Simulation of Turbulent Particulate Flow on HPC Systems

Konstantin Fröhlich, Lennart Schneiders, Matthias Meinke, Wolfgang Schröder

k.froehlich@aia.rwth-aachen.de

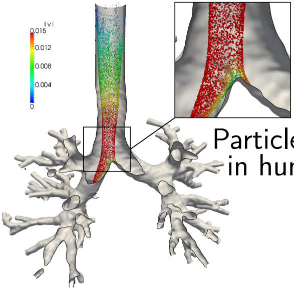
Institute of Aerodynamics
RWTH Aachen University
Germany

24th Workshop on Sustained Simulation
Stuttgart, Germany, December 5, 2016

Motivation

Research projects involing turbulent particle-laden flow

- Models verified and improved via particle-resolved simulations

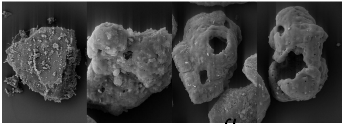


Particle deposition in human airways

extremeinstability.com

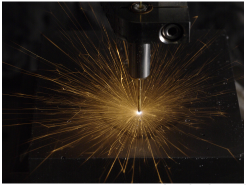
Precipitation modeling

Coal/biomass combust.



LEAT, RU Bochum, *oxyflame.com*

Electrical discharge machining

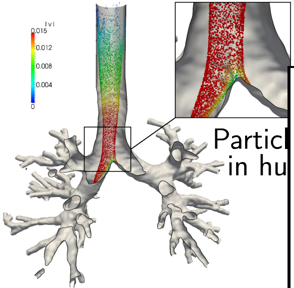


edm-huber.de

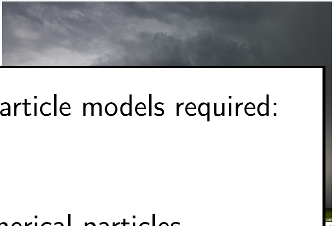
Motivation

Research projects involing turbulent particle-laden flow

- Models verified and improved via particle-resolved simulations



Partic
in hu



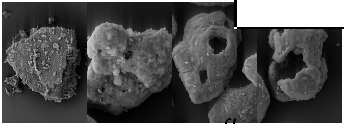
Precipitation modeling

m

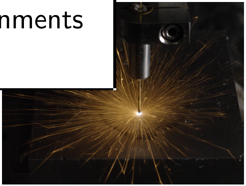
Improved particle models required:

- $d_p \sim \eta$
- non-spherical particles
- high-temperature environments

Coal/biomass combu



LEAT, RU Bochum, *oxyflame.com*

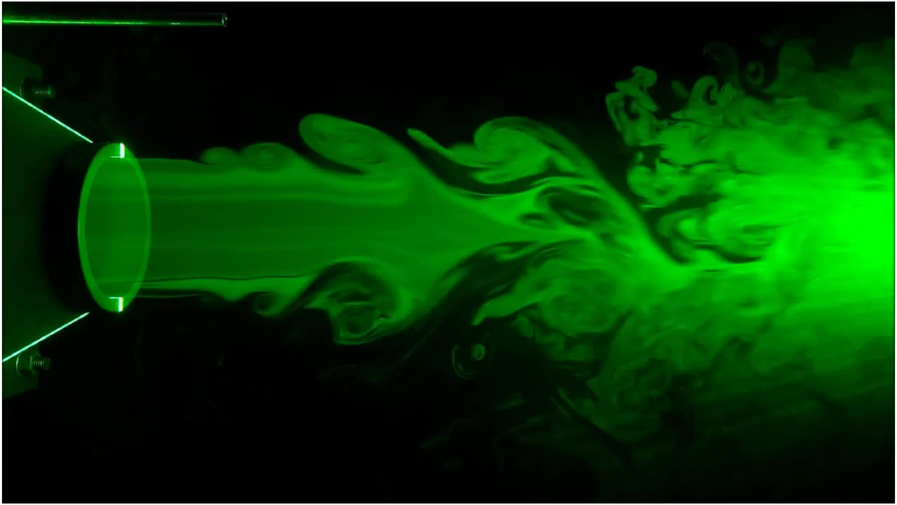


edm-huber.de

Outline

1. Motivation
2. Sharp-interface Cartesian method
   - Multiple level-set/cut-cell boundary representation
   - Dynamic mesh refinement
   - Dynamic load balancing
3. Application to particulate turbulent flow
   - Modulation of isotropic turbulence by spherical particles
   - Quantification of particle-induced dissipation
4. Summary

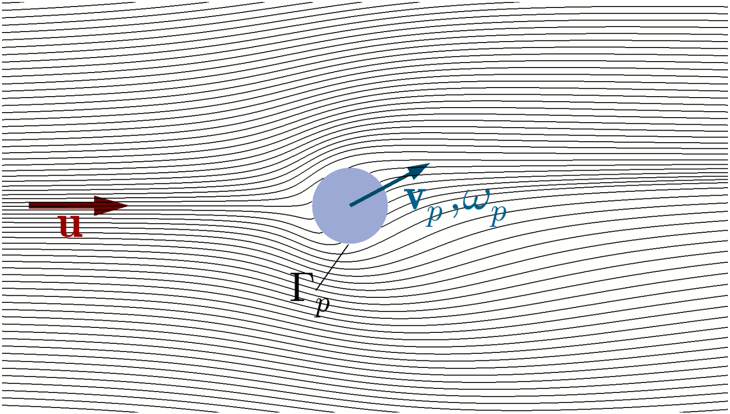
Turbulent flow



Laminar-turbulent transition in a round jet

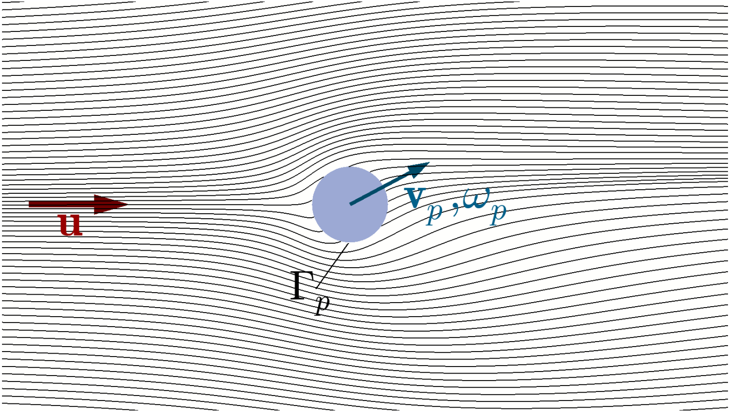
B. O. Andersen, TU Denmark

Governing equations



Fluid motion: $$\frac{d}{dt}\int_{V(t)} \boldsymbol{Q}\, dV + \oint_{\partial V(t)} \underline{\boldsymbol{H}} \cdot \boldsymbol{n}\, dA = \mathbf{0}, \quad \boldsymbol{Q} = [\rho, \rho\boldsymbol{u}, \rho E]$$

