1} T_i^n\right) \qquad \theta_i^{n+1} = \theta_i^n + \Delta t \omega_i^n$$

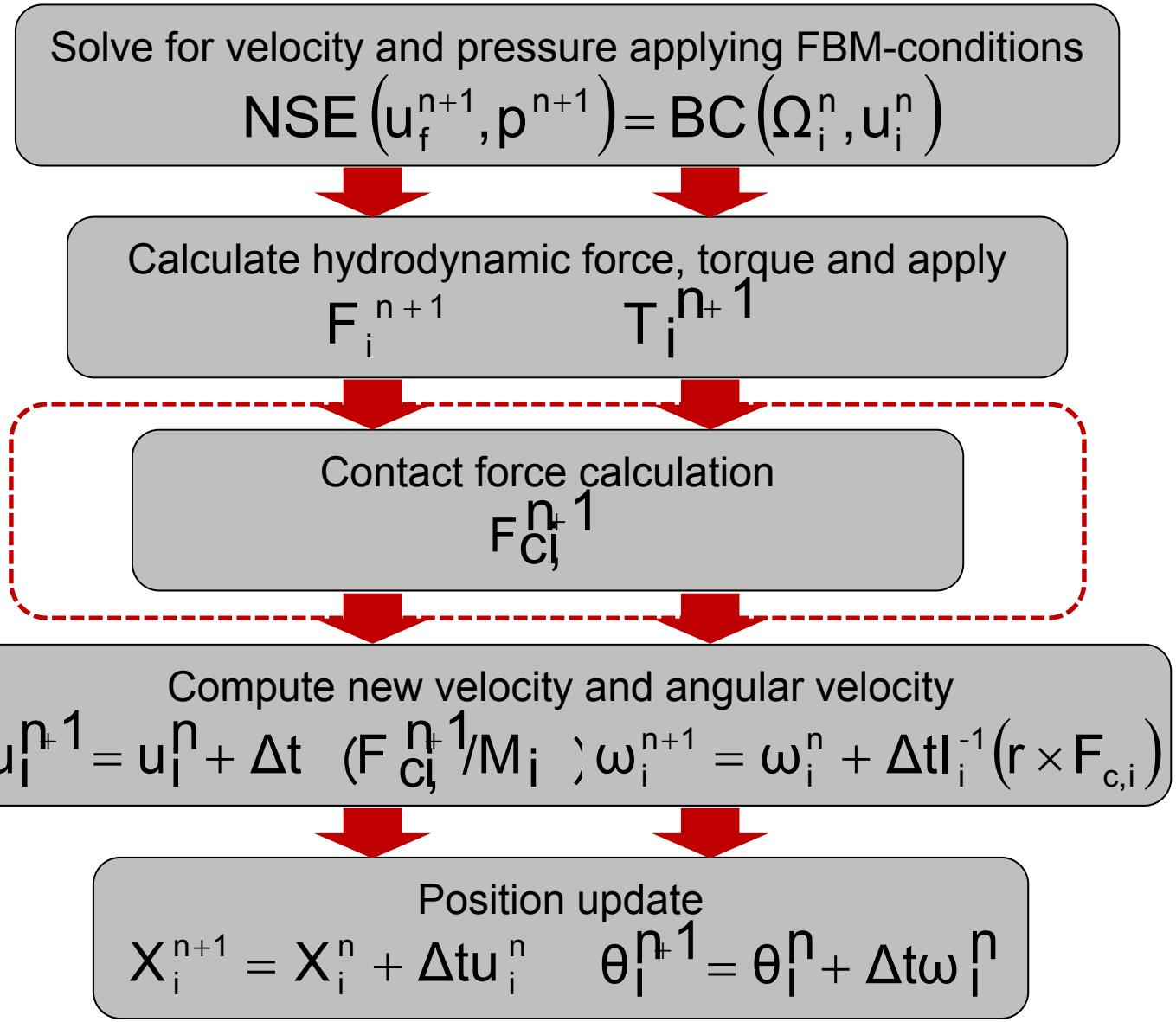
Boundary Conditions

We apply the velocity u(X) as no-slip boundary condition at the interface $\partial\Omega_i$ between the i-th particle and the fluid, which for $X \in \Omega_i$ is defined by:

$$u(X) = U_i + \omega_i \times (X - X_i)$$



Numerical Solution Scheme

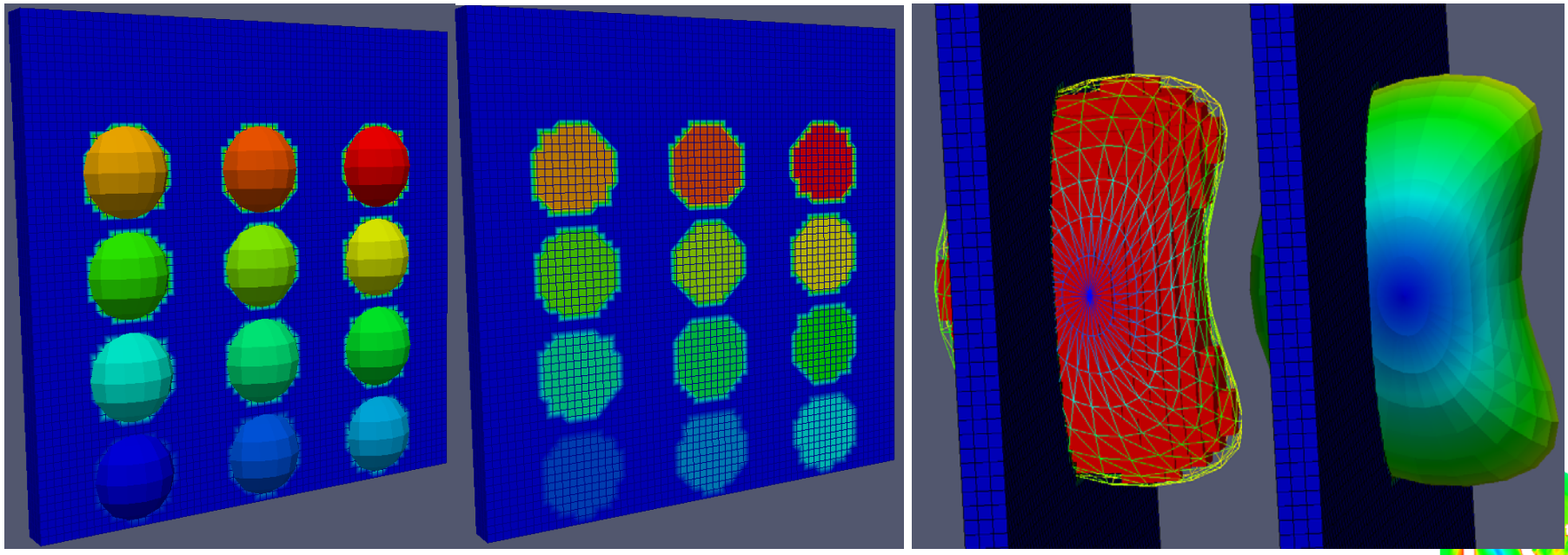


Fictitious Boundary Method

technische universität dortmund

Eulerian Approach:

- FBM = special case of (scaled) Penalty method
- **Objects are represented as a boolean (in/out) function on the mesh**
- Complex shapes are possible (**surface triangulation, implicit functions**)
- Use of a fixed mesh possible → only first order accurate
- **But:** Higher accuracy possible by (local) **mesh adaptation** techniques



Dynamic ALE-Mesh Adaptation

technische universität dortmund

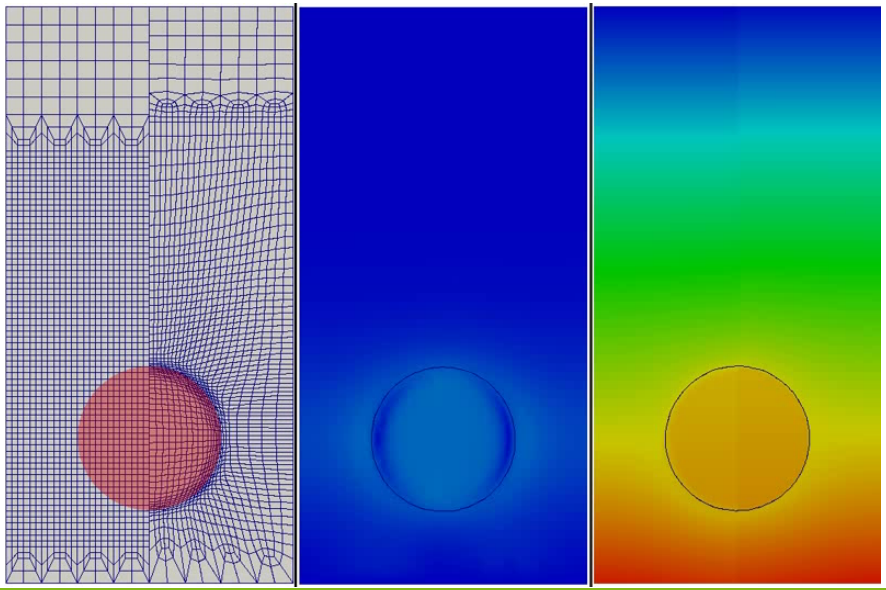
Advantages:

- Constant mesh/data structure → **GPU**
- Increased resolution in regions of interest ("r-adaptivity")
- Anisotropic 'umbrella' smoother (with snapping/projection) or GDM
- Straightforward usage on general meshes in 2D / 3D

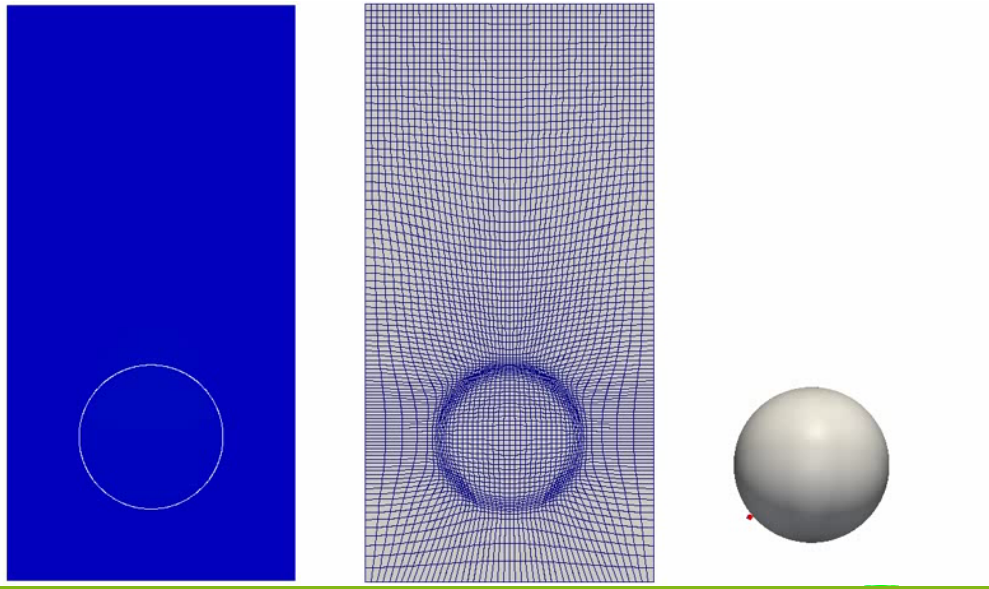
Quality of the method depends on the construction of the monitor function

- Geometrical description (solid body, interface triangulation)
- Field oriented description (steep gradients, fronts) → numerical stabilization

Validation: 2.5D Rising bubble – light setup



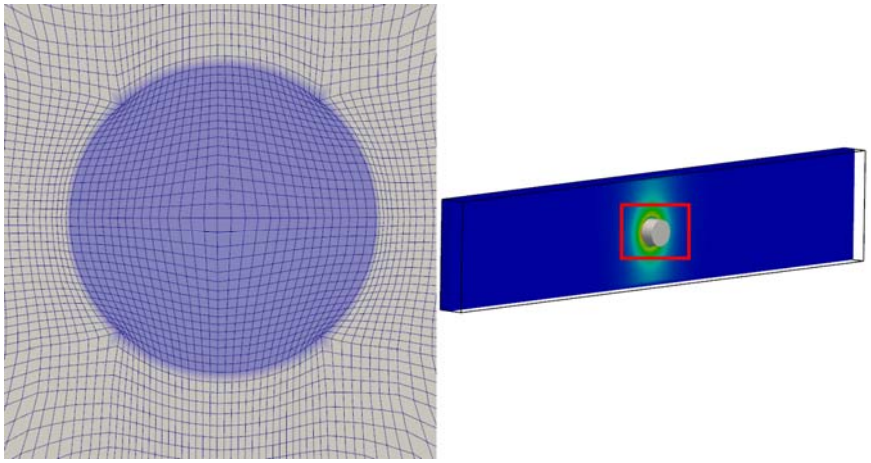
Testing: 3D Rising bubble - hard setup

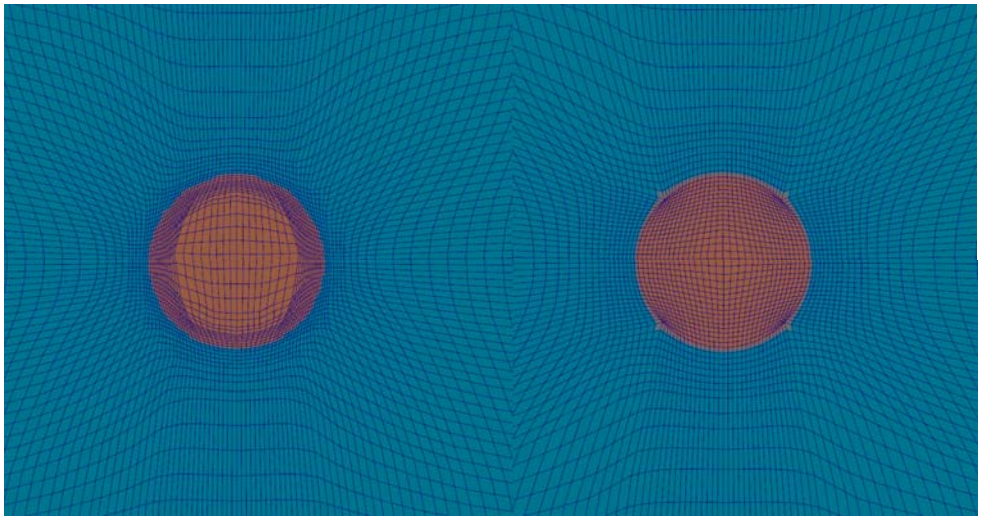


Test: Oscillating Cylinder

technische universität dortmund

- Measure Drag/Lift Coefficients for a sinusoidally oscillating cylinder
- Compare results for FBM, adapted FBM and adapted FBM + boundary projection/parametrization



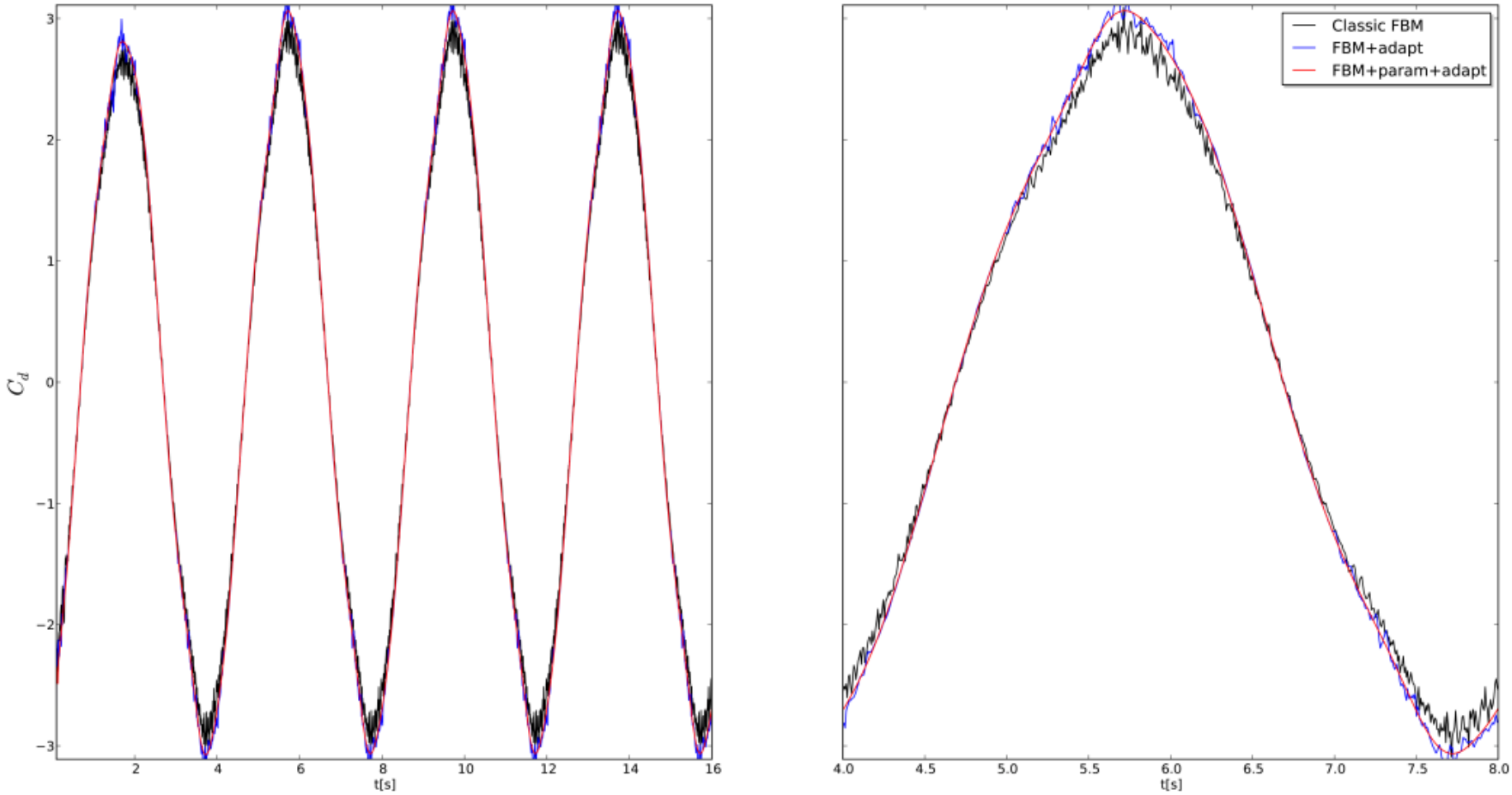


Nodes concentrated near liquid-solid interface

Nodes projected and parametrized on boundary plus concentration of nodes near boundary

