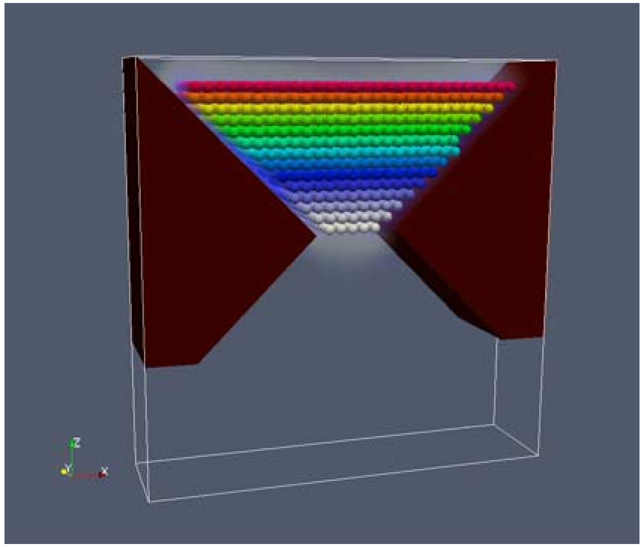
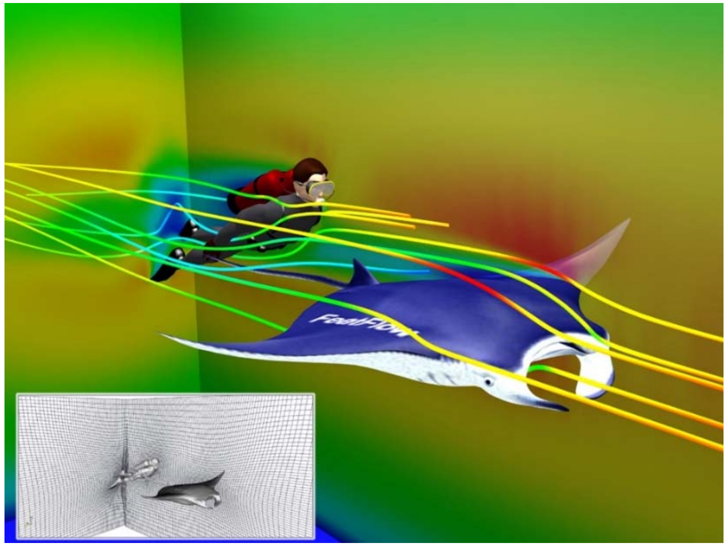
Hardware-oriented Numerics for PDE and Unconventional High Performance Computing Techniques

---

Motivation, Concepts, Applications to CFD





Stefan Turek and the FEAST Group
Institut für Angewandte Mathematik (LS3), TU Dortmund
http://www.mathematik.tu-dortmund.de/LS3
http://www.featflow.de

Motivation

I) **Scientific Computing has been facing a paradigm shift**

II) **Unconventional hardware has to be taken into account**

III) **Realistic applications: *Virtual Labs* for Multiphase flow**

Main message: *Future solvers* for ***simulations*** of ***PDEs***

are influenced by

HWON + (UC)HPC Techniques on **Exascale Hardware**

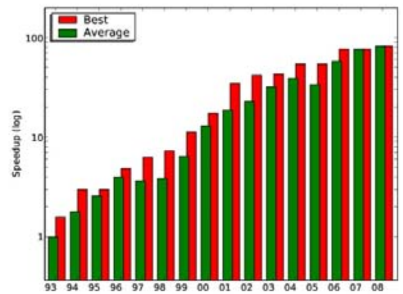
Motivation

I) Scientific Computing has been facing a paradigm shift

► Adaptive Finite Element Methods (AFEM) and Multigrid Solvers: most flexible, efficient and accurate simulation tools for PDEs nowadays, but **software realization no longer runs faster automatically on newer hardware**

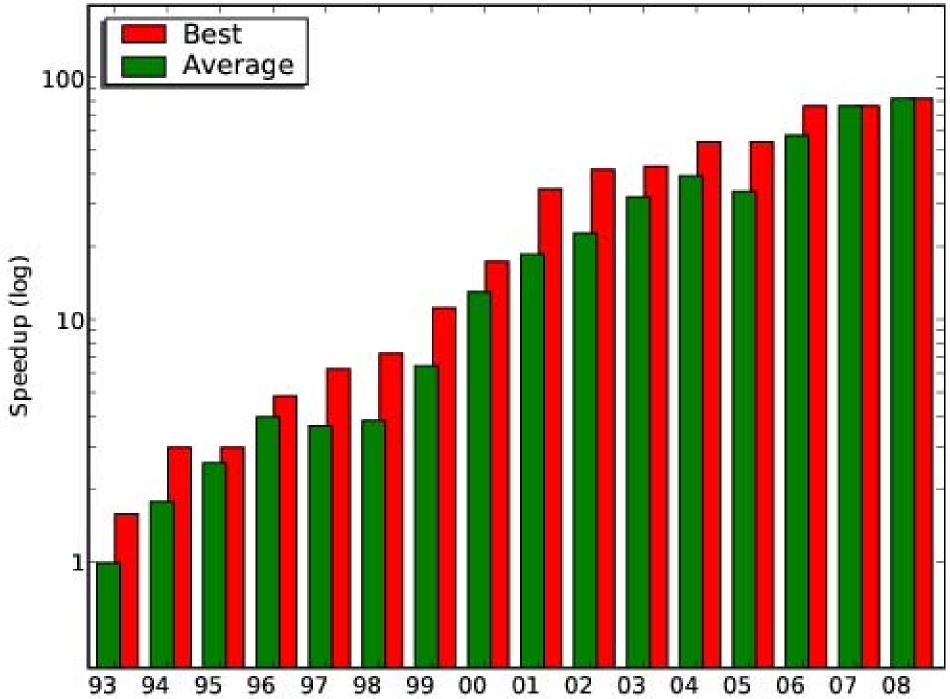
► Single CPU cores are not getting so much faster (frequency scaling), while significant speed-up is obtained via different levels of parallelism

► Data movement is usually more expensive than computations (memory wall) in particular for Linear Algebra problems



FeatFlow-Benchmark 1993-2008: FEM-MG F77-code

Speed-up (logscale)



→ Speed-up of 80x for free in 16 years

→ Stagnation for standard simulation tools

→ **Absolute performance?**

Motivation

II) *Unconventional* hardware has to be taken into account

Parallelism and *heterogeneity* ubiquitous in modern hardware:

→ Multicore CPUs, multicore accelerators (XEON Phi), GPUs

→ Heterogeneity on a node (e.g. CPUs + GPUs)

→ Heterogeneous chips (ARM SoCs, Cell in PS3, Jaguar in PS4)

→ Many levels of parallelism (fine-grained SIMD to coarse grained MP)

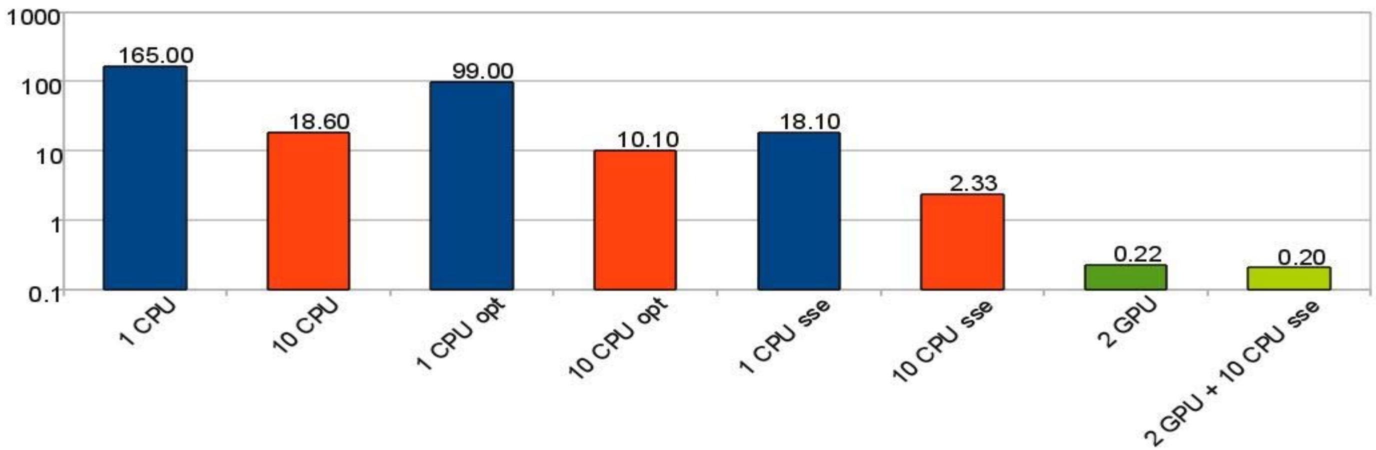
Consequences from *modern* and *unconventional* hardware:

→ Algorithms and whole software-frameworks have to be re-tailored

→ Hardware-aware implementation vs. numerical efficiency gets crucial

→ Insufficient compiler-/library-/autotuning-support ➔ careful tuning

→ Myths vs. reality: 100x speedups via GPGPU, 'prototype codes', ...

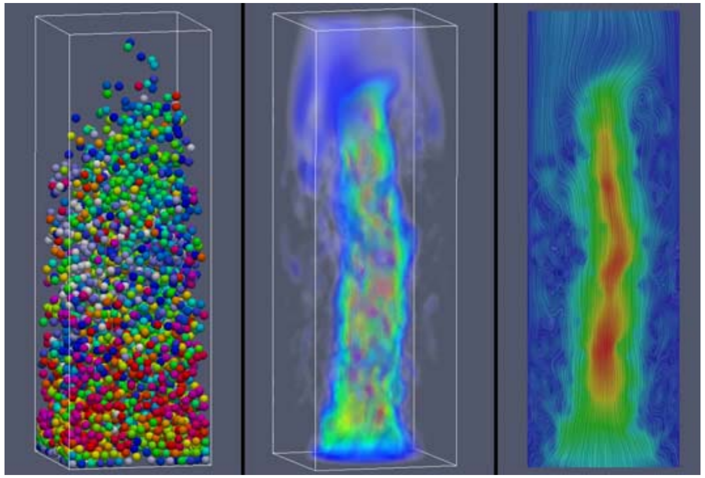


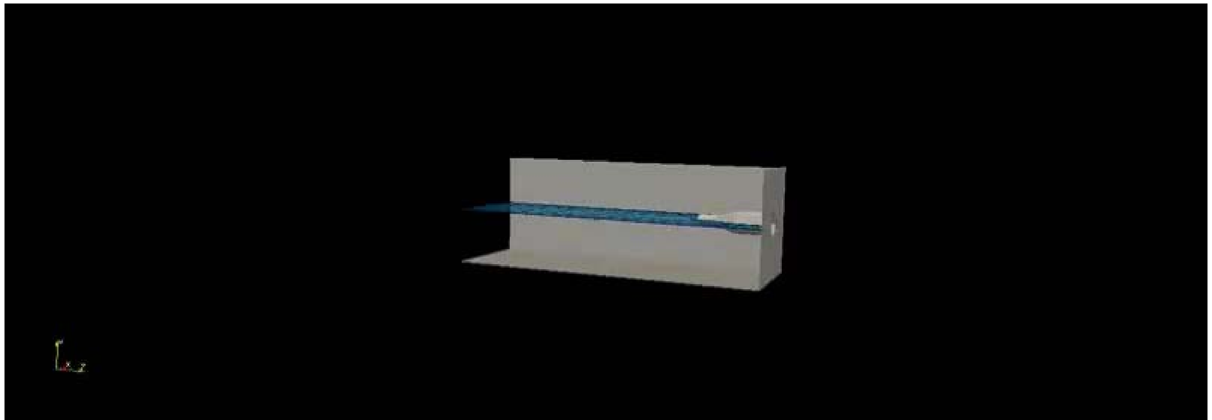
III) Realistic (= industrial) applications: *Virtual Labs*

- ► How to design algorithms and software on these architectures for <u>complete</u> *Virtual Labs* for realistic applications?

- ► Vision: *Highly efficient, flexible and accurate „real life“ simulation based on <u>modern Numerics</u> and algorithms while exploiting <u>modern hardware</u>!*

- ► Here: Multiphase-CFD as prototype for complex problems





Aim of this Talk

technische universität
dortmund

High Performance Computing

meets

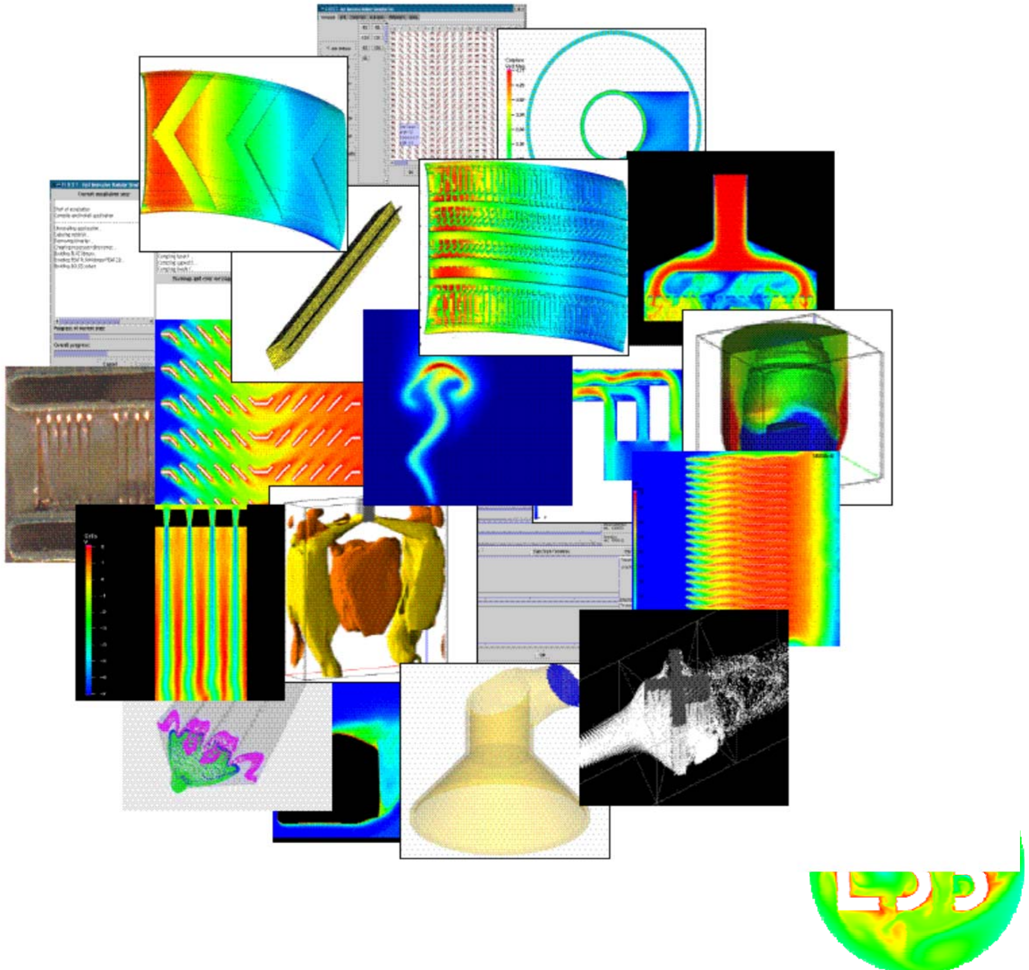
Hardware-oriented Numerics

on

Unconventional Hardware

for

Multiphase Flow Problems



Use the "best" numerical & algorithmic concepts while exploiting modern hardware <u>at the same time</u>!

- It is more than '*good Numerics*' and '*good Implementation*' (and hence '*good scheduling of building blocks*') on modern (parallel) hardware architecture
- Consider '*short-term hardware developments*' now, but '*long-term hardware trends*' for designing efficient numerical schemes
- 'Total Numerical Efficiency' as critical quantity for balancing numerical efficiency vs. hardware efficiency

`Total Numerical Efficiency’

technische universität dortmund

- ‘**High** (guaranteed) **accuracy** for user-specific quantities with minimal #d.o.f. (~ *N*) via **fast and robust solvers** – for a wide class of parameter variations – with **optimal numerical complexity** (~ *O*(*N*)) …
  But: while exploiting a significant percentage of the **available huge single node/full machine TeraFLOP/ExaFLOP rates** at the same time’

- Is it easy to achieve high ‘Total Numerical Efficiency’? How to measure?

What is the problem since ‘Exascale Computing seems to be easy’ (as stated in literature…)..☺

Problem: *Adaptive Space-time-FEM-Multigrid-like solvers are the candidates from a mathematical point of view*

Why Exascale Computing is ‘easy’…..

- **If parallel efficiency is bad, take a much less efficient serial algorithm**

→ Easier to parallelize

→ More impressive speedups and scaling

- **Show very high “Macho Flop” rates per iteration or time step**

→ Take explicit approaches without solvers of (non)linear systems

- **Never show total CPU time-to-solution and obtained accuracy**

→ Who wants to solve real problems?

→ Only the Macho Flops count

- **Never try to realize complex, but numerically efficient algorithms**

→ Spending millions of CPU hours allows you not to read scientific papers

Fast Poisson FEM-MG-Solvers

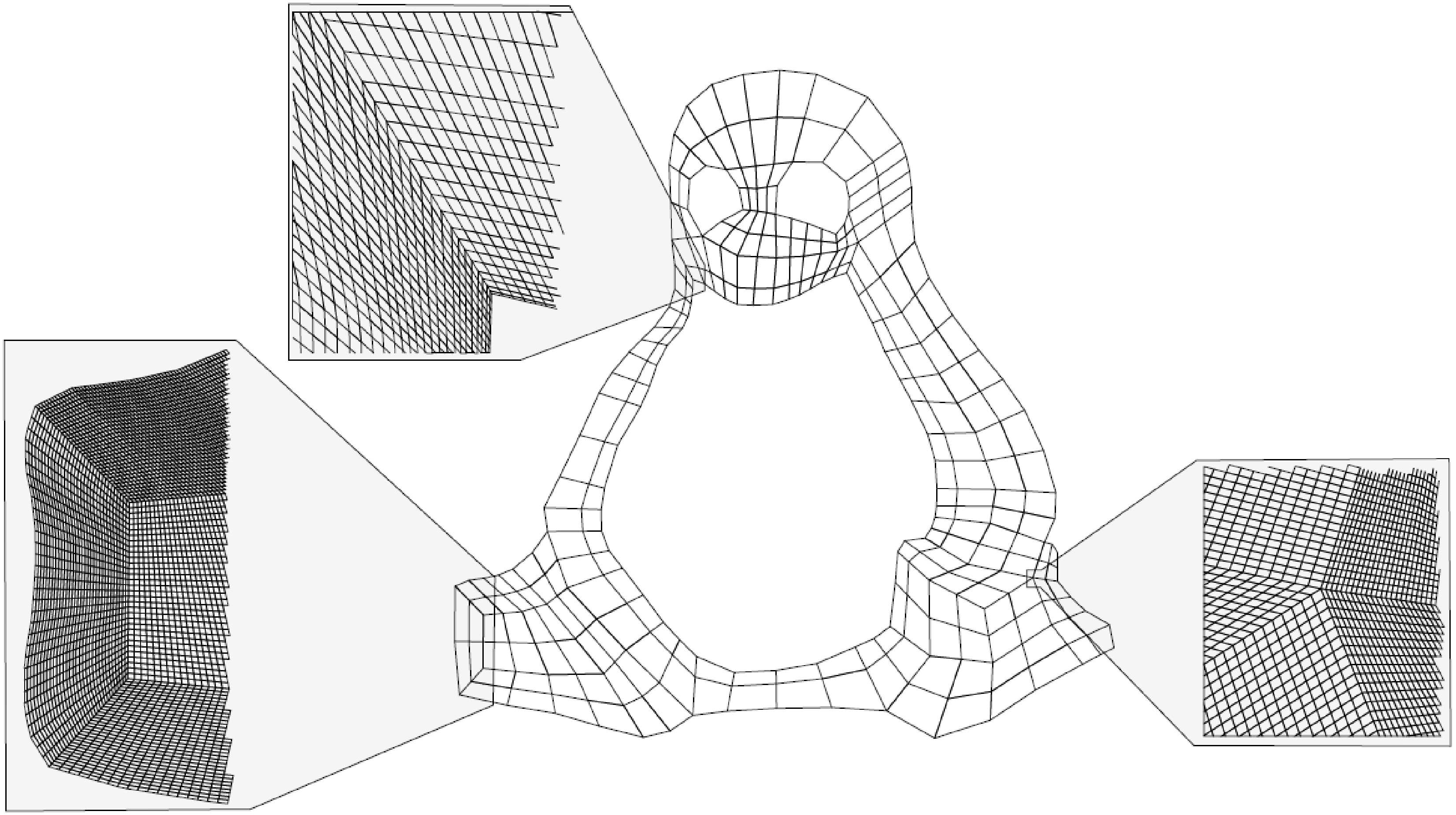
- 'Optimized' Multigrid methods for **scalar PDE** problems (≈Poisson problems) on **general meshes** should require appr. 1000 FLOPs per unknown (in contrast to single-grid Krylov-space methods or direct solvers a la UMFPACK)

- Problem size 10^6 : Much less than 1 sec on PC (???)
- Problem size 10^{15}: Less than 1 sec on ExaFLOP/s computer

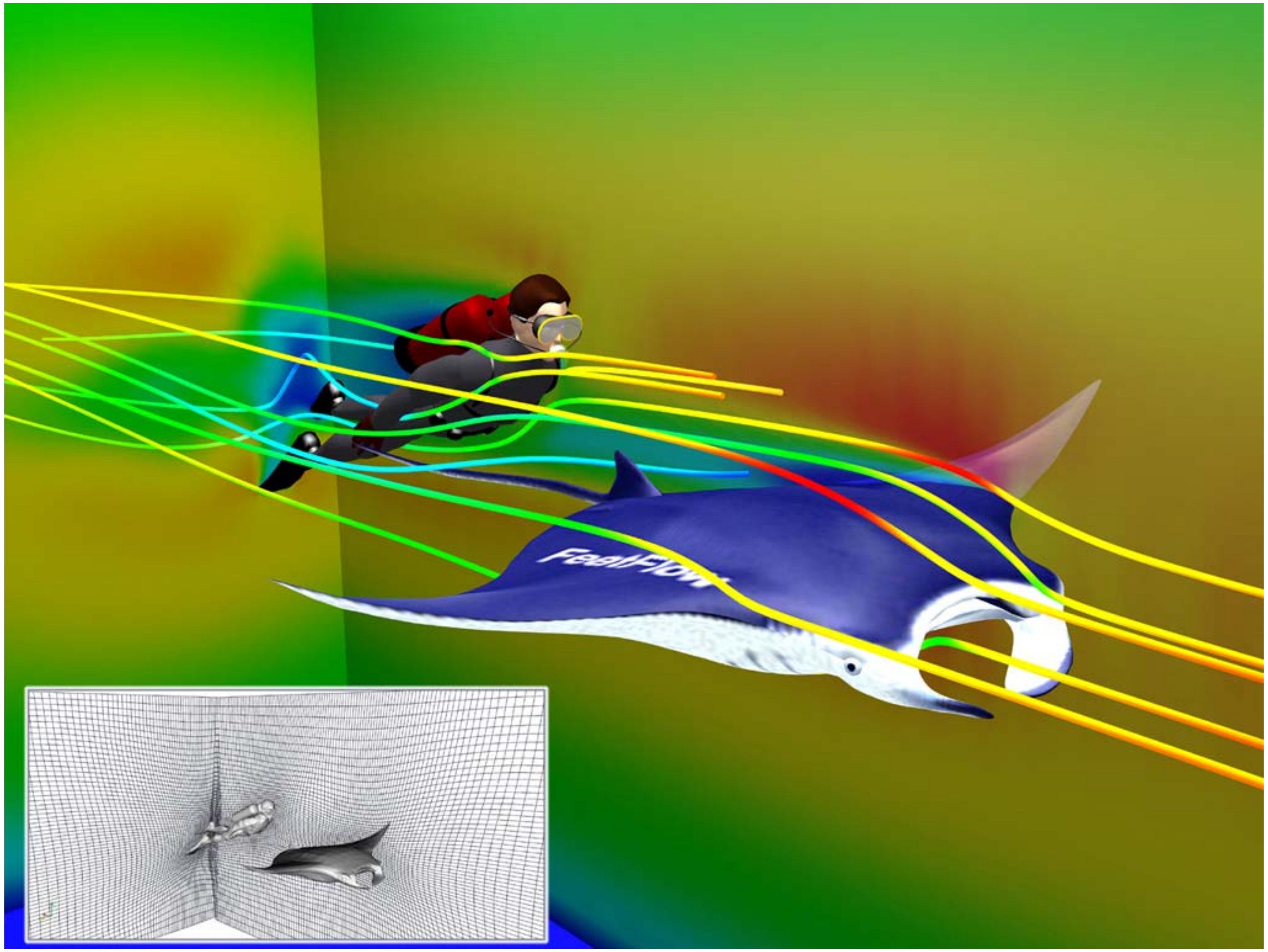
➔ **More realistic (and much harder) 'Criterion' for Exascale Computing in Technical Simulations**

➔ **Necessary component:** Sparse Matrix-Vector applications on **general (= "unstructured") grids**

Unstructured Meshes with locally regular substructures due to hierarchical data structures for adaptive MG-FEM