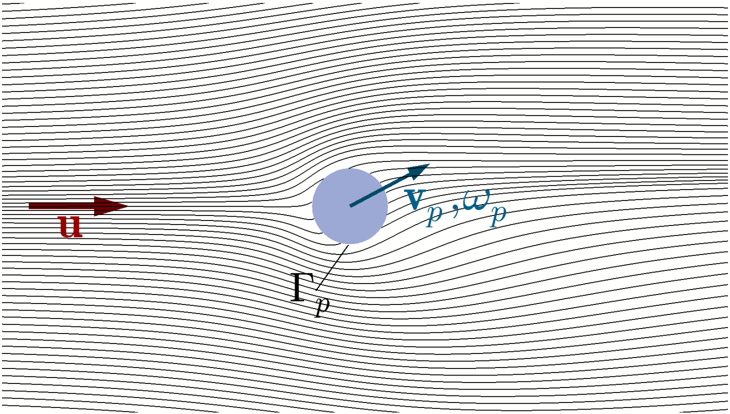
Governing equations



Rigid body acceleration: $$m\frac{d\boldsymbol{v}}{dt} = \boldsymbol{F}$$

Angular acceleration: $$\underline{\widetilde{\mathcal{I}}}\frac{d\widetilde{\omega}}{dt} + \widetilde{\omega} \times (\underline{\widetilde{\mathcal{I}}}\widetilde{\omega}) = \widetilde{\mathcal{T}}$$

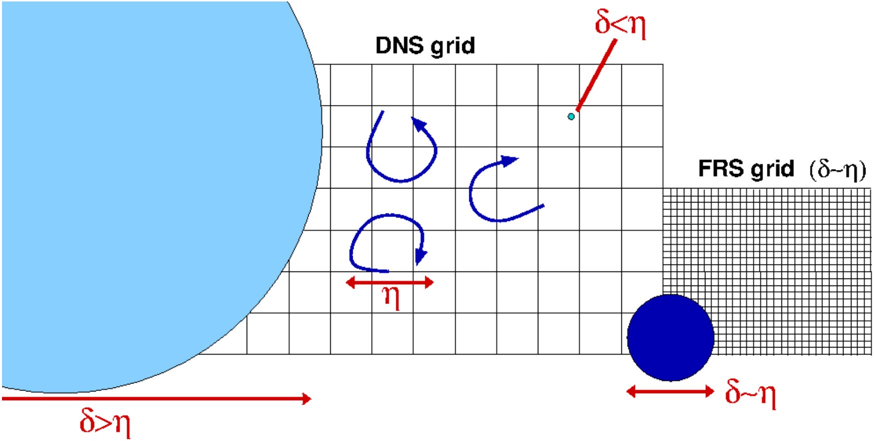
Governing equations



Surface force: $$\boldsymbol{F}_p = \oint_{\Gamma_p} (-p\boldsymbol{n} + \underline{\boldsymbol{\tau}} \cdot \boldsymbol{n}) dA,$$

Surface torque: $$\boldsymbol{\mathcal{T}}_p = \oint_{\Gamma_p} (\boldsymbol{x} - \boldsymbol{r}_p) \times (-p\boldsymbol{n} + \underline{\boldsymbol{\tau}} \cdot \boldsymbol{n}) dA$$

Particle phase representation

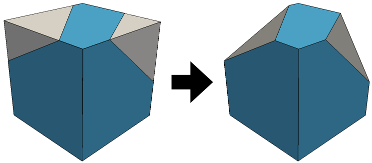


- $d < \eta$: Lagrangian "point-mass" approach justified
- $d \sim \eta$: need extra resolution for particles boundary layers
- $d > \eta$: DNS grid is sufficient to resolve particles

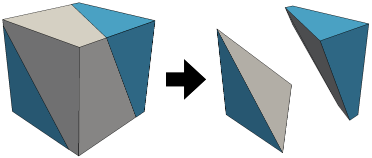
Cut-cell discretization

Sharp resolution of complex particles shapes – cut-cell method

- Intersection of Cartesian mesh with zero level set gives discrete boundary
- Enables sharp and conservative resolution of immersed boundaries
- Complex geometries by multiple level-set/multi cut cell
- Stabilization of small cut cells neccessary



multiply-cut cell



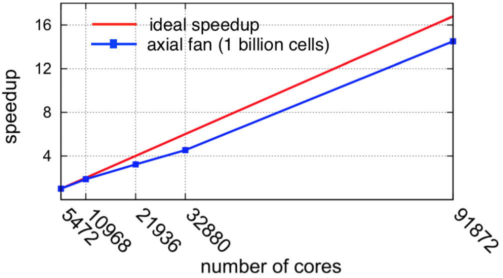
split cut cell

Schneiders et al., An accurate moving boundary formulation in cut-cell methods, J. Comput. Phys. 235 (2013)

Strong scaling cut cells

Solution 3D Navier-Stokes equations including cut cells:

- 5-stage explicit Runge-Kutta time stepping method, $\mathcal{O}(\Delta t^2)$
- Advective terms: AUSM (Advection Upstream Splitting Method) with modified pressure splitting, $\mathcal{O}(\Delta x^2)$
- Viscous terms: central differences, $\mathcal{O}(\Delta x^2)$

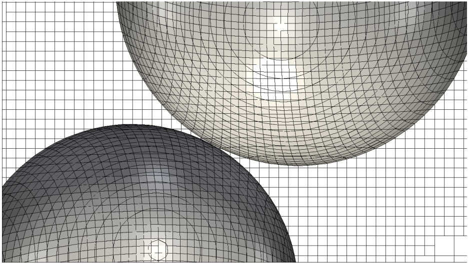


A. Pogorelov et al., Cut-cell method based large-eddy simulation of tip-leakage flow, Physics of Fluids 27 (2015)

Examples

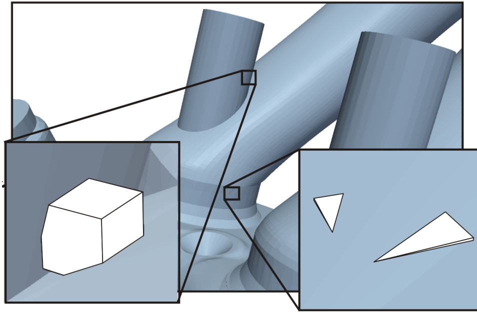
Particle-particle collisions

- Sharp resolution of the gap in between colliding particles
- Conservation: no loss of mass pushed out of the gap