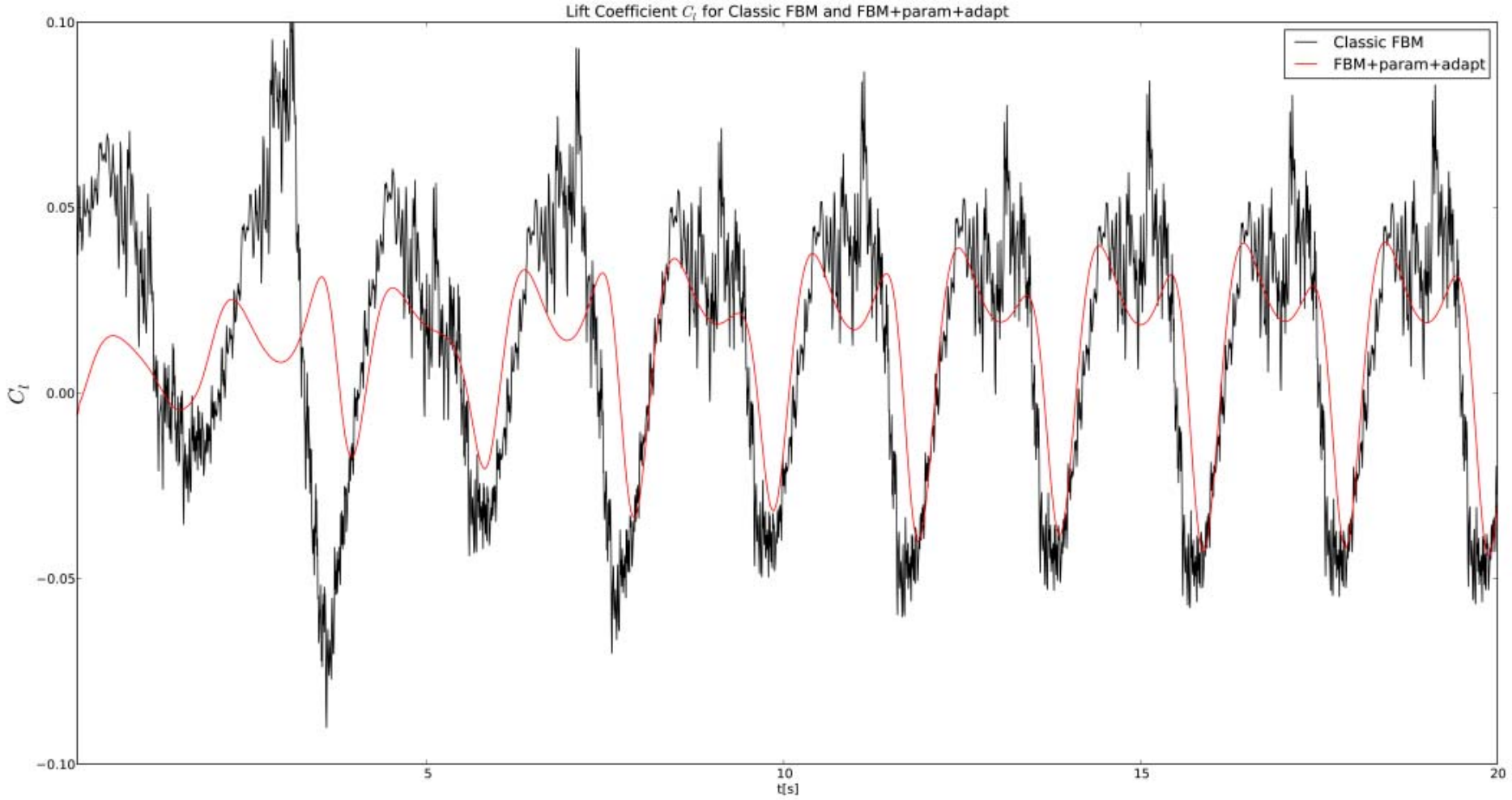
Oscillating Cylinder Results

Drag Coefficient C_d for Classic FBM, FBM+adapt, FBM+param+adapt



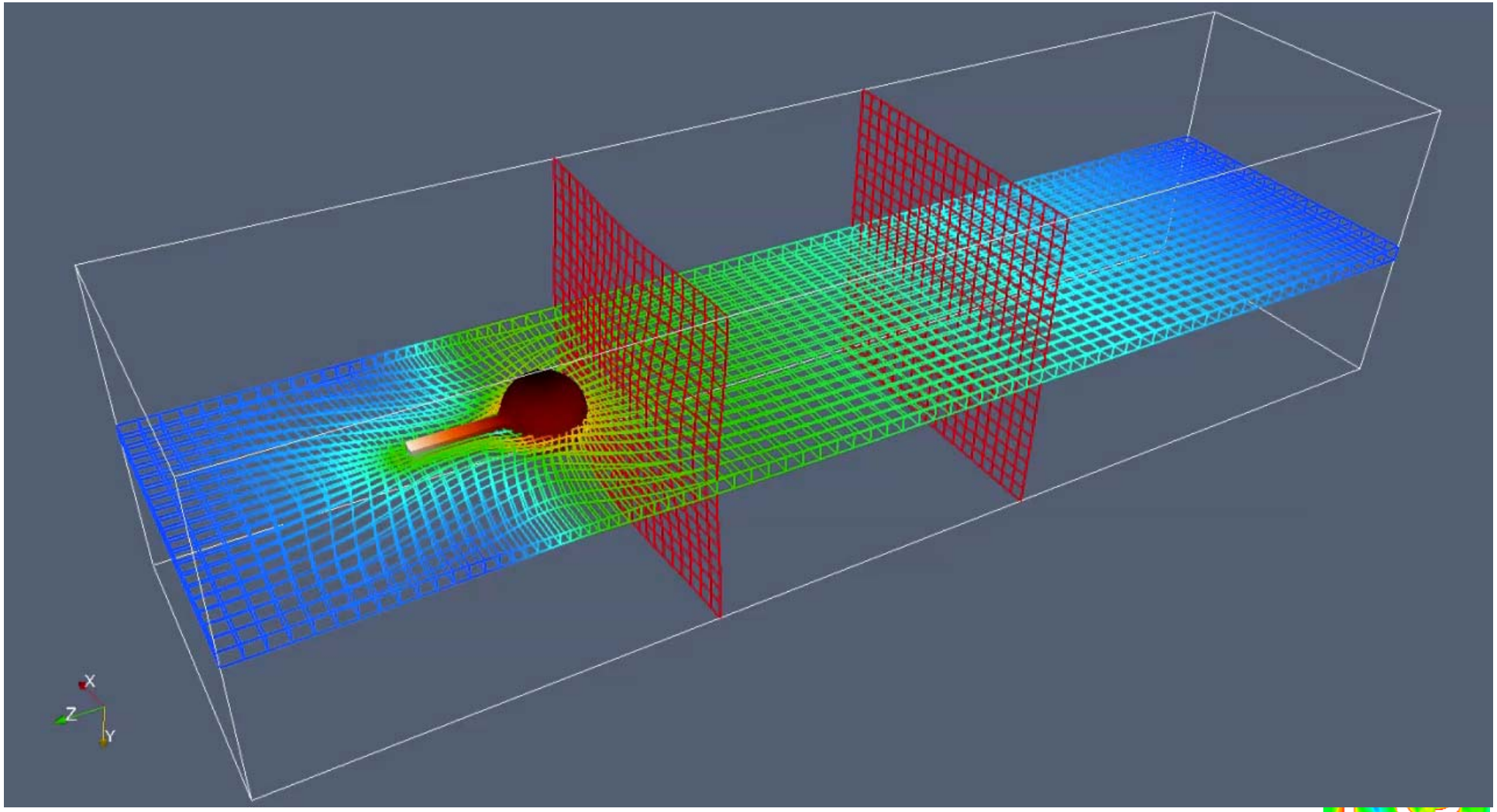
Oscillating Cylinder Results

technische universität dortmund



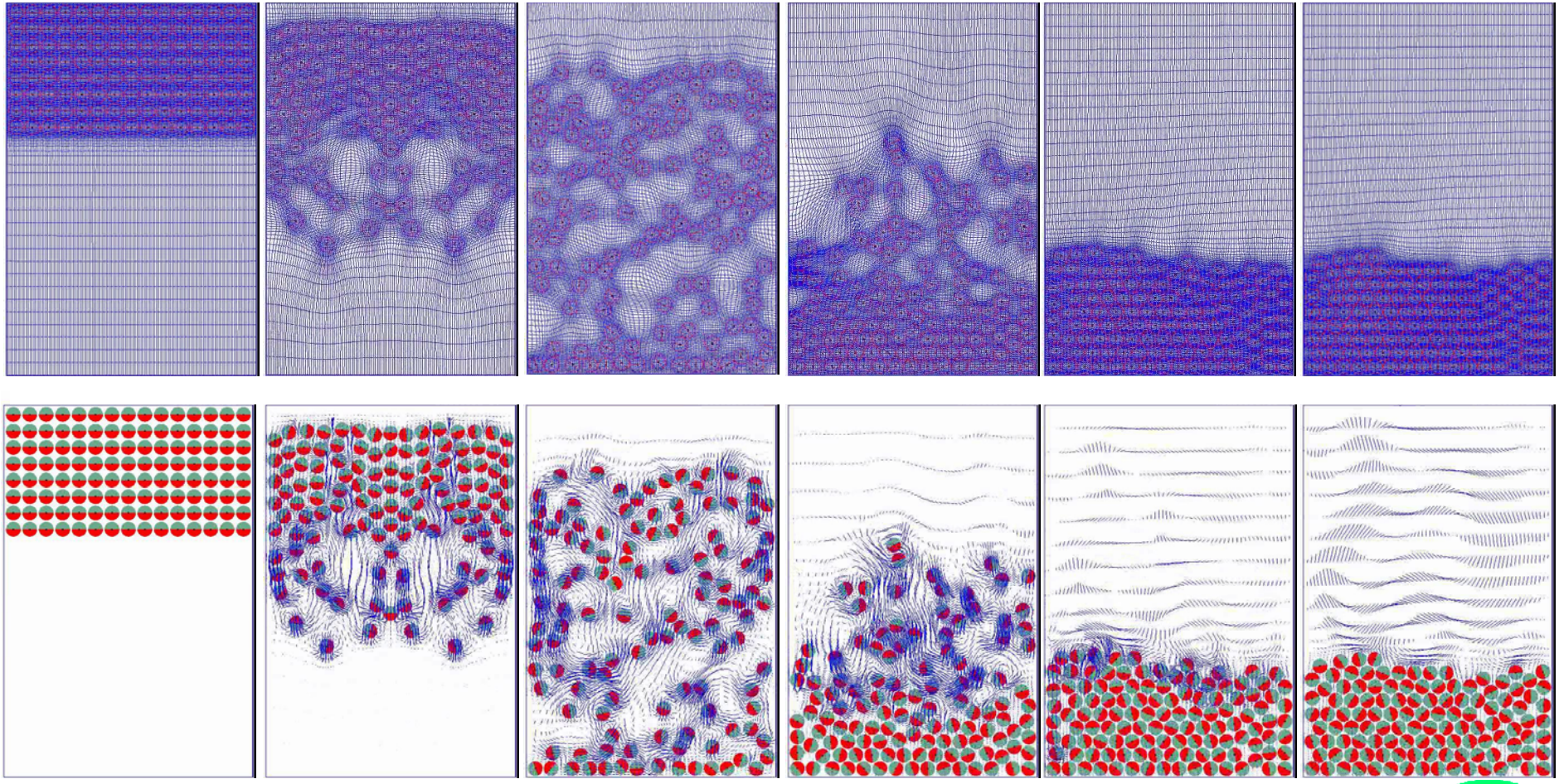
(Passive) Microswimmer

technische universität dortmund



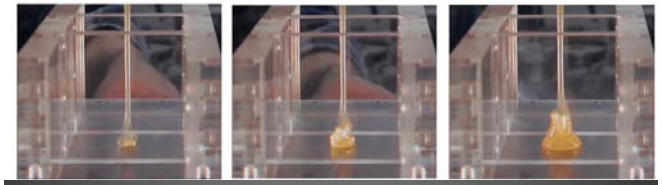
Sedimentation

technische universität dortmund



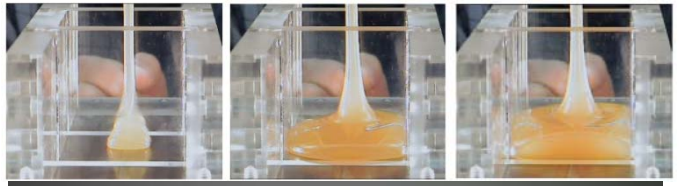
Viscous Liquid Jets

J. M. Nóbrega et al.: The phenomenon of jet buckling: Experimental and numerical predictions

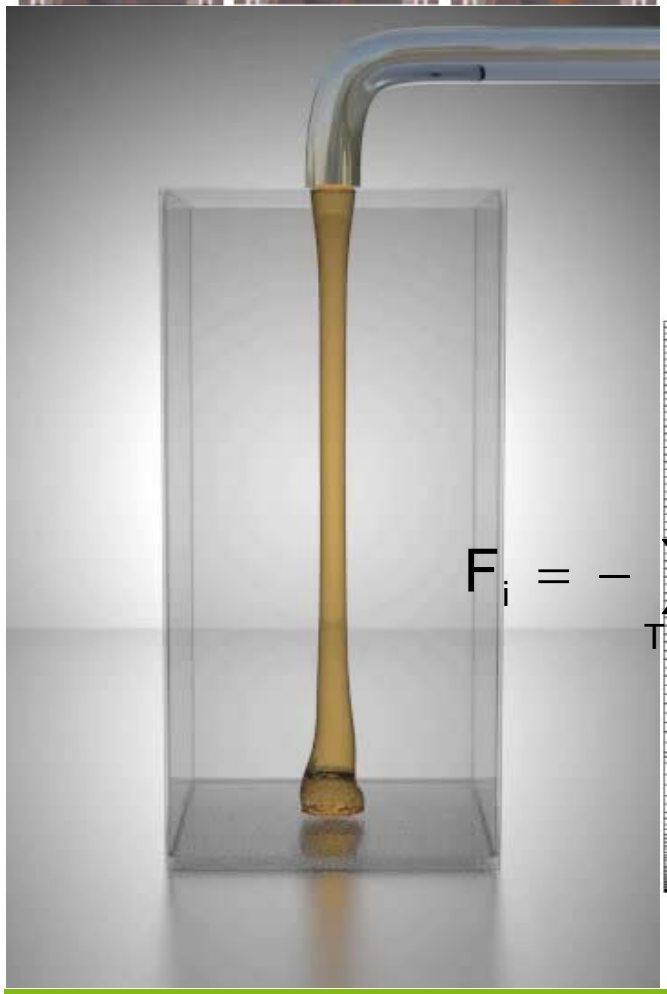


Corn syrup-air system

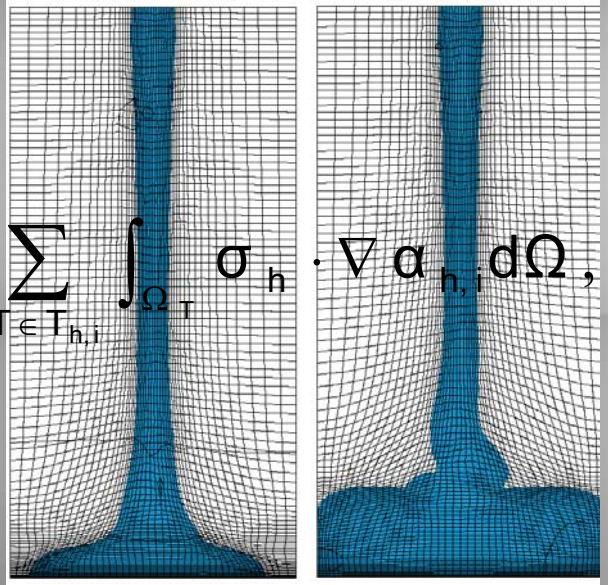
24x24x48 mesh



Interface triangulation:
T_0: ~100,000 triangles
T_N: ~300,000 triangles

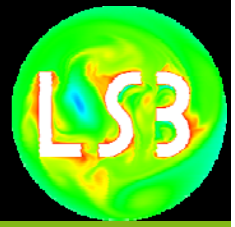


$$F_i = -\sum_{T \in T_{h,i}} \int_{\Omega_T} \sigma_h \cdot \nabla \alpha_{h,i} \, d\Omega,$$



Rendering: Raphael Münster / Blender

How to calculate the hydrodynamic forces???



technische universität dortmund

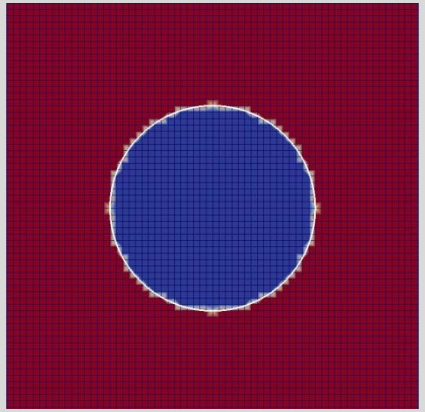
Hydrodynamic force and torque acting on the i-th particle

$$F_i = -\int_{\partial\Omega_i} \sigma \cdot n_i \, d\Gamma_i, \qquad T_i = -\int_{\partial\Omega_i} (X - X_i) \times (\sigma \cdot n_i) \, d\Gamma$$

Force Calculation with Fictitious Boundary Method

Problems: The FBM can only decide:

- `INSIDE`(1) and `OUTSIDE`(0)
- No description of the surface



Alternative:

Replace the surface integral by a volume integral

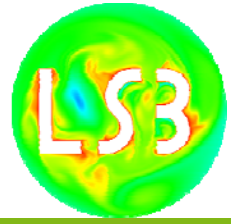
Numerical Force Evaluation (I)

Define an ***indicator function*** α_i:

$$\alpha_i(X) = \begin{cases} 1 & \text{for } X \in \Omega_i \\ 0 & \text{for } X \in \Omega_f \end{cases}$$

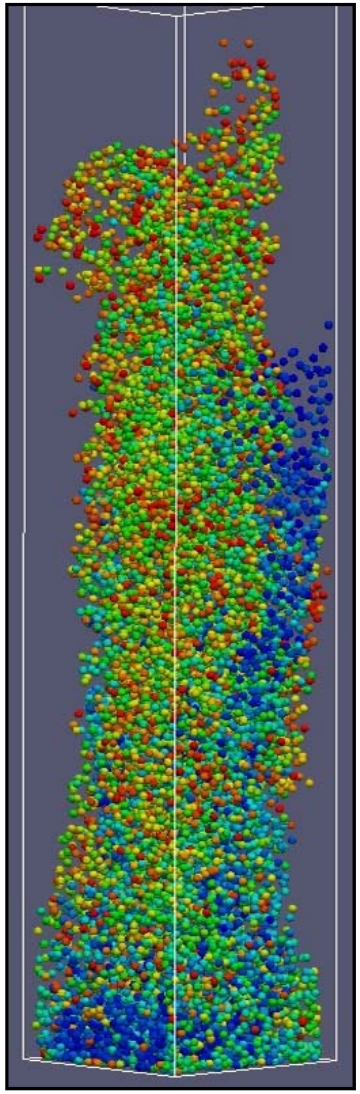
Remark: The gradient of α_i is zero everywhere except at the surface of the i-th Particle and approximates the normal vector (in a weak sense), allowing us to write:

$$F_i = -\int_{\Omega_T} \sigma \cdot \nabla \alpha_i \, d\Omega \;, \quad T_i = -\int_{\Omega_T} (X - X_i) \times (\sigma \cdot \nabla \alpha_i) \, d\Omega$$



Numerical Force Evaluation (II)

Large-scale FBM-Simulations



Integration over Ω_T *too* expensive:

- Gradient is non-zero on $\partial\Omega_i$
- Information available from FBM
- Evaluate boundary cells only
- Visit each cell only once

$$F_i = -\sum_{T \in T_{h,i}} \int_{\Omega_T} \sigma_h \cdot \nabla \alpha_{hj} d\Omega ,$$

$$T_i = -\sum_{T \in T_{h,i}} \int_{\Omega_T} (x - x_i) \times (\sigma_h \cdot \nabla \alpha_{h,i}) d\Omega$$

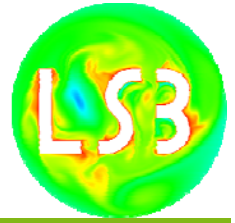
$\alpha_{h,i}(x)$: finite element interpolant of $\alpha(x)$

$T_{h,i}$: elements intersected by i-th particle

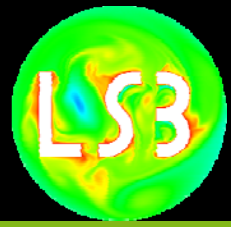
Contact/Collision Modelling

- **Contact determination** for rigid bodies A and B:
  - → Distance d(A,B)
  - → Relative velocity $v_{AB} = (v_A + \omega_A \times r_A - (v_B + \omega_B \times r_B))$
  - → Collision normal $N = (X_A(t) - X_B(t))$
  - → Relative normal velocity $N \cdot (v_A + \omega_A \times r_A - (v_B + \omega_B \times r_B))$
- **Contact force calculation** realized as a three step process
  - → Broad phase
  - → Narrow phase
  - → Contact/Collision force calculation

  ▶ Several single, resp., multibody collision models are realized and tested, but…..



How to validate???