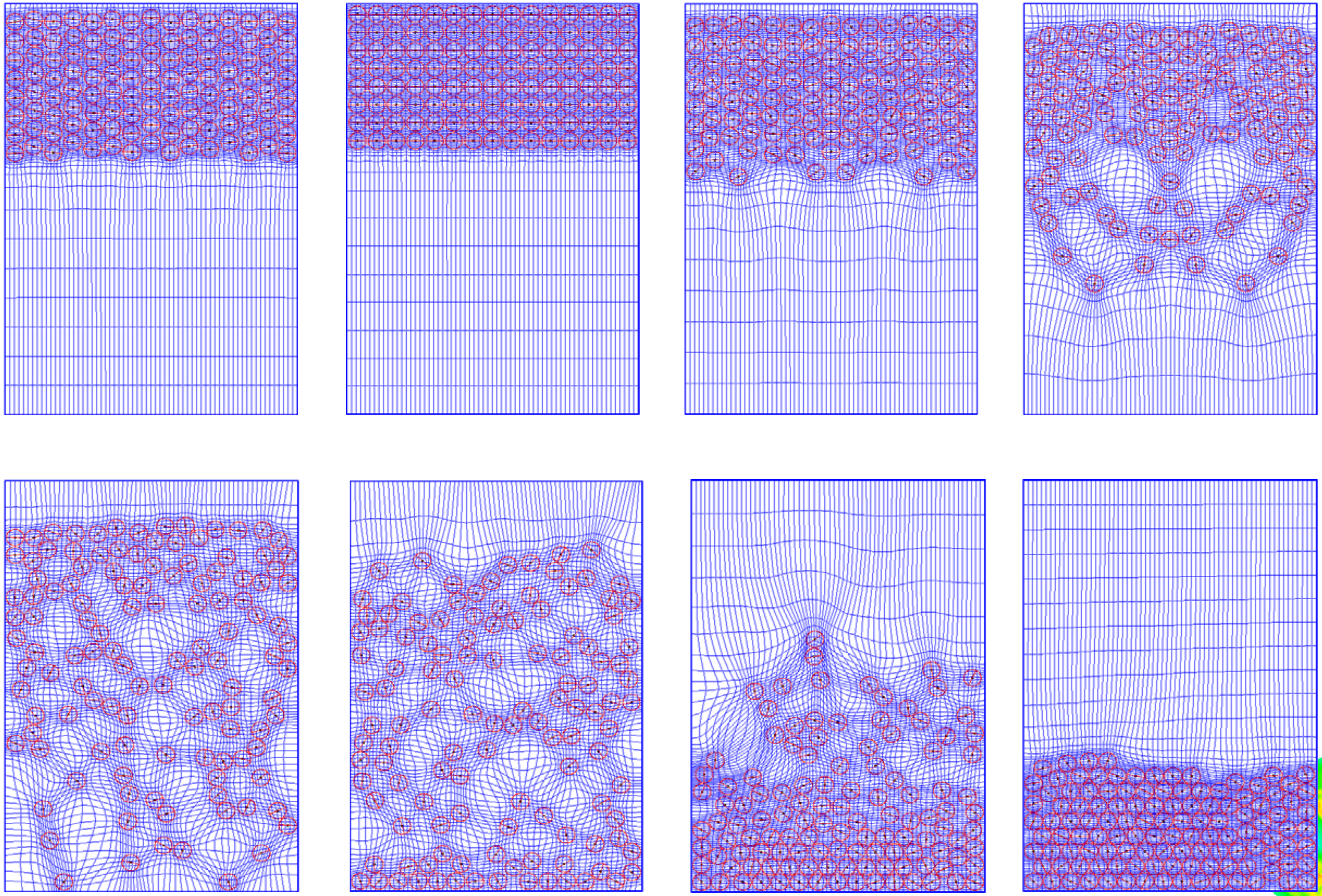


Tensorproduct *(TP) Meshes (static)*

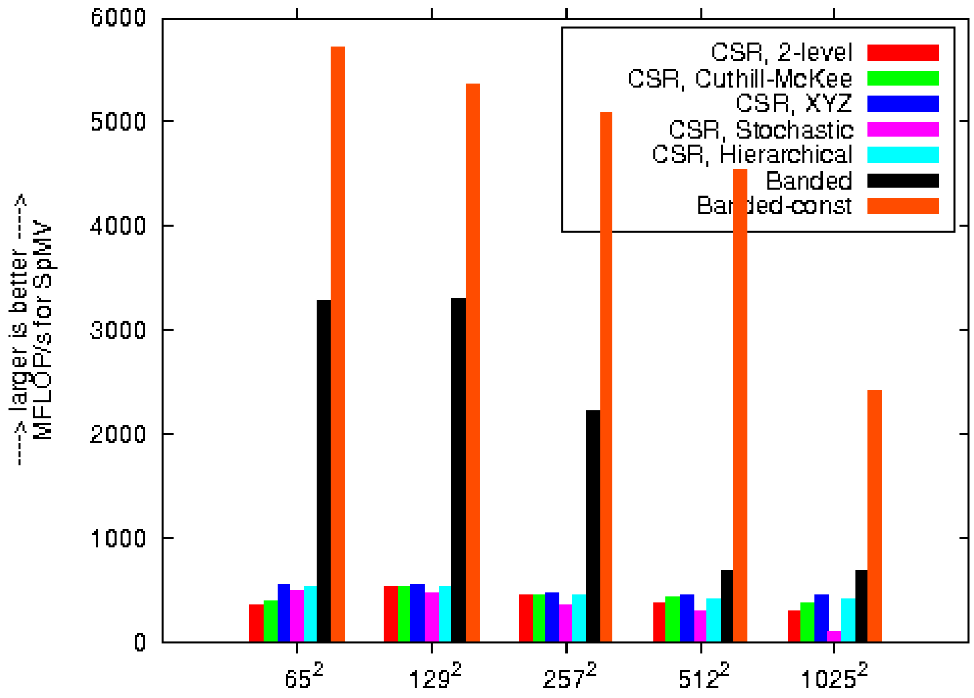


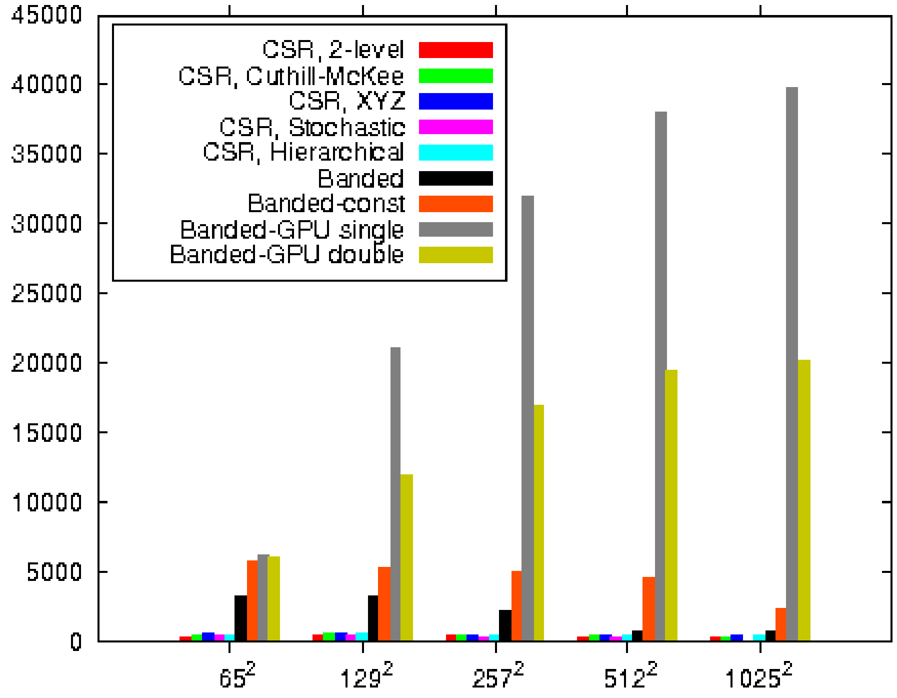
…with **Fictitious Boundary Methods (FBM)** for complex objects

Tensorproduct (TP) Meshes (dynamic)



Sparse MV on TP Grids: Old, but…





0.1 – 0.7/5.8 – 2.4 GFLOP/s

Xeon E5450

20 - 40 GFLOP/s

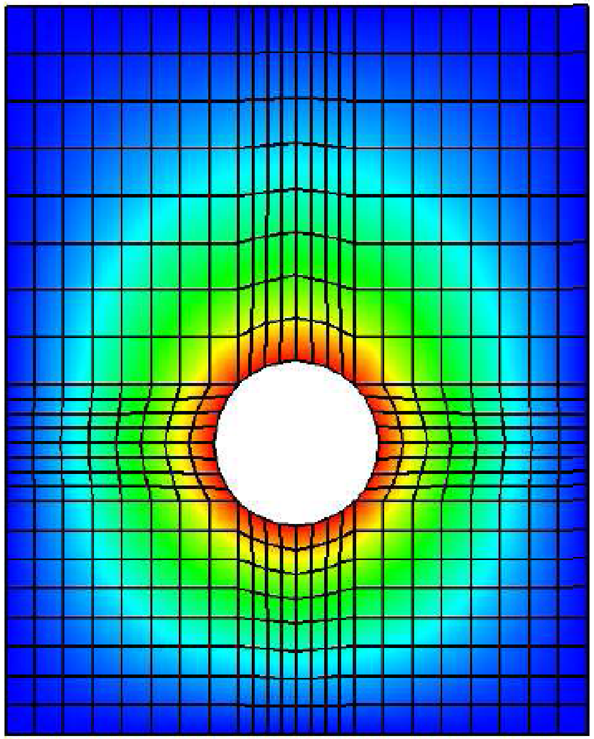
GeForce GTX 280

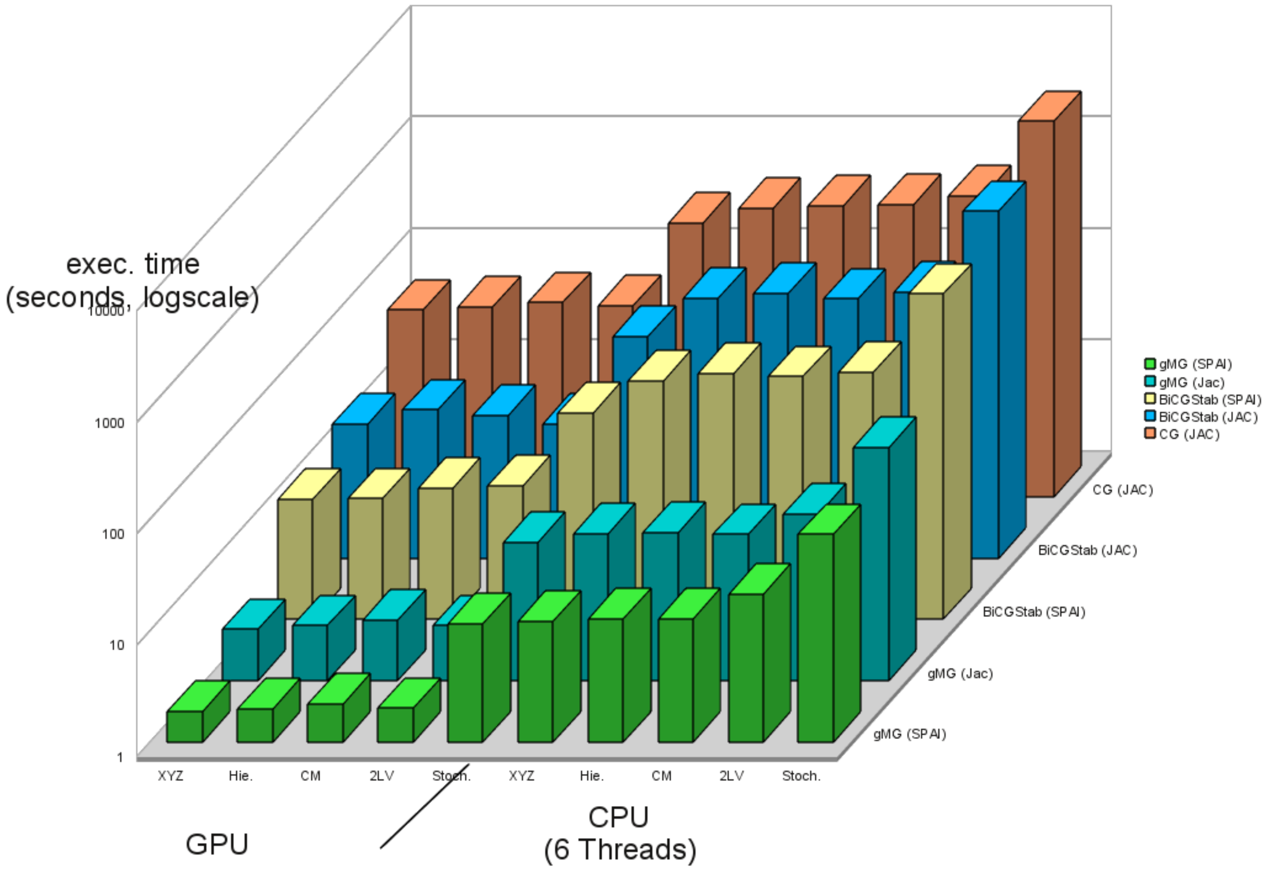
…as typical memory-bound Algorithms

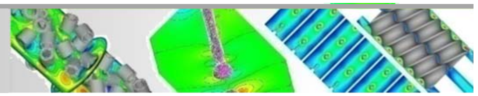
Poisson Solver (Basic) Tests

$$
\begin{aligned}
-\Delta u &= 0 \quad \text{in } \Omega, \\
u &= 0 \quad \text{on } \Gamma_1 \\
u &= 1 \quad \text{on } \Gamma_2
\end{aligned}
$$

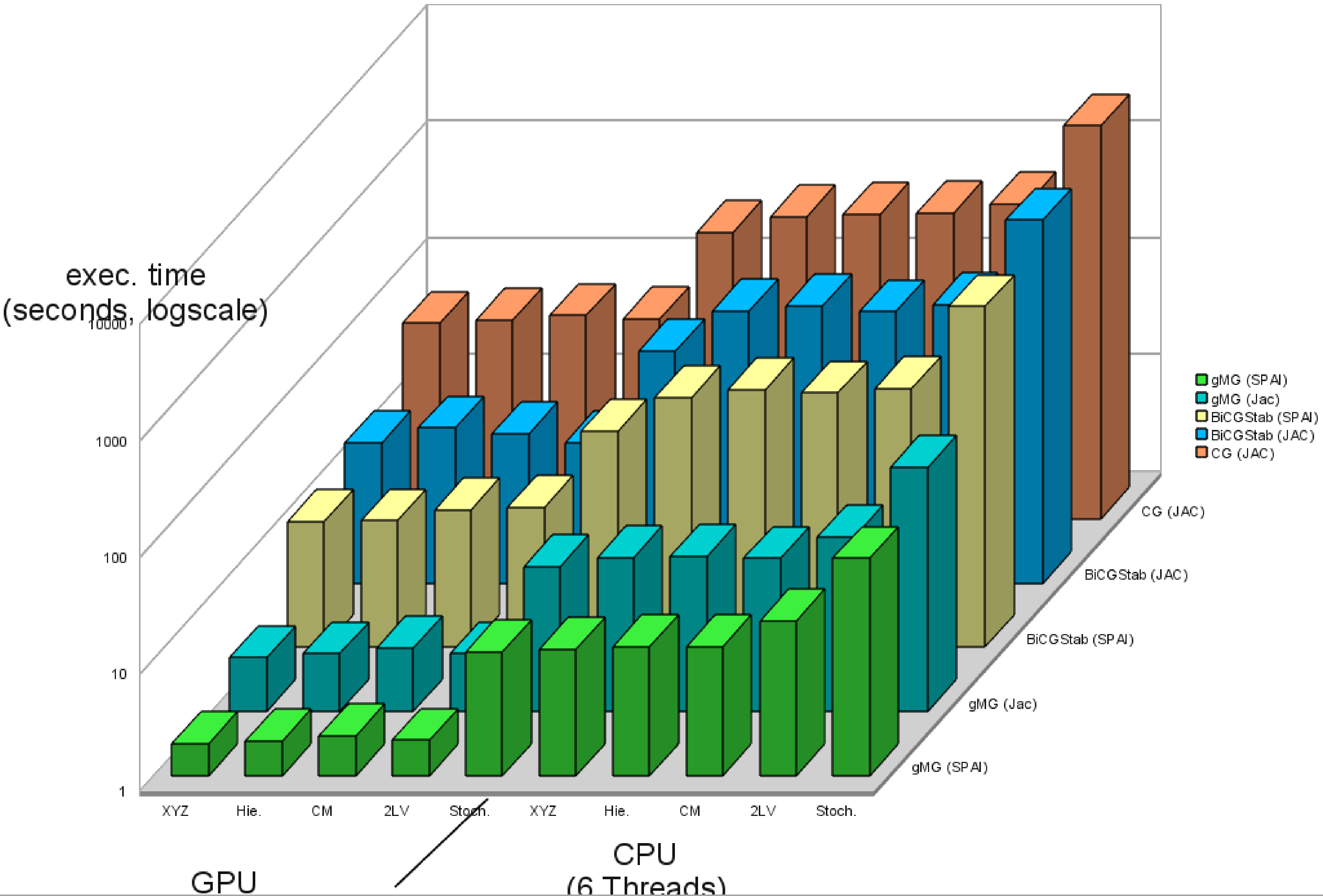
L	Q_1 N	Q_1 non-zeros	Q_2 N	Q_2 non-zeros
4	576	4552	2176	32192
5	2176	18208	8448	128768
6	8448	72832	33280	515072
7	33280	291328	132096	2078720
8	132096	1172480	526336	8351744
9	526336	4704256	2101248	33480704
10	2101248	18845696	-	-







Poisson Solver (Basic) Tests



Poisson Solver Tests

Identical solution, but differences of more than a factor 1000x regarding the CPU time for one „simple“ (small) subproblem after „optimization“ on all levels!

…not enough for TFLOP results (higher order FEM, ScaRC solvers,…)

…this is only 1 component in complex PDE applications…

Summary: Extensive Tests show…..

- Even for `basic problems’ (Poisson solver) the combination of numbering strategies + numerical components + hardware leads to differences in total efficiency of factor 1000x and more on a single node
- (Massively) `Parallel Peak Performance’ with modern Numerics is even harder, already for moderate processor numbers
- Besides the mathematical part, the realization of flexible (and user-friendly?) mathematical software is very challenging
- **Absolute performance ratings are necessary and must take the Numerics into account!**

Some HWON Rules of Thumb (so far…)

- Realize all MG components via sparse MV (preconditioners, grid transfer) & Optimize sparse MV w.r.t. FEM space, numbering and hardware

→ **Generic and hardware-optimized gMG-FEM-BLAS Toolbox**

- Use higher order in time (large time steps) + space (large FEM stencils)

→ **High arithmetic intensity via dominant solution part (→ gMG)**

- Design strongly coupled schemes (globally) with Operator-Splitting components (locally)

→ **Combine (outer) high robustness & (inner) high efficiency**

- Exploit locally regular structures to improve global convergence

→ **Strong local solvers cost nothing & Hide irregularities locally**

→ **Patchwise adaptivity, generalized TP meshes, Grid Deformation, FBM,…**

New HWON Exascale Challenges

- Energy efficiency (see next example…)

- **Parallel in time (PARAREAL-like…)?**

- **Numerical Scientific Computing regarding dynamically changing heterogeneous hardware?**

→ Asynchronicity and resilience

→ Robustness of numerical solvers

→ Numerical loadbalancing vs. scheduling

- **Uncertainty Quantification including stochastic effects due to**

→ **Numerical errors**

→ **Modeling errors**

→ **Propagation of input uncertainty**

- **Realization as flexible, re-usable, scalable software (SPP EXA)?**

Wir bauen den 'grünsten' Supercomputer der Welt!

→ 80qm Photovoltaik-Solarpaneele: Mathe-Dach
→ 2t Gel-Blei Solar-Akkus
→ 3 kW Leistung / 18 TFlop/s
→ Speicherung, Pufferung, Wandlung, Messung:
modernste Technik
→ keine Energie-Folgekosten

Mobilprozessoren + HWON machen es möglich

→ NVIDIA Tegra K1
→ sparsam, effizient
→ günstig
→ experimentell!

Für Studierende, die 'basteln' wollen ...nicht nur mathematisch!

HWON Challenges (I) – Basic Level

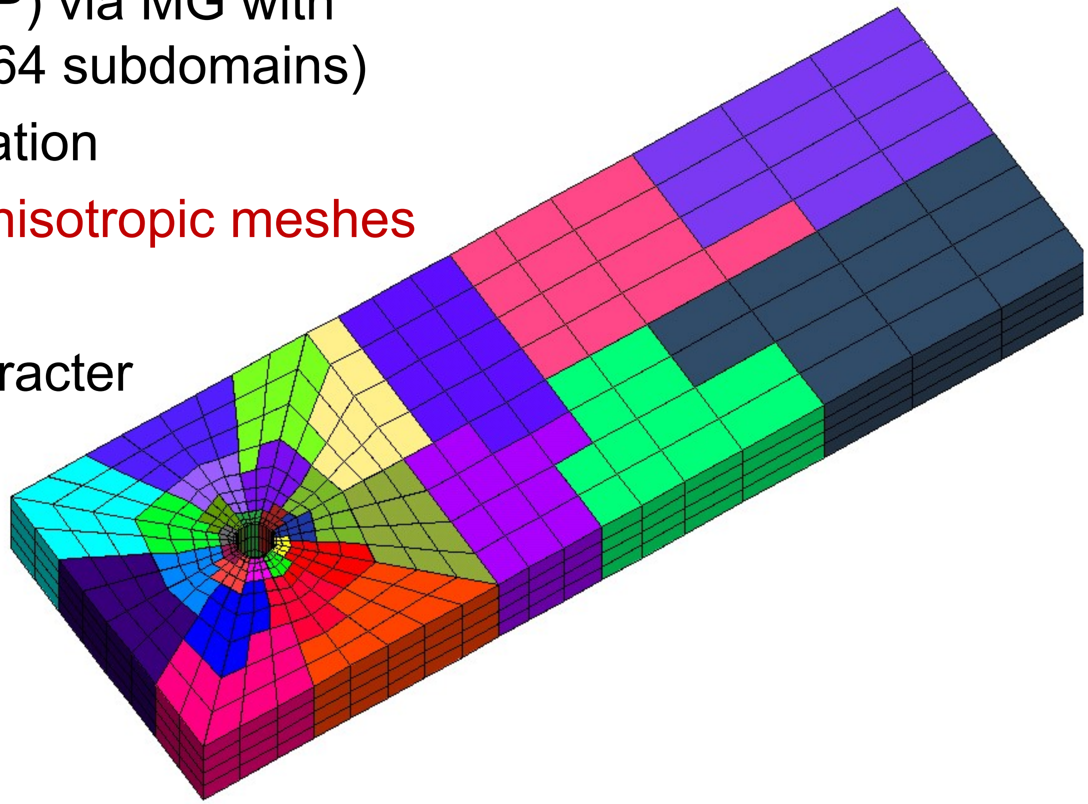
- **Strong ILU-like smoothers?**
  - ► ILU directly on GPUs?
  - ► SPAI – FSAI – AINV: Numerical properties?
  - ► Exploiting local structures: Linelet-GS, linewise GS-ADI?
  - ► 3D ???
- **Basic components for different FEM?**
  - ► Optimal numbering for nonconforming FEM?
  - ► FEM-adapted grid transfer via sparse MV?
- **Realization of a FEM-gMG library**
  - ► BLAS-like: Generic vs. Hardware-optimized?

Parallel Performance

- Pressure Poisson Problem (PPP) via MG with **blockwise** ILU smoothing (1 – 64 subdomains)
  - ► Problems due to communication
  - ► Numerical problems w.r.t. anisotropic meshes

→ Increasing block-Jacobi character

→ ScaRC as hierarchically clustered recursive MG-DD solver



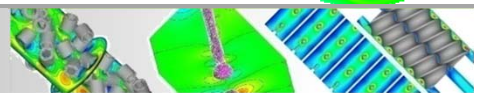
	1 P.	2 P.	4 P.	8 P.	16 P.	32 P.	64 P.
%Comm.	10%	24%	36%	45%	47%	55%	56%
# PPP-IT	2.2	3.0	3.9	4.9	5.2	5.7	6.2

HWON Challenges (II) – Advanced

- **Scalable (= robust & efficient) parallel solvers?**
  - Globally unstructured – locally structured
  - Exploit structured subdomains for scalable efficiency
  - Hide anisotropies locally to increase global robustness
  - Higher local arihtmetic costs, but less global communication

- **(Recursive) solver expert system?**
  - ► numerical + computational a priori knowledge!

- **Load balancing?**
  - ► due to 'total CPU time per accuracy per processor'?
  - ► dynamical a posteriori process?