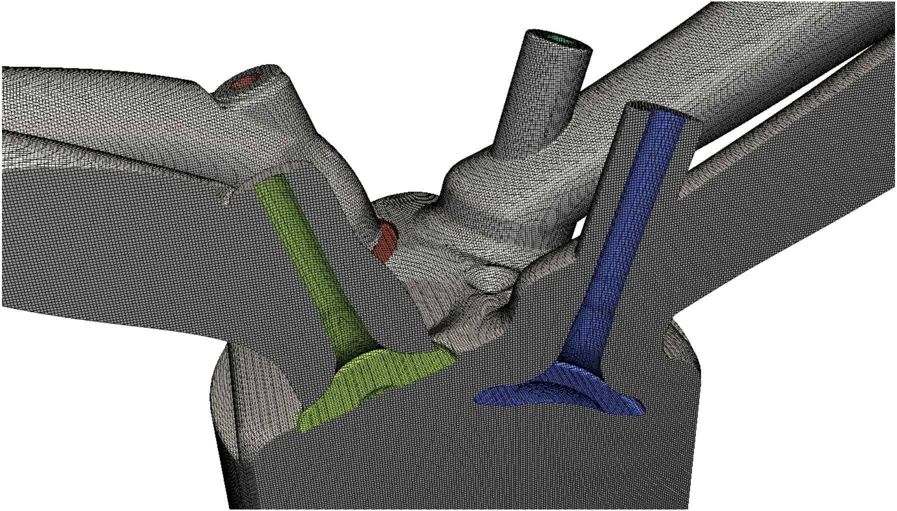


Technical flows

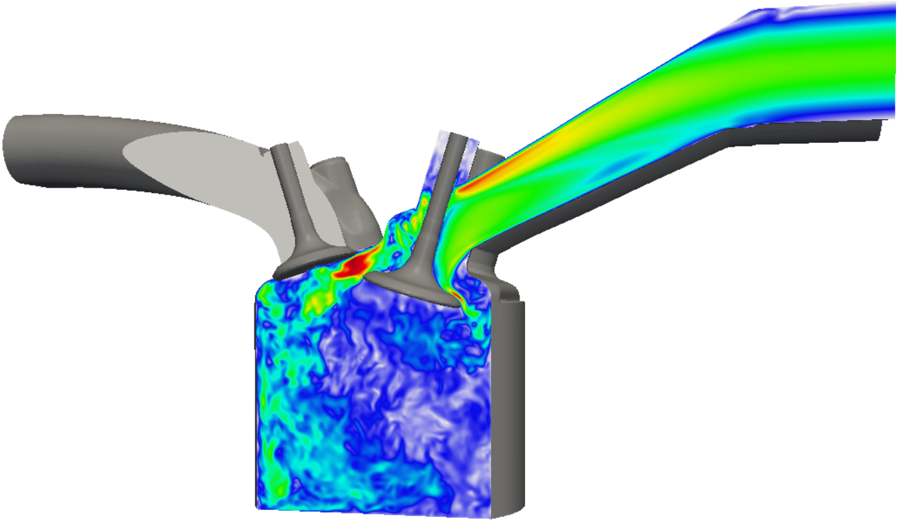
- Accurate and robust handling of sharp geometric features
- No mass leaks by moving parts



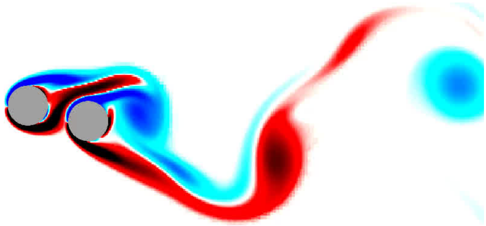
Combustion engine

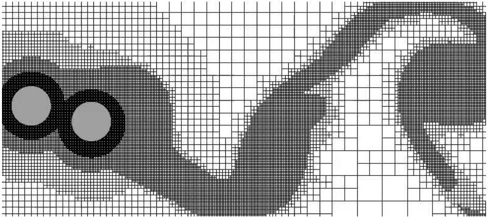


Combustion engine



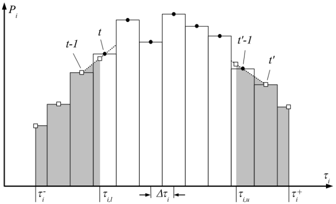
Adaptive mesh refinement





Sensor-based adaptation

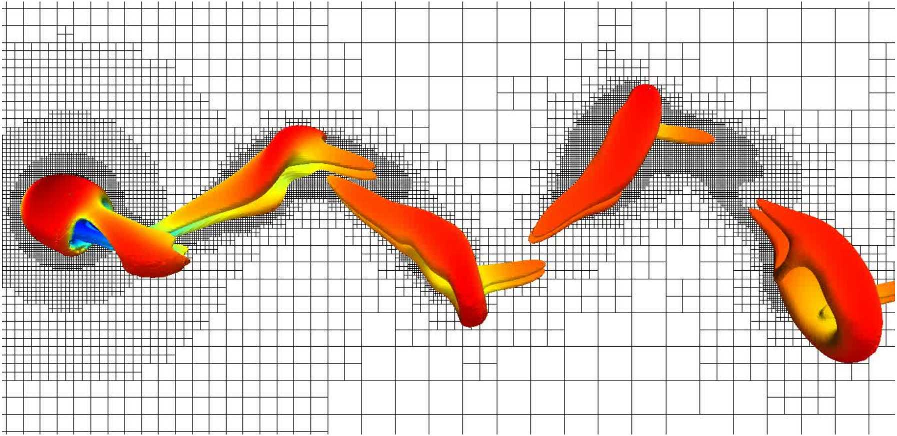
- Vorticity:
  $\tau_c = |\nabla \times \mathbf{u}|\Delta x^{3/2}$
- Entropy gradient:
  $\tau_e = |\nabla p - a^2 \nabla \rho|\Delta x^{3/2}$
- Wall distance: $\tau_\delta = |\phi|/\delta$



Hartmann at al., An adaptive multilevel multigrid formulation for Cartesian hierarchical grid methods, Comput. Fluids 37 (2008)

Elastically mounted sphere

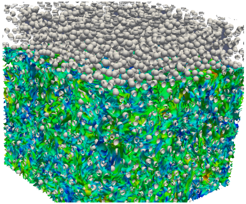
Elastically mounted sphere, 3 DOF, $Re_D = 300$, $U_{red} = 7$