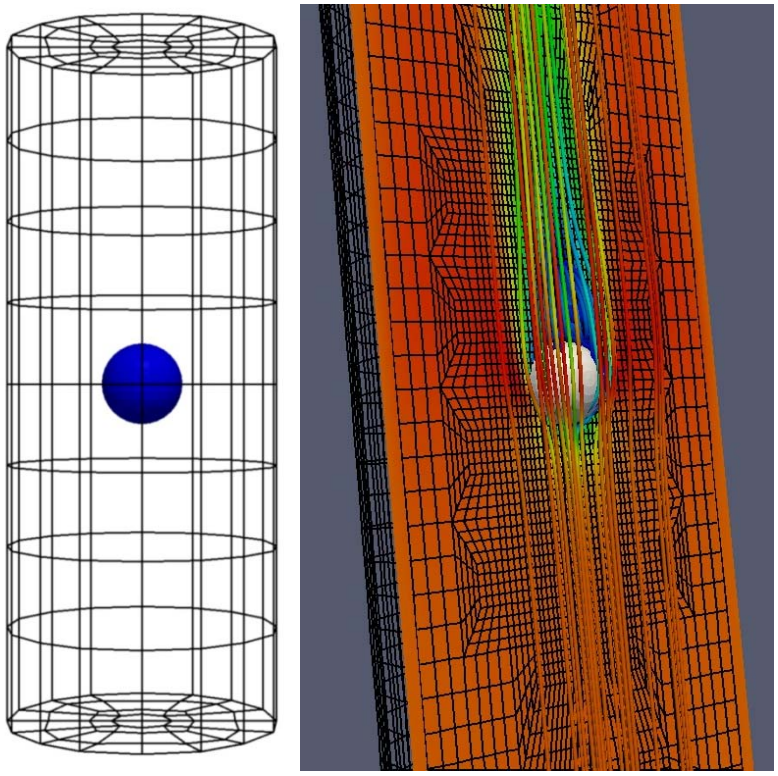


Benchmarking and Validation (I)

technische universität dortmund

Free fall of particles:

- Terminal velocity
- Different physical parameters
- Different geometrical parameters



Münster, R.; Mierka, O.; Turek, S.: Finite Element fictitious boundary methods (FEM-FBM) for 3D particulate flow, IJNMF, 2011

$d_s = 0.3, \quad \rho_s = 1.14$

ν	$U_{featflow}$	U_{exp}	Relative error (%)
0.02	5.885	6.283	6.33
0.05	4.133	3.972	4.05
0.1	2.588	2.426	6.66
0.2	1.492	1.401	6.50

$d_s = 0.2, \quad \rho_s = 1.14$

ν	$U_{featflow}$	U_{exp}	Relative error (%)
0.02	4.370	4.334	0.83
0.05	2.699	2.489	8.44
0.1	1.649	1.552	6.25
0.2	0.946	0.870	8.74

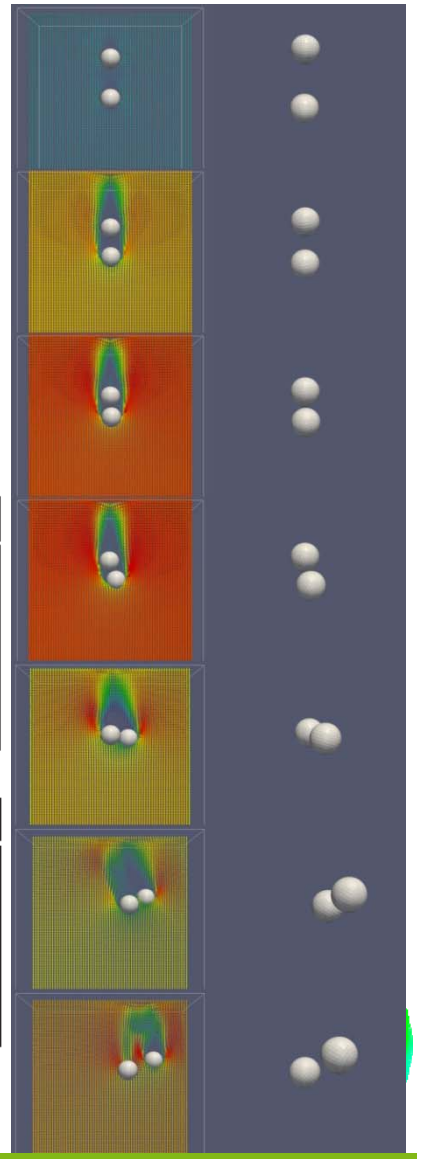
$d_s = 0.3, \quad \rho_s = 1.02$

ν	$U_{featflow}$	U_{exp}	Relative error (%)
0.01	2.167	2.107	2.84
0.02	1.495	1.436	4.11
0.05	0.809	0.749	8.01
0.1	0.402	0.404	0.44
0.2	0.218	0.216	1.02

$d_s = 0.2, \quad \rho_s = 1.02$

ν	$U_{featflow}$	U_{exp}	Relative error (%)
0.01	1.4660	1.4110	3.90
0.02	0.9998	0.9129	9.52
0.05	0.4917	0.4603	6.82
0.1	0.2637	0.2571	2.57
0.2	0.1335	0.1317	1.37

Source: Glowinski et al. 2001

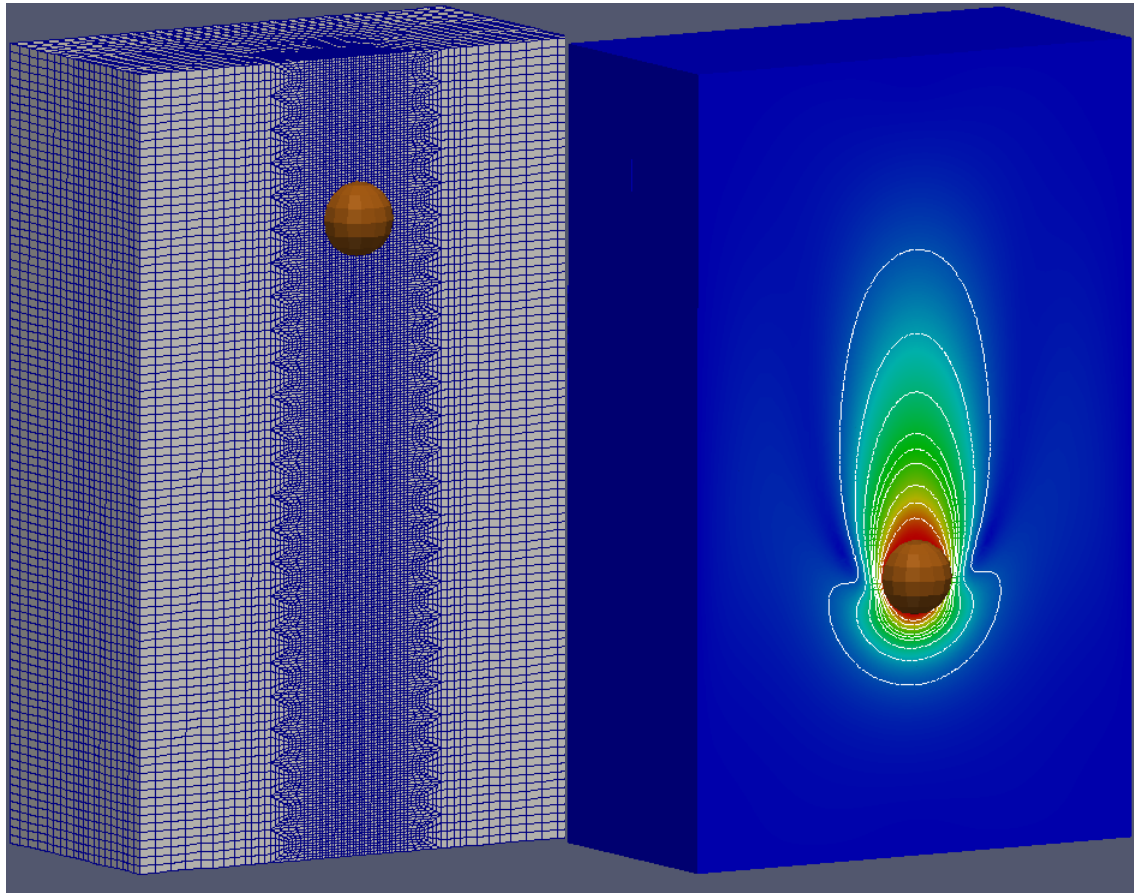


Benchmarking and Validation (II)

technische universität dortmund

Settling of a sphere towards a plane wall:

- Sedimentation Velocity
- Particle trajectory
- Kinetic Energy
- Different Reynolds numbers



Setup

Computational mesh:

- 1.075.200 vertices
- 622.592 hexahedral cells
- Q2/P1:

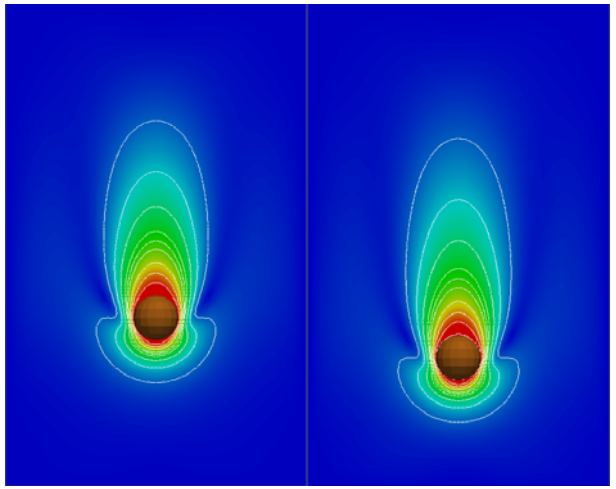
  → 50.429.952 DoFs

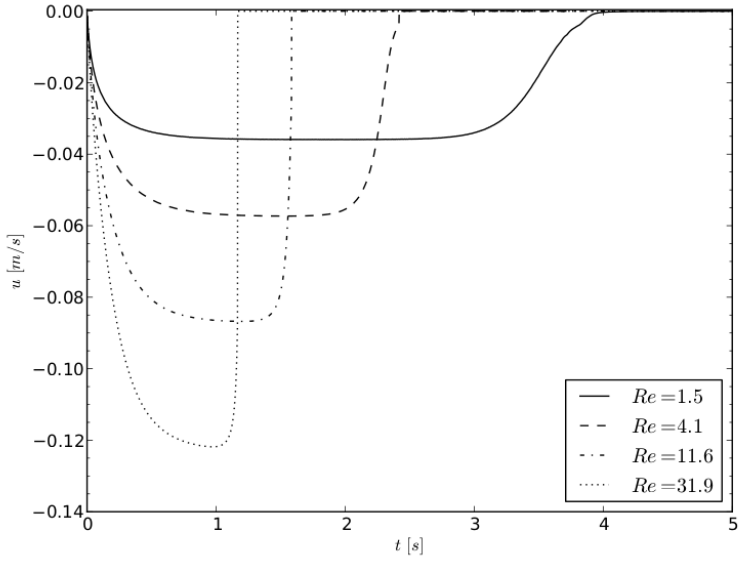
Sedimentation Benchmark (I)

technische universität dortmund

Observations

- Velocity profiles compare well to ten Cate‘s data
- Maximum velocity close to experiment
- Flow features are accurately resolved

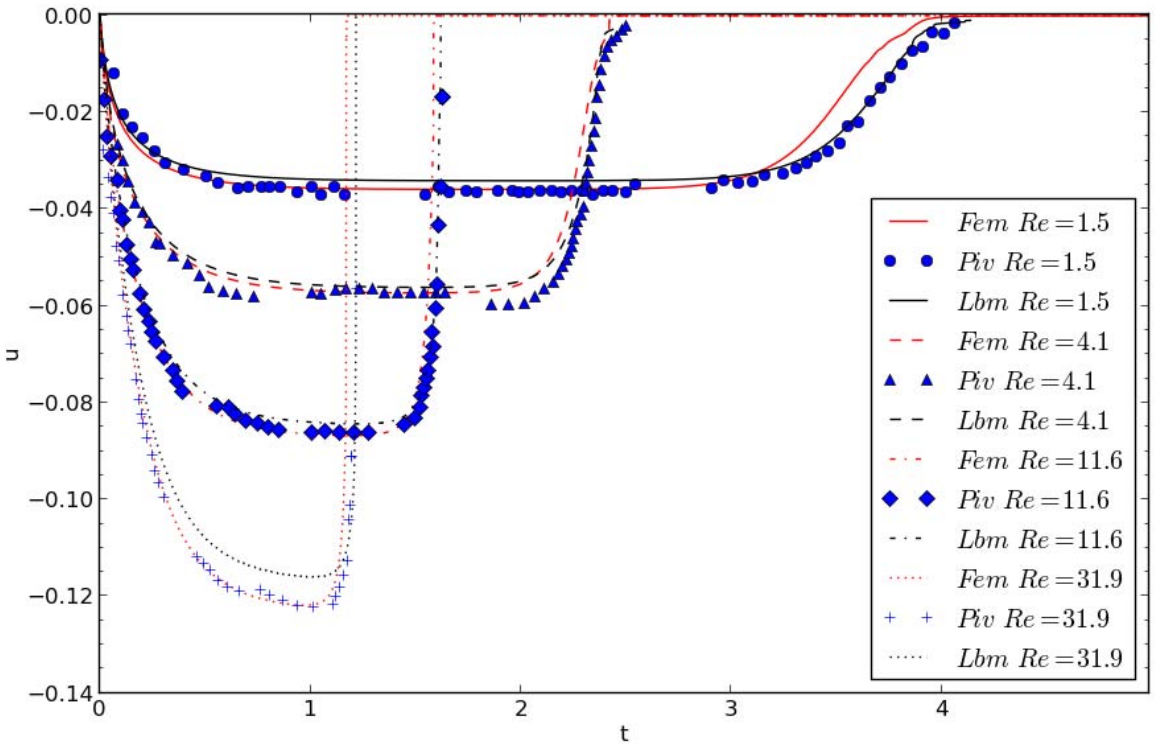




Re	u_{max}/u_∞	u_{max}/u_∞ *ten Cate*	u_{max}/u_∞ *exp*
1.5	0.945	0.894	0.947
4.1	0.955	0.950	0.953
11.6	0.953	0.955	0.959
31.9	0.951	0.947	0.955

Tab. 1 Comparison of the u_{max}/u_∞ ratios between the FEM-FBM, ten Cate's simulation and ten Cate's experiment

Sedimentation Benchmark (II)



Comparison of FEM-FBM and the experimental values and the LBM results of the group of Sommerfeld

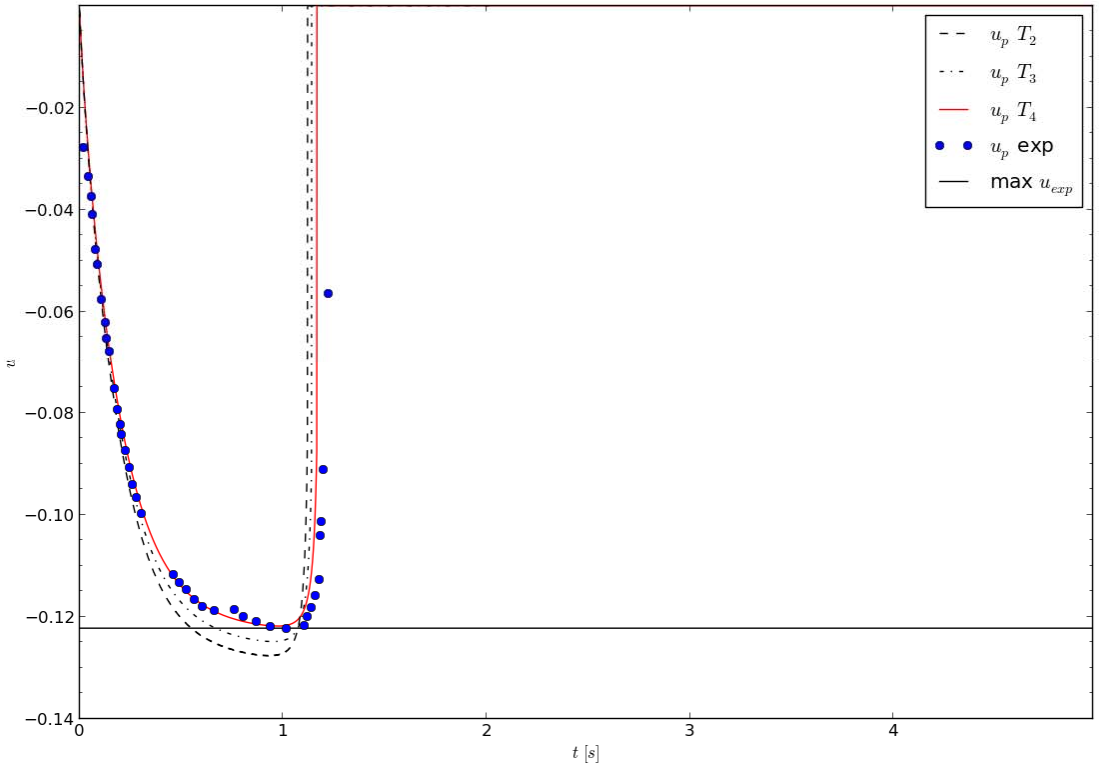
Source: 13th Workshop on Two-Phase Flow Predictions 2012
Acknowledgements: Ernst, M., Dietzel, M., Sommerfeld, M.

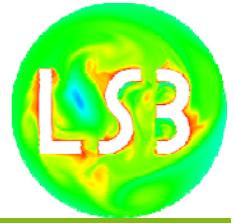
Sedimentation Benchmark (III)

technische universität dortmund

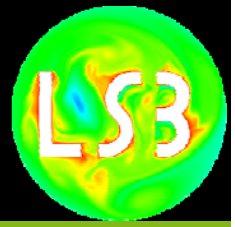
FEM-Multigrid Framework

- *Increasing the mesh resolution produces more accurate results*
  Test performed at different mesh levels
  - Maximum velocity is approximated better ✓
  - Shape of the velocity curve matches better ✓