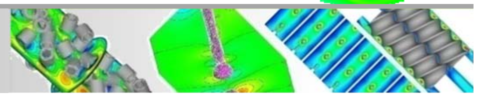
- **Adaptive meshing & complex (time dependent) geometries**
  - ► Grid Deformation: Flexible deformation & preserving logical structures
  - ► Fictitious Boundary Method as filter process for geometrical details

- **Coupling mechanisms**
  - ► Decoupled vs. Fully Coupled
  - ► Monolithic vs. Segregated

  → **Design new algorithms due to high arithmetic intensity**

CPU(Solver)	Method	#NT	Lift		Drag	
			mean	peak	mean	peak
14,358(81%)	Impl. MPSC	39	1%	1%	0%	2%
42,679(51%)	Semi-impl. DPM	165	0%	0%	0%	0%
64,485(54%)	Semi-expl. DPM	889	0%	8%	0%	0%

- **Higher order discretization in space and time**
  - ► Higher order time stepping schemes for increasing the solution part
  - ► Higher order FEM for more dense matrices

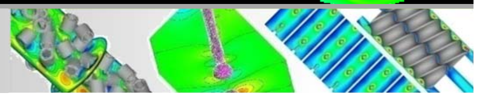


HWON Challenges (IV) – Benchmarks

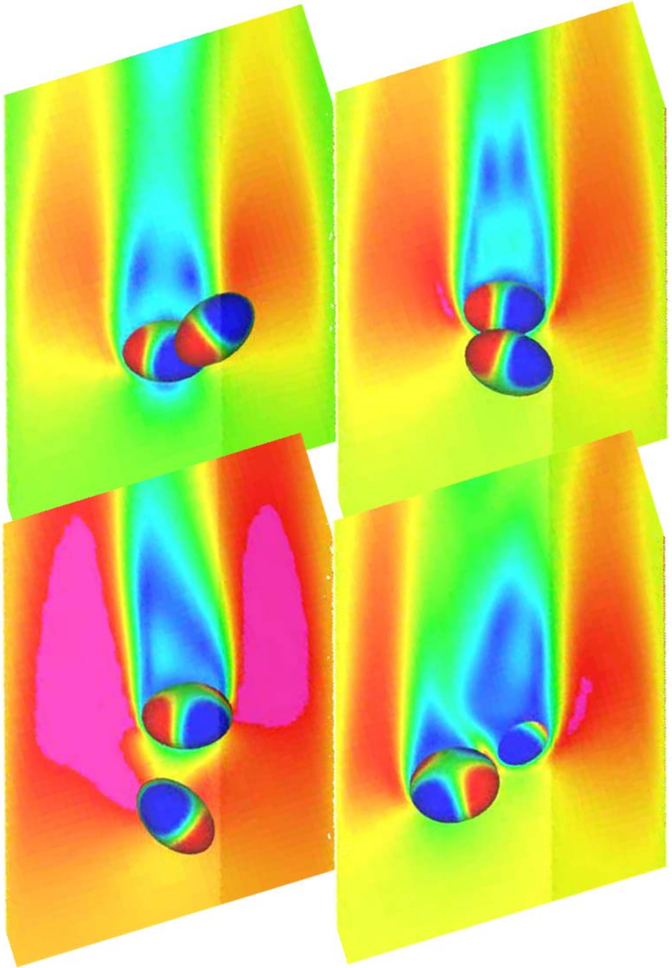
- **How to define benchmarking scenarios which allow to measure the absolute performance???**
- **We have to consider absolute timings w.r.t. (virtually) optimal algorithms!**

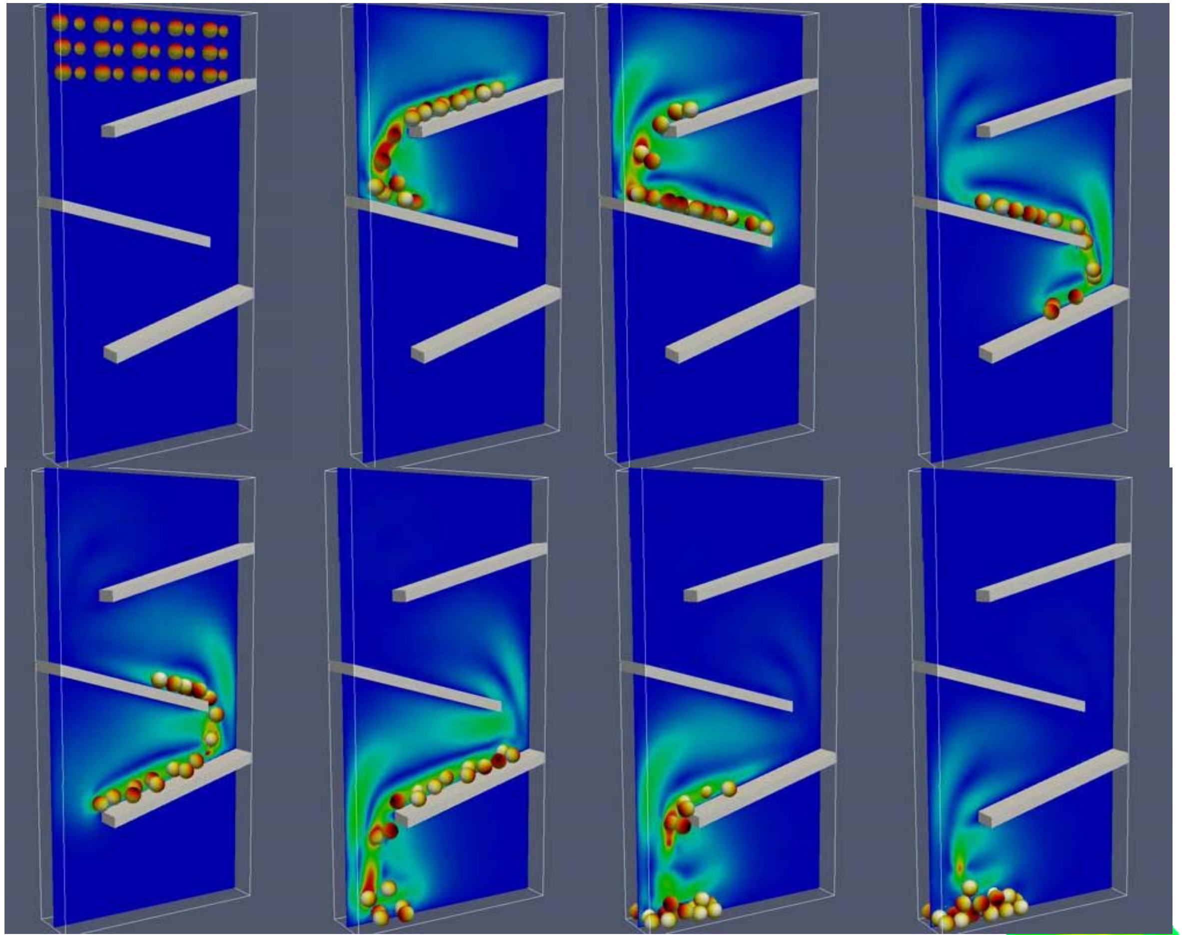
Summary: Extensive Tests show…..

- Even for `basic problems' (Poisson solver) the combination of numbering strategies + numerical components + hardware leads to differences in total efficiency of factor 1000x and more
- `Parallel Peak Performance' with modern Numerics is even harder, already for moderate processor numbers
- Besides the mathematical part, the realization of flexible (and user-friendly?) mathematical software is very challenging
- **Absolute performance ratings are necessary!**
- Applying HWON to complex algorithms and applications is another story…



Application to Multiphase Flow





FeatFlow-PRO (by IANUS Simulation)

HPC features:

- Moderately parallel
- GPU computing
- Open source

Numerical features:

- Higher order (Q2P1) FEM schemes in space
- Implicit FD/FEM in time
- FCT & EO FEM stabilization techniques
- Use of semi-unstructured meshes
- Dynamic adaptive grid deformation
- **Fictitious Boundary (FBM) methods**
- Newton-Multigrid solvers

Non-Newtonian flow module:

- generalized Newtonian model (Power-law, Carreau,...)
- viscoelastic model (Giesekus, FENE, Oldroyd,...)

Multiphase flow module (resolved interfaces):

- ℓ/ℓ – interface tracking (Level Set)
- s/ℓ – interface capturing (FBM)
- $s/\ell/\ell$ – combination of ℓ/ℓ and s/ℓ

Engineering aspects:

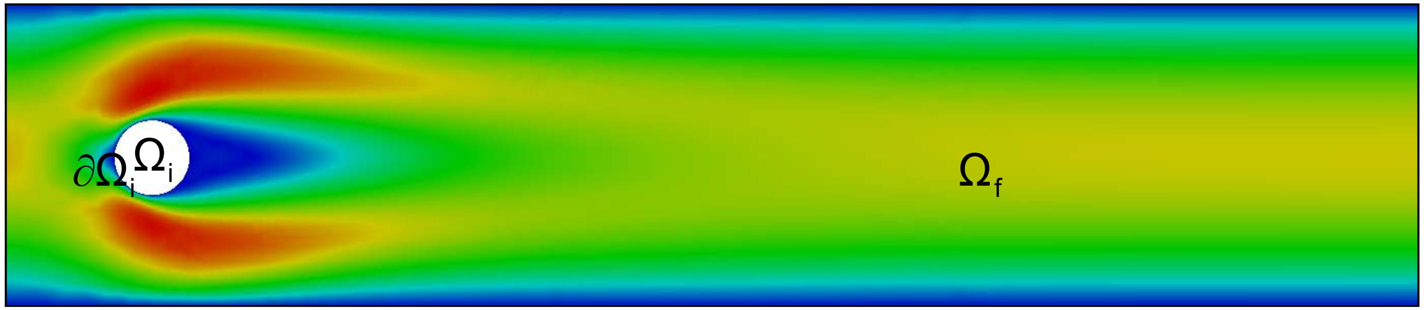
- Geometrical design
- Modulation strategy
- Optimization

FFpro: 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:

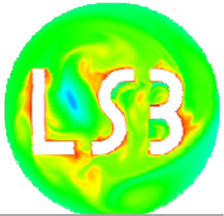
$$\sigma(X,t) = -pI + \mu[\nabla u + (\nabla u)^T]$$

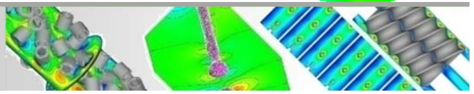
Equations of Motion (I)

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





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

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

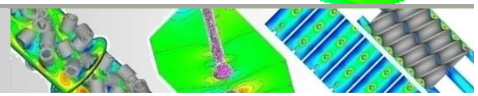
which can be done numerically by an explicit Euler scheme:

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

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

Solve for velocity and pressure applying FBM-conditions

$$\mathrm{NSE}\left(u_f^{n+1}, p^{n+1}\right) = \mathrm{BC}\left(\Omega_i^n, u_i^n\right)$$

Calculate hydrodynamic force, torque and apply

$$F_i^{n+1} \qquad T_i^{n+1}$$

Contact force calculation

$$F_{c,i}^{n+1}$$

Compute new velocity and angular velocity

$$u_i^{n+1} = u_i^n + \Delta t \left(F_{c,i}^{n+1}/M_i\right) \qquad \omega_i^{n+1} = \omega_i^n + \Delta t I_i^{-1}\left(r \times F_{c,i}\right)$$

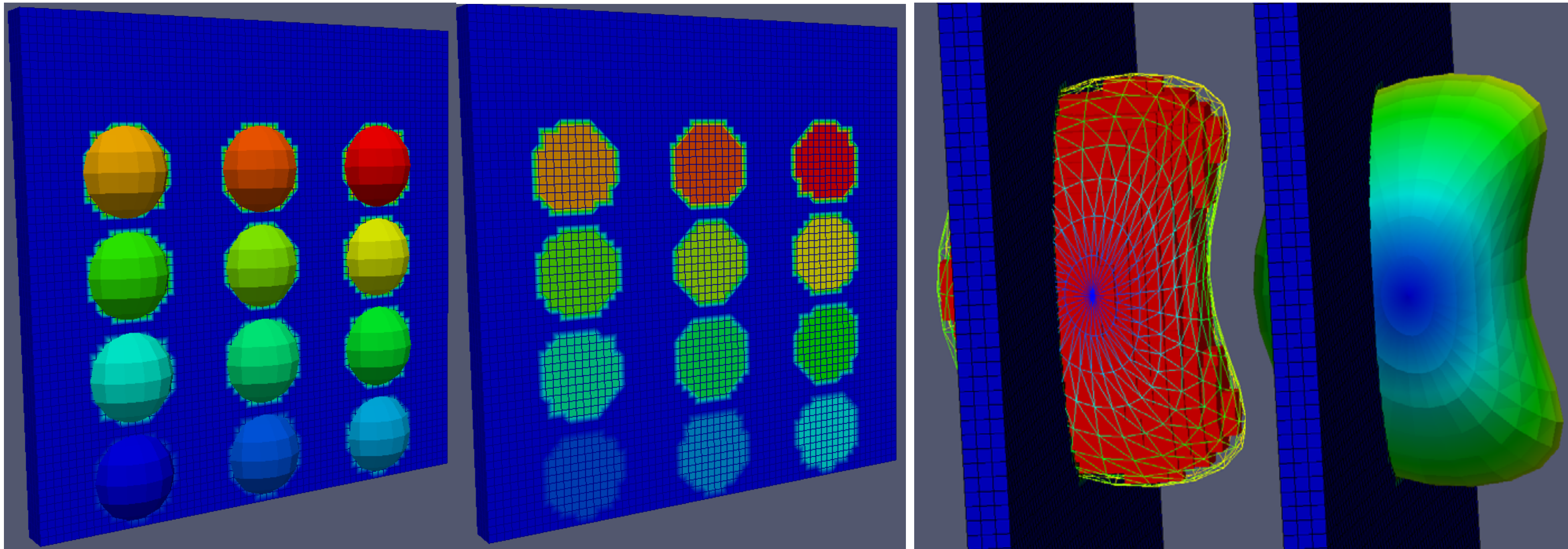
Position update

$$X_i^{n+1} = X_i^n + \Delta t u_i^n \qquad \theta_i^{n+1} = \theta_i^n + \Delta t \omega_i^n$$

Fictitious Boundary Method

Eulerian Approach:

- FBM = special case of (scaled) Penalty method
- Internal 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 using mesh adaptation techniques



Dynamic ALE-Mesh Adaptation

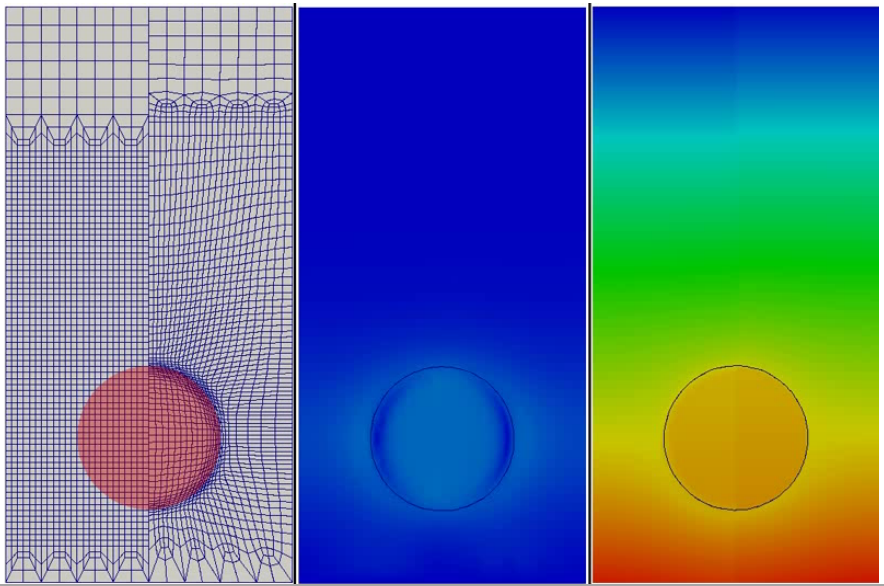
Advantages:

- Constant mesh/data structure → **GPU**
- Increased resolution in regions of interest
- PDE approach ←→ anisotropic ‘umbrella’ smoother, snapping/projection
- Straightforward usage on 3D unstructured meshes

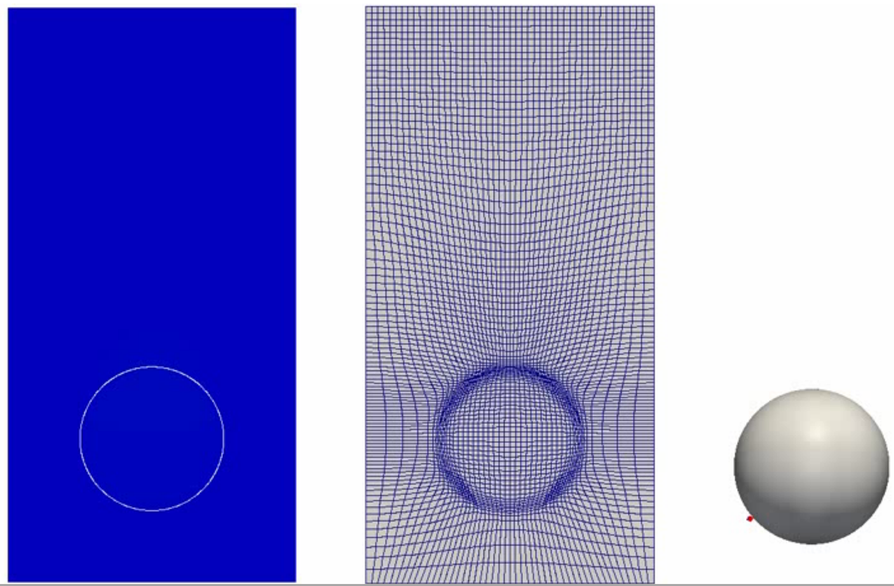
Quality of the method depends on the construction of the monitor function

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

Validation: 2.5D Rising bubble – light setup

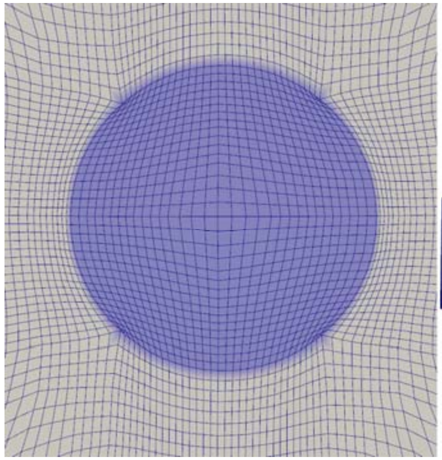


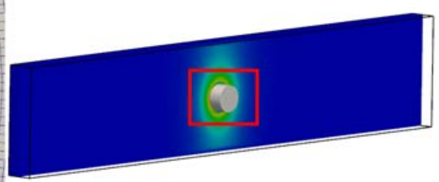
Testing: 3D Rising bubble - hard setup

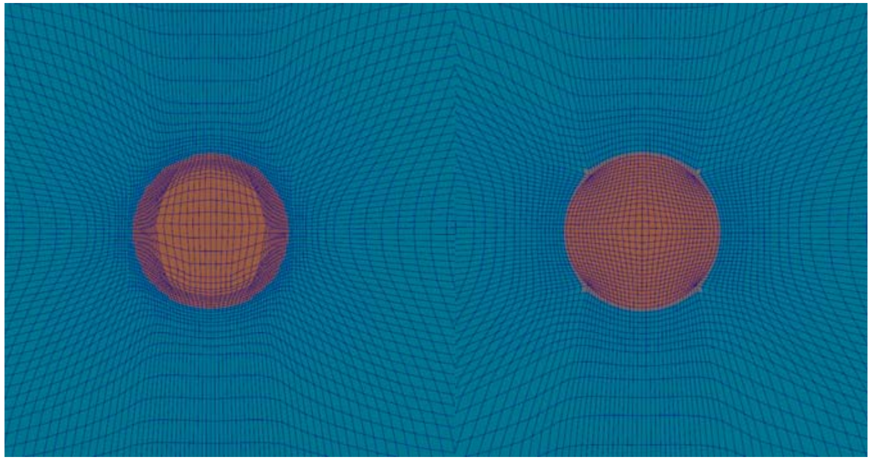


Oscillating Cylinder

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



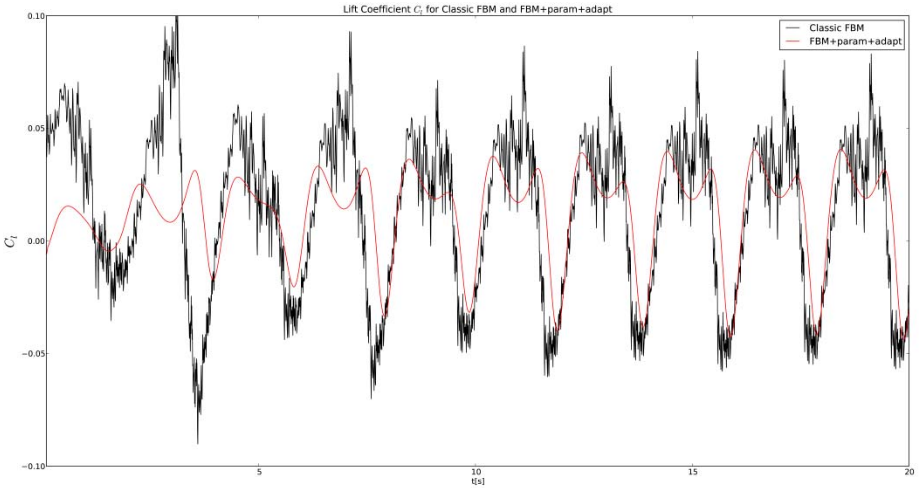


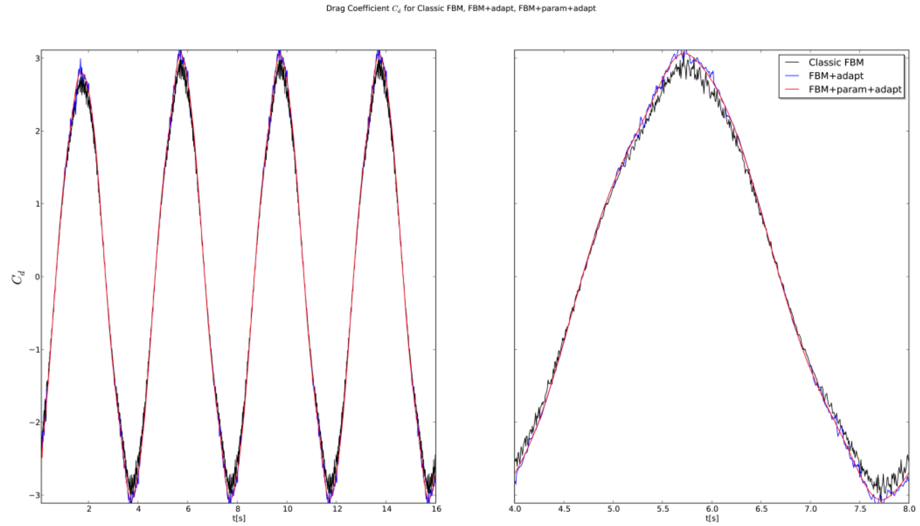


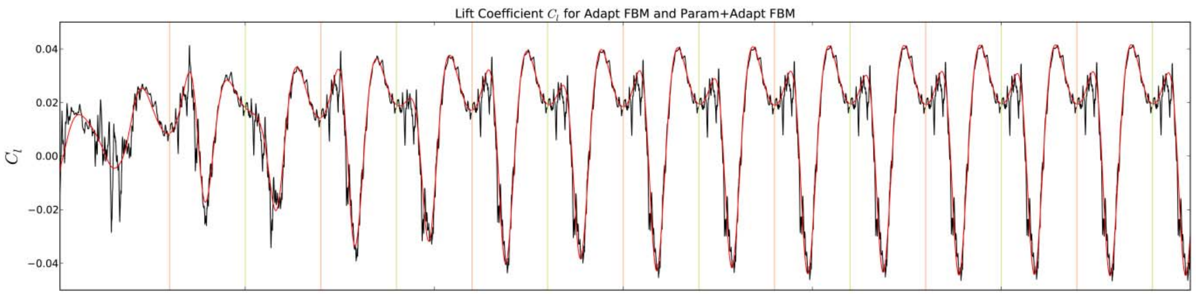
Nodes concentrated near liquid-solid interface

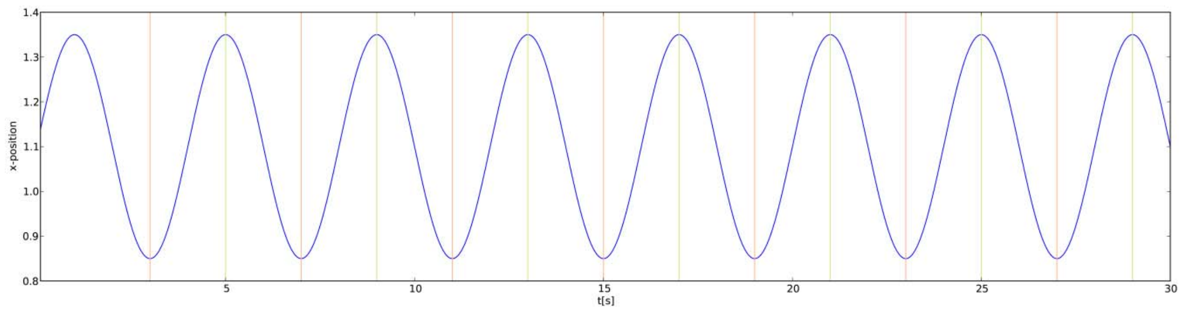
Nodes projected and parametrized on boundary plus concentration of nodes near boundary