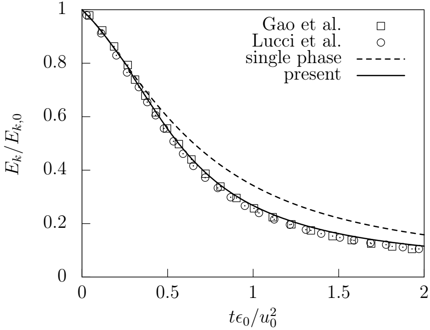


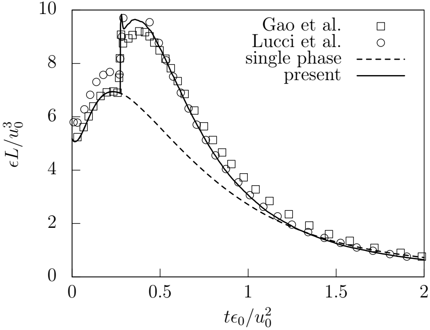
Spherical particles

Spheres in decaying isotropic turbulence

- $Re_\lambda \approx 75$, $N_p = 256^3$ cells, 6400 spheres
- $d_p \sim \lambda$, $\rho_p/\rho_f = 2.56$, $\phi_m = 0.25$

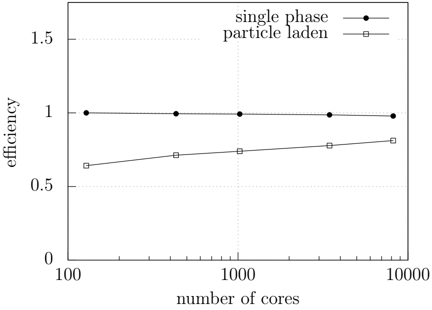




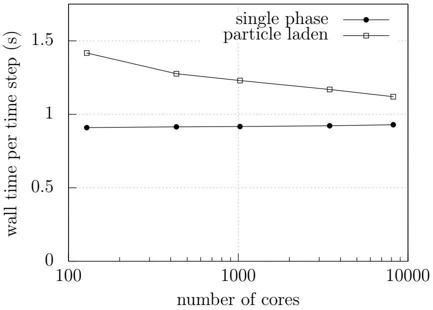


Lucci, Ferrante, Elghobashi: J. Fluid Mech. (2010), Gao, Li, Wang: Comp. Math. App. (2013)

Weak scaling (unbalanced)



efficiency



walltime per time step

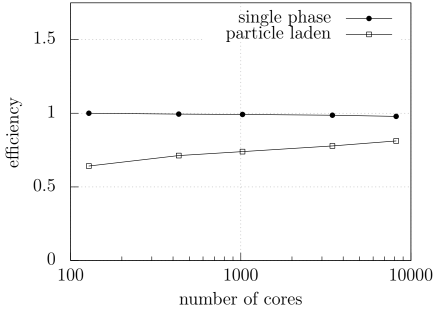
uniform mesh, $256^3 \rightarrow 1024^3$ cells

$128 \rightarrow 8192$ cores

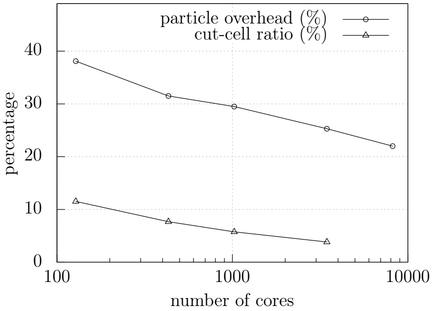
131 072 cells per core

Schneiders, Günther, Meinke, Schröder, An efficient conservative cut-cell method for rigid bodies interacting with viscous compressible flows, J. Comput. Phys. 311 (2016)

Weak scaling (unbalanced)



efficiency



particle overhead, cut-cell ratio

uniform mesh, $256^3 \rightarrow 1024^3$ cells

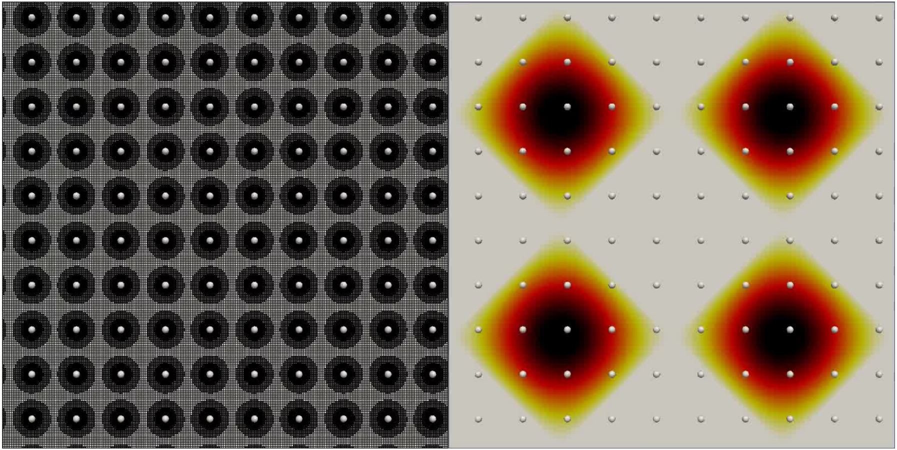
$128 \rightarrow 8192$ cores

131 072 cells per core

Schneiders, Günther, Meinke, Schröder, An efficient conservative cut-cell method for rigid bodies interacting with viscous compressible flows, J. Comput. Phys. 311 (2016)

Dynamic mesh refinement

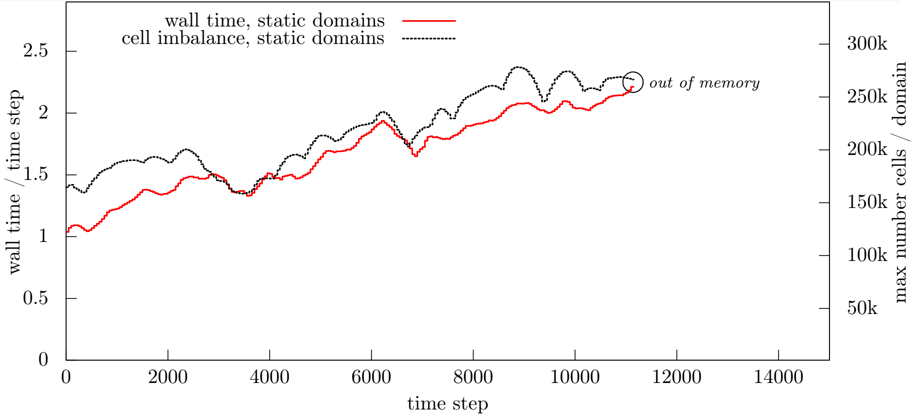
Taylor-Green vortex; $Re = 1600$; showing 100 particles in z-plane