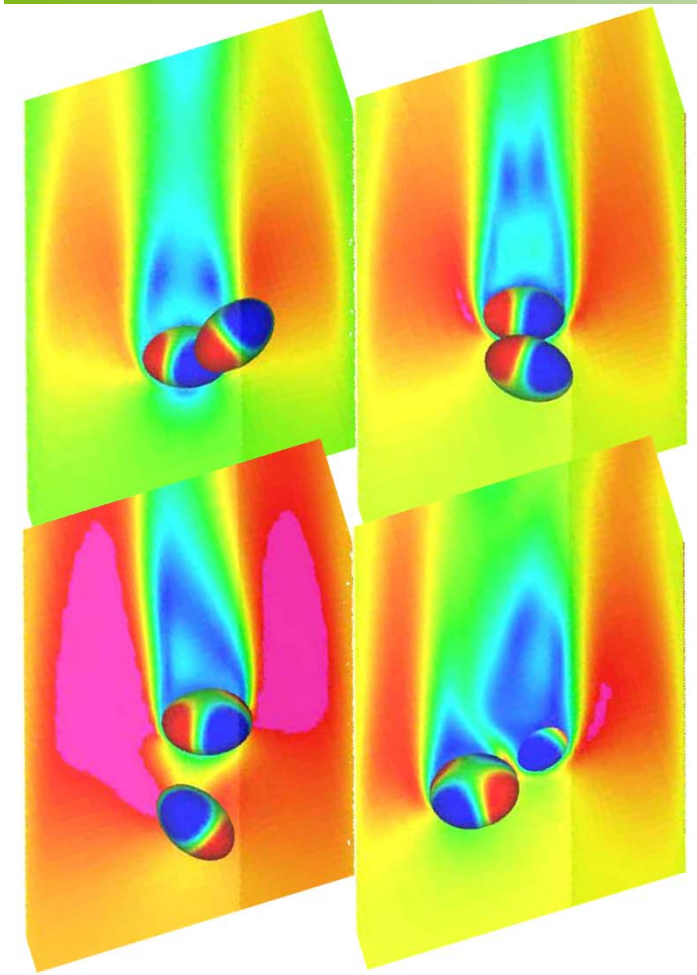


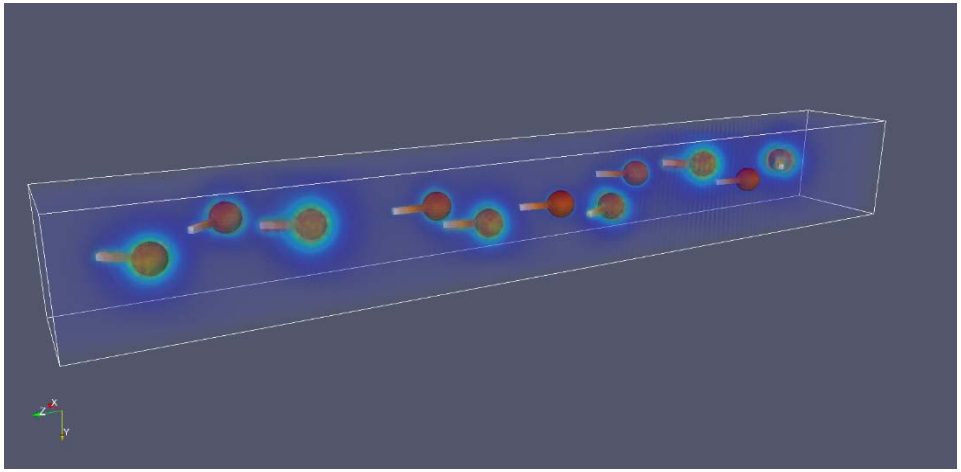
Now let‘s get it „colorful“…

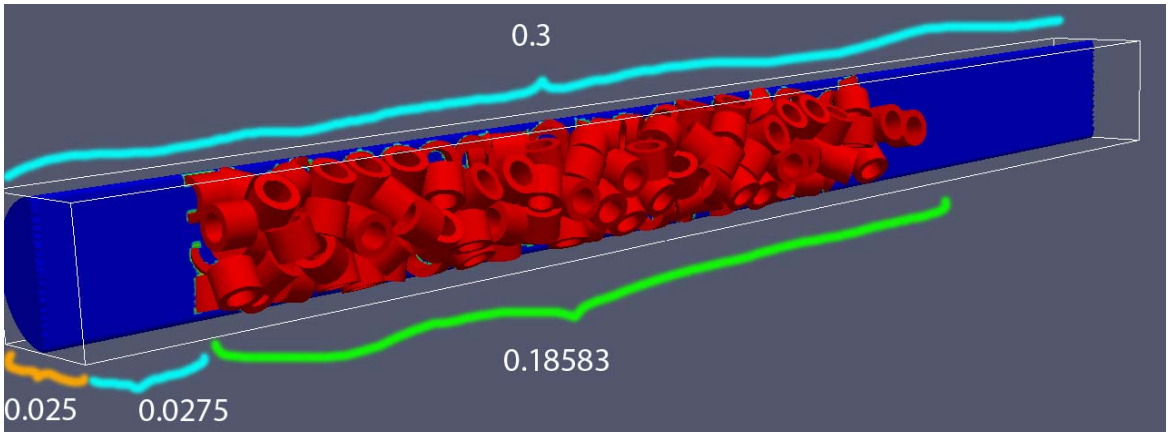


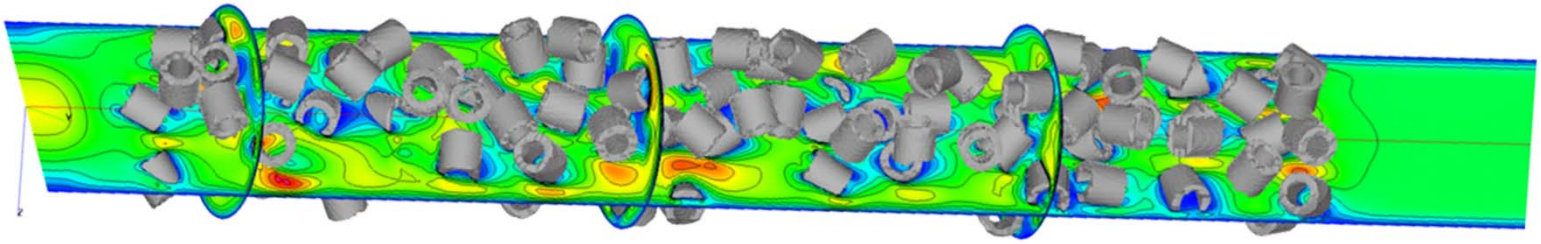
Some More **Complex** *Examples*

technische universität dortmund



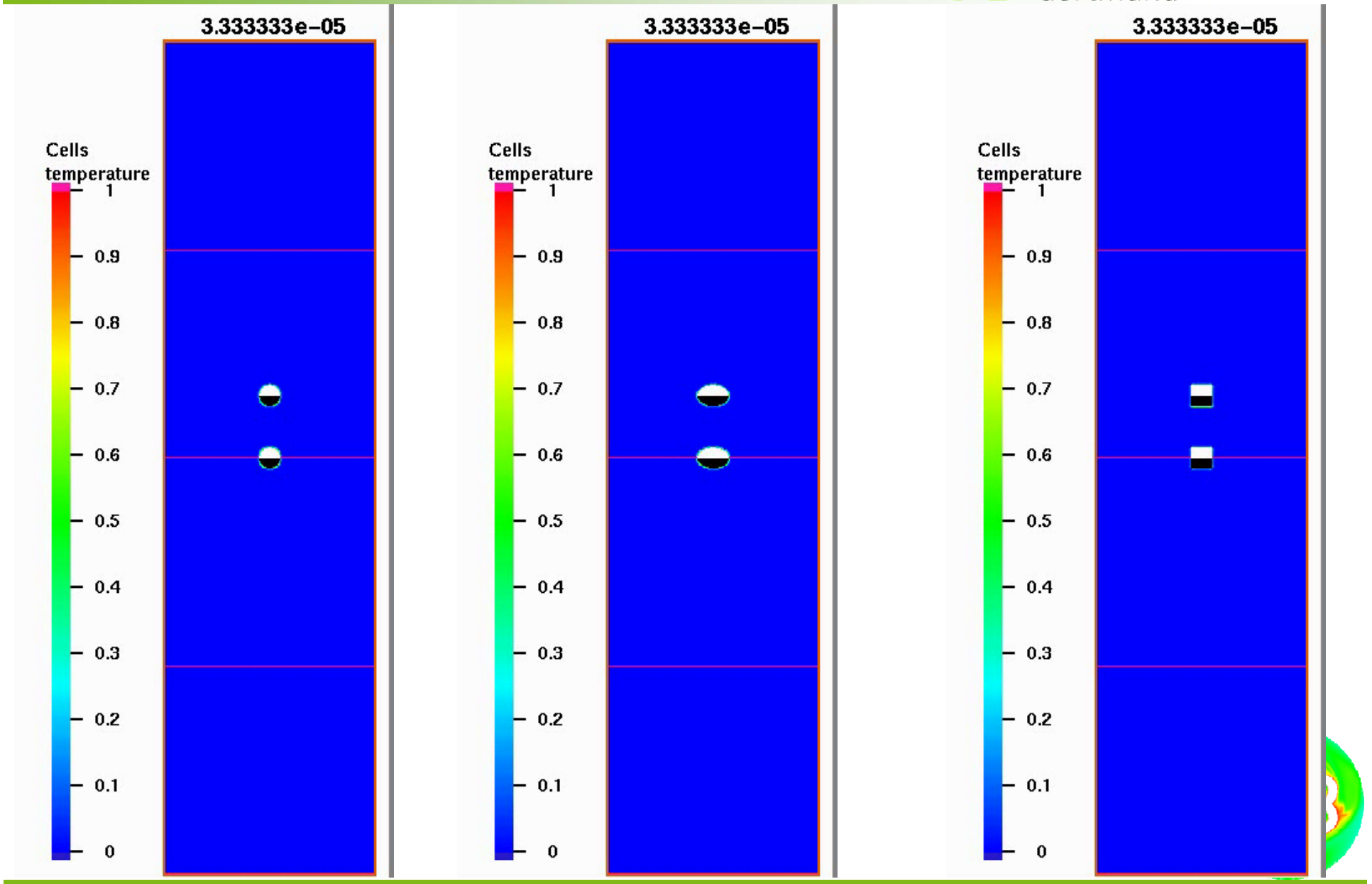






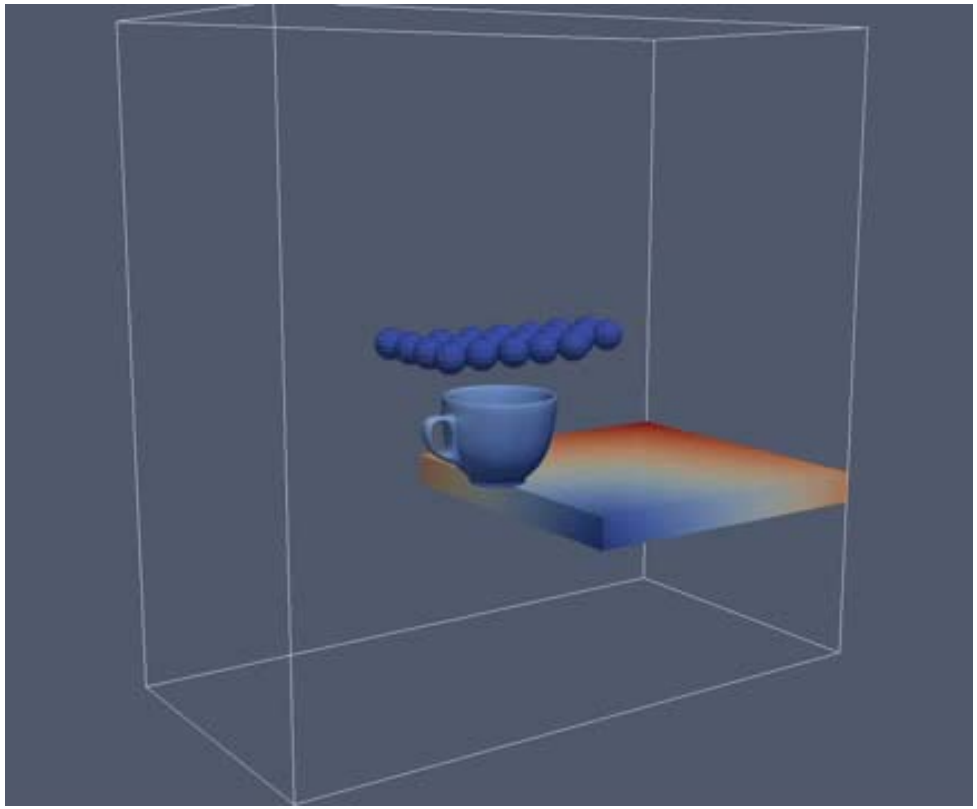
‘Kissing, Drafting, Thumbling’

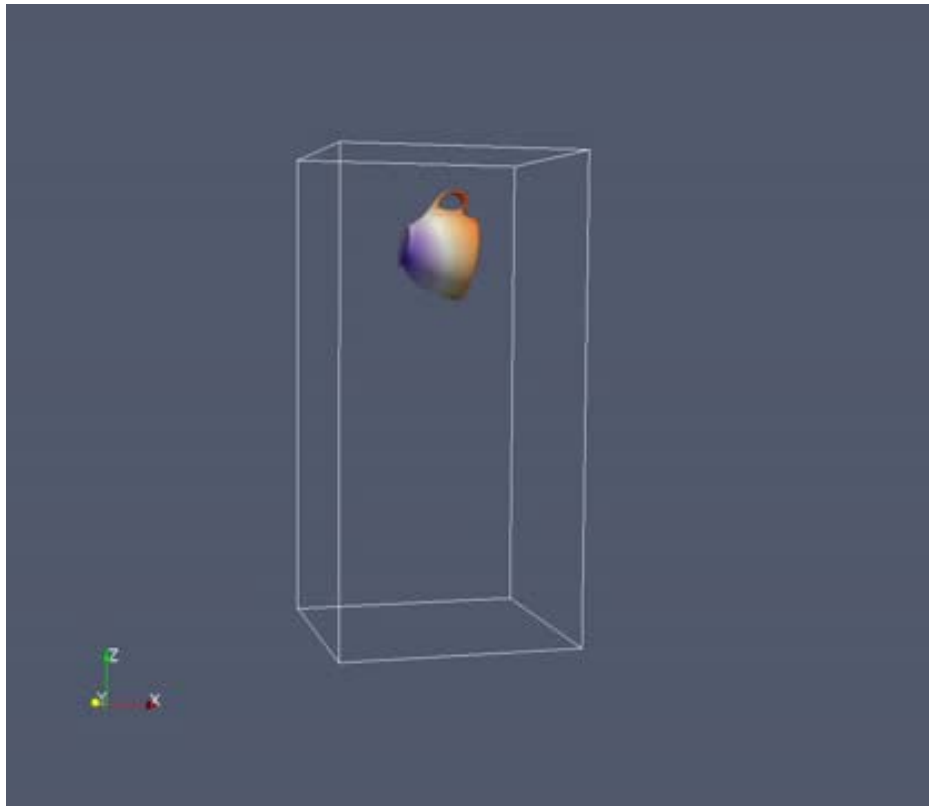
technische universität dortmund



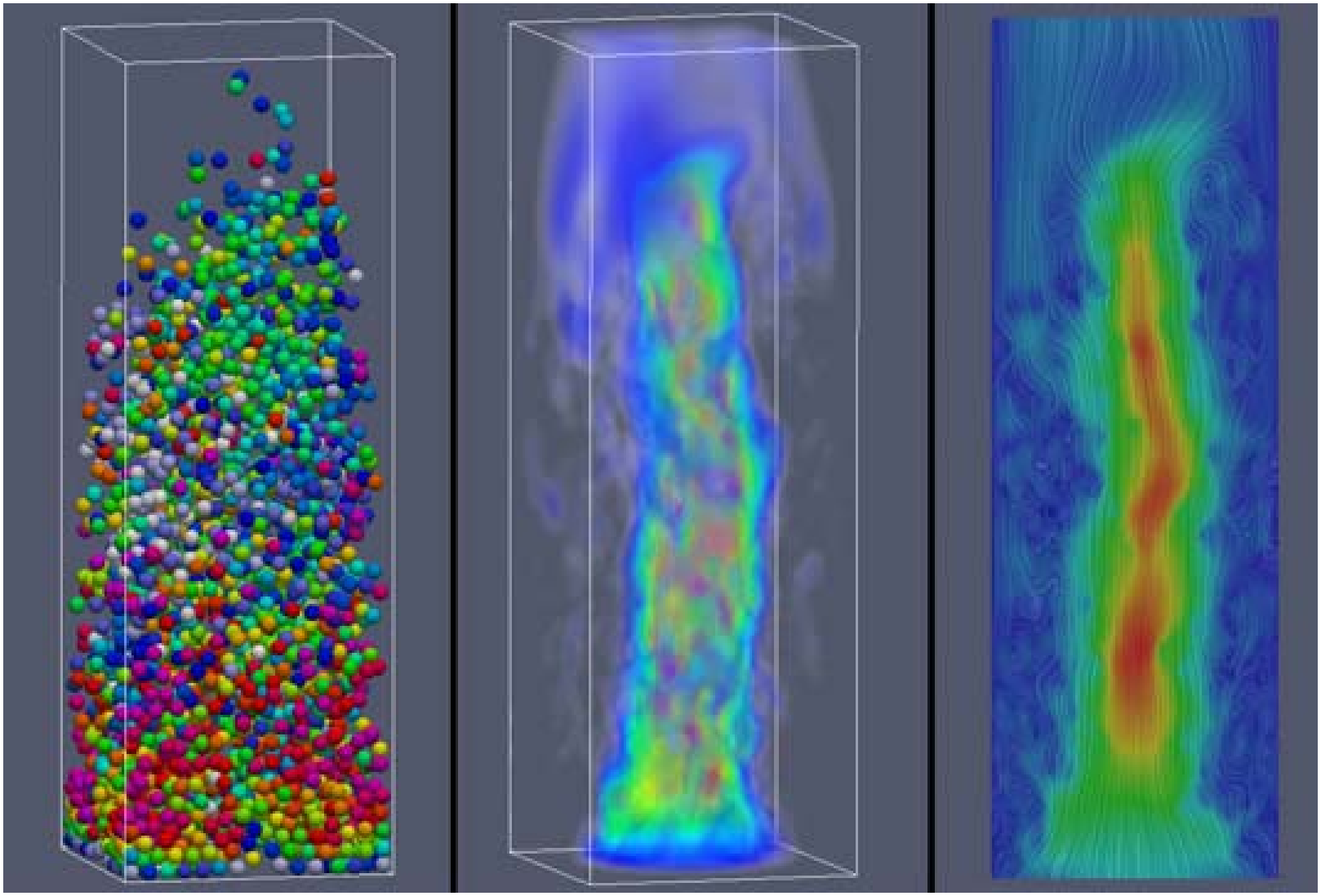
(More) Complex Geometry Examples

technische universität dortmund

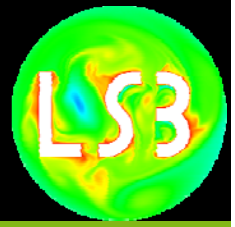




Fluidized Bed Example



…and finally some applications of FBM…

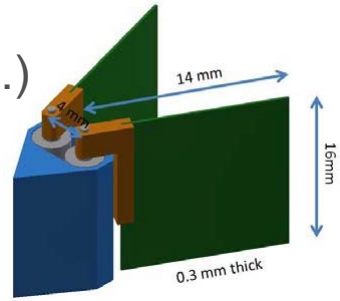


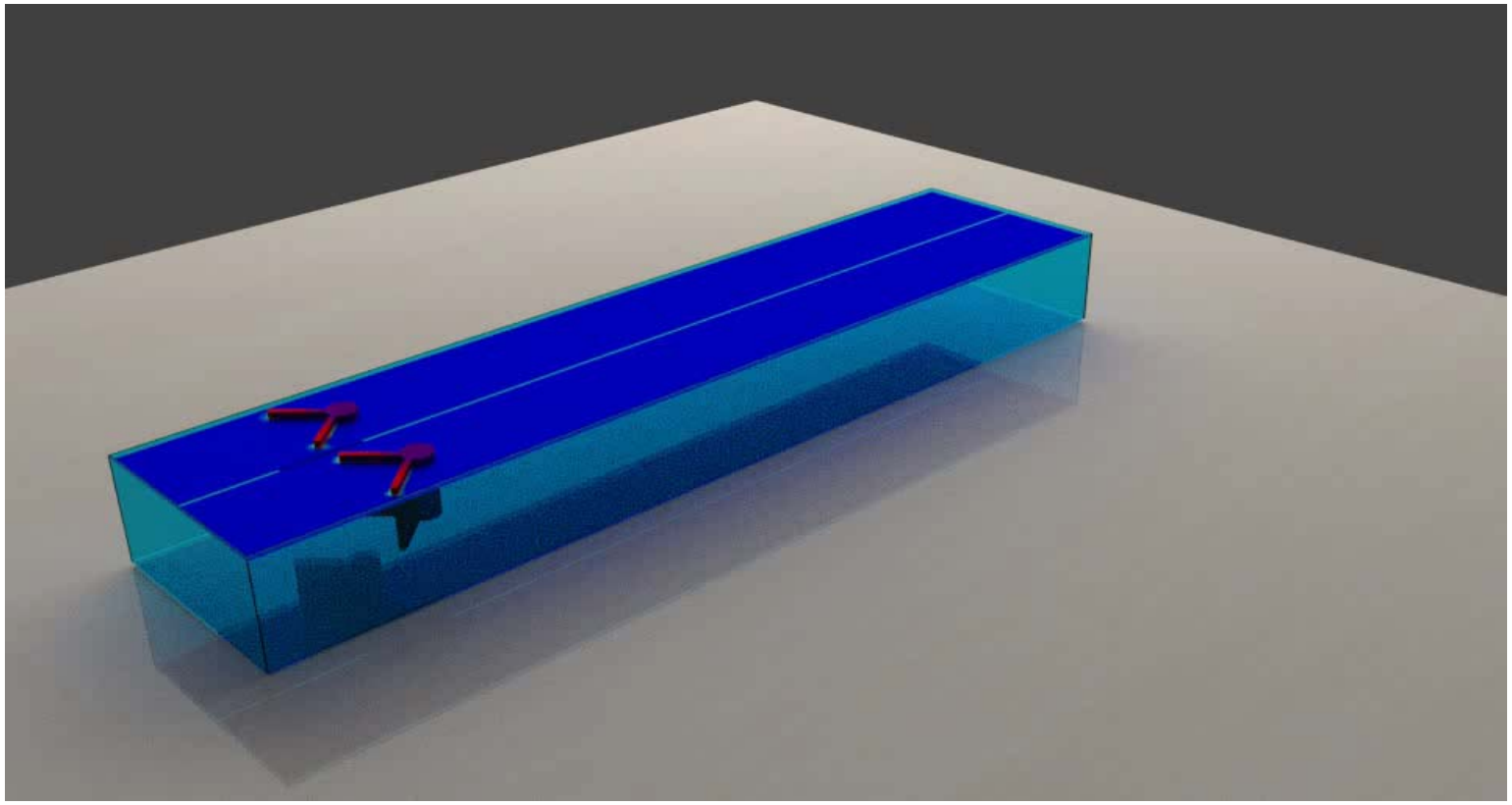
Example: (Active) Microswimmer

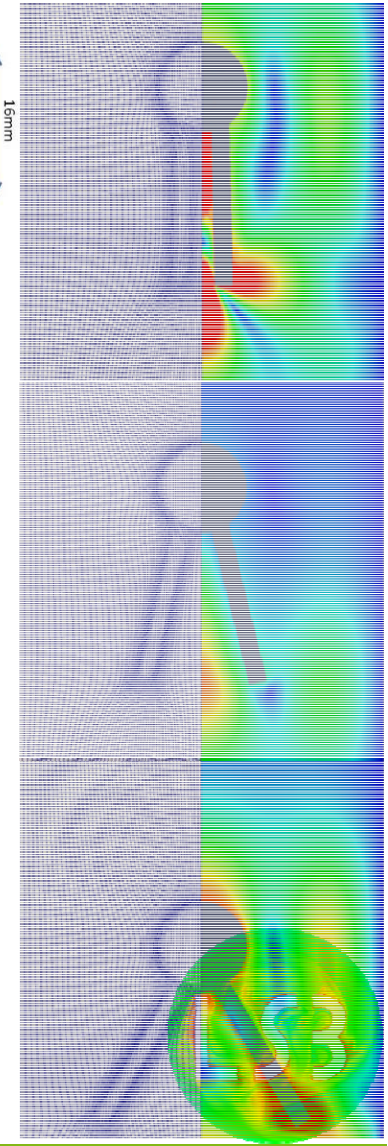
technische universität dortmund

Application to microswimmers (published: Nature Comm.)