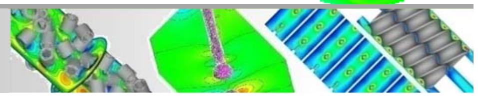
Oscillating Cylinder Results



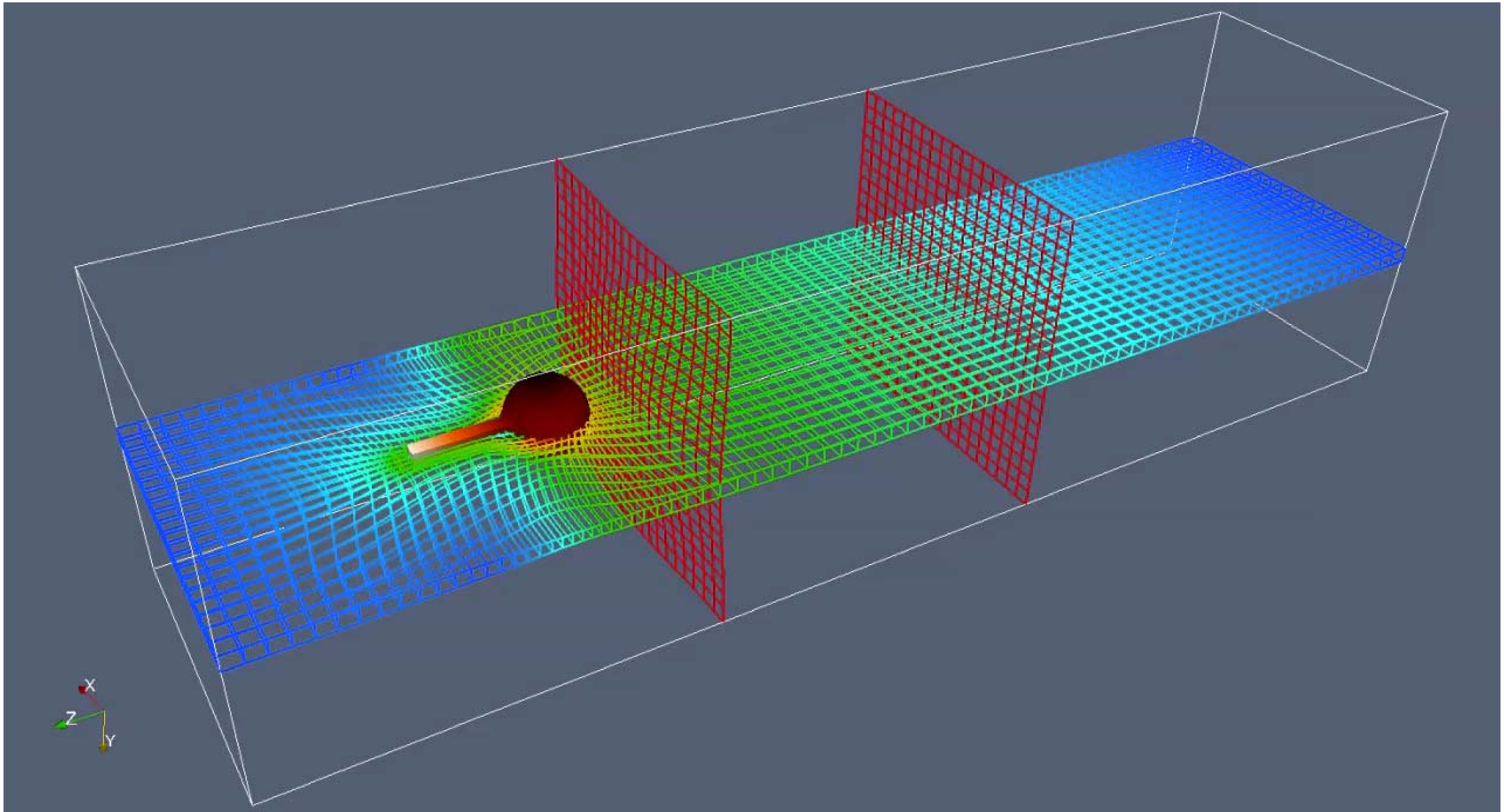




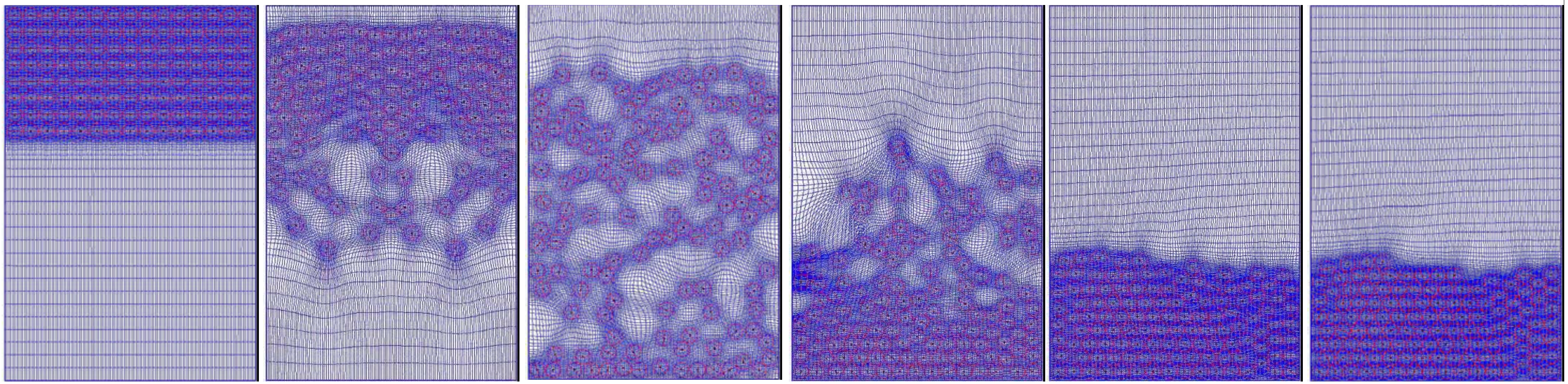


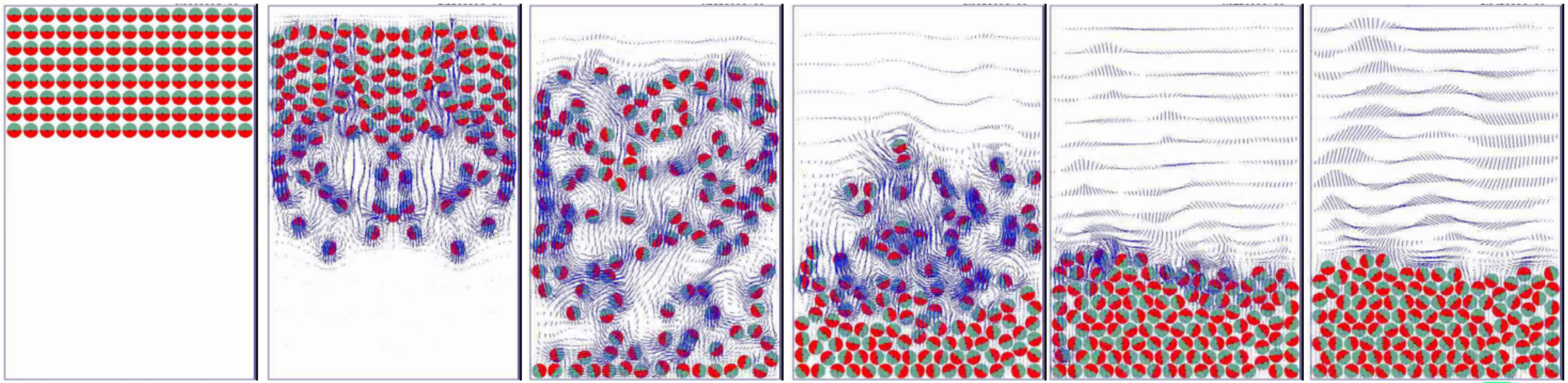


(Passive) Microswimmer



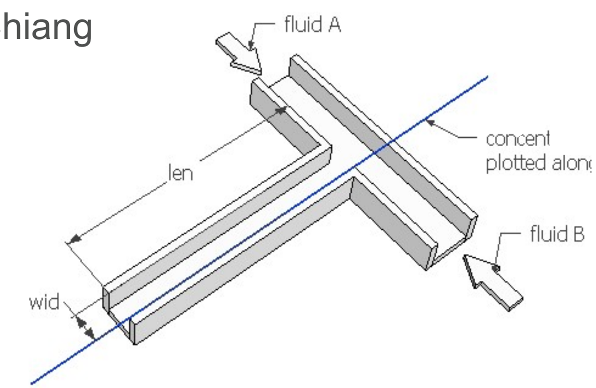
Sedimentation





Chemical Reactions

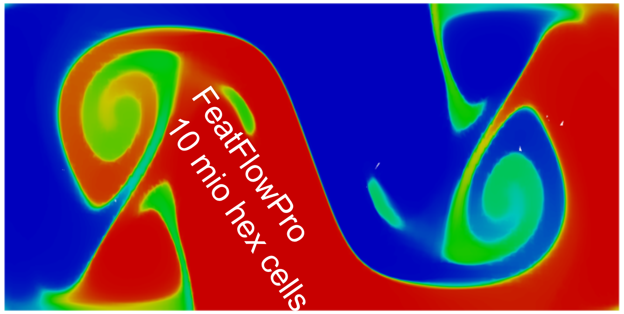
$A + B \xrightarrow{k\to\infty} P$ $\partial_t\phi + u\cdot\nabla\phi = \nabla\cdot(d\nabla\phi)$ with $\phi = c_A\text{-}c_B$ Toor and Chiang



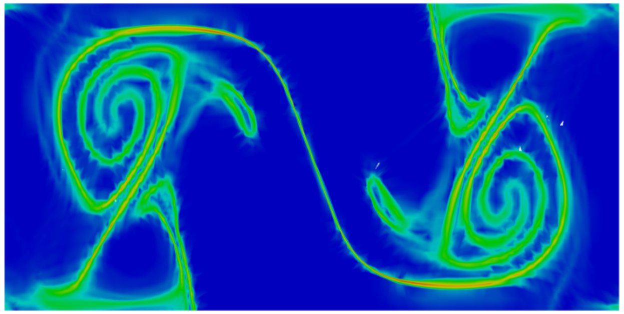
Challenges:

- Extreme resolution requirements → AFC + GD
- Extremely different time-scales → Operator splitting
- Micromixing–subgrid mixing models → MEMM (Fox et al.)

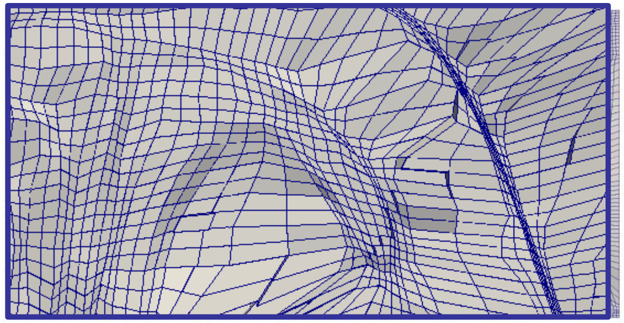
Transported scalar field ϕ



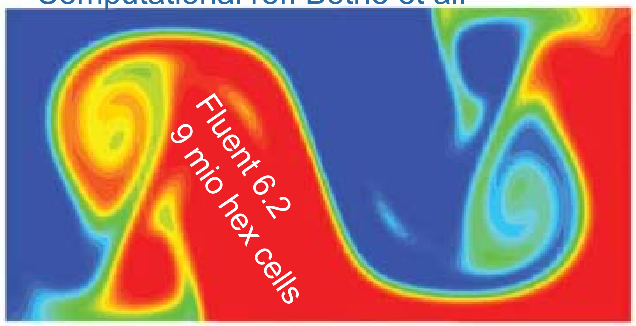
Monitor function

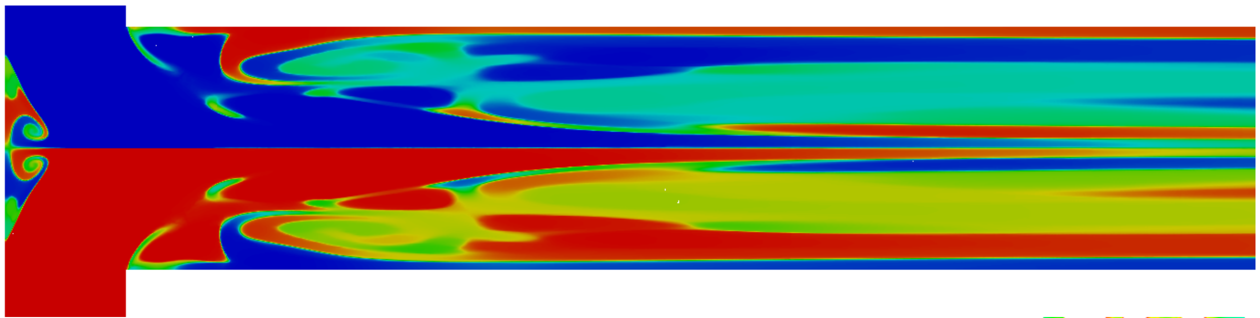


Computational mesh



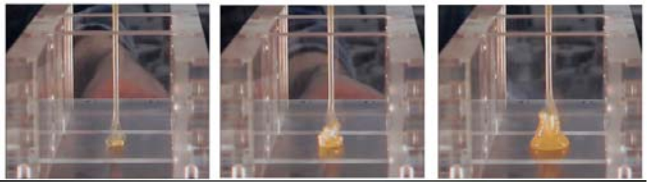
Computational ref. Bothe et al.





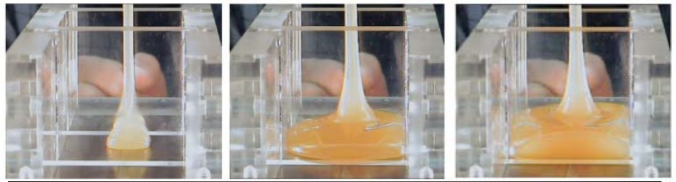
Viscous Liquid Jets

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

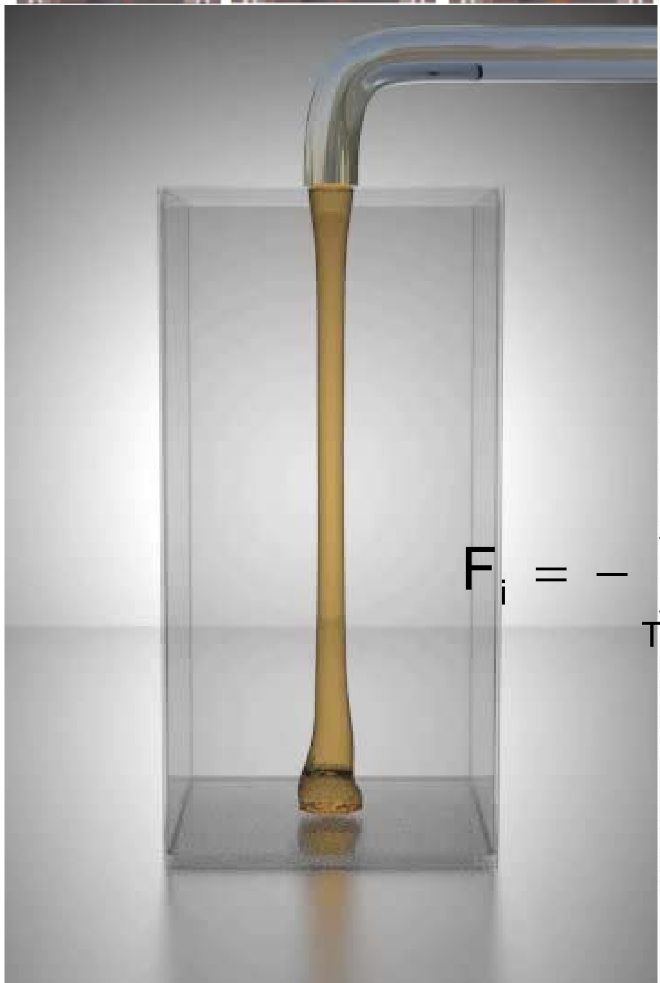


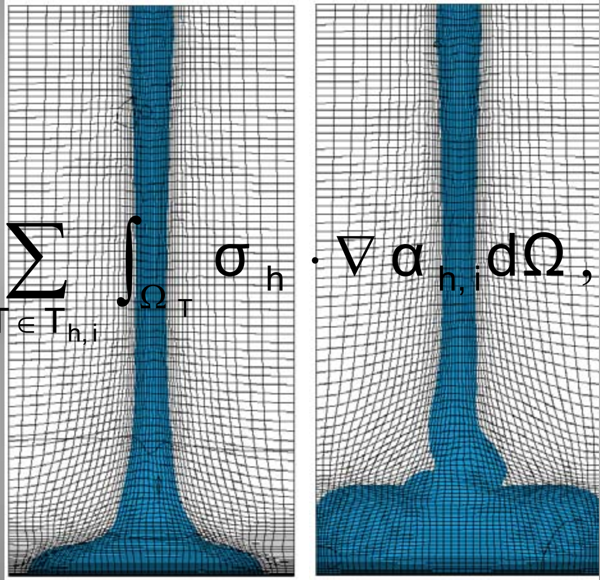
Corn syrup-air system

24x24x48 mesh



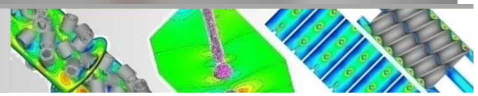
Interface triangulation:
T_0: ~100,000 triangles
T_N: ~300,000 triangles

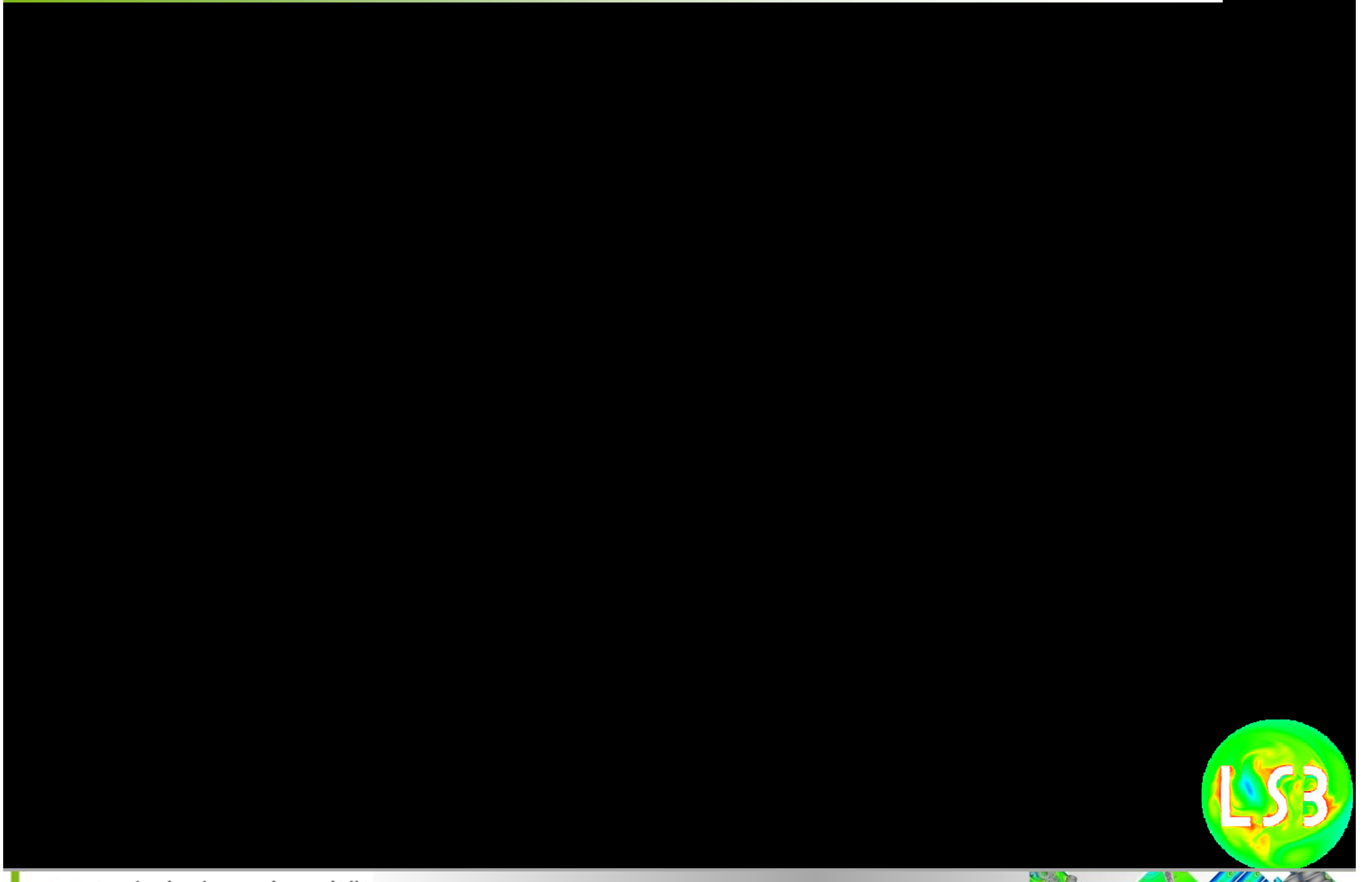


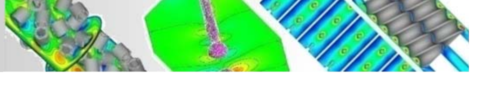


$$F_i = -\sum_{T \in T_{h,i}} \int_{\Omega_T} \sigma_h \cdot \nabla \alpha_{h,i} \, d\Omega,$$

Rendering: Raphael Münster / Blender







Hydrodynamic Forces

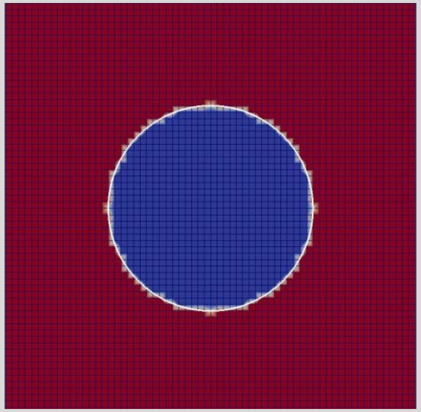
Hydrodynamic force and torque acting on the i-th particle

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

Force Calculation with Fictitious Boundary Method

<u>Problems: The FBM can only decide:</u>

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



Alternative:

Replace the surface integral by a volume integral

Numerical Force Evaluation (I)

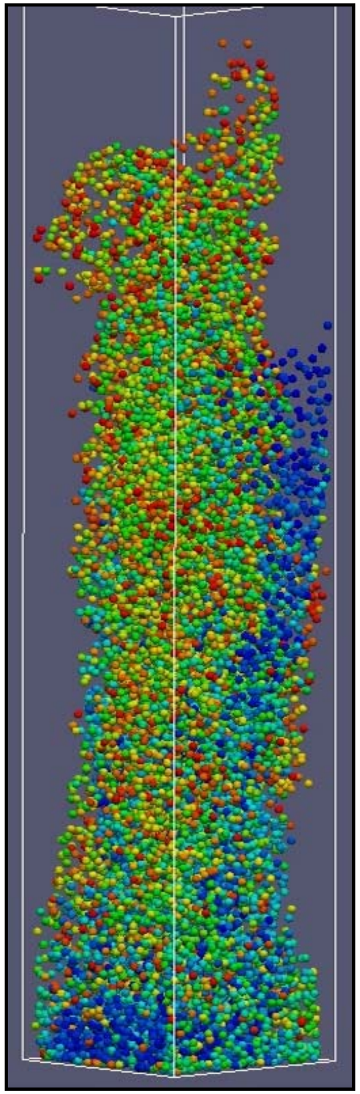
Define an ***indicator function*** α_i:

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

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:

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

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

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

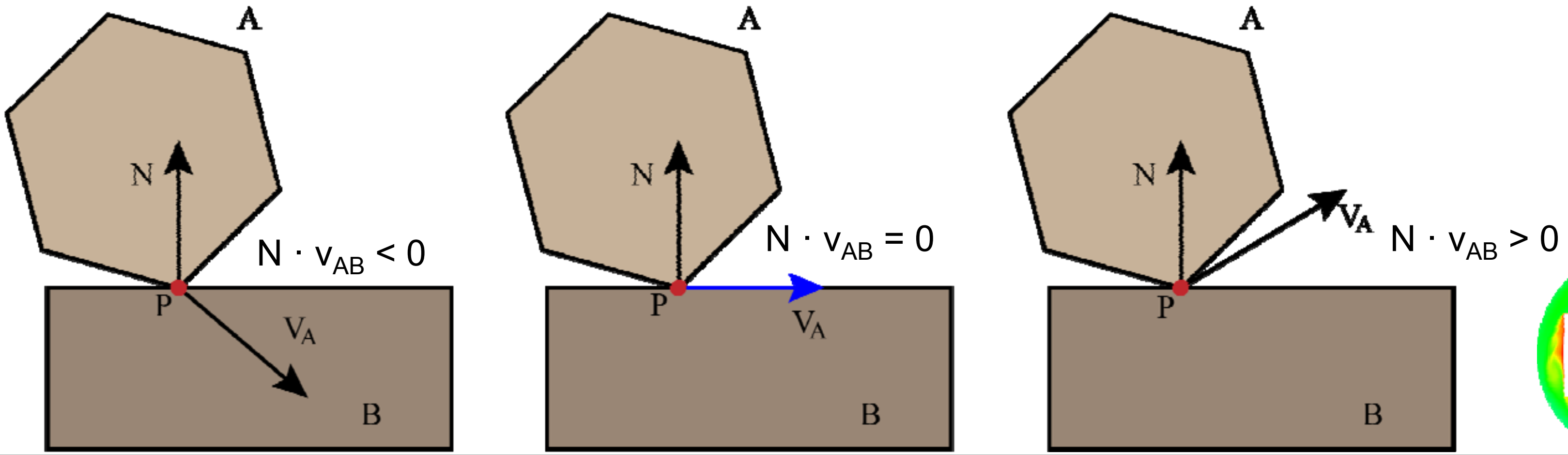
$$T_i = -\sum_{T \in T_{h,i}} \int_{\Omega_T} (X - X_i) \times \left(\sigma_h \cdot \nabla \alpha_{h,i}\right) d\Omega$$

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

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

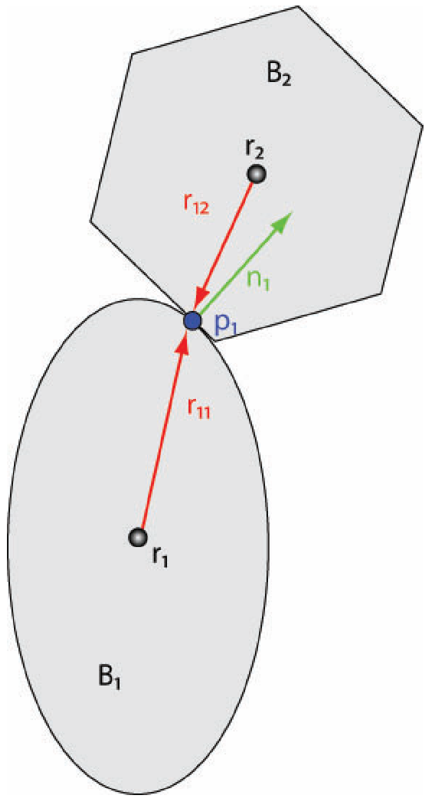
Contact/Collision Modelling

- **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 $N \cdot (v_A + \omega_A \times r_A - (v_B + \omega_B \times r_B))$
- 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

- Contact force calculation realized as a three step process
  → **Broadphase**
  → **Narrowphase**
  → 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



► Special single, resp., multibody collision models

Single Body Collision Model

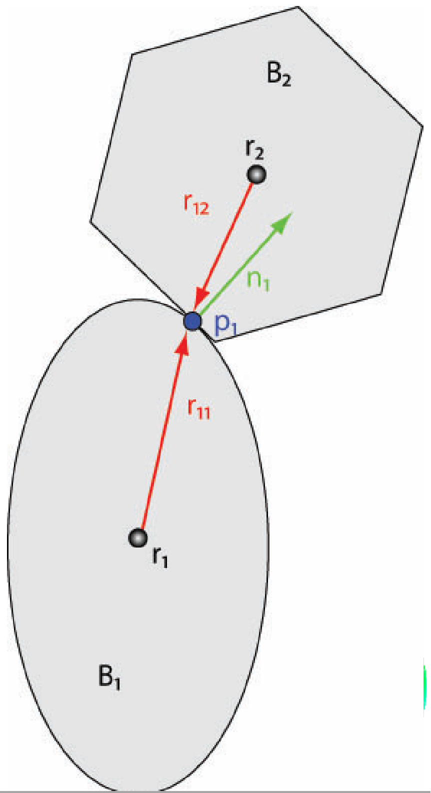
For a single pair of colliding bodies we compute the impulse f that causes the velocities of the bodies to change:

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

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.)