Dynamic load balancing

Taylor-Green vortex; 4000 particles; $\mathcal{O}(10^9)$ cells; 20,000 cores

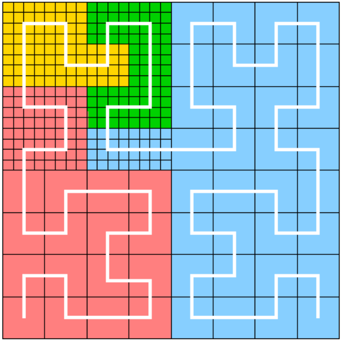
- Mesh adaptation every 50th time step $\rightarrow$ overhead $\sim 8\%$
- Out of memory and performance drop due to particle clustering



Dynamic load balancing

Parallel domain decomposition

- Hilbert curve on background mesh weighted by number of offsprings
- Depth-first ordering on hierarchical octree data structure
- Fully automated
- Recompute domain boundaries and redistribute cells if imbalance too high

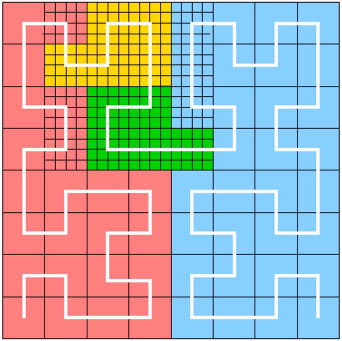


Lintermann et al., Massively parallel grid generation on HPC systems, Comput. Meth. Appl. Mech. Eng. (2014)

Dynamic load balancing

Parallel domain decomposition

- Hilbert curve on background mesh weighted by number of offsprings
- Depth-first ordering on hierarchical octree data structure
- Fully automated
- Recompute domain boundaries and redistribute cells if imbalance too high

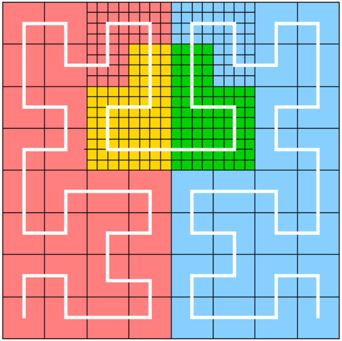


Lintermann et al., Massively parallel grid generation on HPC systems, Comput. Meth. Appl. Mech. Eng. (2014)

Dynamic load balancing

Parallel domain decomposition

- Hilbert curve on background mesh weighted by number of offsprings
- Depth-first ordering on hierarchical octree data structure
- Fully automated
- Recompute domain boundaries and redistribute cells if imbalance too high



Lintermann et al., Massively parallel grid generation on HPC systems, Comput. Meth. Appl. Mech. Eng. (2014)

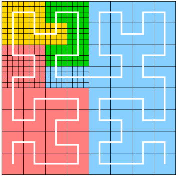
Dynamic load balancing

Strategy 1

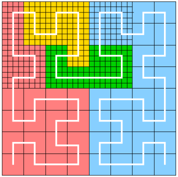
- Adapt mesh to reach specified target number of cells
- Redistribute cells to keep load approx. constant
- Con: target cell number case-dependent parameter

Strategy 2

- Adapt mesh as needed (number of cells free param.)
- Redistribute cells to balance load
- Restart using more cores if average load too high
- Domain decomposition fully automatic, MPI I/O



4 domains, 192 cells



4 domains, 272 cells

7 domains, 352 cells

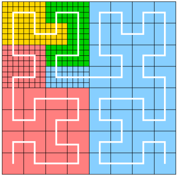
Dynamic load balancing

Strategy 1

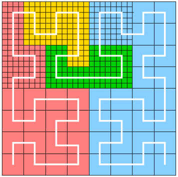
- Adapt mesh to reach specified target number of cells
- Redistribute cells to keep load approx. constant
- Con: target cell number case-dependent parameter

Strategy 2

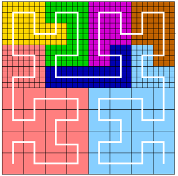
- Adapt mesh as needed (number of cells free param.)
- Redistribute cells to balance load
- Restart using more cores if average load too high
- Domain decomposition fully automatic, MPI I/O



4 domains, 192 cells



4 domains, 272 cells

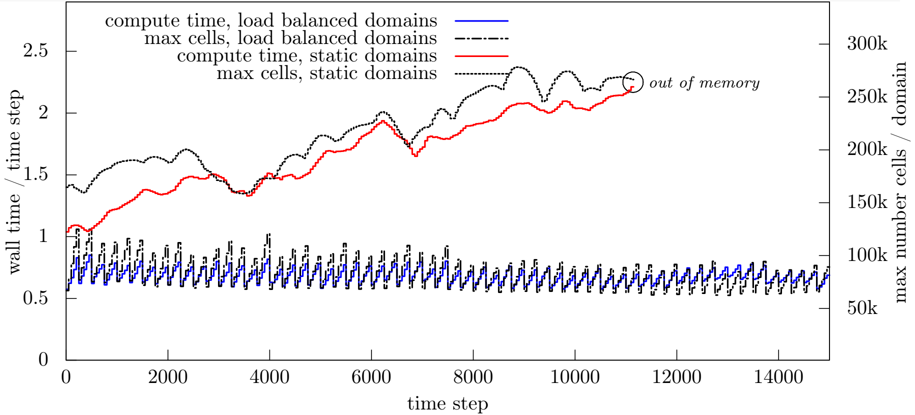


7 domains, 352 cells

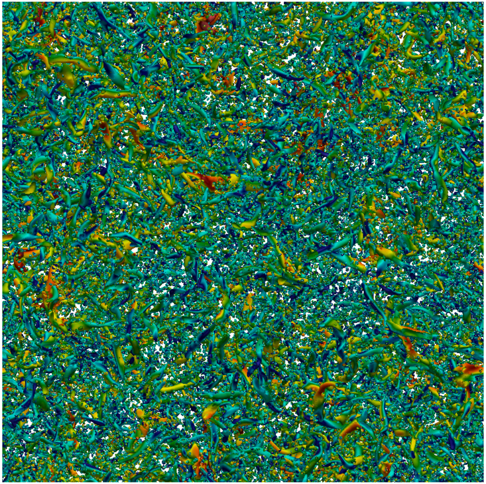
Dynamic load balancing

Taylor-Green vortex; 4000 particles; $\mathcal{O}(10^9)$ cells; 20,000 cores

- Mesh adaptation every 50th time step → overhead ∼ 8%
- Load balancing every 250th time step → overhead ∼ 6%