- Exp.: Cooperation with AG Fischer (MPI IS Stuttgart)
- Analysis with respect to shear thickening/thinning
- Use of grid deformation to resolve *s/l* interface



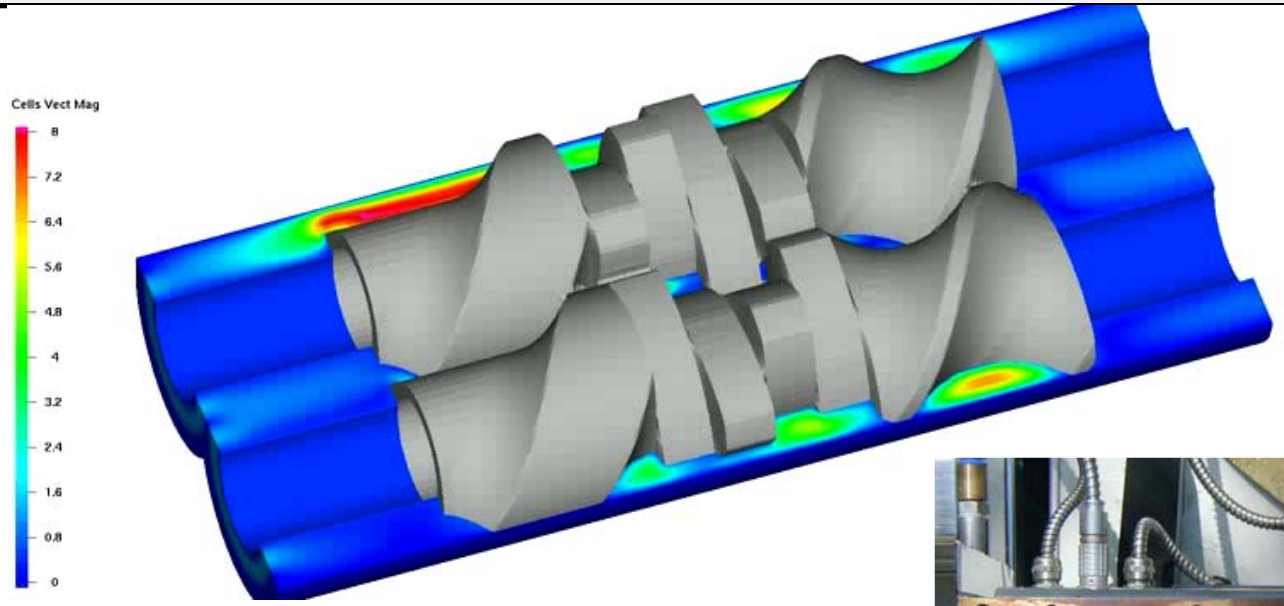




Example: Screw Extruder (I)

technische universität dortmund

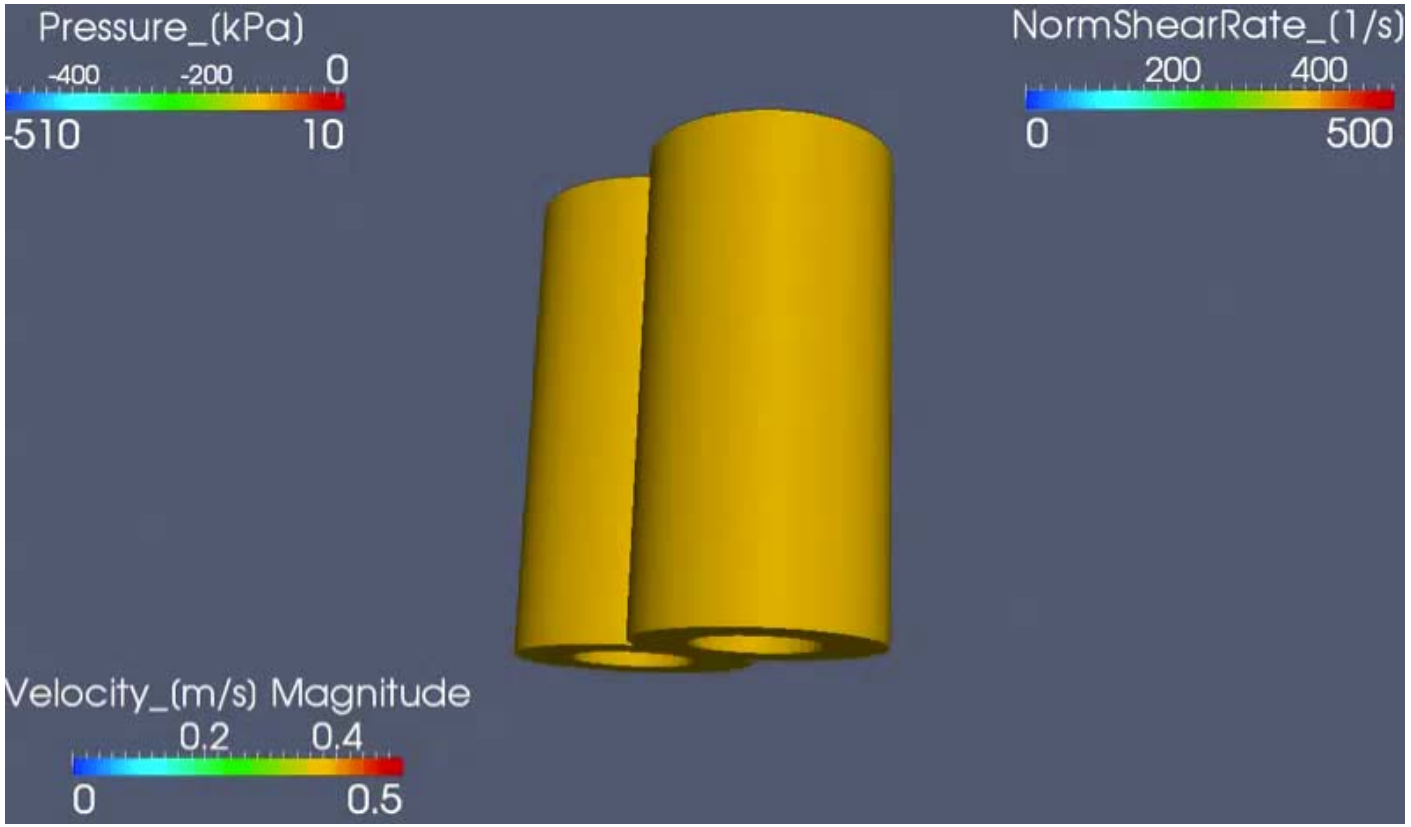
- Numerical simulation of (partially filled) *(twin)screw extruders*
- *Non-Newtonian rheological* models (shear & temperature dependent) with *non-isothermal* conditions (cooling from outside, heat production)
- Analysis of the influence of local characteristics on the global product quality, prediction of hotspots and maximum shear rates
- Optimization of torque acting on the screws, energy consumption

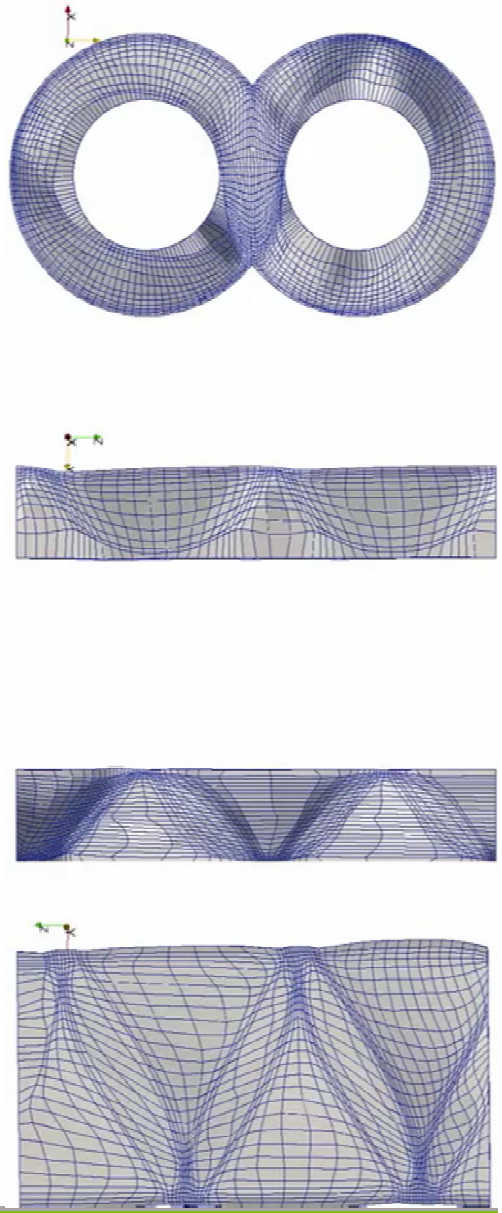


Example: Screw Extruder (II)

technische universität dortmund

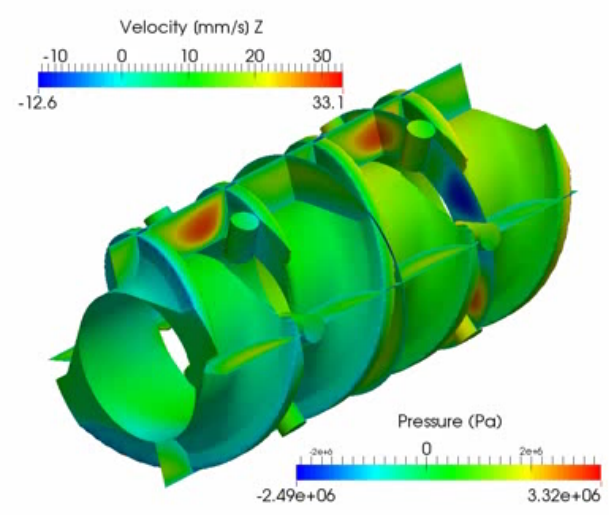
ALE based Mesh Deformation

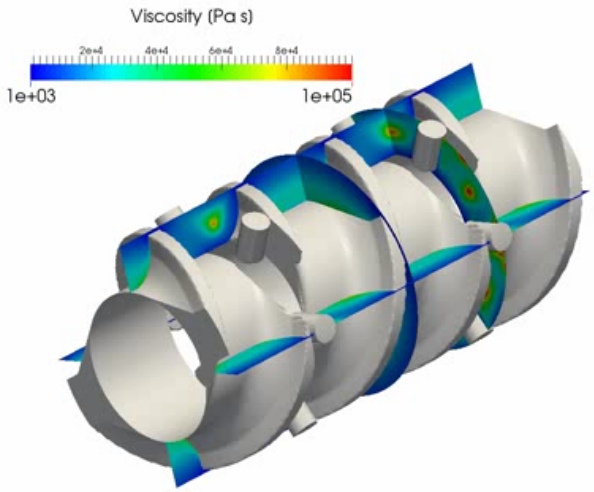




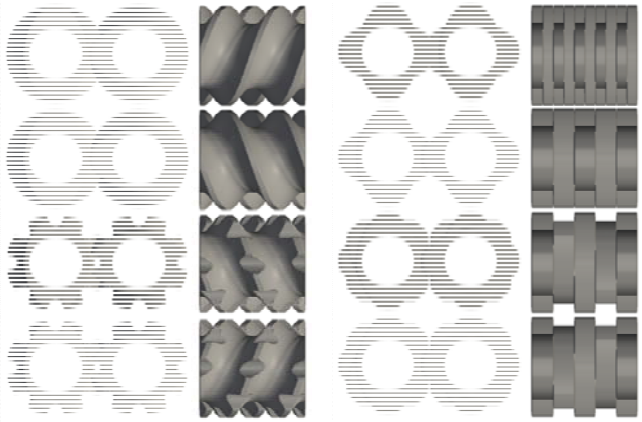
Example: Screw Extruder (III)

technische universität dortmund

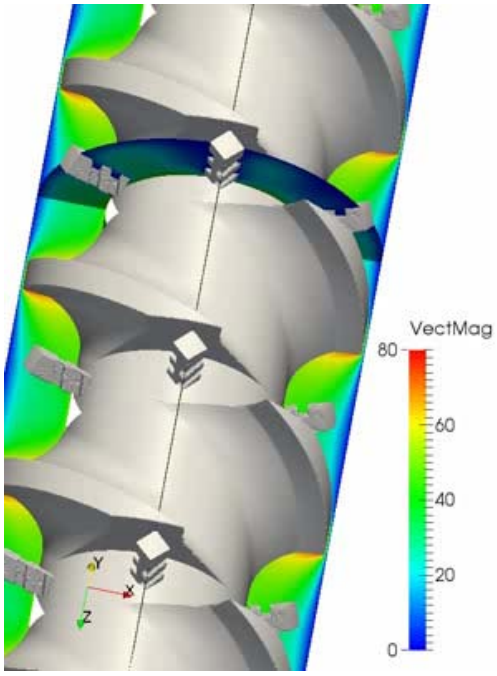


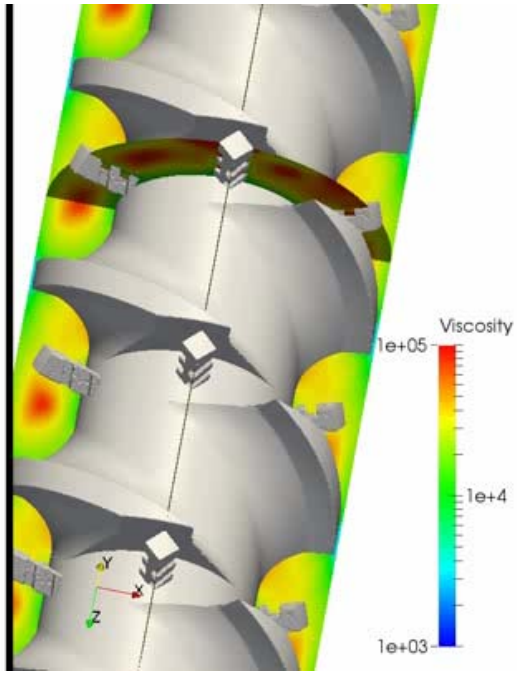


Twin-screw-element library



Combination of screw segments



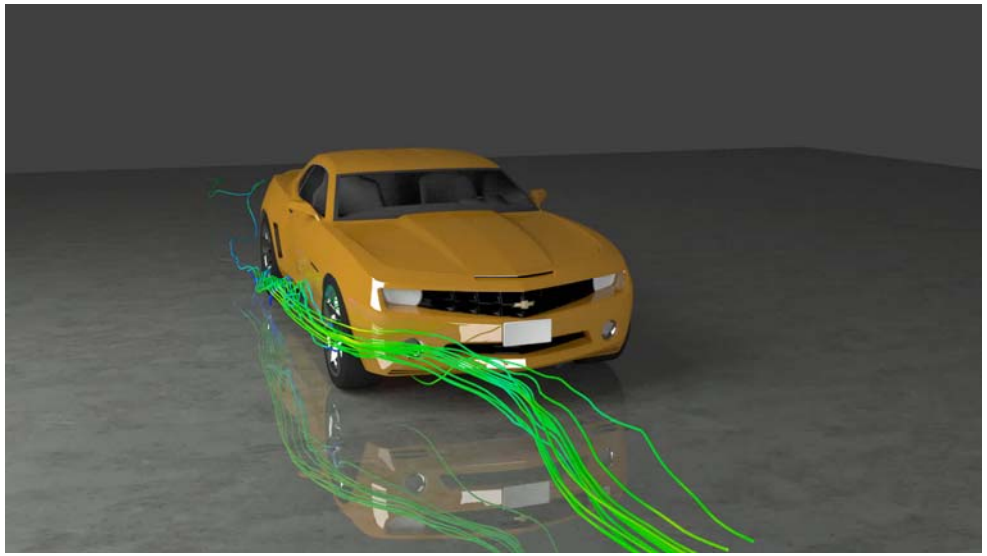


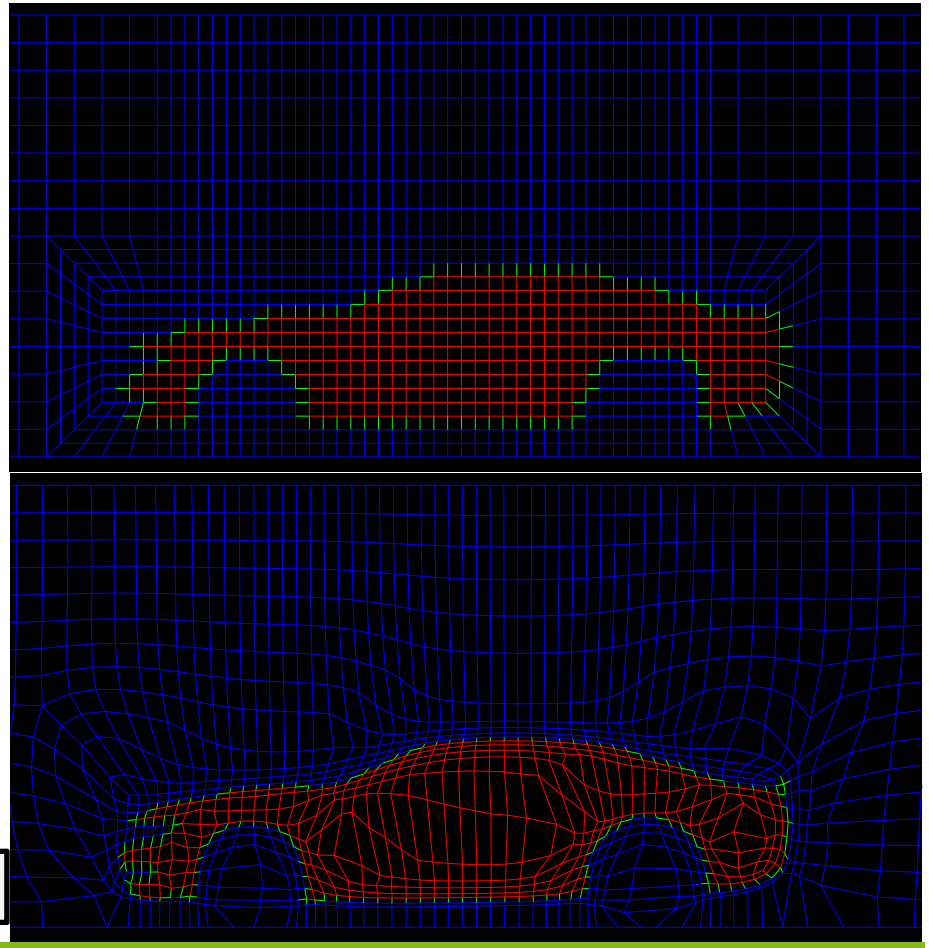
Example: Virtual Wind Tunnel (I)

technische universität dortmund

- Numerical simulation of complex geometries
- Use of a regular semi-unstructured preadapted mesh
- Resolution of small scale details by local mesh adaptation

Streamline visualization of the flow field around a car





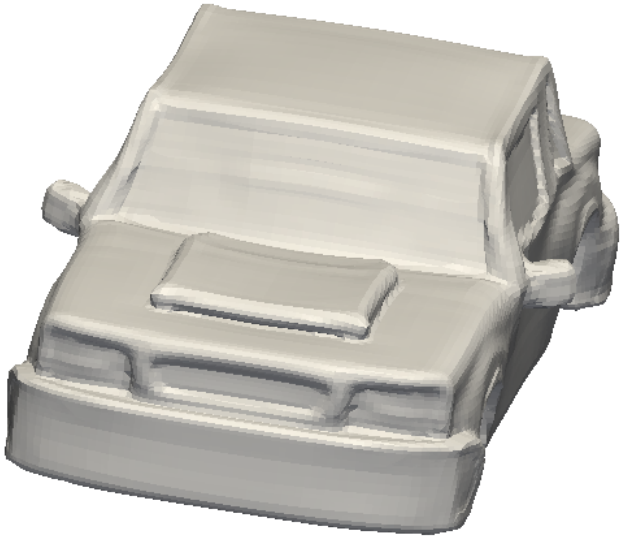
Mesh Slices with and without adaptation

Influence of Mesh Adaptation (II)