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

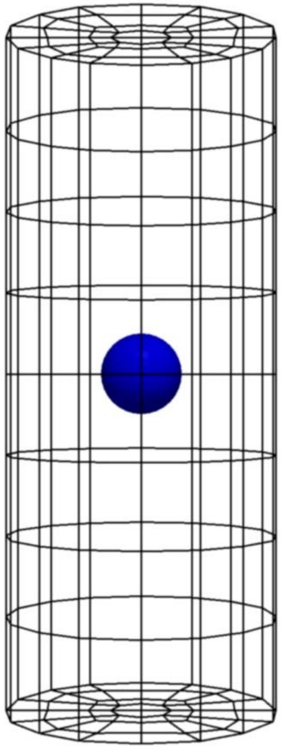
A problem of this form is called a **Linear Complementarity Problem (LCP)** which can be solved with efficient iterative methods like the **Projected Gauss-Seidel solver (PGS)**.

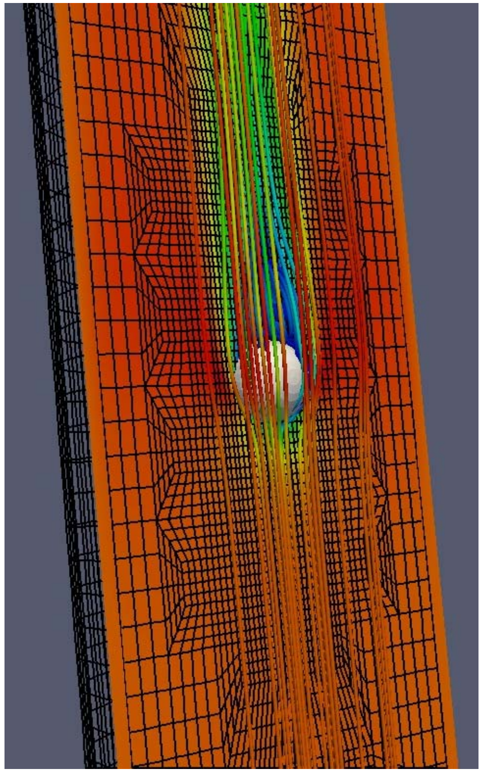
Kenny Erleben,*Stable, Robust, and Versatile Multibody Dynamics Animation*

Benchmarking and Validation (I)

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

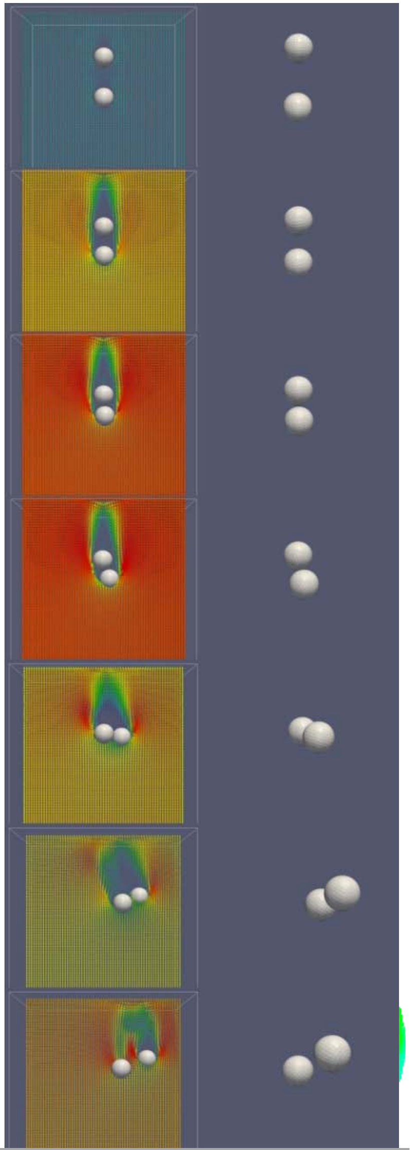
$d_s = 0.3, \quad \rho_s = 1.02$

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

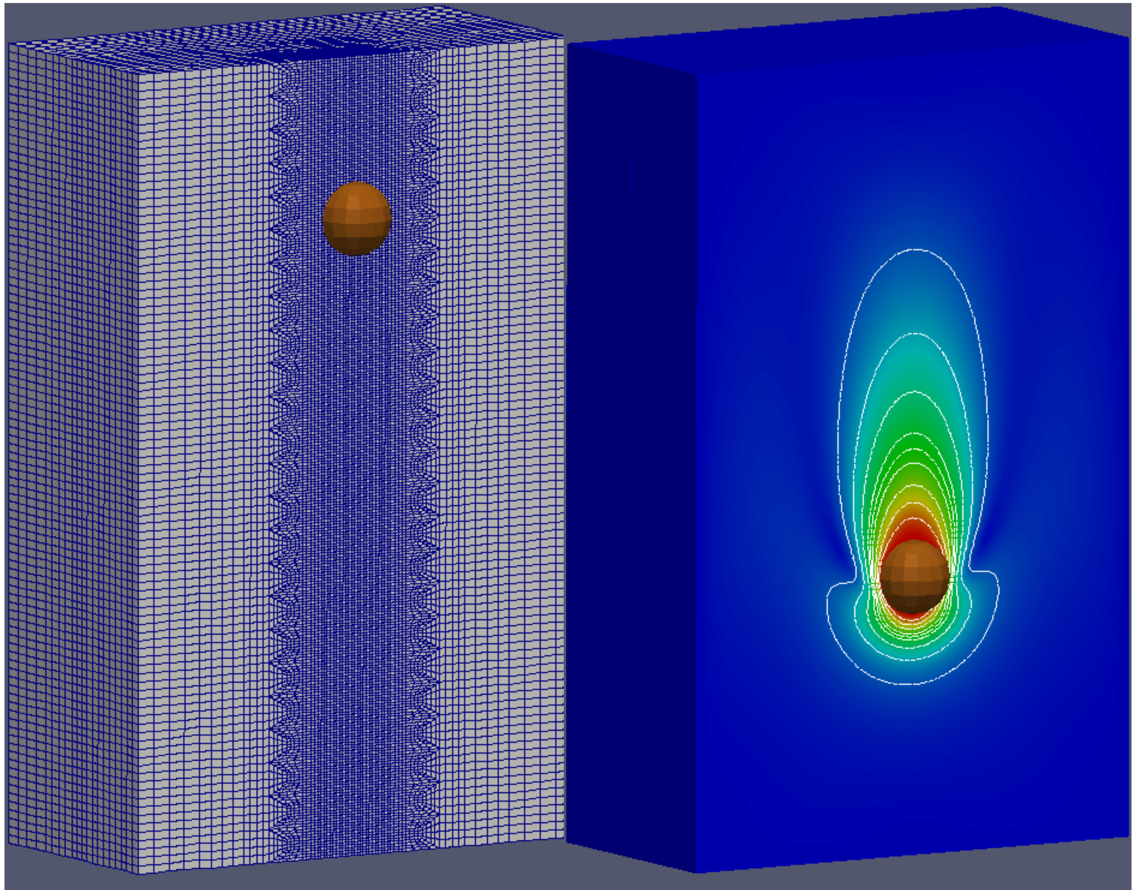
Source: Glowinski et al. 2001



Benchmarking and Validation (II)

<u>Settling of a sphere towards a plane wall:</u>

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



Setup

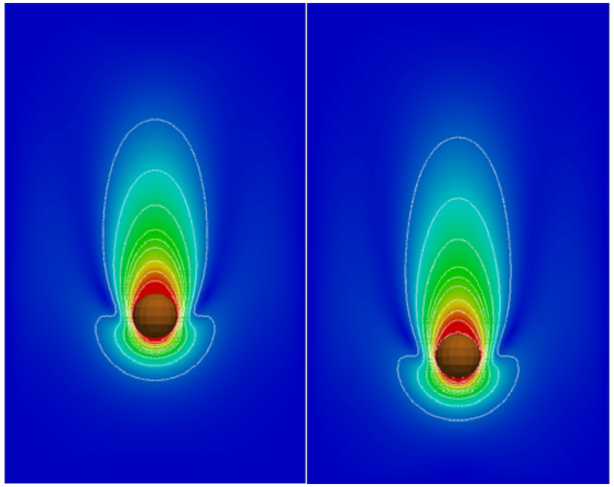
Computational mesh:

- 1.075.200 vertices
- 622.592 hexahedral cells
- Q2/P1:
  → 50.429.952 DoFs

Hardware Resources:

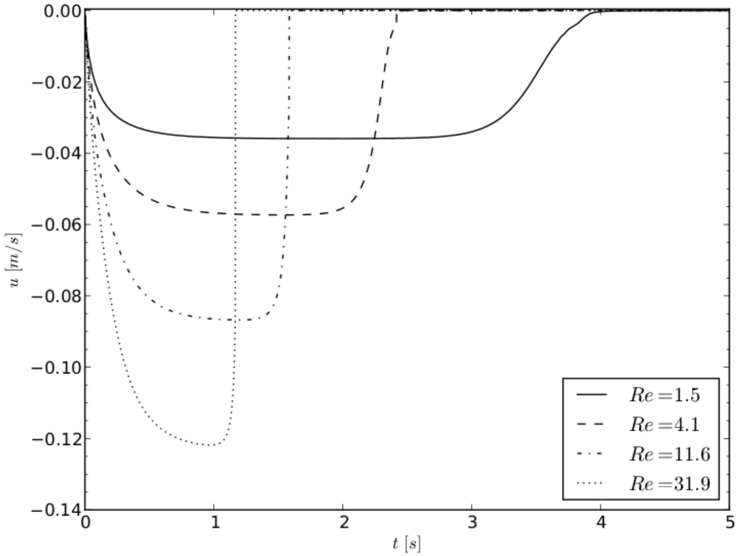
- 32 Processors

Sedimentation Benchmark (I)



Observations

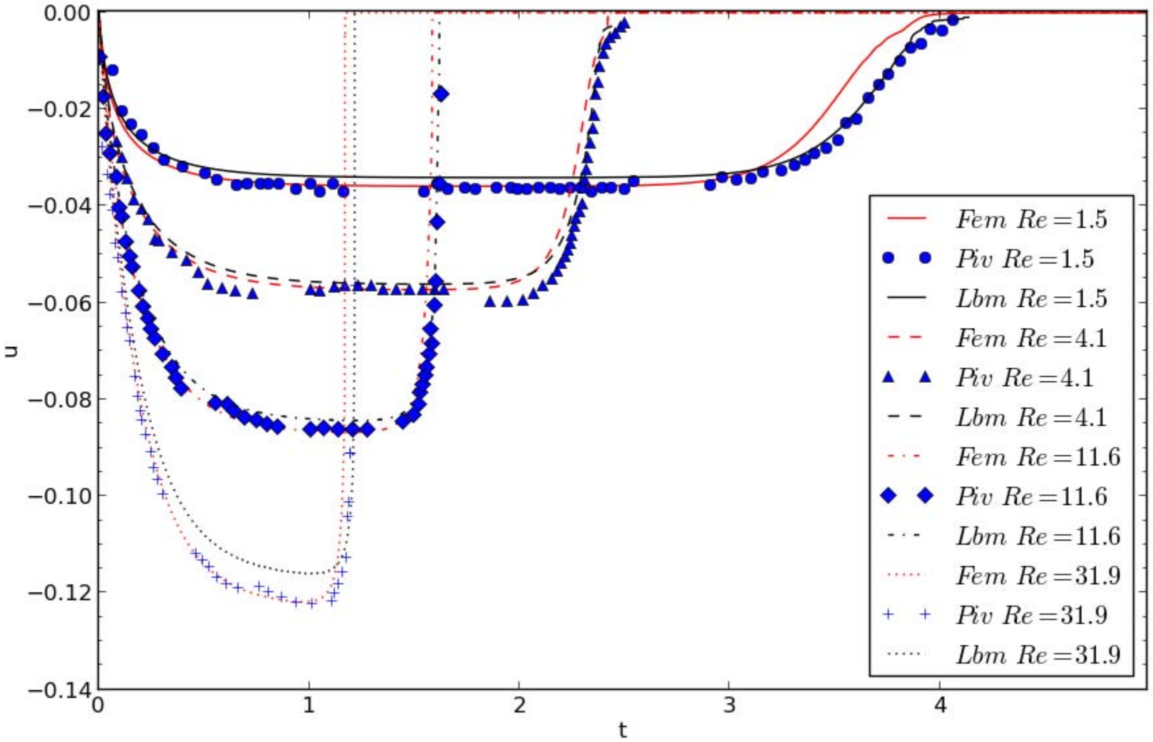
- Velocity profiles compare well to ten Cate's data
- Maximum velocity close to experiment
- Flow features are accurately resolved



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

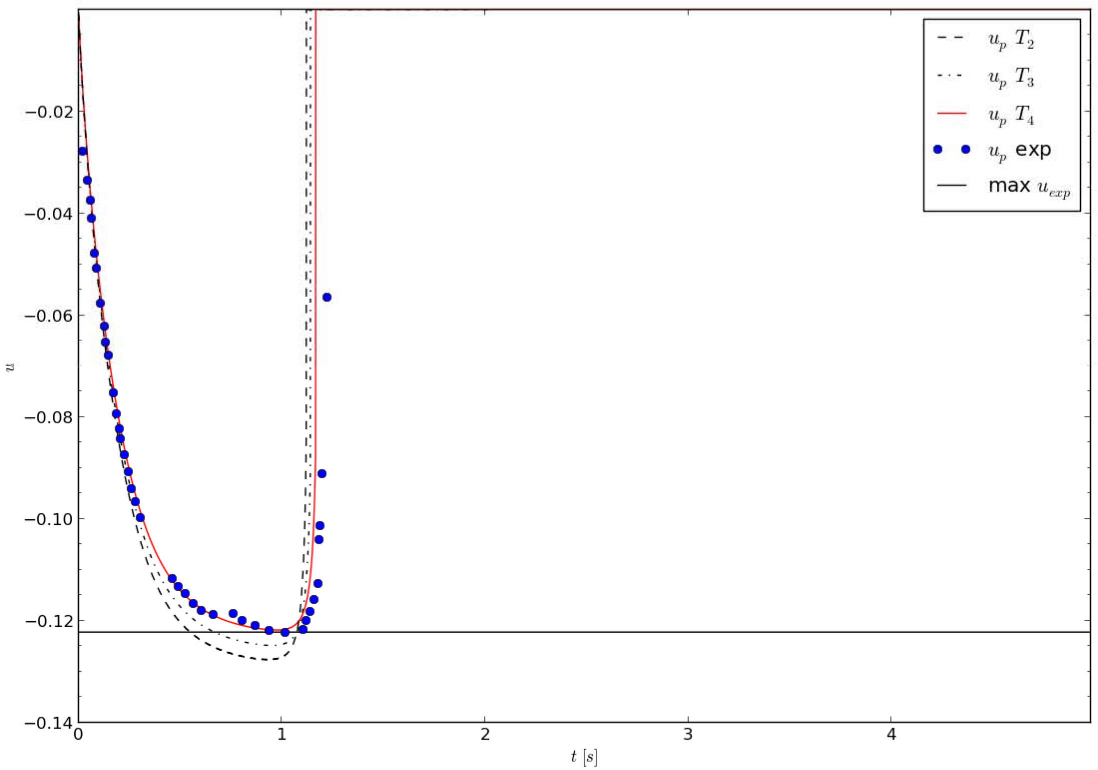
Source: 13th Workshop on Two-Phase Flow Predictions 2012
Acknowledgements: Ernst, M., Dietzel, M., Sommerfeld, M.

Sedimentation Benchmark (III)

FEM-Multigrid Framework

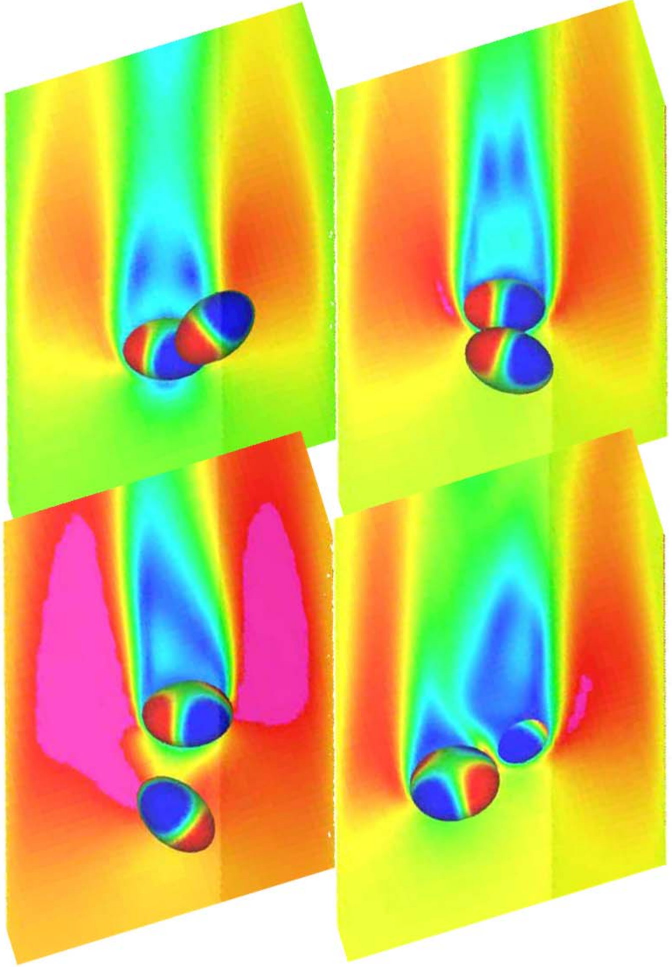
- *Increasing the mesh resolution produces more accurate results*

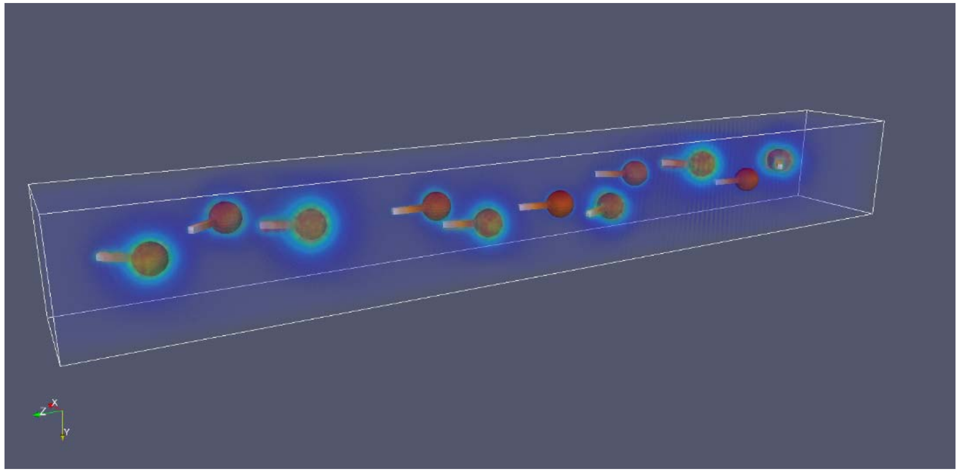
  Test performed at different mesh levels
  - Maximum velocity is approximated better ✓
  - Shape of the velocity curve matches better ✓

