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
Research projects involving turbulent particle-laden flow

- Models verified and improved via particle-resolved simulations

Particle deposition in human airways

Coal/biomass combust.

Precipitation modeling

Electrical discharge machining

LEAT, RU Bochum, oxyflame.com

edm-huber.de
Research projects involving turbulent particle-laden flow

- Models verified and improved via particle-resolved simulations

Improved particle models required:

- \( d_p \sim \eta \)
- non-spherical particles
- high-temperature environments
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
Laminar-turbulent transition in a round jet

B. O. Andersen, TU Denmark
Fluid motion:

\[
\frac{d}{dt} \int_{V(t)} Q \, dV + \int_{\partial V(t)} H \cdot n \, dA = 0, \quad Q = [\rho, \rho u, \rho E]
\]
Rigid body acceleration: \[ m \frac{dv}{dt} = F \]

Angular acceleration: \[ \mathcal{I} \frac{d\tilde{\omega}}{dt} + \tilde{\omega} \times (\mathcal{I}\tilde{\omega}) = \tilde{T} \]
Surface force: \[ F_p = \oint_{\Gamma_p} (-p\mathbf{n} + \mathbf{\tau} \cdot \mathbf{n}) dA, \]

Surface torque: \[ T_p = \oint_{\Gamma_p} (\mathbf{x} - \mathbf{r}_p) \times (-p\mathbf{n} + \mathbf{\tau} \cdot \mathbf{n}) dA \]
Motivation

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
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 necessary

Solution 3D Navier-Stokes equations including cut cells:

- 5-stage explicit Runge-Kutta time stepping method, $O(\Delta t^2)$
- Advective terms: AUSM (Advection Upstream Splitting Method) with modified pressure splitting, $O(\Delta x^2)$
- Viscous terms: central differences, $O(\Delta x^2)$
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
Sharp-interface Cartesian method
Combustion engine
Combustion engine
Sensor-based adaptation

- Vorticity:
  \[ \tau_c = |\nabla \times u| \Delta x^{3/2} \]

- Entropy gradient:
  \[ \tau_e = |\nabla p - a^2 \nabla \rho| \Delta x^{3/2} \]

- Wall distance:
  \[ \tau_\delta = |\phi|/\delta \]

Elastically mounted sphere, 3 DOF, $Re_D = 300$, $U_{red} = 7$
Spheres in decaying isotropic turbulence

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

Gao et al.  Lucci et al.  single phase  present

Weak scaling (unbalanced)

efficiency

wall time per time step

uniform mesh, $256^3 \rightarrow 1024^3$ cells
128 $\rightarrow$ 8192 cores
131072 cells per core

Simulation of Turbulent Particulate Flow on HPC Systems | 24th Workshop for Sustained Simulation

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

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

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

![Graph showing wall time and max number of cells/time step with an indication of out of memory issue.](image)
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

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

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

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

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

![Graph showing wall time, load balanced domains, max cells, load balanced domains, compute time, static domains, max cells, static domains, with out of memory marker.](image)
Simulation details: $N_p = 45\,000 \left( d_p \sim \eta \right); 2 \cdot 10^9$ cells; 48,000 cores at Hazel Hen (HLRS)
Near-particle statistics

\[ \frac{\rho_p}{\rho_f} = 40 \]

\[ \frac{\rho_p}{\rho_f} = 200 \]

\[ \frac{\rho_p}{\rho_f} = 1000 \]

\[ \frac{\rho_p}{\rho_f} = 5000 \]

\[ \int \epsilon \, dV = F_p \cdot (U_p - v_p) + T_p \cdot (\Omega_p - \omega_p), \]

\[ \Sigma_p(t) \]

\[ \rho_p/\rho_f = 200 \]

\[ \rho_p/\rho_f = 5000 \]

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
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

- point-mass approach
- particle-resolved approach

$A_i$
Boundary conditions

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

Small-cell treatment

- \( \tilde{Q} = Q + (1 - \kappa)(Q^i - Q) + E \)
- Interpolated update \( Q^i \) provides stability
- Conservation defect \( D = (1 - \kappa)(Q - Q^i) \)
- Flux exchange term \( E_c = \sum_{l \in N_c} \sigma_{l,c} V_l D_l / V_c \)
- \( \kappa \) continuously differentiable, \( \kappa \to 0 \) as \( V \to 0 \)

Near-boundary discretization

Boundary conditions

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

Small-cell treatment

- $\tilde{Q} = Q + (1 - \kappa)(Q^i - Q) + E$
- Interpolated update $Q^i$ provides stability
- Conservation defect $D = (1 - \kappa)(Q - Q^i)$
- Flux exchange term $E_c = \sum_{l \in Nc} \sigma_{l,c} V_l D_l / V_c$
- $\kappa$ continuously differentiable, $\kappa \to 0$ as $V \to 0$

## Efficient time stepping

\[
\begin{array}{c|ccc}
0 & \alpha_1 & \alpha_2 & \alpha_3 \\
\alpha_1 & 0 & \alpha_2 & \alpha_3 \\
\alpha_2 & 0 & 0 & \alpha_3 \\
\vdots & \vdots & \ddots & \vdots \\
\alpha_{s-1} & 0 & \ldots & \alpha_{s-1} \\
1 & 0 & \ldots & 0 \end{array}
\]

Table: Multi-stage Runge-Kutta scheme (\textit{MS-RK})

\[
\begin{array}{c|ccc}
0 & 1 & 1 & 1 \\
1 & 1-\alpha_1 & \alpha_1 & 1 \\
1 & 1-\alpha_2 & 0 & \alpha_2 \\
1 & 1-\alpha_{s-2} & 0 & \ldots \quad 0 \quad \alpha_{s-2} \\
1 & 1-\alpha_{s-1} & 0 & \ldots \quad 0 \quad \alpha_{s-1} \\
\end{array}
\]

Table: Predictor-corrector Runge-Kutta scheme (\textit{PC-RK})

Let overhead for solver reinitialization \(\sigma \equiv t_{\text{init}}/(t_{\text{init}} + t_{\text{exec}})\)

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

Here: \(s = 5, \sigma = 0.38\), \(\rightarrow\) speedup = 2.5
Near-particle statistics

![Graphs showing pdf distributions for alignment of particle and fluid velocity, and alignment of particle and fluid rotation.]

Summary

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)
## Multi-stage Runge-Kutta scheme \((MS-RK)\)

\[
(QV)^{(0)} = (QV)^n, \\
(QV)^{(k)} = (QV)^{(0)} - \alpha_k \Delta t \ R(t^n + \alpha_{k-1} \Delta t; \ Q^{(k-1)}), \quad k = 1, \ldots, s, \\
(QV)^{n+1} = (QV)^{(s)}. \\
\]

\(\alpha = \{1/4, 1/6, 3/8, 1/2, 1\}\)


---

## Predictor-corrector Runge-Kutta scheme \((PC-RK)\)

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

\(k = 2, \ldots, s\)
Multi-stage Runge-Kutta scheme (MS-RK)

\[
(QV)^{(0)} = (QV)^n, \quad \text{Van der Houwen (1972), Jameson (1983)}
\]

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

\[
(QV)^{n+1} = (QV)^{(s)}.
\]

E.g. \(\alpha = \{1/4, 1/6, 3/8, 1/2, 1\}\)

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

\[
(QV)^{(0)} = (QV)^n,
\]

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

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

\[
(QV)^{n+1} = (QV)^{(s)}. \quad k = 2, \ldots, s
\]