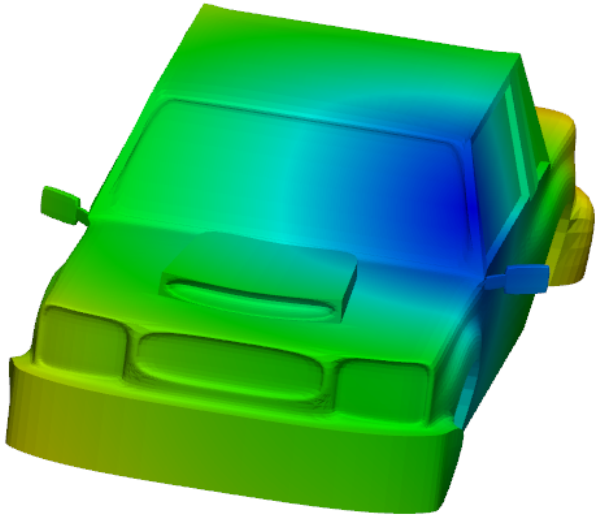
technische universität dortmund

Car representation by the computational mesh

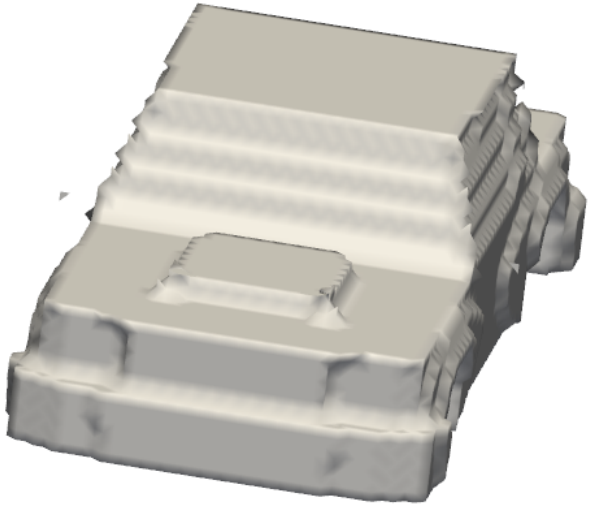
with adaptation



original



no adaptation

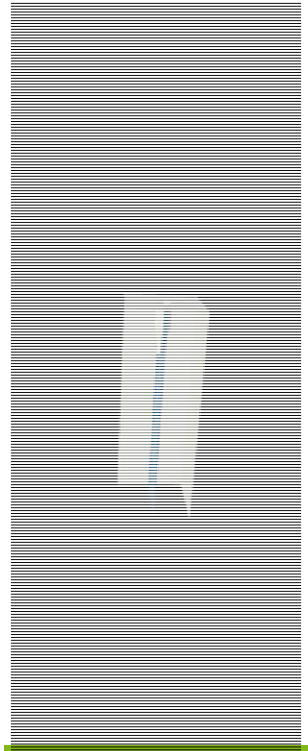


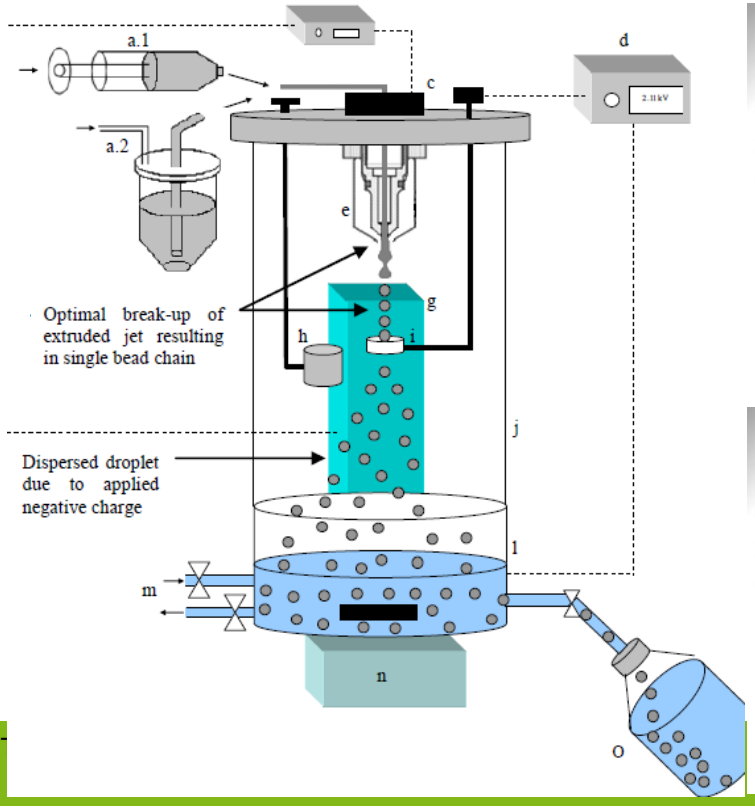
- Details may be lost without adaptation
- Better resolution with the same number of DOFs
- Mesh adaptation saves at least one refinement level

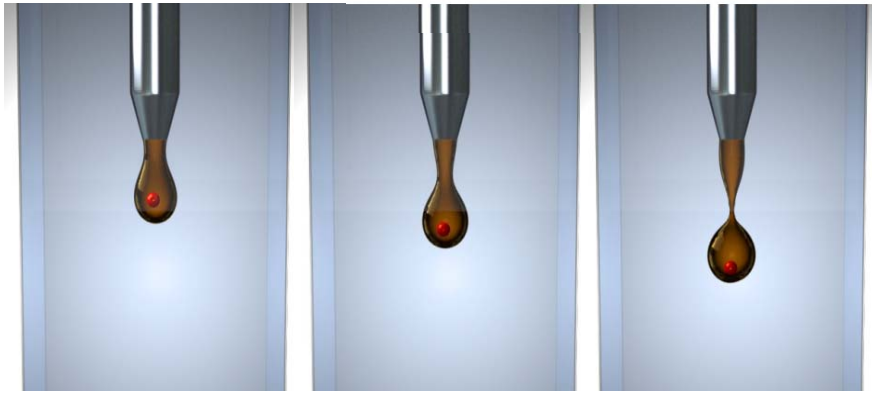
Example: Fluid Prilling&Encapsulation

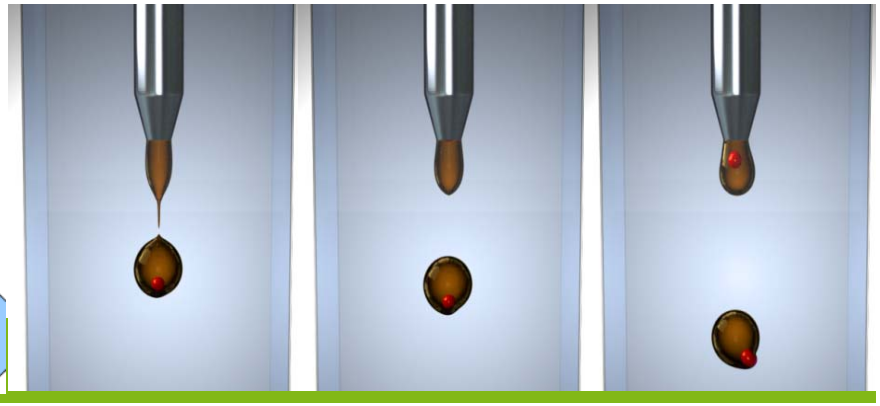
technische universität dortmund

- Numerical simulation of *micro-fluidic drug encapsulation ("monodisperse compound droplets")*
- Polymeric "bio-degradable" outer fluid with generalized Newtonian behaviour
- Optimization w.r.t. boundary conditions, flow rates, droplet size, geometry









Extensions & Future Activities

technische universität dortmund

Fluidics

- Viscoelastic fluids
- Multiphase problems
  - → Liquid-Liquid-Solid
  - → Melting/Solidification

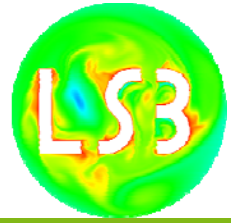
Hardware-Oriented Numerics

- Improve parallel efficiency of collision detection and force computation on GPU
- Dynamic grid adaptation with snapping

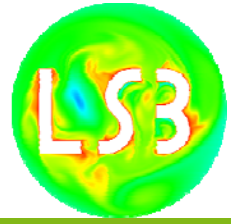
More communication between the scientific communities is required!

Benchmarking

- 1 „complex“ object
- „Many“ (=2) objects!

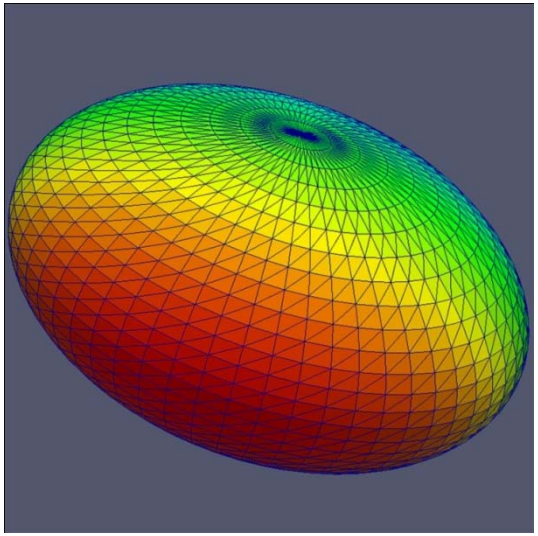


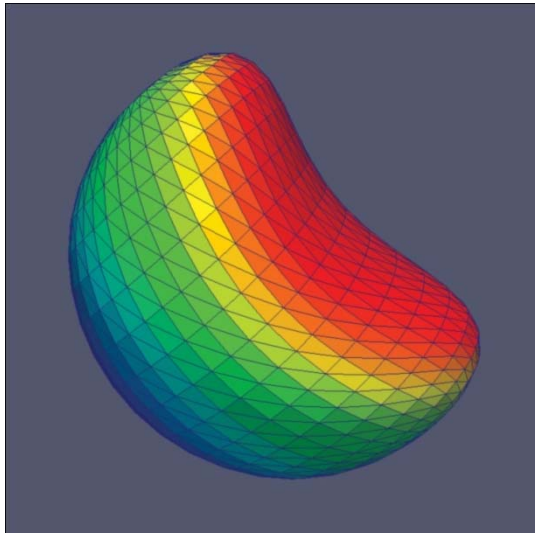
Backup slides …



Complex Rigid Bodies (I)

- **Geometric representation**:
  → Surface triangulation



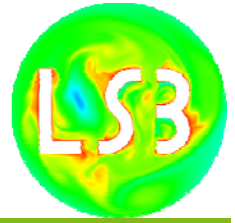


- **Important Properties**:
  → Center of Mass

$$\frac{\int_V \rho(x)x\,dV}{m}$$

  → Moment of Inertia tensor

$$\begin{bmatrix} \frac{1}{5}m(b^2+c^2) & 0 & 0 \\ 0 & \frac{1}{5}m(a^2+c^2) & 0 \\ 0 & 0 & \frac{1}{5}m(a^2+b^2) \end{bmatrix}$$



Complex Rigid Bodies (II)

Moment of Inertia Tensor:

- Can be calculated in a preprocessing step using FEM software
- Moment of inertia tensor changes as the body moves and rotates

  → Transform the moment of inertia tensor by the transformation matrix of the rigid body

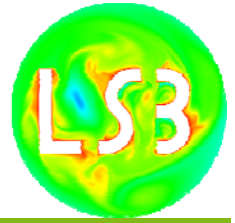
$$\begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \qquad \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix}$$

$$I_{xx} = \int_P (y^2 + z^2)\rho(x,y,z)dP \qquad I_{xy} = \int_P xy \cdot \rho(x,y,z)dP$$

$$I_{yy} = \int_P (x^2 + z^2)\rho(x,y,z)dP \qquad I_{xz} = \int_P xz \cdot \rho(x,y,z)dP$$

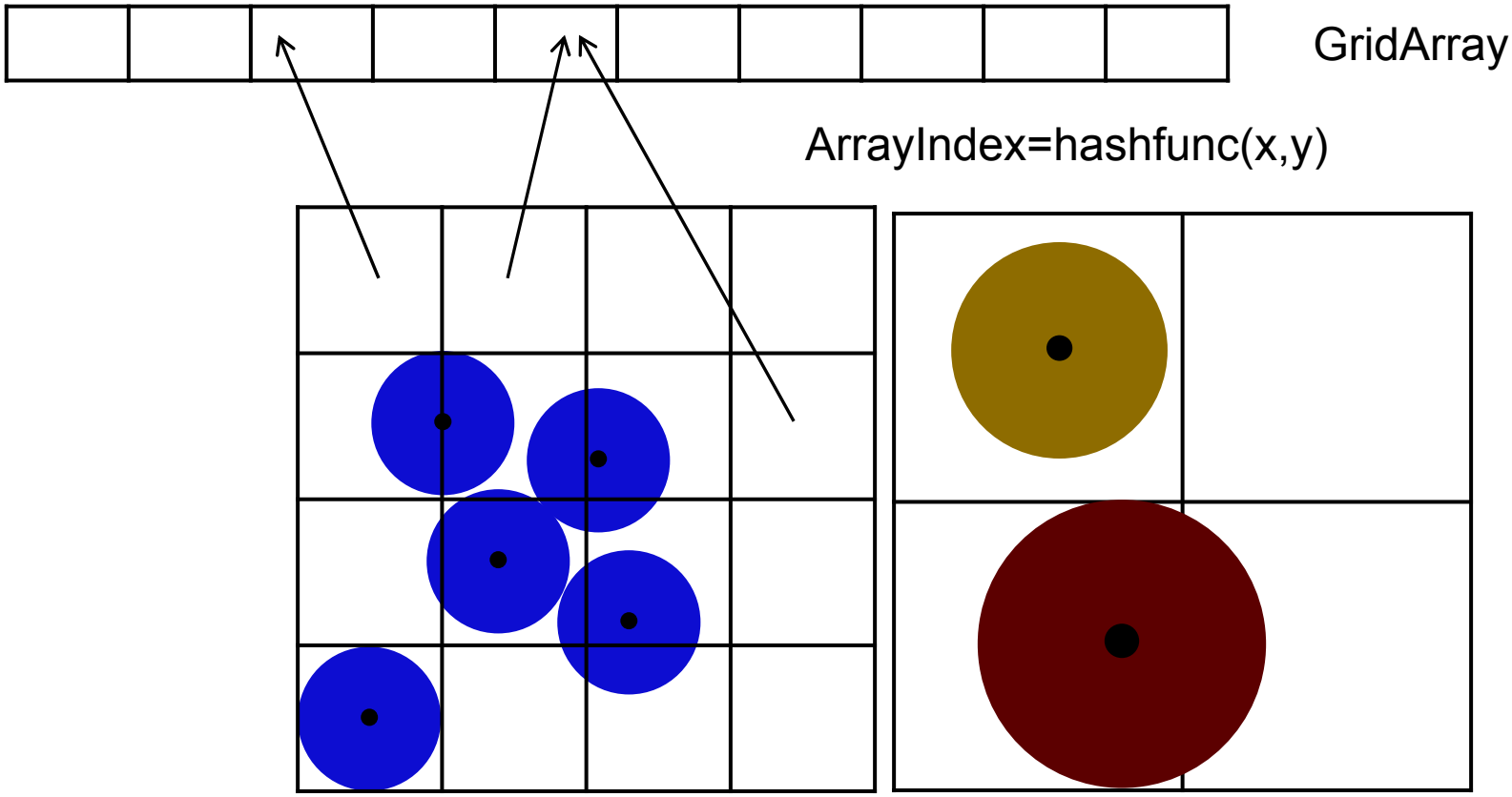
$$I_{zz} = \int_P (x^2 + y^2)\rho(x,y,z)dP \qquad I_{yz} = \int_P yz \cdot \rho(x,y,z)dP$$

Transformation of I: $I_w = M_{lw} \cdot I \cdot M_{lw}^T$



Spatial Partitioning/Uniform Grids

- Cell size relates to radius of (bounding) sphere
- Only objects in neighbouring cells are candidates for collision
- Multiple grid levels for objects of different size
- Efficient storage by using a hash mapping



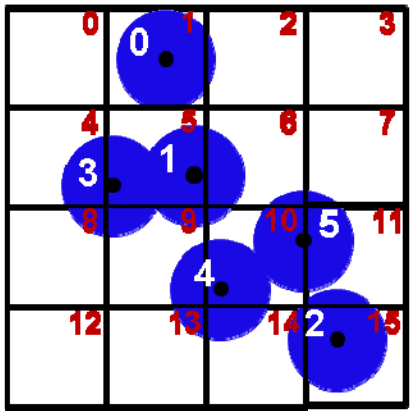
Collision Pipeline on the GPU (I)

technische universität dortmund

Collision Detection

- Domain is covered by a uniform grid or Aspatial hash
  - → use cell size equal to particle diameter
- Sort particles with GPU Radix sort by grid cell index
  - → Need to check only neighbouring cells for collisions
  - → Sorting leads to cache efficient grid traversal

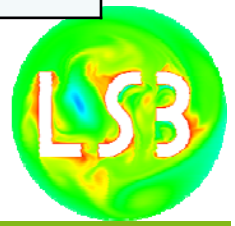
Uniform Grid → **Data Structure** → **Sorted Data Structure**



array	cell	particle
0	1	0
1	5	1
2	15	2
3	5	3
4	10	4
5	10	5

array	cell	particle
0	1	0
1	5	3
2	5	1
3	10	4
4	10	5
5	15	2

Cell array	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

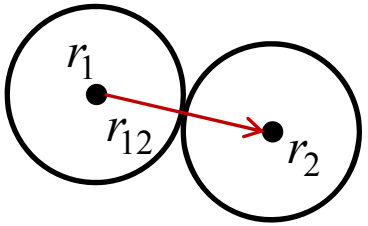


Collision Pipeline on the GPU (II)

technische universität dortmund

Collision forces

- Use a DEM approach that can be easily evaluated in parallel
- Consider only the 3x3x3 neighbouring cells for each particle

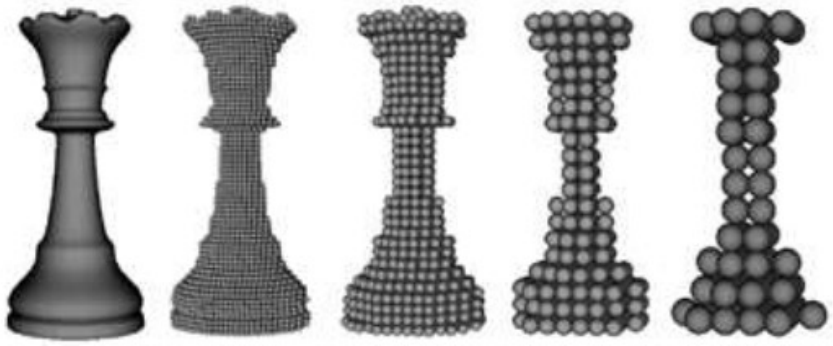


Forces acting on each particle

$$F_{i,s} = -k\left(d - \left|r_{ij}\right|\right)\frac{r_{ij}}{\left|r_{ij}\right|} \qquad F_{i,d} = \eta \cdot u_{ij}$$

$$F_{i,t} = k_t \cdot u_{ij,t} \qquad u_{ij,t} = u_{ij} - \left(u_{ij} \cdot \frac{r_{ij}}{\left|r_{ij}\right|}\right)\frac{r_{ij}}{\left|r_{ij}\right|}$$

k,η: material constants

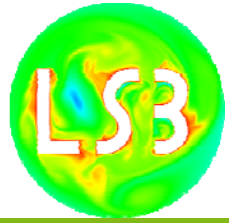


Sum up for each collision

$$F_{i,c} = \sum_{\text{collisions}(i)} \left(F_{i,s} + F_{i,d} + F_{i,t}\right)$$

$$T_{i,c} = \sum_{\text{collisions}(i)} \left(r_i \times \left(F_{i,s} + F_{i,d} + F_{i,t}\right)\right)$$

- Can be extended to rigid bodies
- Details: GPU Gems 3 (Takahiro Harada)

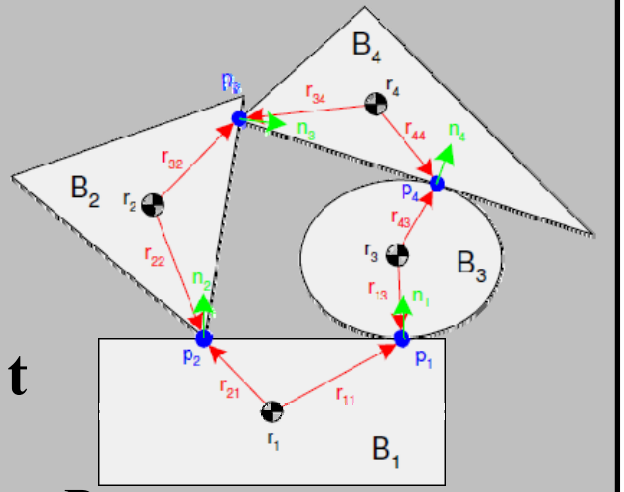


Multi-Body Collision Model (III)

Sequential Impulses

- Apply pairwise impulses iteratively
- Normal impulse

$$P_n = \max\left(\frac{-\Delta\bar{\mathbf{v}}\cdot\mathbf{n}}{k_n}, 0\right)$$



- Tangential (frictional) impulse

$$v_t = \Delta\mathbf{v}\cdot\mathbf{t}$$

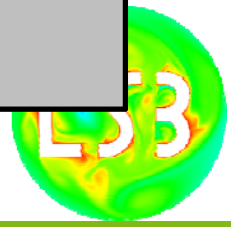
$$-\mu P_n \le P_t \le \mu P_n$$

- Terminate when:
  - Impulses become small
  - Iteration limit is reached

$$P_t = \text{clamp}\left(\frac{-\Delta\bar{\mathbf{v}}\cdot\mathbf{t}}{k_t}, -\mu P_n, \mu P_n\right)$$

$$k_t = \frac{1}{m_1} + \frac{1}{m_2} + \left[I_1^{-1}(\mathbf{r}_1\times\mathbf{t})\times\mathbf{r}_1 + I_2^{-1}(\mathbf{r}_2\times\mathbf{t})\times\mathbf{r}_2\right]\cdot\mathbf{t}$$

Details: Guendelman, *Nonconvex rigid bodies with stacking*



Multi-Body Collision Model (II)

Alternative LCP formulation:

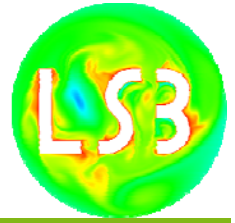
$$w = Ax + b, \quad w, x, b \in R^n, A \in R^{n \times n}, w, x, b \geq 0$$

$$(w_1 x_1, \ldots, w_n x_n) = (0, \ldots, 0)$$

$$A_{lk} = \delta_{i_l i_k} n_l^T \left(\frac{1}{m_{i_k}} 1_{3\times 3} - r_{li_l}^{\times} I_{i_k}^{-1} r_{ki_k}^{\times} \right) n_k - \delta_{i_l j_k} n_l^T \left(\frac{1}{m_{j_k}} 1_{3\times 3} - r_{li_l}^{\times} I_{j_k}^{-1} r_{kj_k}^{\times} \right) n_k$$

$$- \delta_{j_l i_k} n_l^T \left(\frac{1}{m_{i_k}} 1_{3\times 3} - r_{lj_l}^{\times} I_{i_k}^{-1} r_{ki_k}^{\times} \right) n_k + \delta_{j_l j_k} n_l^T \left(\frac{1}{m_{j_k}} 1_{3\times 3} - r_{lj_l}^{\times} I_{j_k}^{-1} r_{kj_k}^{\times} \right) n_k$$

This LCP can be solved with efficient iterative methods like the projected Gauss-Seidel solver (PGS).