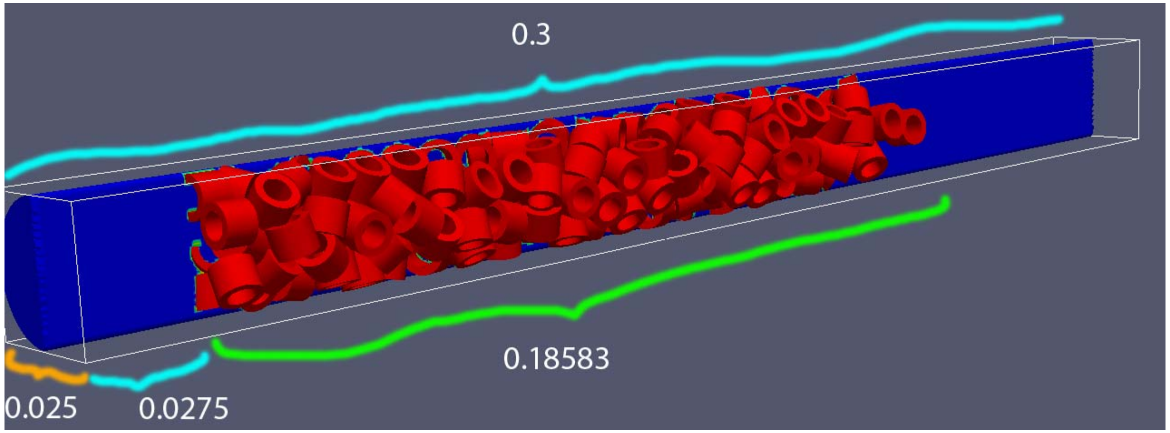


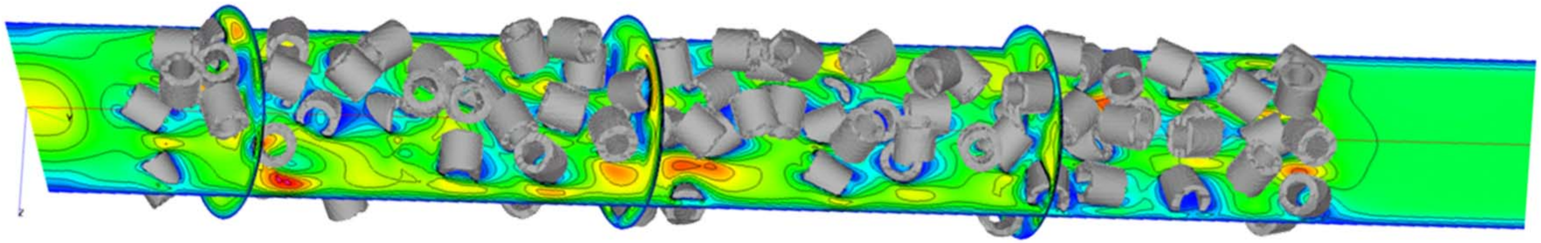
Some More Complex Examples

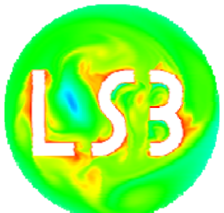
technische universität
dortmund



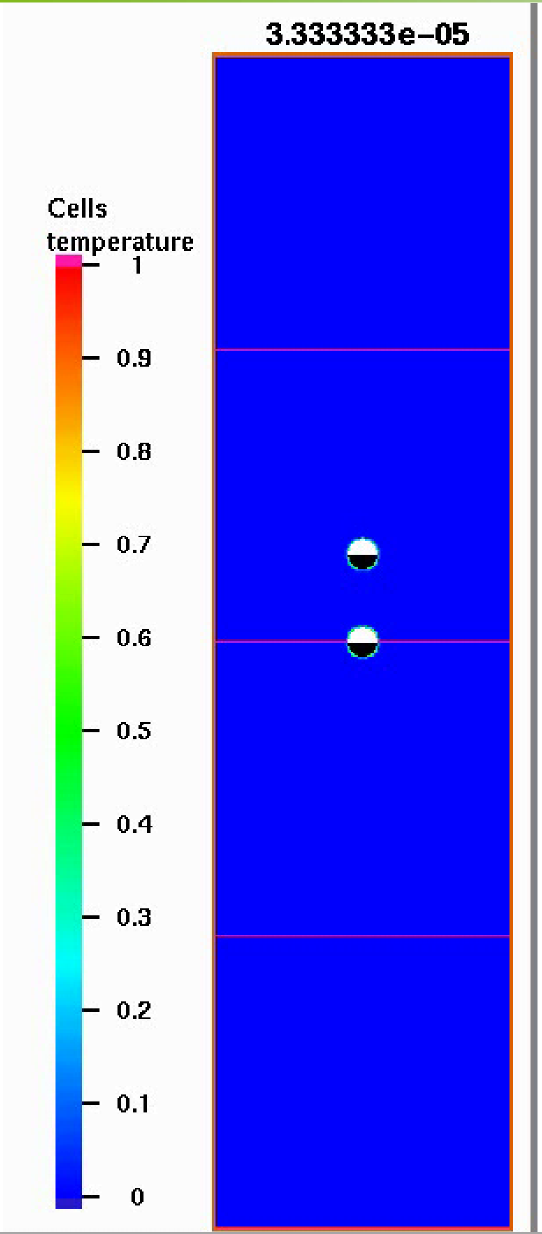


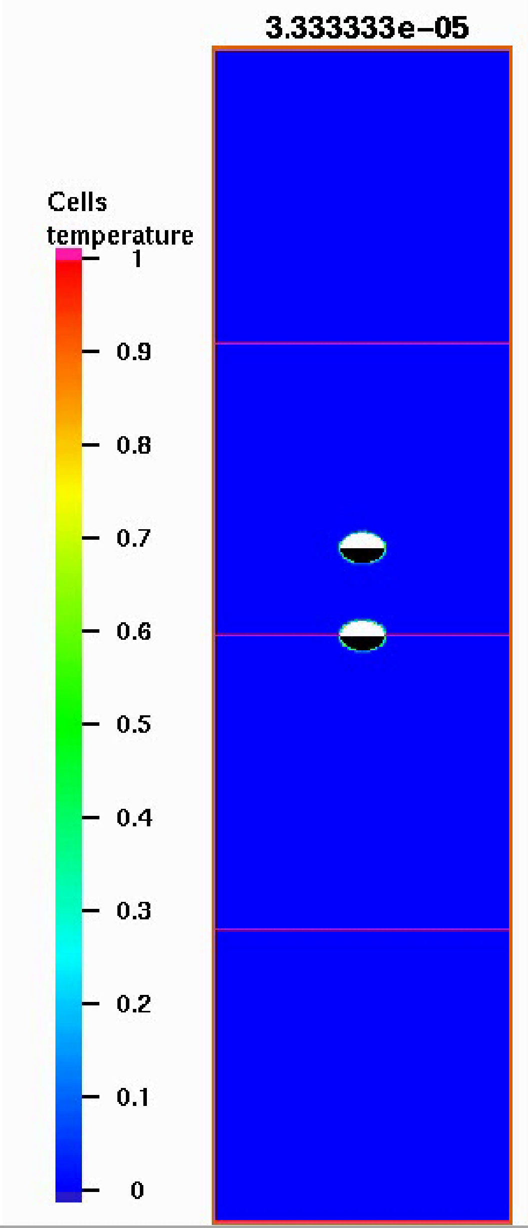


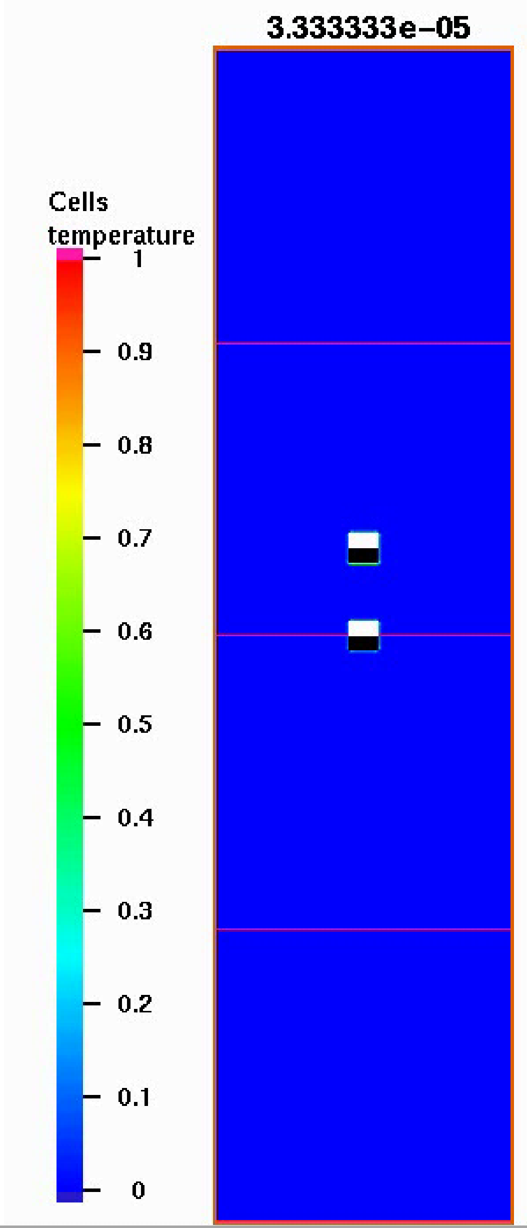




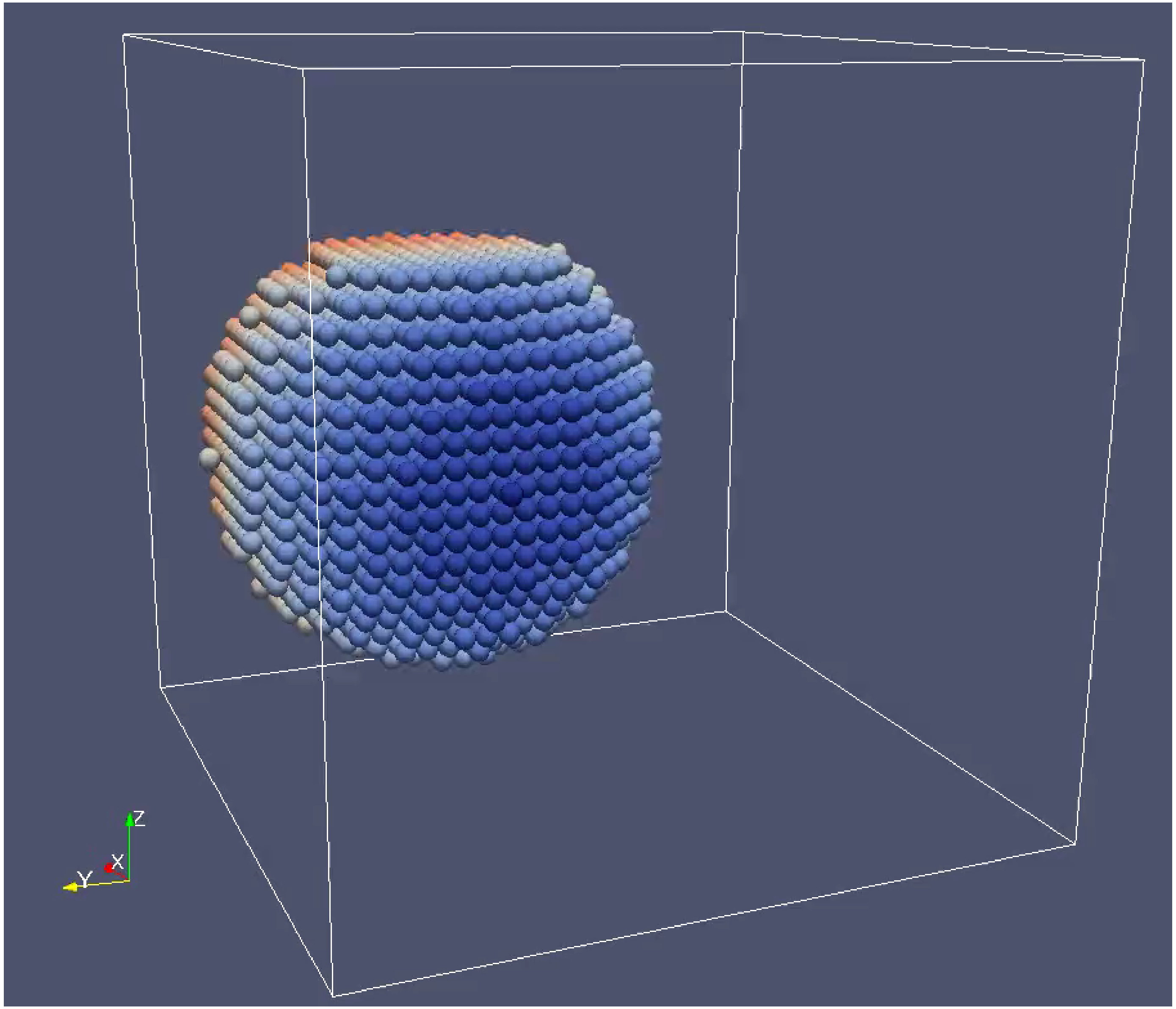
‘Kissing, Drafting, Thumbling’