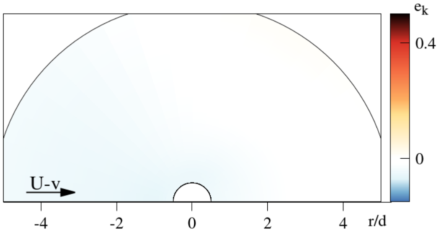


Results

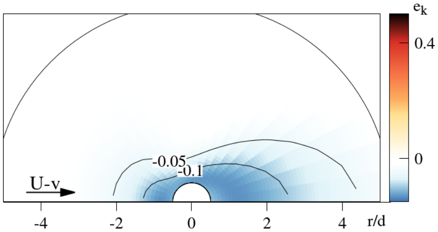


Simulation details: $N_p = 45\,000$ ($d_p \sim \eta$); $2 \cdot 10^9$ cells; $48\,000$ cores at Hazel Hen (HLRS)

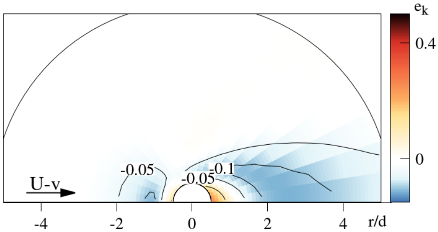
Near-particle statistics



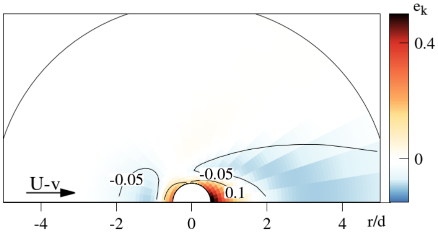
$\rho_p/\rho_f = 40$



$\rho_p/\rho_f = 200$



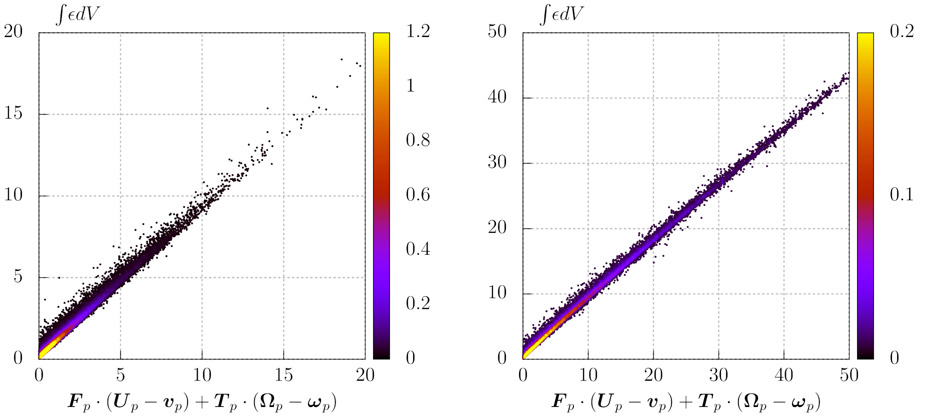
$\rho_p/\rho_f = 1000$



$\rho_p/\rho_f = 5000$

Schneiders, Meinke, Schröder, Interaction of isotropic turbulence with particles of Kolmogorov-length scale size, submitted to Journal of Fluid Mechanics (2016)

Particle-induced dissipation



$\rho_p/\rho_f = 200$ $\rho_p/\rho_f = 5000$

$$\int_{\Sigma_p(t)} \epsilon \, dV = \boldsymbol{F}_p \cdot (\boldsymbol{U}_p - \boldsymbol{v}_p) + \mathcal{T}_p \cdot (\boldsymbol{\Omega}_p - \boldsymbol{\omega}_p),$$

Schneiders, Meinke, Schröder, Interaction of isotropic turbulence with particles of Kolmogorov-length scale size, submitted to Journal of Fluid Mechanics (2016)

Summary and outlook

Summary

- Strictly conservative cut-cell method for complex moving geometries
- Dynamic load balancing to enable dynamic mesh refinement
- Novel results for turbulence modulation by particles at $d_p \sim \eta$

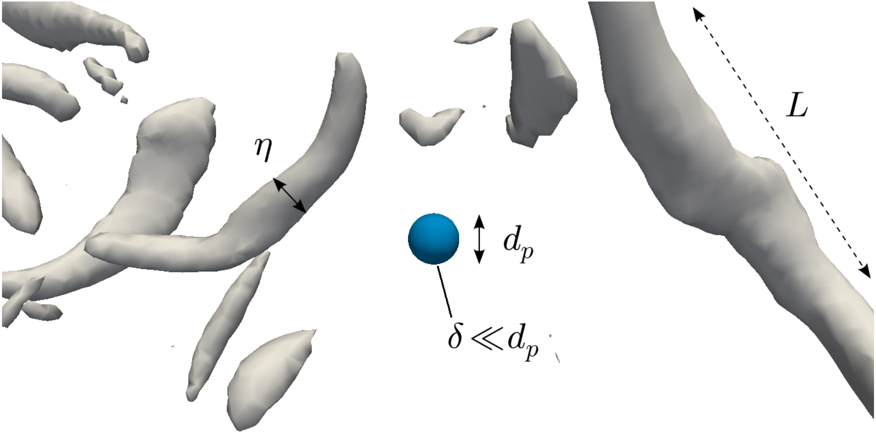
Performance issues

- I/O overhead since dynamic mesh has to be stored
- Load-balancing does not anticipate the future
- Computations highly memory-bound, low peak performance
- Large data sets for sampling, on-the-fly statistics expensive

Funding: DFG (SFB/TRR 129 "Oxyflame")