Grid Deformation Method

technische universität dortmund

Idea : construct transformation $\phi, x = \phi(\xi, t)$ with $\det \nabla \phi = f$

$\Rightarrow$ local mesh area $\approx f$

1. Compute monitor function $f(x,t) > 0, f \in C^1$ and

$$\int_\Omega f^{-1}(x,t)dx = |\Omega|, \quad \forall t \in [0,1]$$

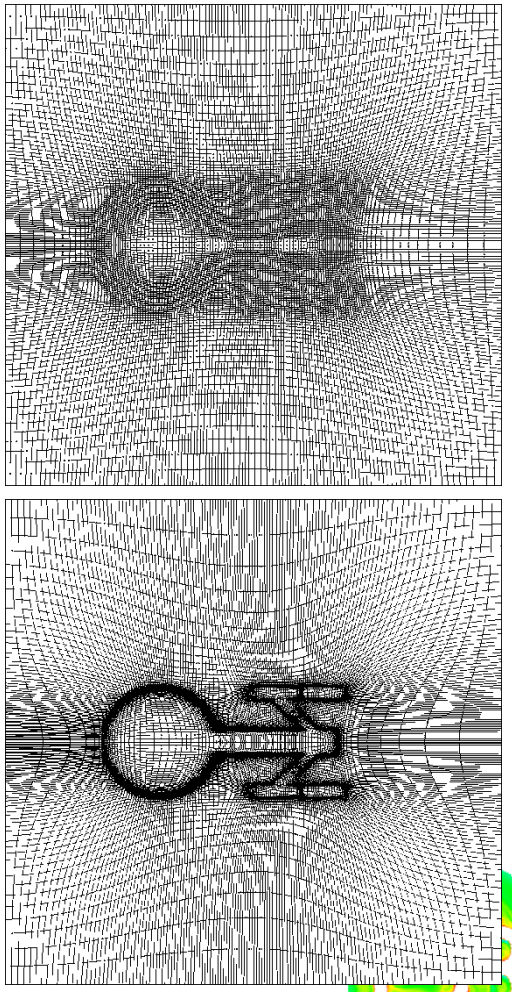
2. Solve $(t \in [0,1])$

$$\Delta v(\xi,t) = -\frac{\partial}{\partial t}\left(\frac{1}{f(\xi,t)}\right), \quad \left.\frac{\partial v}{\partial n}\right|_{\partial\Omega} = 0$$

3. Solve the ODE system

$$\frac{\partial}{\partial t}\phi(\xi,t) = f(\phi(\xi,t),t)\nabla v(\phi(\xi,t),t)$$

new grid points: $x_i = \phi(\xi_i, 1)$



Grid deformation preserves the (local) logical structure

Multi-Body Collision Model (II)

Alternative LCP formulation:

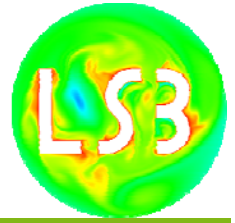
$$w = Ax + b, \quad w, x, b \in R^n, A \in R^{n \times n}, w, x, b \geq 0$$

$$(w_1 x_1, \ldots, w_n x_n) = (0, \ldots, 0)$$

$$A_{lk} = \delta_{iik} n_l^T \left(\frac{1}{m_{ik}} 1_{3\times 3} - r_{il}^{\times} I_{ik}^{-1} r_{ki}^{\times} \right) n_k - \delta_{ijk} n_l^T \left(\frac{1}{m_{jk}} 1_{3\times 3} - r_{il}^{\times} I_{jk}^{-1} r_{kj}^{\times} \right) n_k$$

$$- \delta_{jik} n_l^T \left(\frac{1}{m_{ik}} 1_{3\times 3} - r_{jl}^{\times} I_{ik}^{-1} r_{ki}^{\times} \right) n_k + \delta_{jjk} n_l^T \left(\frac{1}{m_{jk}} 1_{3\times 3} - r_{jl}^{\times} I_{jk}^{-1} r_{kj}^{\times} \right) n_k$$

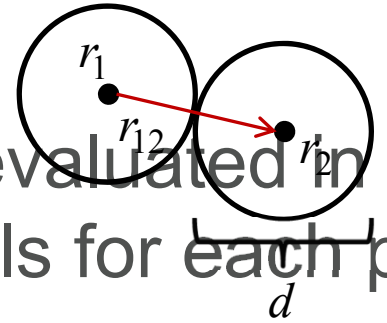
This LCP can be solved with efficient iterative methods like the projected Gauss-Seidel solver (PGS).



Particle Simulation on the GPU(II)

Collision forces

- Use a DEM approach that can be easily evaluated in pa
- Consider only the 3x3x3 neighbouring cells for each pa



Forces acting on each particle

$$F_{i,s} = -k\left(d - \left|r_{ij}\right|\right)\frac{r_{ij}}{\left|r_{ij}\right|} \qquad F_{i,d} = \eta \cdot u_{ij}$$

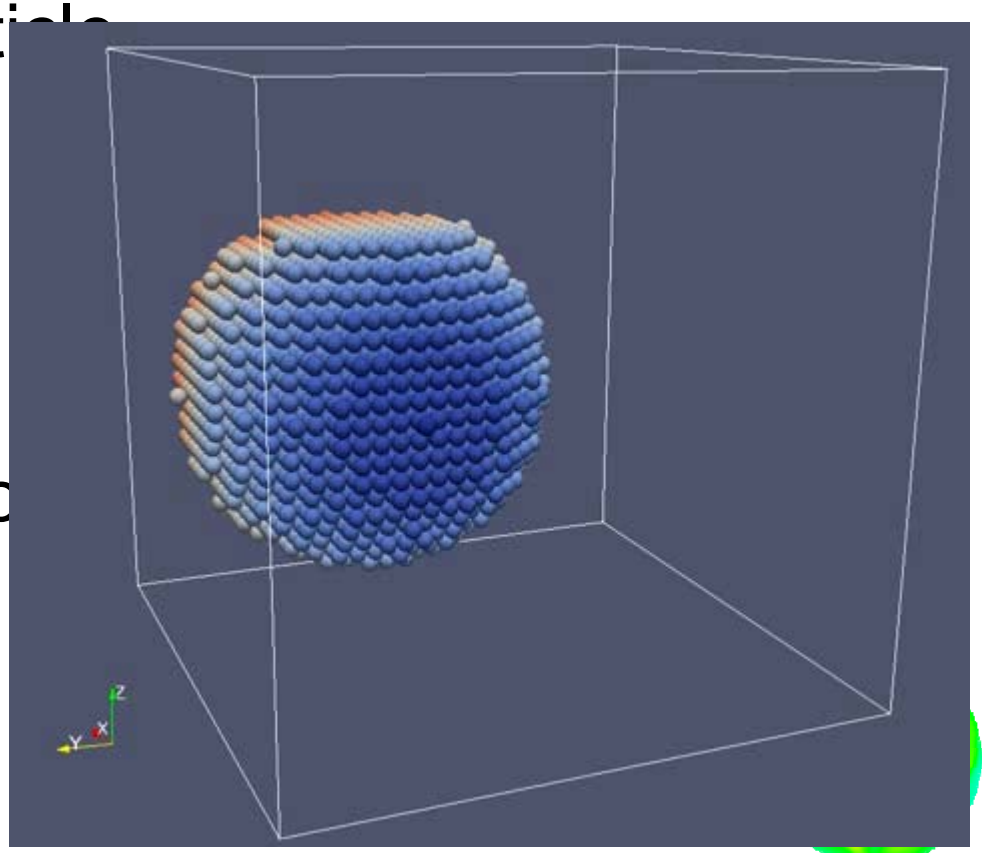
$$F_{i,t} = k_t \cdot u_{ij,t} \qquad u_{ij,t} = u_{ij} - \left(u_{ij} \cdot \frac{r_{ij}}{\left|r_{ij}\right|}\right)\frac{r_{ij}}{\left|r_{ij}\right|}$$

k,η: material constants

Sum up for each collisio

$$F_{i,c} = \sum_{collisions(i)} \left(F_{i,s} + F_{i,d} + F_{i,t}\right)$$

$$T_{i,c} = \sum_{collisions(i)} \left(r_i \times \left(F_{i,s} + F_{i,d} + F_{i,t}\right)\right)$$



Efficient FBM Calculation

Efficient large-scale FBM

For *n* particles and *m* dofs a naive calculation of the inside/outside function on the grid has a complexity of *O(n·m)*

Neccessary only if the grid data structure does not have links between the cell index and the geometric location of the cell

Exploit temporal coherency

- The particle trajectory is relatively smooth
- Change in position between timesteps is small

FBM Update Strategy

→ Store the cells intersected by a particle in t_n

→ Start neighbour search from these cells in t_{n+1}

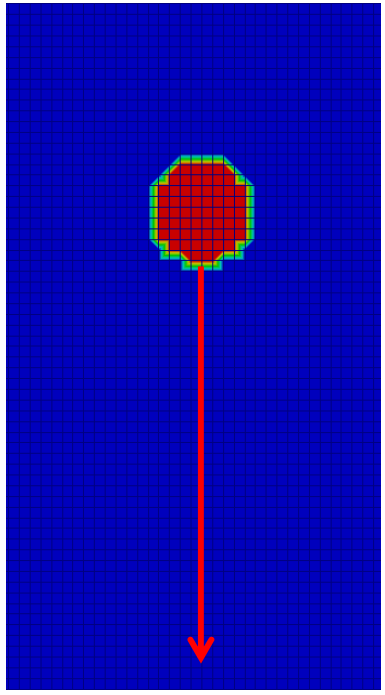
→ While cell is in object and not visited:

  → Store cell, mark as visited

  → Search next neighbour

→ Number of cells visited is a constant:

  → Update complexity: *O(n)*

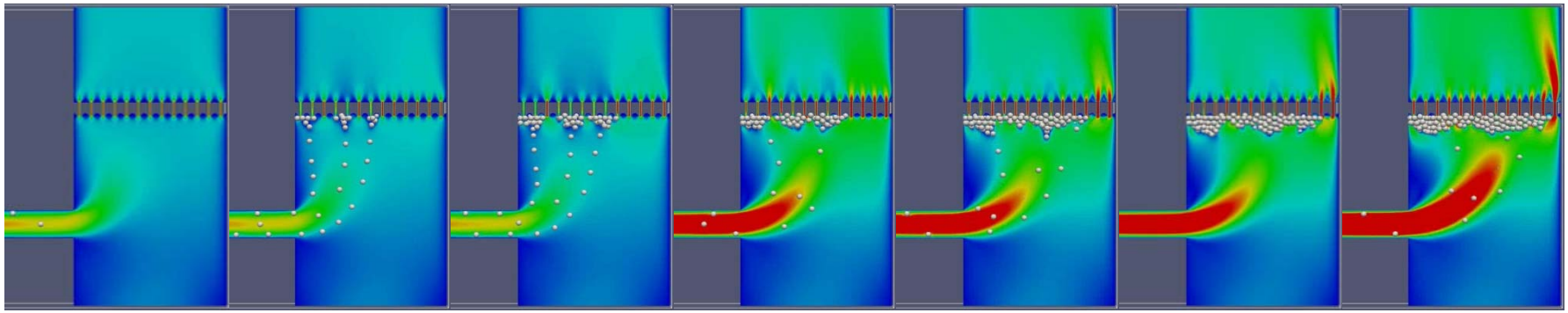


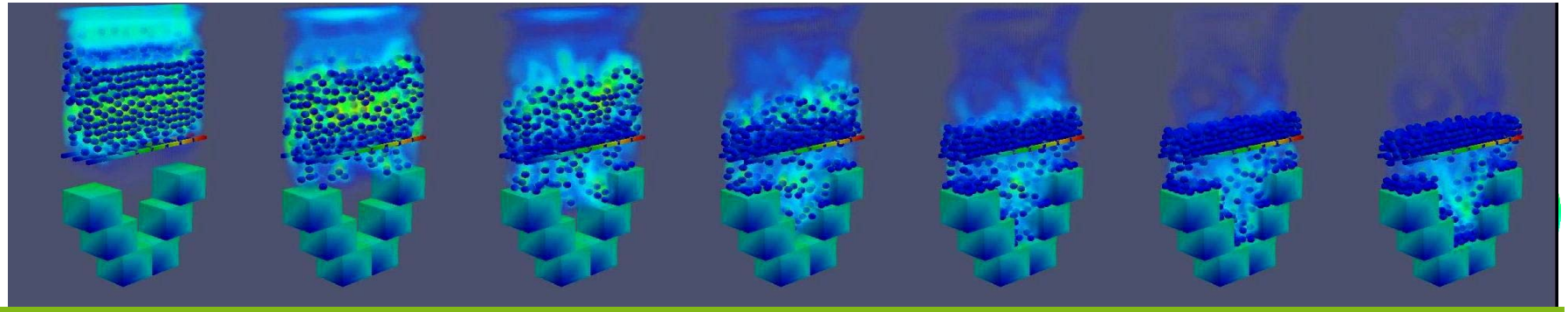
Alternative:

Hierarchical Uniform Hash Grids

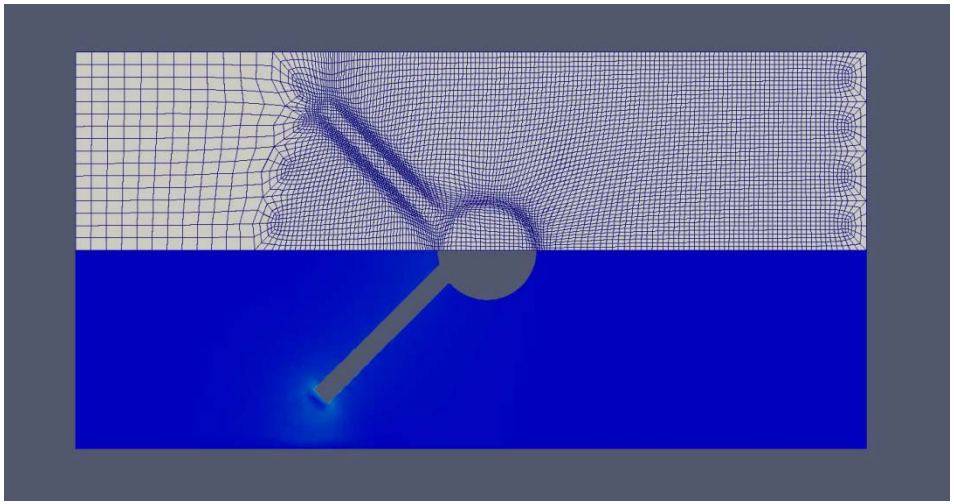
Examples

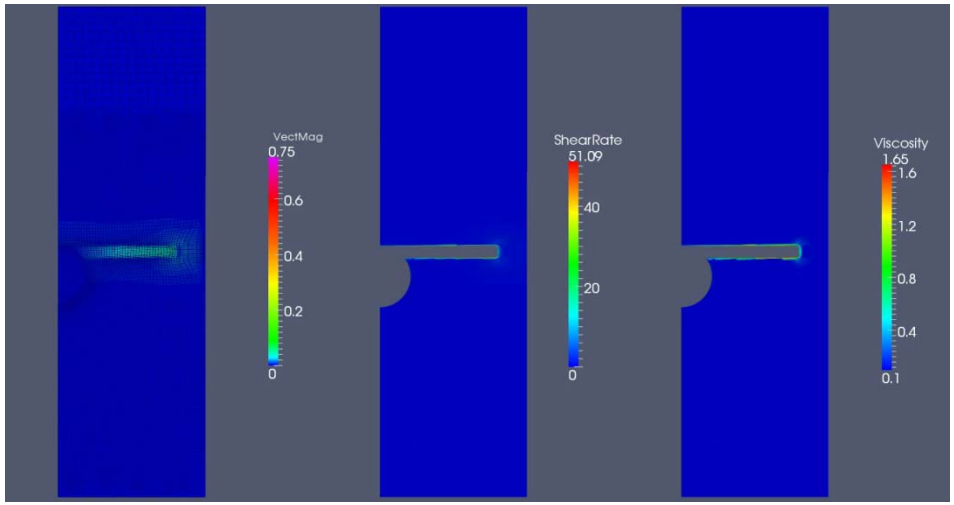
- Flows with complex geometries
- Fluidized bed
- Particulate flow demonstrating incompressibility
- GPU sedimentation example
- Numerical results and benchmark test cases
- Comparison of results with other groups