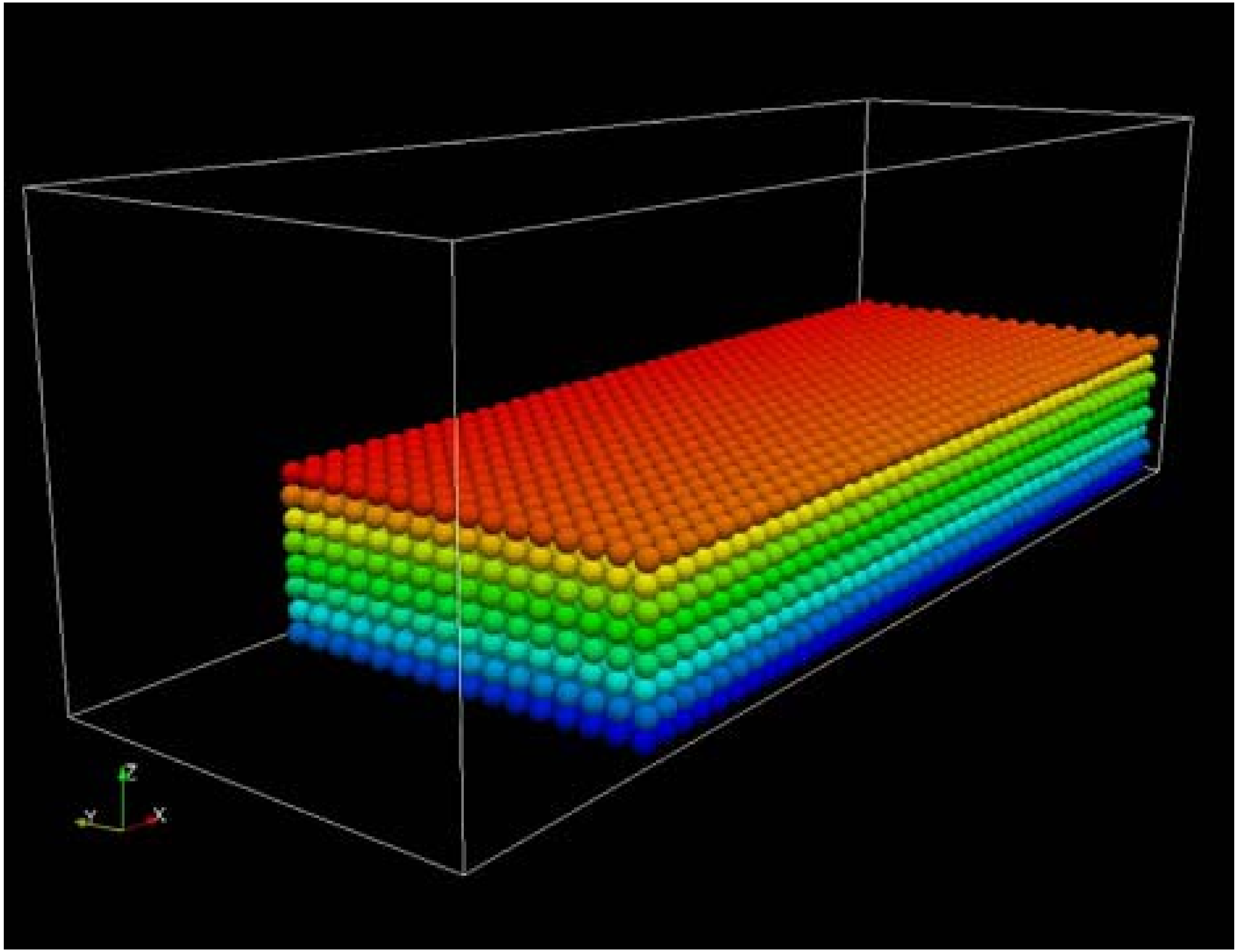




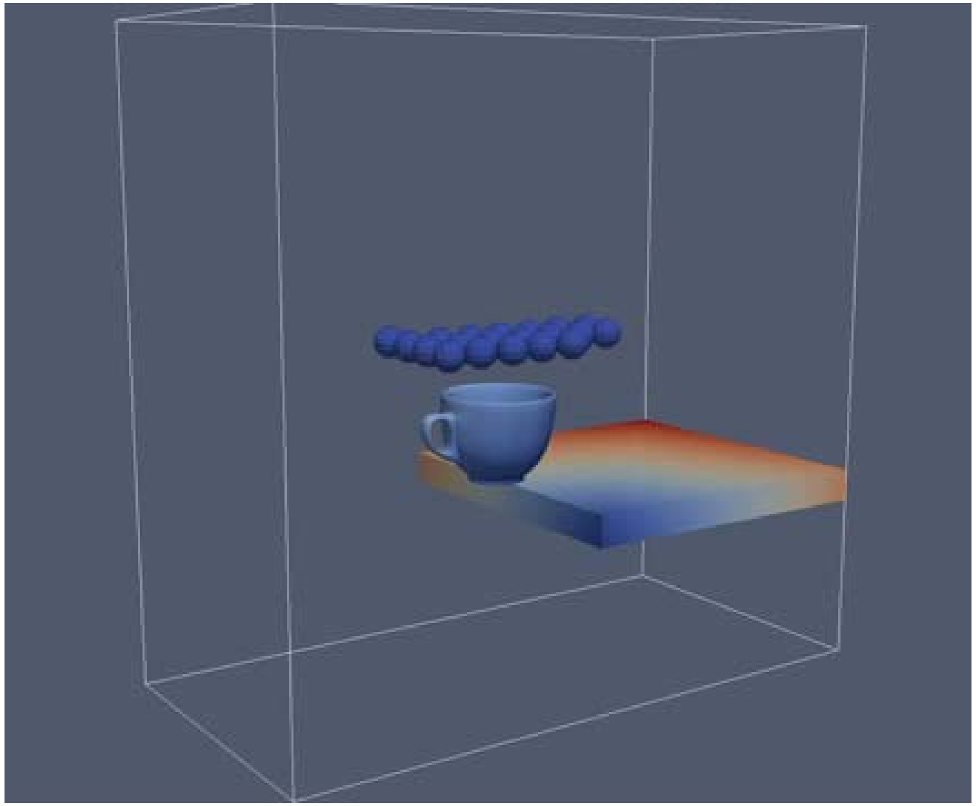
Many Particle Sedimentation

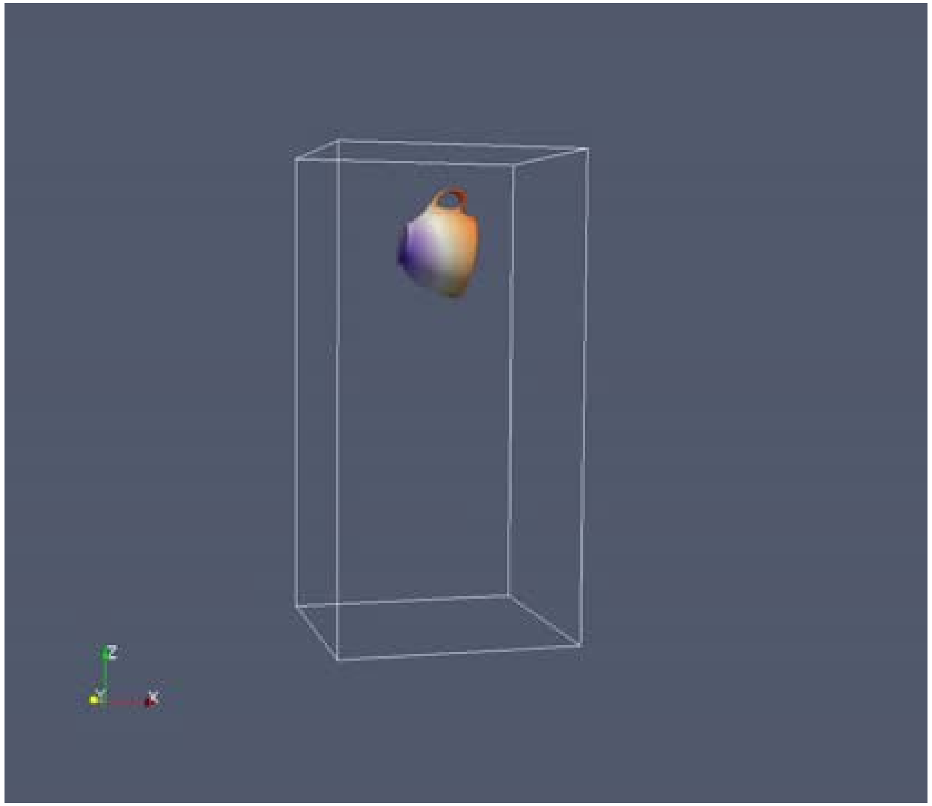


Driven Cavity with Particles

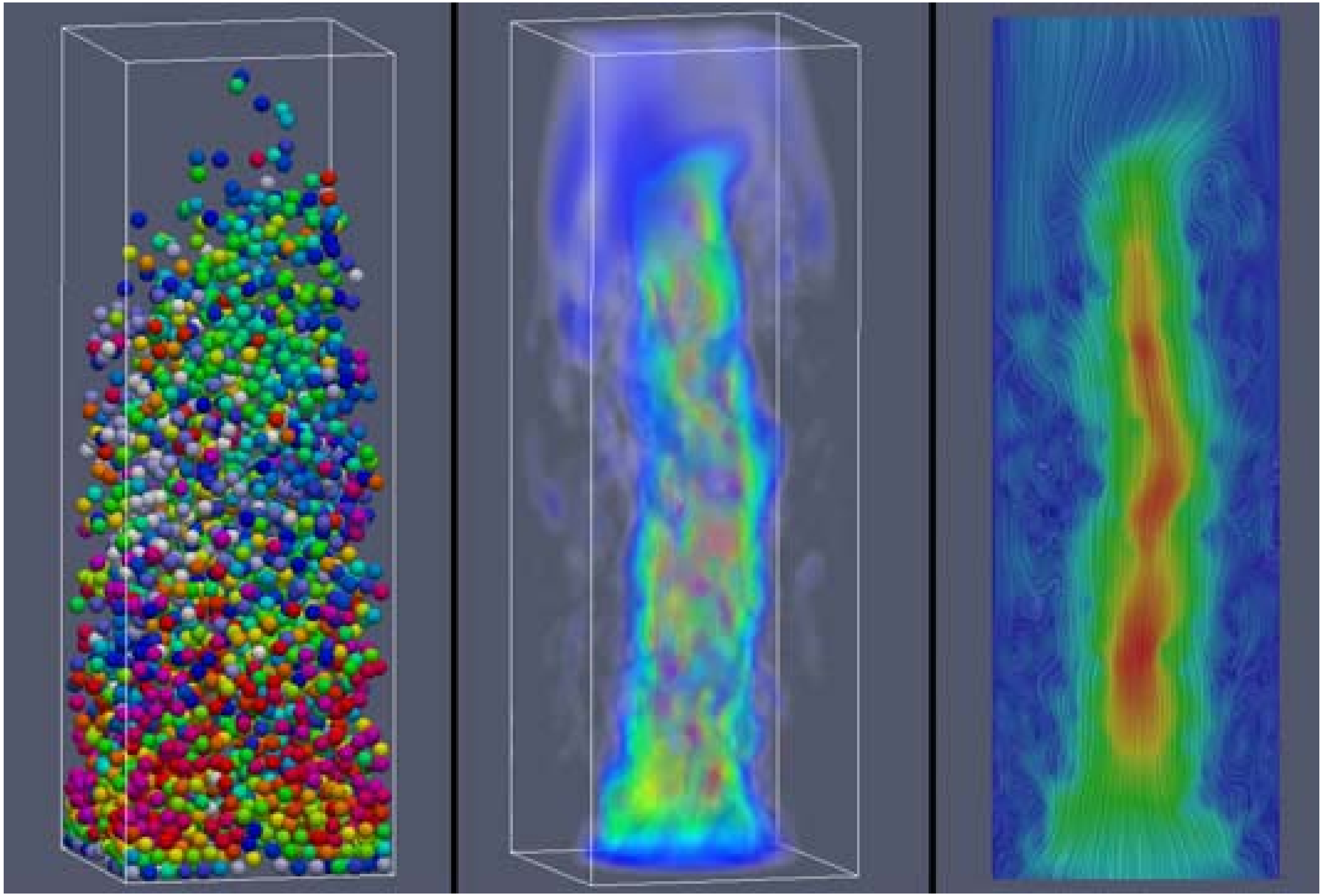


(More) Complex Geometry Examples

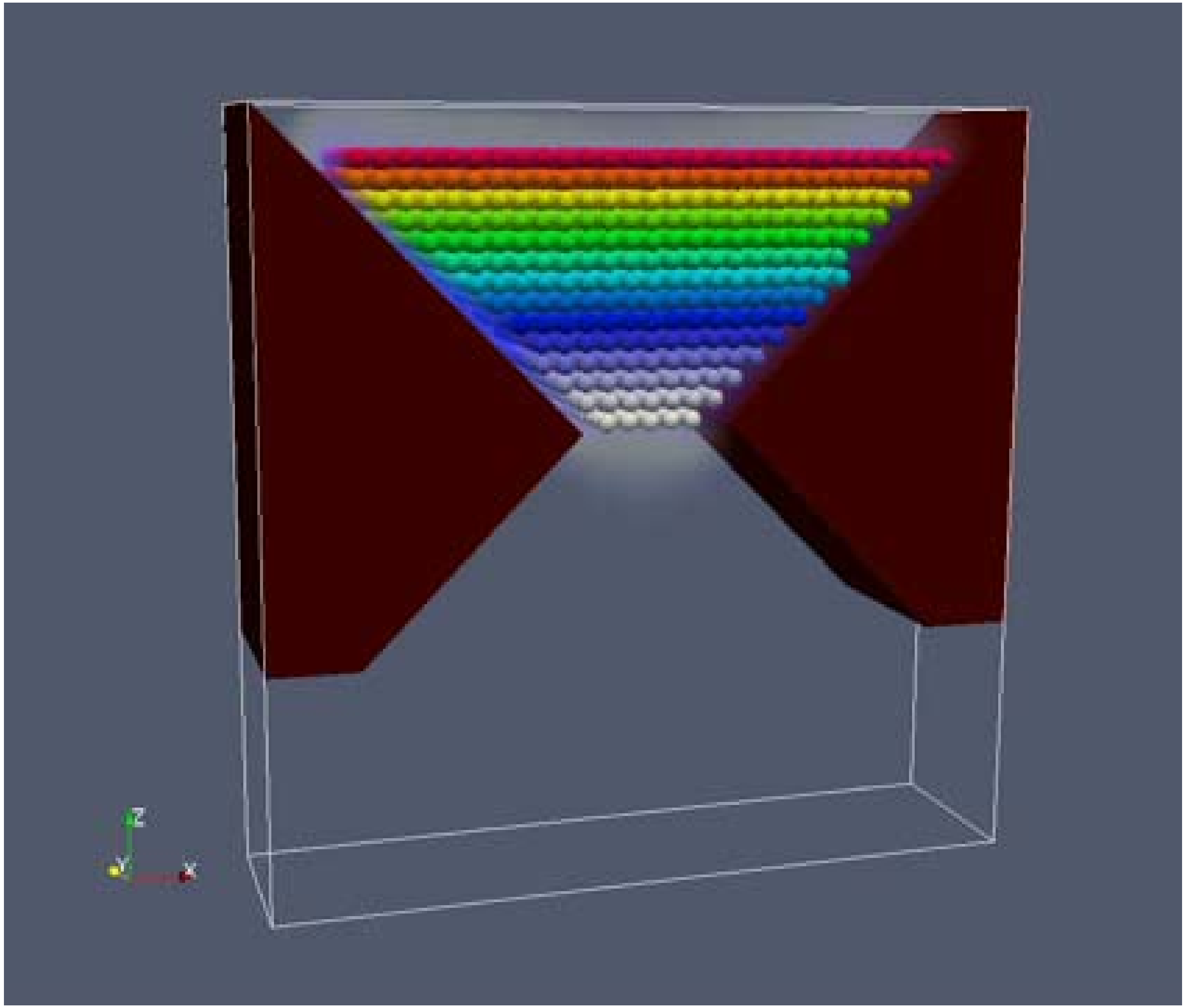




Fluidized Bed Example

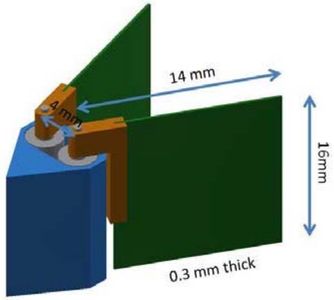


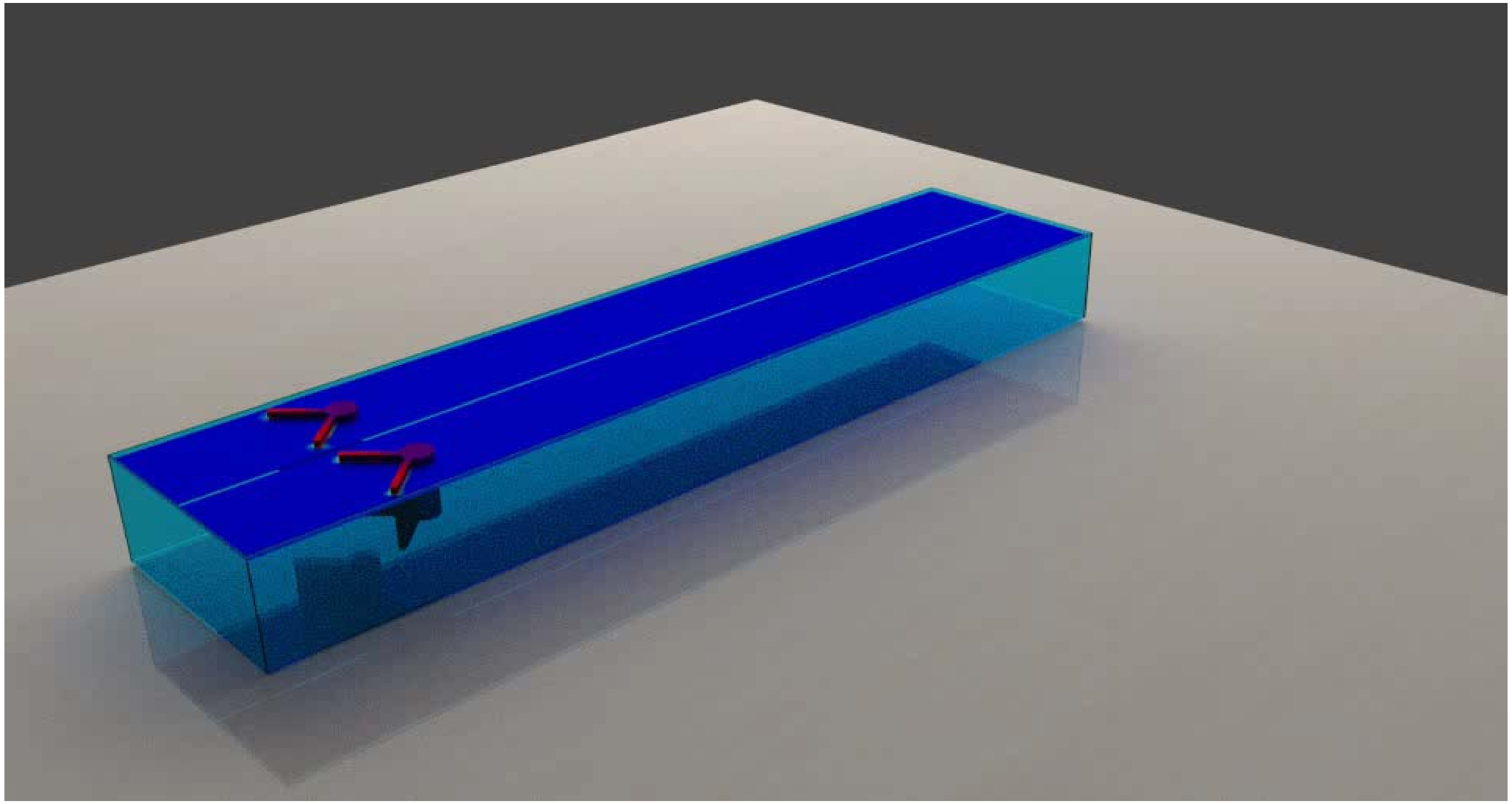
DGS Configuration

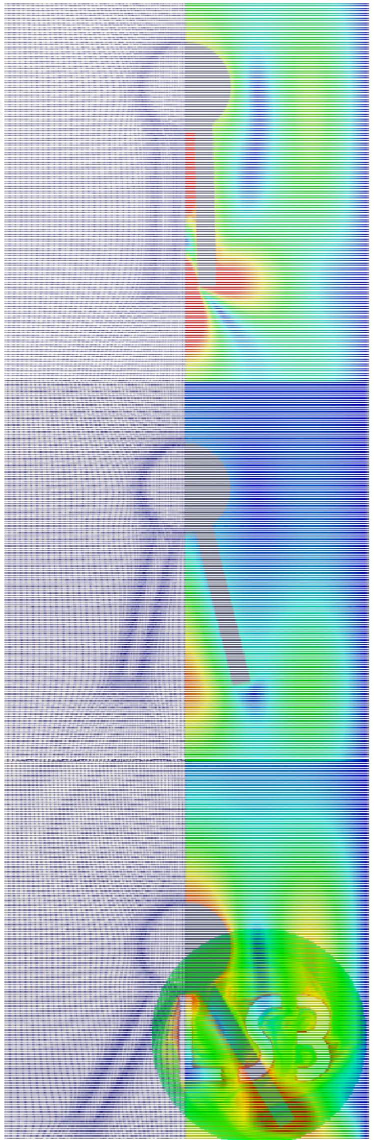


Microswimmer Example

Application to microswimmers (in: 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

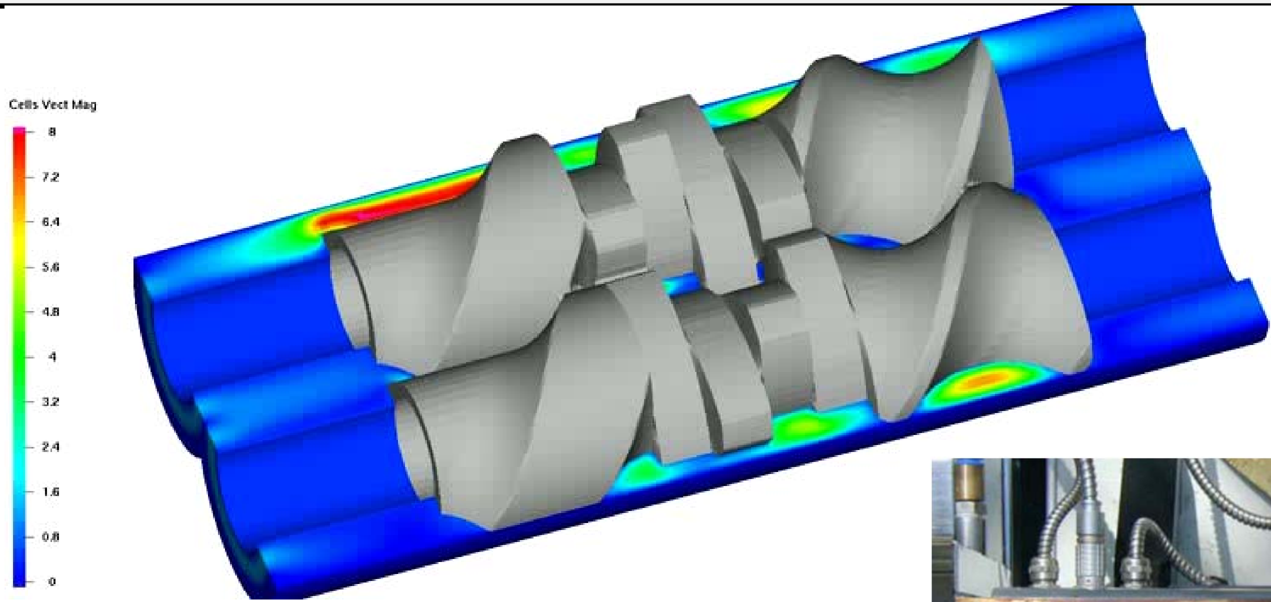


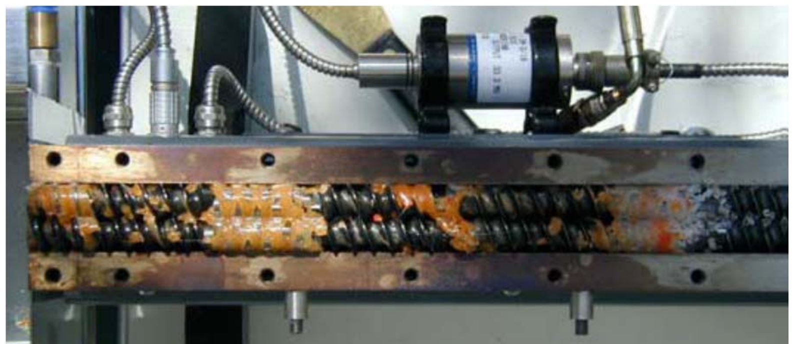




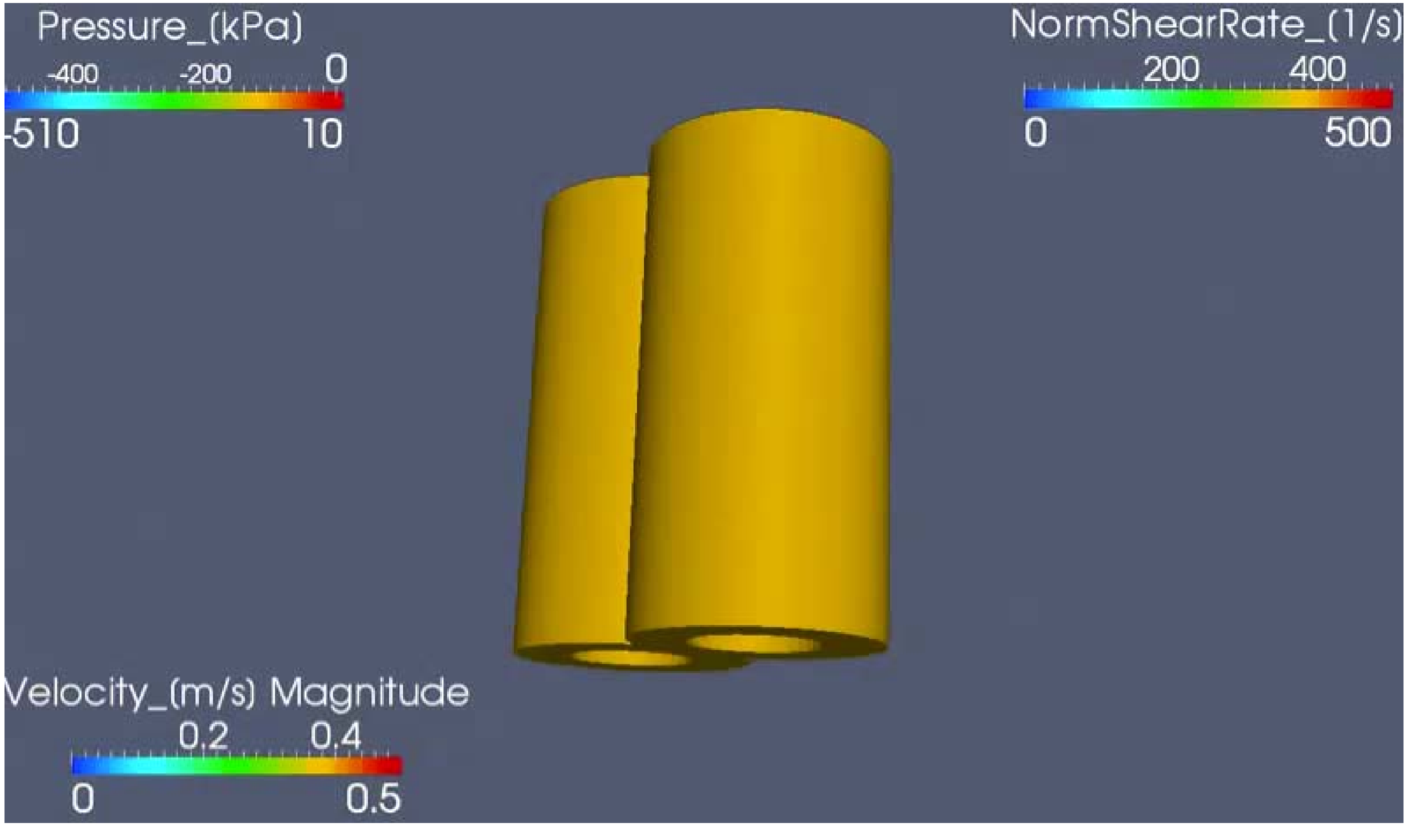
Example: Twinscrew Extruder (I)

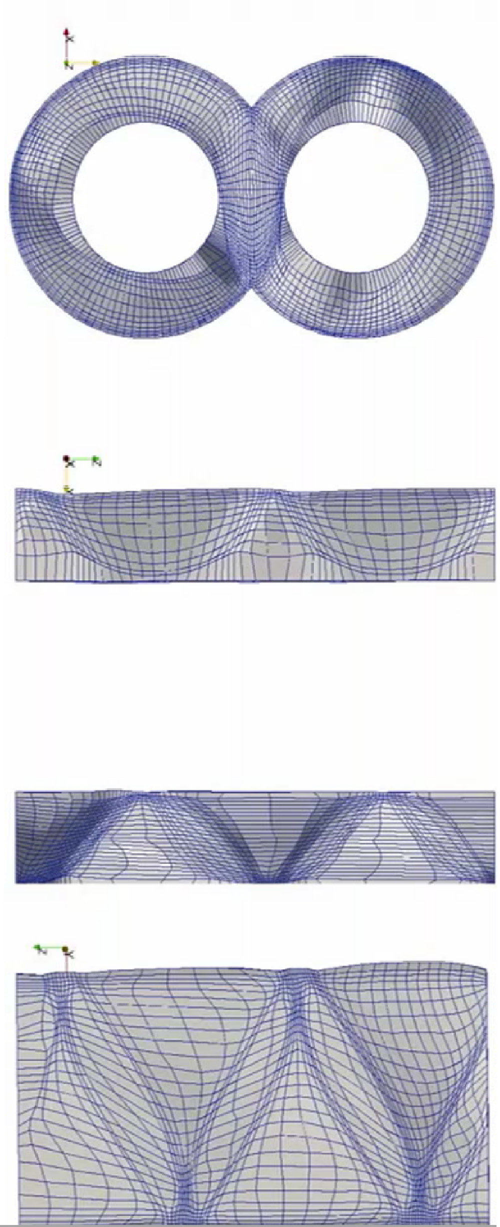
- Numerical simulation of (partially filled) *twinscrew extruders*
- *Non-Newtonian rheological* models (shear & temperature dependent) with *non-isothermal* conditions (cooling from outside, heat production)
- Analysis of the influence of local characteristics on the global product quality, prediction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



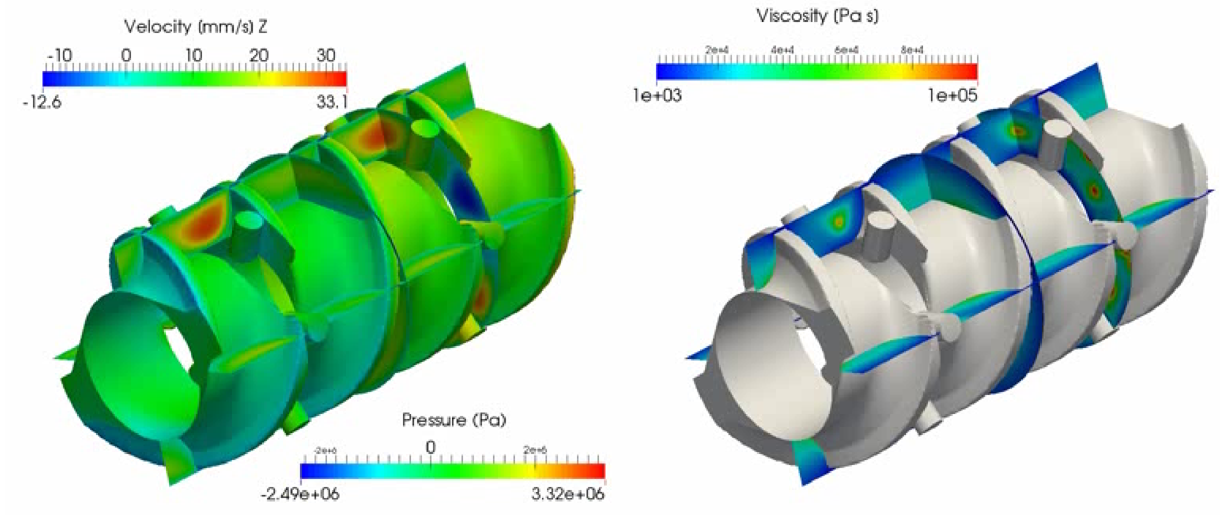


Example: Twinscrew Extruder (II)

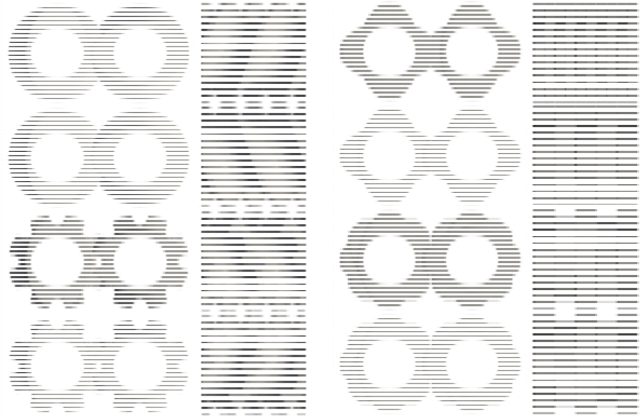




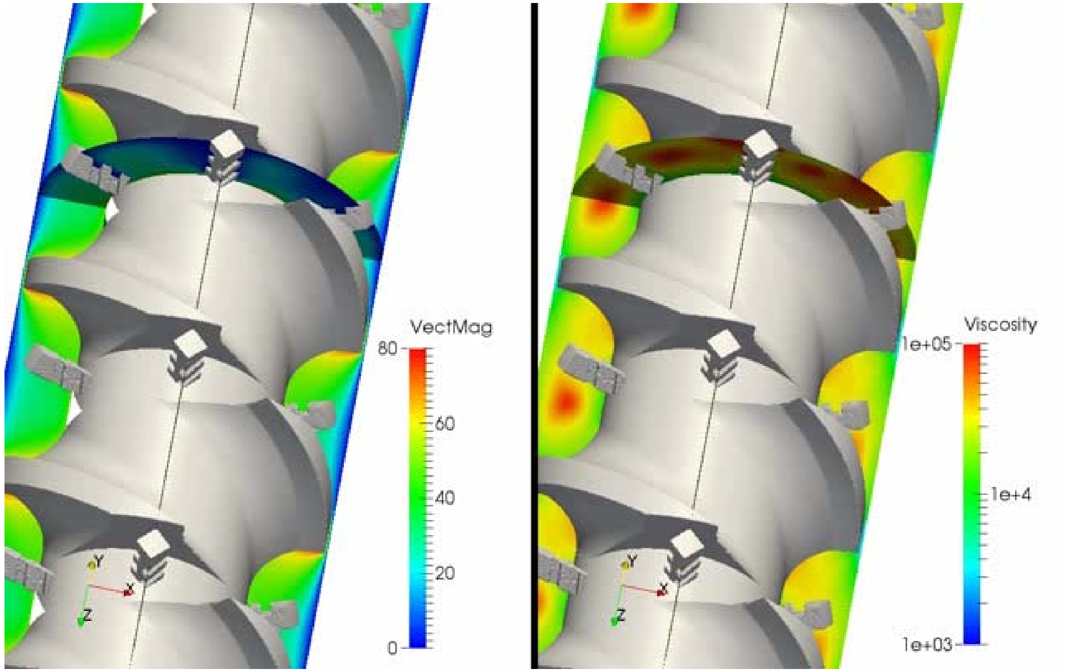
Example: Screw Extruder (III)



Twin-screw-element library



Combination of screw segments

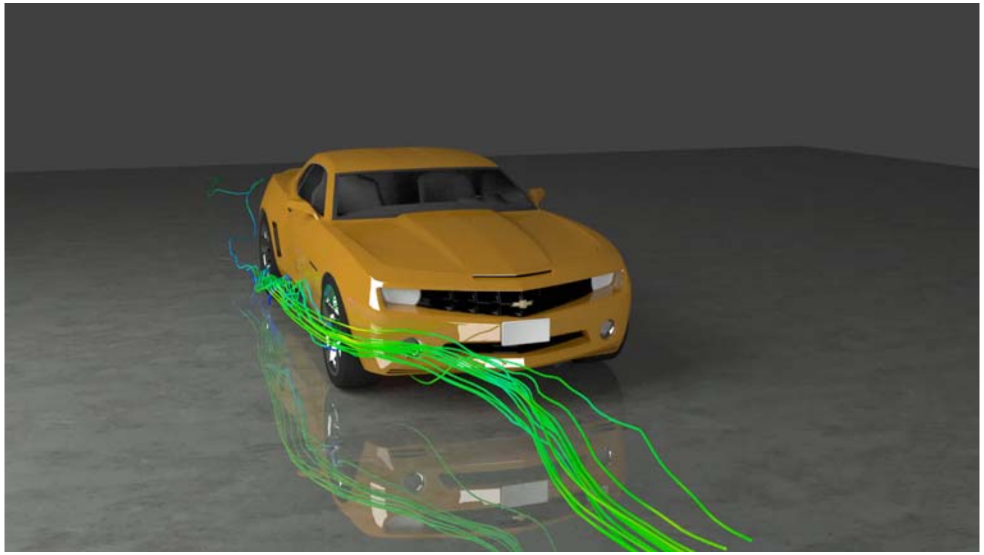


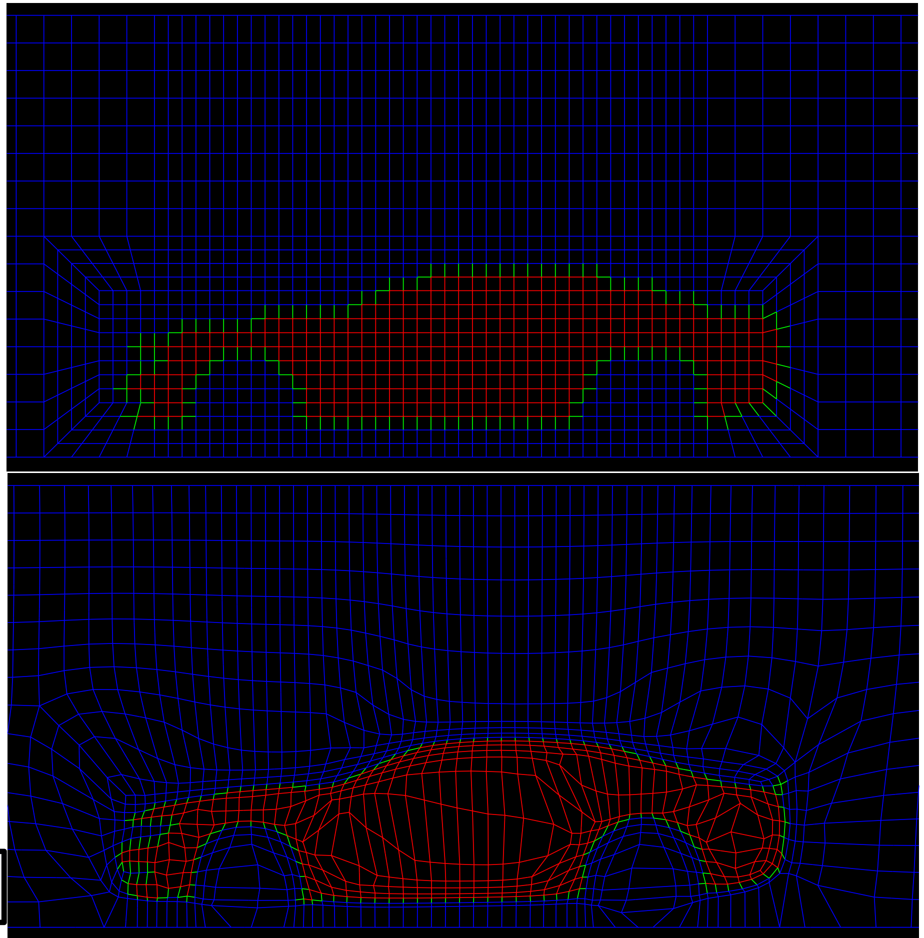
Example: Virtual Wind Tunnel (I)

technische universität dortmund

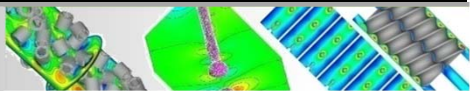
- Numerical simulation of complex geometries
- Use of a regular semi-unstructured preadapted mesh
- Resolution of small scale details by local mesh adaptation