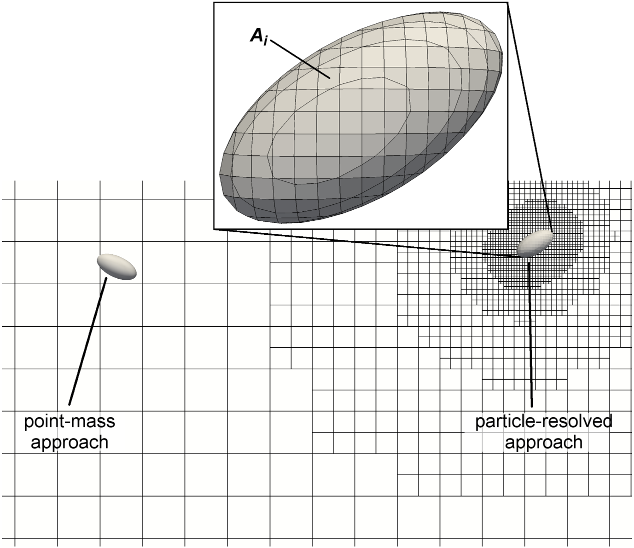
Compute time: HLRS Stuttgart

Particle phase representation

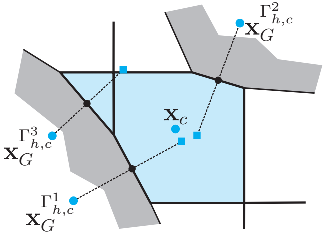


Direct numerical simulation (DNS): $\Delta x \sim \eta$
Particle-resolved simulation (PRS): $\Delta x \sim \delta_p$ $(\ll \eta$ if $d_p \sim \eta)$
Literally no studies for the case $d_p \sim \eta$ due to enormous comp. costs

DNS mesh



Near-boundary discretization



Boundary conditions

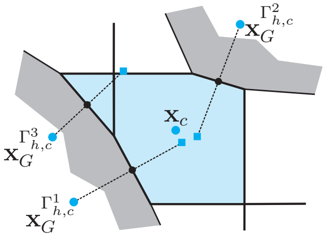
- Interpolation of primitive variables at image points (2nd order WLSQ)
- Extrapolation to mirror points/ghost cells

Small-cell treatment

- $\widetilde{\mathbf{Q}} = \mathbf{Q} + (1-\kappa)(\mathbf{Q}^i - \mathbf{Q}) + \mathbf{E}$
- Interpolated update $\mathbf{Q}^i$ provides stability
- Conservation defect $\mathbf{D} = (1-\kappa)(\mathbf{Q} - \mathbf{Q}^i)$
- Flux echange term $\mathbf{E}_c = \sum_{I \in N_c} \sigma_{I,c} V_I \mathbf{D}_I / V_c$
- κ continuously differentiable, $\kappa \to 0$ as $V \to 0$

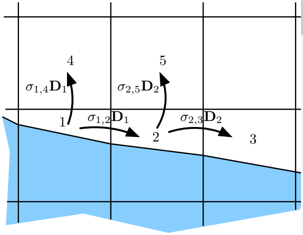
L. Schneiders, D. Hartmann, M. Meinke, W. Schröder, An accurate moving boundary formulation in cut-cell methods, J. Comput. Phys. 235 (2013) 786–809.

Near-boundary discretization



Boundary conditions

- Interpolation of primitive variables at image points (2nd order WLSQ)
- Extrapolation to mirror points/ghost cells



Small-cell treatment

- $\widetilde{\mathbf{Q}} = \mathbf{Q} + (1-\kappa)(\mathbf{Q}^i - \mathbf{Q}) + \mathbf{E}$
- Interpolated update $\mathbf{Q}^i$ provides stability
- Conservation defect $\mathbf{D} = (1-\kappa)(\mathbf{Q} - \mathbf{Q}^i)$
- Flux echange term $\mathbf{E}_c = \sum_{l \in N_c} \sigma_{l,c} V_l \mathbf{D}_l / V_c$
- κ continuously differentiable, $\kappa \to 0$ as $V \to 0$

L. Schneiders, D. Hartmann, M. Meinke, W. Schröder, An accurate moving boundary formulation in cut-cell methods, J. Comput. Phys. 235 (2013) 786–809.

Efficient time stepping

$$
\begin{array}{c|cccccc}
0 & & & & & & \\
\alpha_1 & \alpha_1 & & & & & \\
\alpha_2 & 0 & \alpha_2 & & & & \\
\alpha_3 & 0 & 0 & \alpha_3 & & & \\
\vdots & \vdots & & \ddots & \ddots & & \\
\alpha_{s-1} & 0 & & \dots & & \alpha_{s-1} & \\
\hline
1 & 0 & & \dots & & 0 & 1
\end{array}
$$

Table : Multi-stage Runge-Kutta scheme (*MS-RK*)

$$
\begin{array}{c|cccccc}
0 & & & & & & \\
1 & 1 & & & & & \\
1 & 1-\alpha_1 & \alpha_1 & & & & \\
1 & 1-\alpha_2 & 0 & \alpha_2 & & & \\
\vdots & \vdots & \vdots & \ddots & \ddots & & \\
1 & 1-\alpha_{s-2} & 0 & \dots & 0 & \alpha_{s-2} & \\
\hline
1 & 1-\alpha_{s-1} & 0 & \dots & & 0 & \alpha_{s-1}
\end{array}
$$

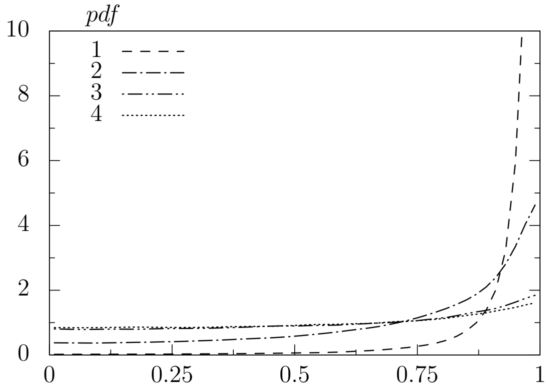
Table : Predictor-corrector Runge-Kutta scheme (*PC-RK*)

Let overhead for solver reinitialization $\sigma := t_{init}/(t_{init} + t_{exec})$

Overall speedup $= 1 + (s-1)\sigma$

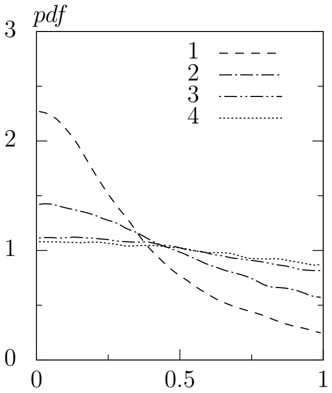
Here: $s = 5$, $\sigma = 0.38$, $\rightarrow$ $\boxed{\text{speedup} = 2.5}$