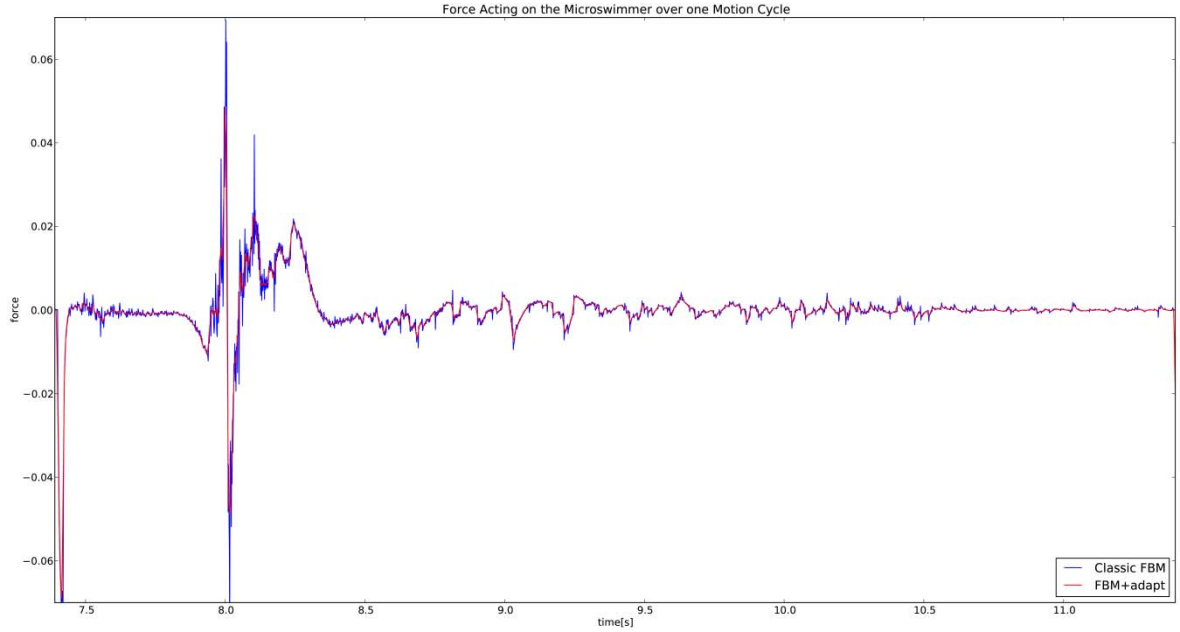




Microswimmer Example

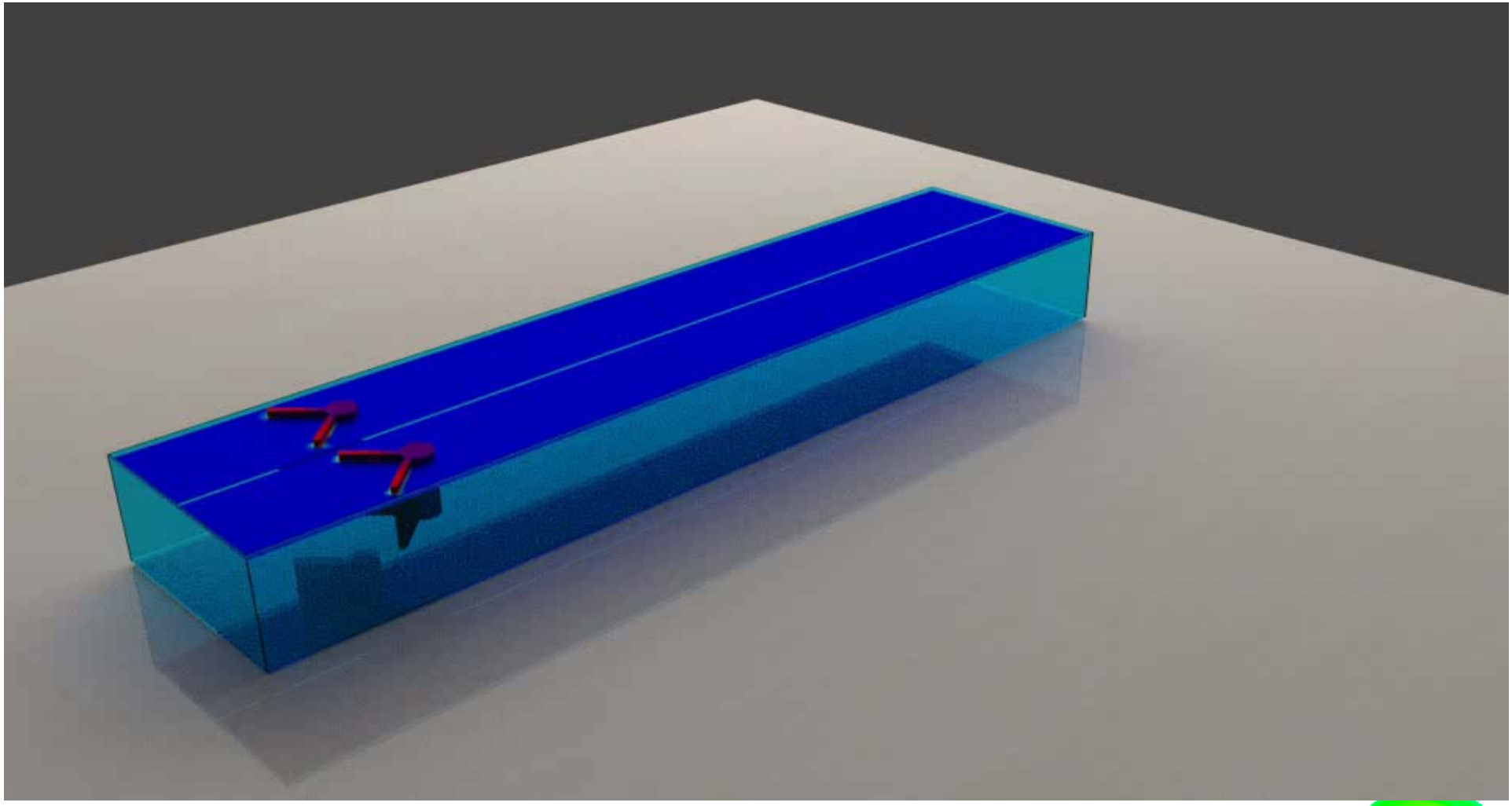


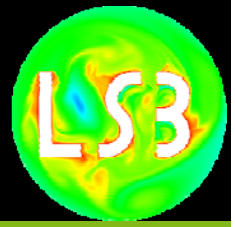




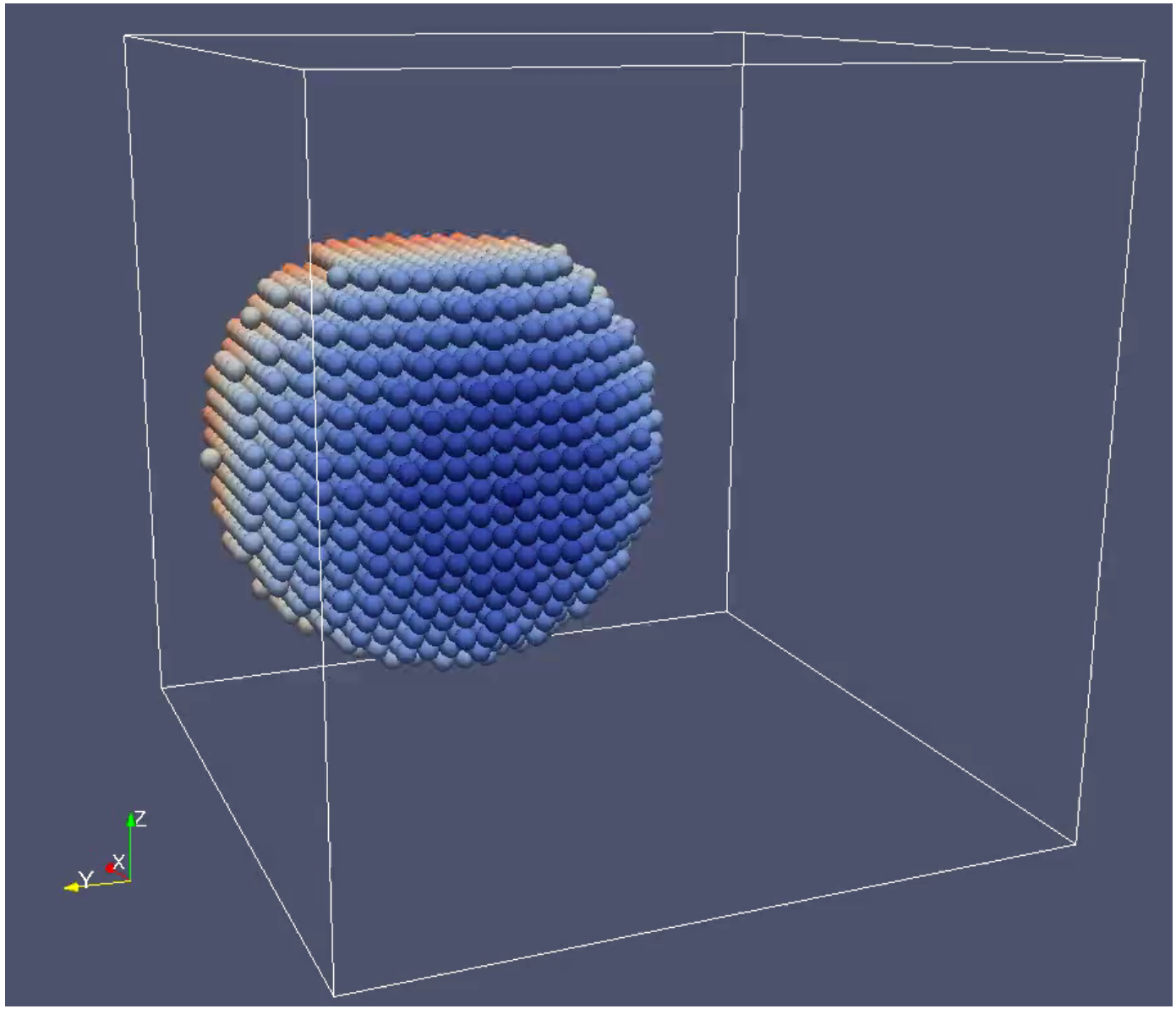
Compared to the classic FBM approach the force curve is much smoother with grid adaptation

Microswimmer Example

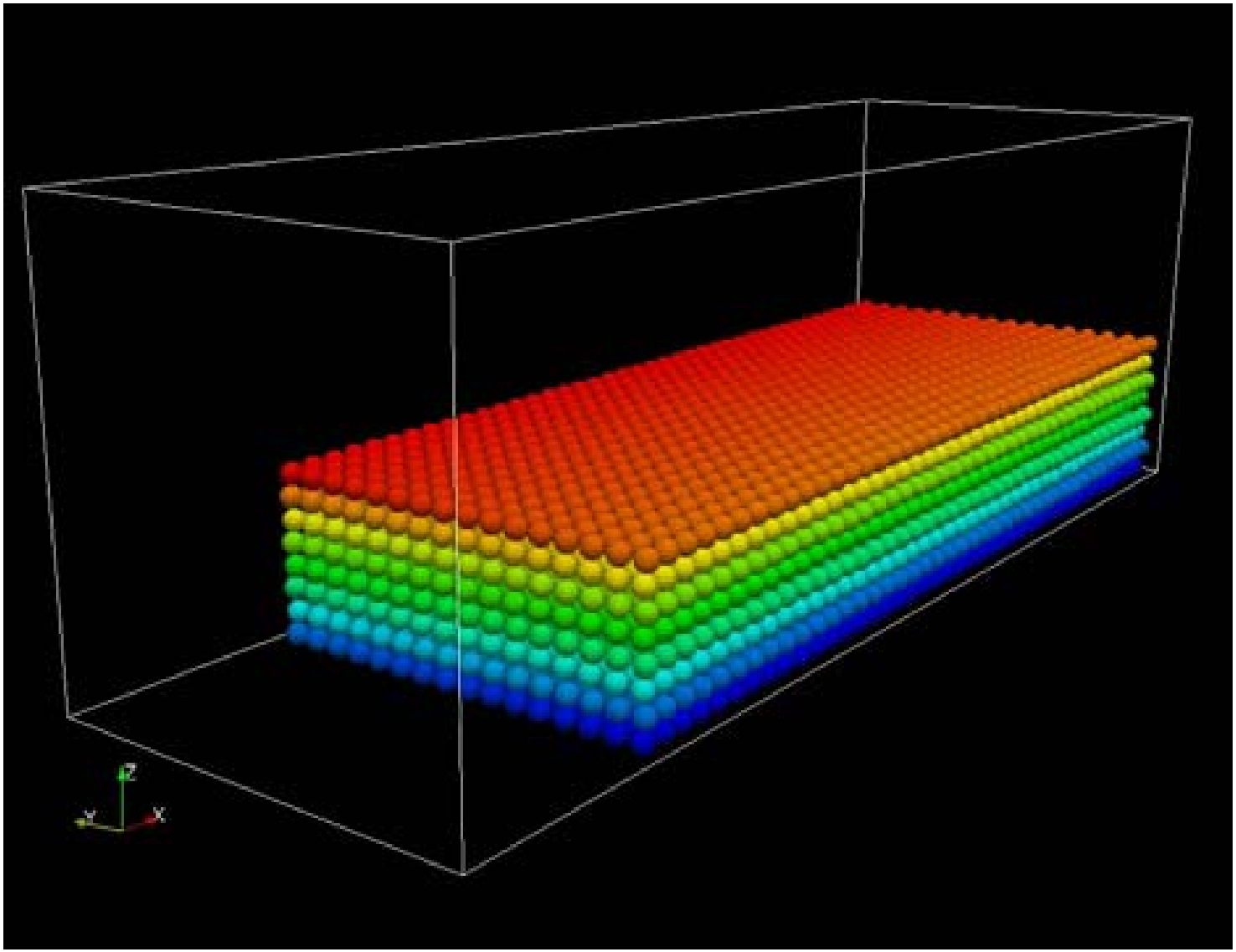



LSTM

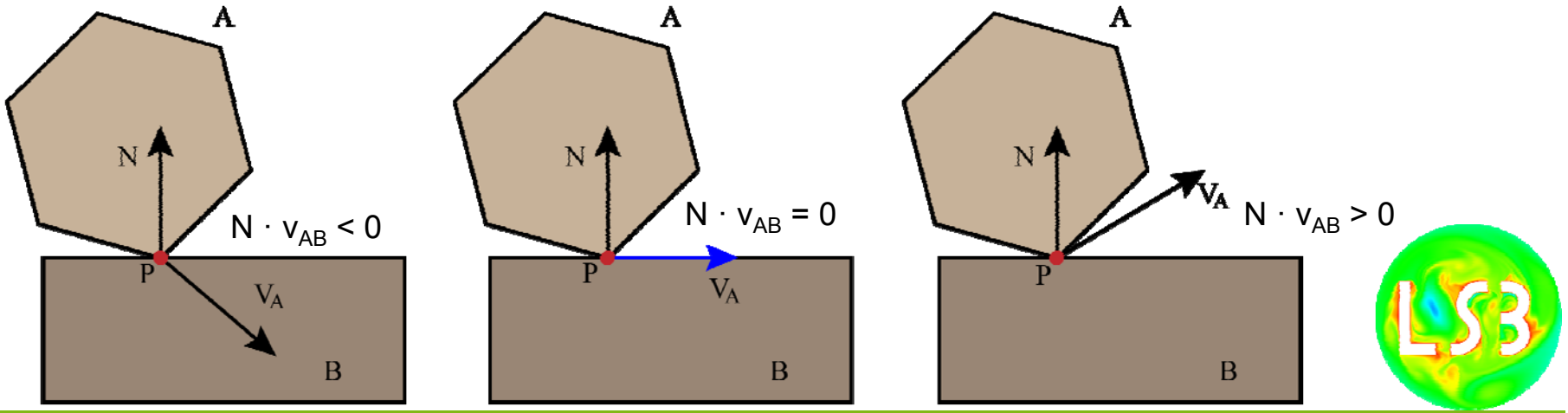
Many Particle Sedimentation



Driven Cavity with Particles



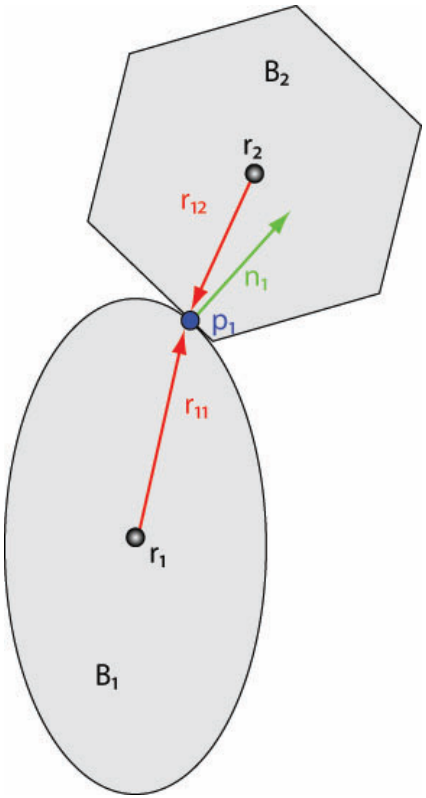
- **Contact determination** for rigid bodies A and B:
  - → Distance d(A,B)
  - → Relative velocity $v_{AB} = (v_A + \omega_A \times r_A - (v_B + \omega_B \times r_B))$
  - → Collision normal $N = (X_A(t) - X_B(t))$
  - → Relative normal velocity $N \cdot (v_A + \omega_A \times r_A - (v_B + \omega_B \times r_B))$
- distinguishes three cases of how bodies move relative to each other:
  - → Colliding contact : $N \cdot v_{AB} < 0$
  - → Separation : $N \cdot v_{AB} > 0$
  - → Touching contact : $N \cdot v_{AB} = 0$



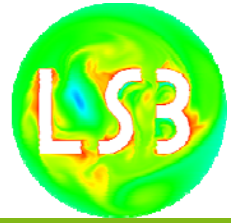
Contact Force Calculation

technische universität dortmund

- Contact force calculation realized as a three step process
  - → **Broad phase**
  - → **Narrow phase**
  - → Contact/Collision force calculation
- Worst case complexity for collision detection is $O(n^2)$
  - → Computing contact information is expensive
  - → Reduce number of expensive tests → Broad Phase
- *Broad phase*
  - → Simple rejection tests exclude pairs that cannot intersect
  - → Use hierarchical spatial partitioning
- *Narrow phase*
  - → Uses Broadphase output
  - → Calculates data neccessary for collision force calculation



► Several single, resp., multibody collision models are realized and tested, but…..



Single Body Collision Model

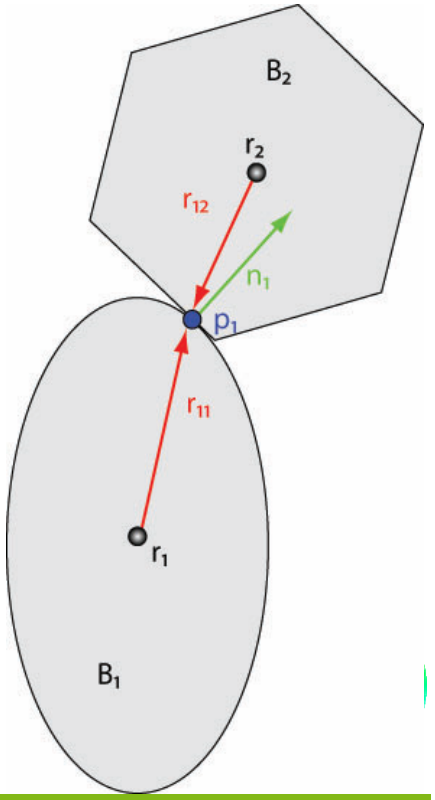
For a single pair of colliding bodies we compute the impulse f that causes the velocities of the bodies to change:

$$f = \frac{-(1+\varepsilon)(n_1(v_1 - v_2) + \omega_1(r_{11} \times n_1) - \omega_2(r_{12} \times n_1))}{m_1^{-1} + m_2^{-1} + (r_{11} \times n_1)^T I_1^{-1}(r_{11} \times n_1) + (r_{12} \times n_1)^T I_2^{-1}(r_{12} \times n_1)}$$

Using the impulse f, the change in linear and angular velocity can be calculated:

$$v_1(t+\Delta t) = v_1(t) + \frac{fn_1}{m_1}, \omega_1(t+\Delta t) = \omega_1(t) + I_1^{-1}(r_{11} \times fn_1)$$

$$v_2(t+\Delta t) = v_2(t) - \frac{fn_1}{m_2}, \omega_2(t+\Delta t) = \omega_2(t) - I_2^{-1}(r_{12} \times fn_1)$$



Multi-Body Collision Model

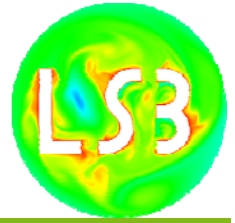
In the case of **multiple colliding bodies** with ***K* contact points** the impulses influence each other. Hence, they are combined into a **system of equations** that involves the following matrices and vectors:

- N: matrix of contact normals
- C: matrix of contact conditions
- M: rigid body mass matrix
- f: vector of contact forces ($f_i \geq 0$)
- f^{ext}: vector of external forces(gravity, etc.)

$$\underbrace{N^T C^T M^{-1} C N}_{A} \cdot \underbrace{\Delta t f^{t+\Delta t}}_{x} + \underbrace{N^T C^T \left(u^t + \Delta t M^{-1} + f^{ext} \right)}_{b} \geq 0, f \geq 0$$

A problem of this form is called a **Linear Complementarity Problem (LCP)** which can be solved with efficient iterative methods like the **Projected Gauss-Seidel solver (PGS)**.

Kenny Erleben, *Stable, Robust, and Versatile Multibody Dynamics Animation*



DGS Configuration