Streamline visualization of the flow field around a car





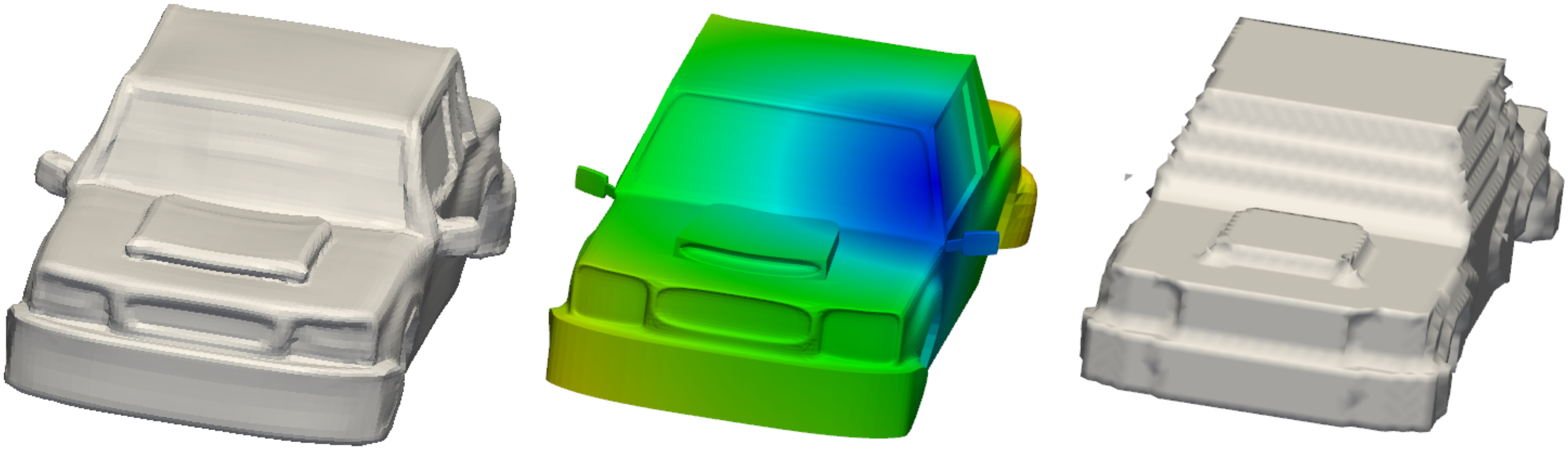
Mesh Slices with and without adaptation



Influence of Mesh Adaptation (II)

Car representation by the computational mesh

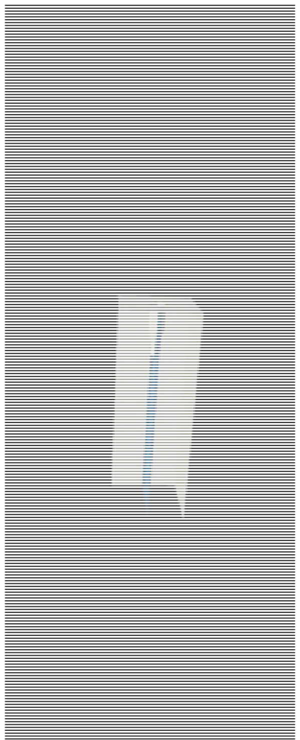
with adaptation	original	no adaptation

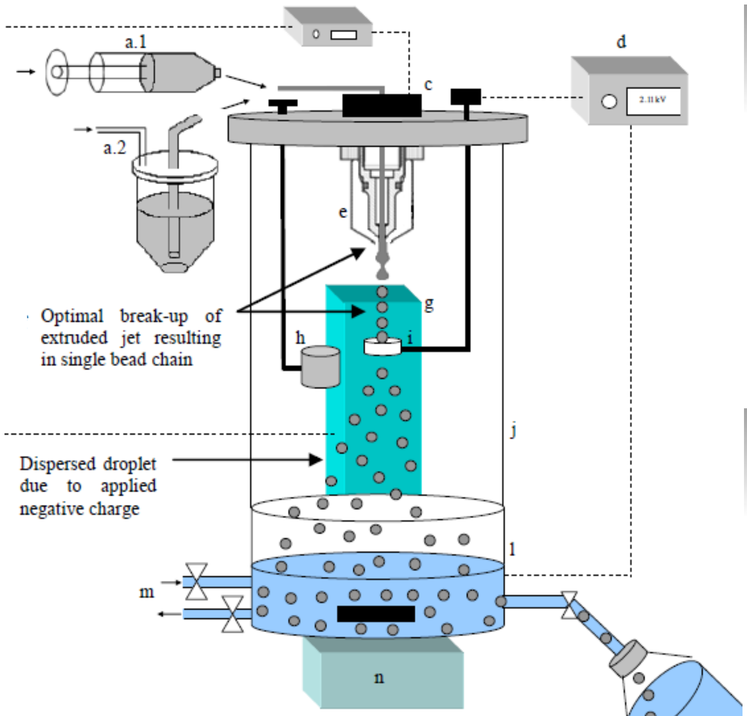


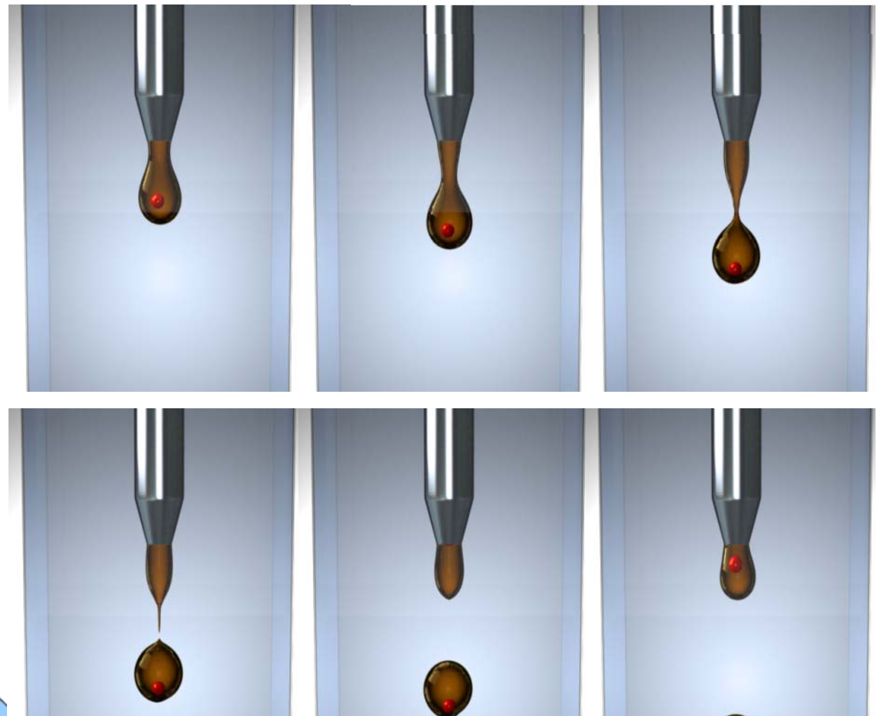
- Details may be lost without adaptation
- Better resolution with the same number of DOFs
- Mesh adaptation saves at least one refinement level

Example: Fluid Prilling&Encapsulation

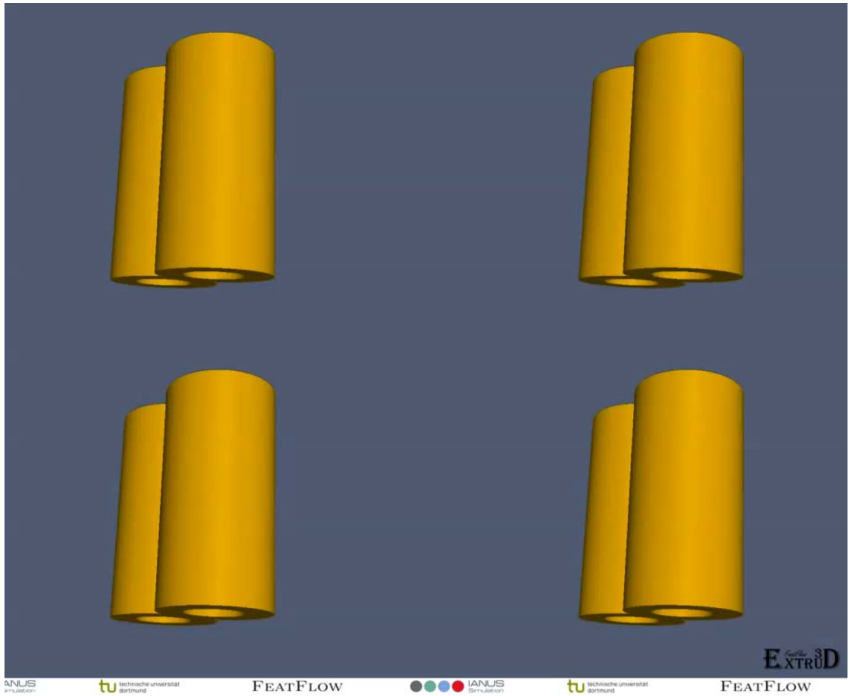
- Numerical simulation of *micro-fluidic drug encapsulation ("monodisperse compound droplets")*
- Polymeric "bio-degradable" outer fluid with generalized Newtonian behaviour
- Optimization w.r.t. boundary conditions, flow rates, droplet size, geometry







Extensions & Future Activities



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

Benchmarking

- More complex object(s)
- „Many“ objects!

Some HWON Rules of Thumb

- Realize all MG components via sparse MV (preconditioners, grid transfer) & Optimize sparse MV w.r.t. FEM space, numbering and hardware

→ **Generic and hardware-optimized `gMG-FEM-BLAS' Toolbox**

- Use higher order in time (large time steps) + space (large FEM stencils)

→ **High arithmetic intensity via dominant `solution part' (→ gMG)**

- Design strongly coupled schemes (globally) with Operator-Splitting components (locally)

→ **Combine (outer) high robustness & (inner) high efficiency**

- Exploit locally regular structures to improve global convergence

→ **Strong local solvers cost nothing & Hide irregularities locally**

→ **Patchwise adaptivity, generalized TP meshes, Grid Deformation, FBM,…**

Huge Potential for Future …

However:

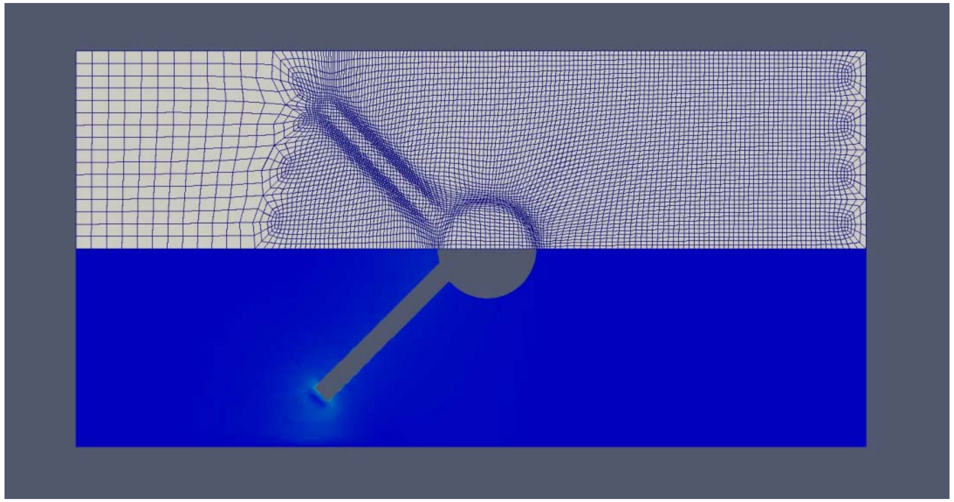
- **Numerical Simulation & High Performance Computing** have to consider recent and future hardware trends, particularly for heterogeneous multicore architectures and massively parallel systems!

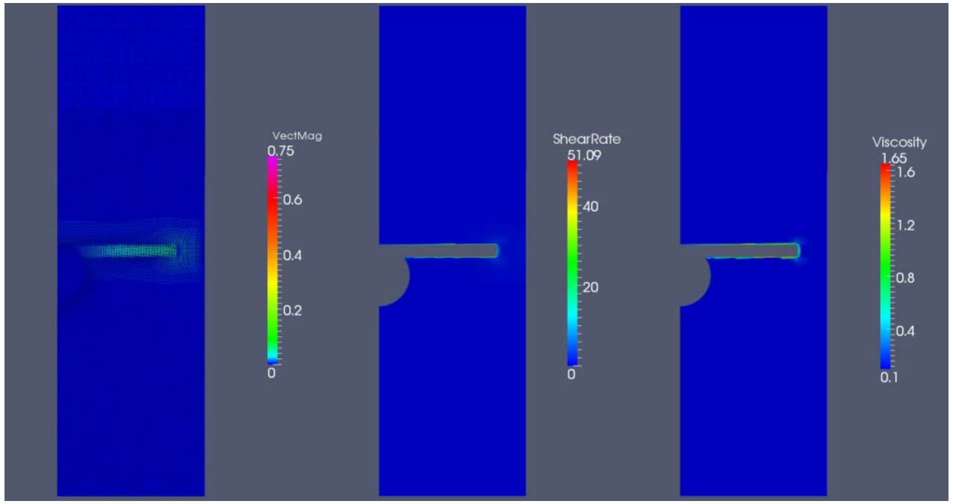
- The combination of **'Hardware-oriented Numerics'** and special **'Data Structures/Algorithms'** and **'Unconventional Hardware'** has to be used!

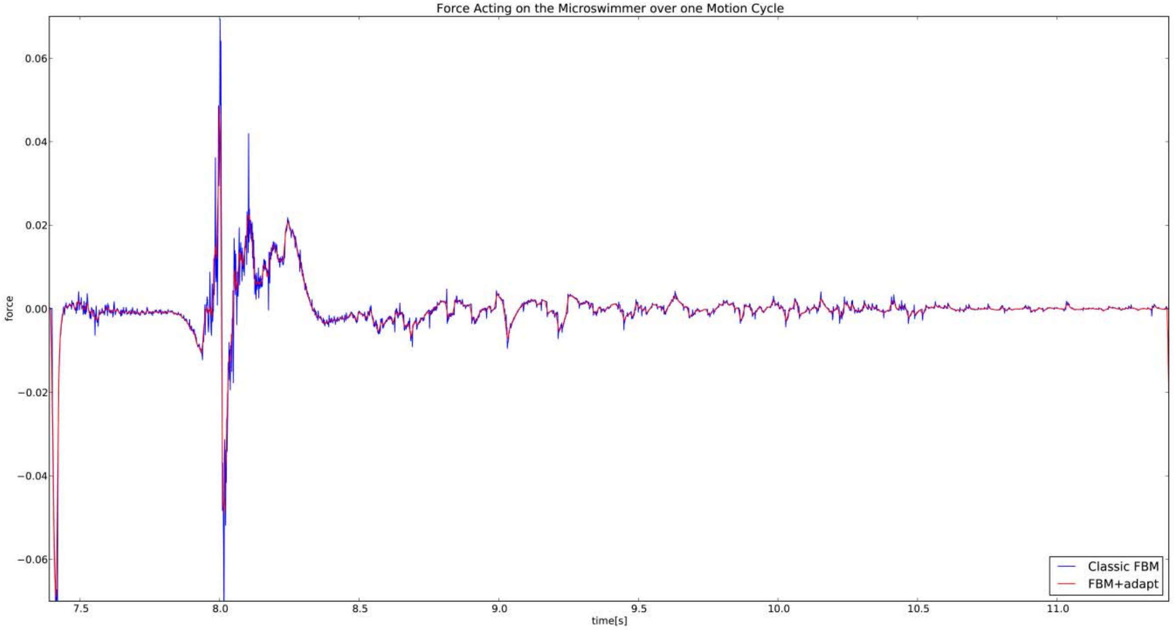
…or many of the existing (academic/commercial) PDE software packages will be 'worthless' in a few years!

Backup slides …

Microswimmer Example

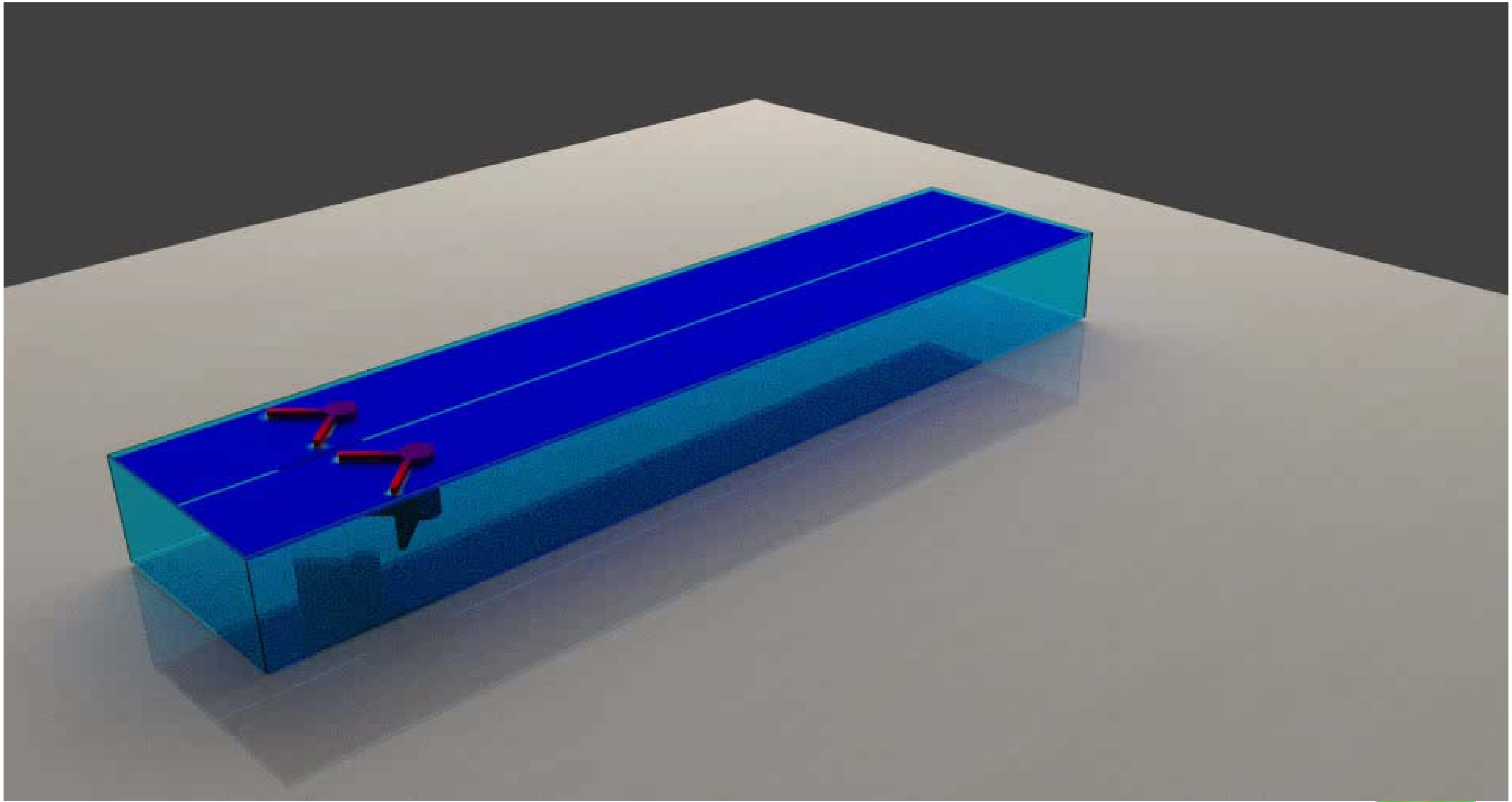




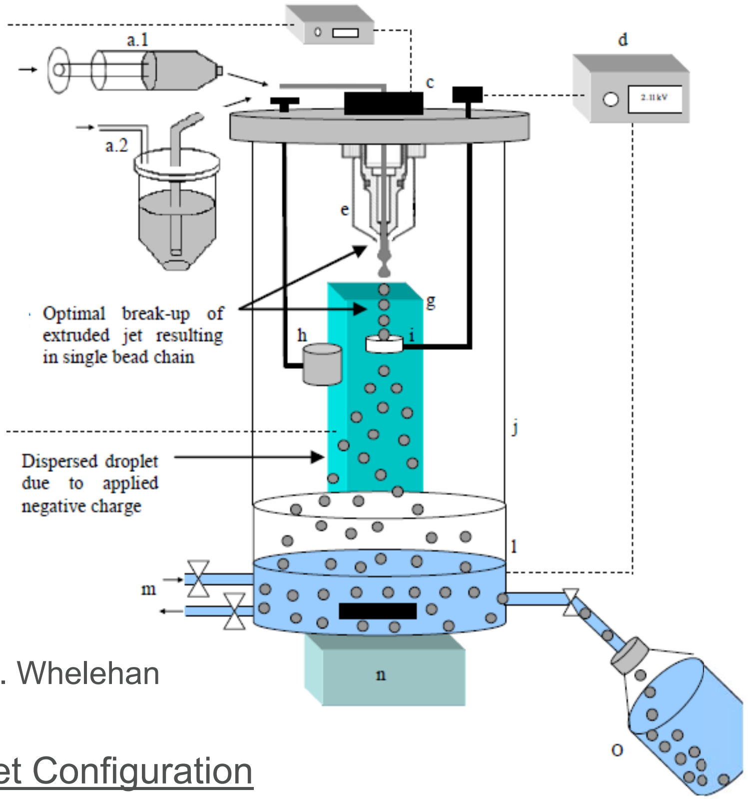


Compared to the classic FBM approach the force curve is much smoother with grid adaptation

Microswimmer Example



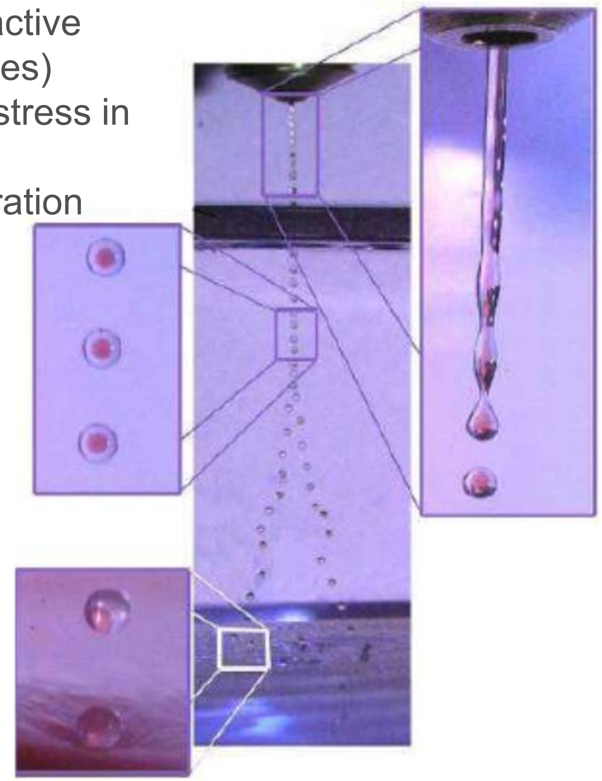
- Numerical simulation of *micro-fluidic drug encapsulation ("monodisperse compound droplets")*
- Polymeric "bio-degradable" outer fluid with generalized Newtonian behaviour
- Optimization w.r.t. boundary conditions, flow rates, droplet size, geometry



M. Whelehan

<u>In Pharmaceutics</u>

- Controlled drug release
- Protection of chemically active ingredients (from both sides)
- Protection against shear stress in stirred reactors
- Protection against evaporation
- Taste or odor masking



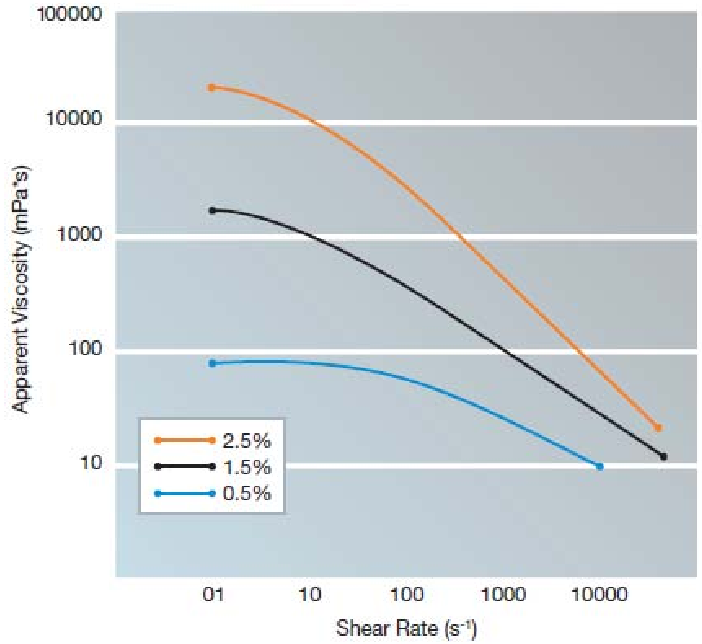
M. Whelehan

<u>Jet Configuration</u>

- Core material is defined as the specific material that requires to be coated (liquid, emulsion, colloid or solid)
- Shell material is present to protect and stabilize the core (Alginate, Chitosan, Gelatin, Pectin, Waxes, Starch)

Fluid Prilling and Encapsulation (II)

technische universität dortmund



Aqueous solutions of
alginates have shear-thinning characteristics