Near-particle statistics



$|\boldsymbol{v}_p \cdot \boldsymbol{U}_p|/(|\boldsymbol{v}_p|\,|\boldsymbol{U}_p|)$

alignment of particle and fluid velocity

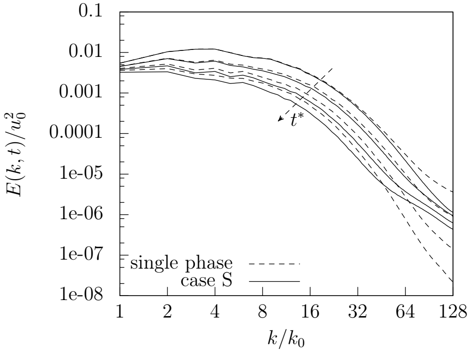


$|\boldsymbol{\omega}_p \cdot \boldsymbol{e}_\gamma|/|\boldsymbol{\omega}_p|$

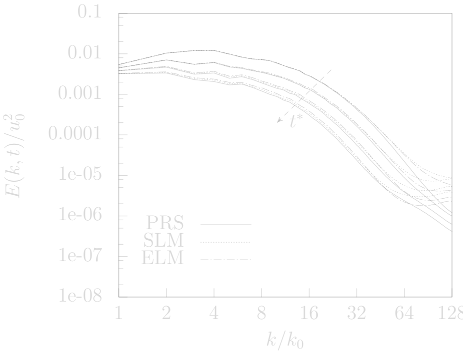
alignment particle and fluid rotation

Schneiders, Meinke, Schröder, Interaction of isotropic turbulence with particles of Kolmogorov-length scale size, submitted to Journal of Fluid Mechanics (2016)

Kinetic energy spectra



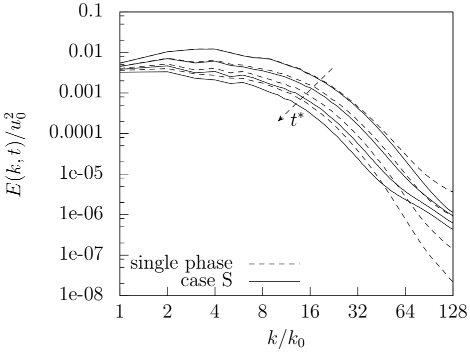
particle-laden vs. particle-free flow



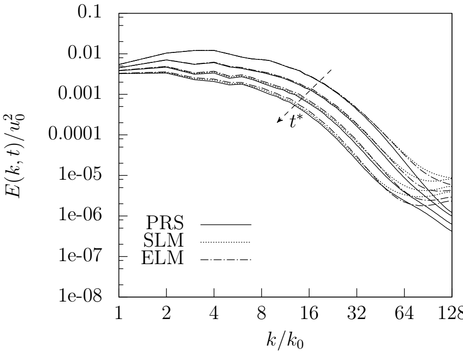
fully-resolved vs. point particle models

Schneiders, Meinke, Schröder, On the accuracy of Lagrangian point-mass models for heavy non-spherical particles in isotropic turbulence, accepted for publication in Fuel (2016)

Kinetic energy spectra



particle-laden vs. particle-free flow



fully-resolved vs. point particle models

Schneiders, Meinke, Schröder, On the accuracy of Lagrangian point-mass models for heavy non-spherical particles in isotropic turbulence, accepted for publication in Fuel (2016)

Efficient time stepping

Multi-stage Runge-Kutta scheme (*MS-RK*)

Van der Houwen (1972), *Jameson* (1983)

$$(\boldsymbol{Q}V)^{(0)} = (\boldsymbol{Q}V)^{n},$$
$$(\boldsymbol{Q}V)^{(k)} = (\boldsymbol{Q}V)^{(0)} - \alpha_k \Delta t\; \boldsymbol{R}(t^n + \alpha_{k-1}\Delta t;\; \boldsymbol{Q}^{(k-1)}), \quad k = 1, \dots, s,$$
$$(\boldsymbol{Q}V)^{n+1} = (\boldsymbol{Q}V)^{(s)}.$$

e.g. $\boldsymbol{\alpha} = \{1/4,\; 1/6,\; 3/8,\; 1/2,\; 1\}$

Predictor-corrector Runge-Kutta scheme (*PC-RK*)

$$(\boldsymbol{Q}V)^{(0)} = (\boldsymbol{Q}V)^{n},$$
$$(\boldsymbol{Q}V)^{(1)} = (\boldsymbol{Q}V)^{(0)} - \Delta t\; \boldsymbol{R}(t^n; \boldsymbol{Q}^{(0)}),$$
$$(\boldsymbol{Q}V)^{(k)} = (\boldsymbol{Q}V)^{(0)} - \Delta t \left[(1-\alpha_{k-1})\boldsymbol{R}(t^n; \boldsymbol{Q}^{(0)}) + \alpha_{k-1}\boldsymbol{R}(t^{n+1}; \boldsymbol{Q}^{(k-1)})\right]$$
$$(\boldsymbol{Q}V)^{n+1} = (\boldsymbol{Q}V)^{(s)}. \qquad k = 2, \dots, s$$

Schneiders et al., An efficient conservative cut-cell method for rigid bodies interacting with viscous compressible flows, J. Comput. Phys. 311 (2016)

Efficient time stepping

Multi-stage Runge-Kutta scheme (*MS-RK*)

$$(QV)^{(0)} = (QV)^{n},$$

Van der Houwen (1972), *Jameson* (1983)

$$(QV)^{(k)} = (QV)^{(0)} - \alpha_k \Delta t\ \boldsymbol{R}(t^n + \alpha_{k-1}\Delta t;\ \boldsymbol{Q}^{(k-1)}), \quad k = 1, \dots, s,$$

$$(QV)^{n+1} = (QV)^{(s)}.$$

e.g. $\boldsymbol{\alpha} = \{1/4,\ 1/6,\ 3/8,\ 1/2,\ 1\}$

Predictor-corrector Runge-Kutta scheme (*PC-RK*)

$$(QV)^{(0)} = (QV)^{n},$$

$$(QV)^{(1)} = (QV)^{(0)} - \Delta t\ \boldsymbol{R}(t^n; \boldsymbol{Q}^{(0)}),$$

$$(QV)^{(k)} = (QV)^{(0)} - \Delta t \left[(1-\alpha_{k-1})\boldsymbol{R}(t^n; \boldsymbol{Q}^{(0)}) + \alpha_{k-1}\boldsymbol{R}(t^{n+1}; \boldsymbol{Q}^{(k-1)})\right]$$

$$(QV)^{n+1} = (QV)^{(s)}. \qquad k = 2, \dots, s$$

Schneiders et al., An efficient conservative cut-cell method for rigid bodies interacting with viscous compressible flows, J. Comput. Phys. 311 (2016)