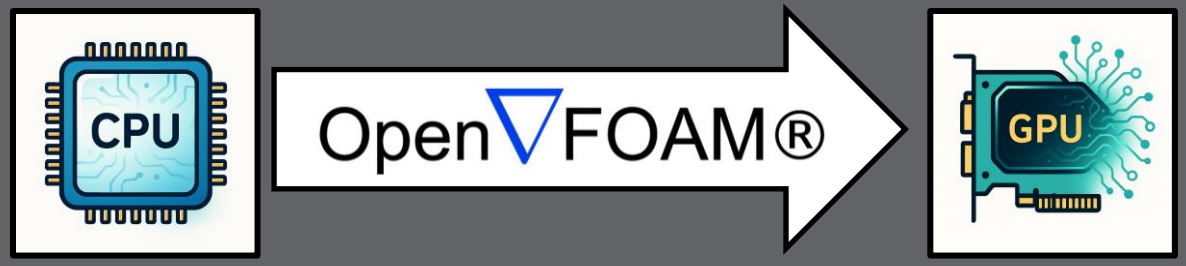
upstreamCFD

Detailed comparison of performance and quality between GPU and CPU simulations with OpenFOAM

Hendrik Hetmann[1], Simone Bnà[2], Giuseppe Giaquinto[2], Marian Fuchs[1], Charles Mockett[1], Alessio Piccolo[2], Tommaso Zanelli[2]

hendrik.hetmann@upstream-cfd.com



5th Automotive CFD Prediction Workshop
8th – 9th October 2026
Bologna, Italy

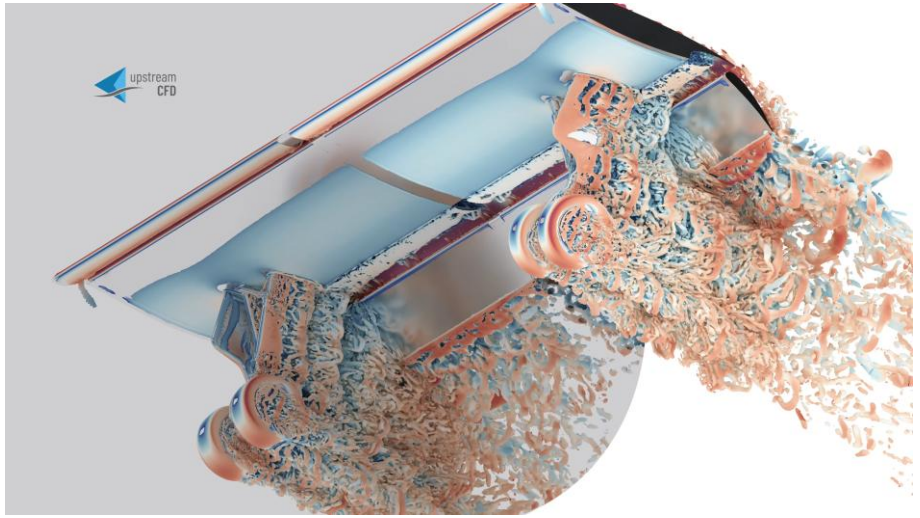
[1] Upstream CFD GmbH, Berlin, Germany

[2] CINECA, Bologna, Italy

Introduction & Motivation

Our motivation at Upstream CFD:

- We are experts in high-fidelity CFD for aerodynamics and aeroacoustics, and offer consulting services for scale-resolving simulations (SRS) methods using open-source CFD tools such as OpenFOAM
- For the adoption of compute-hungry SRS methods in industry, code efficiency is key and GPU computing is an important enabler
- Commercial software vendors and new start-ups started to re-write CFD codes for GPUs in recent years, where a couple of different GPU initiatives also emerged in the OpenFOAM community (e.g. SPUMA, NeoN, gpuFoam)





Outline of talk:

- In our AutoCFD5 contribution, we set out to benchmark the new SPUMA code for the DrivAer test case
- The key questions are:
  - Does SPUMA produce the same results as OpenFOAM?
  - What is the speed/cost/energy benefit of running on GPUs?

Introduction of OpenFOAM GPU port SPUMA

- **Approach and key features:**
  - Port of OpenFOAM to GPU by identifying and grouping three basic low-level algorithm types and replacing them with GPU optimised variants:
    - For-loop
    - Reduction summation
    - Reduction comparison
  - Unified memory + memory pool approach:
    - Almost all OpenFOAM functions will work without error
    - Non-ported features will keep running on CPU and slow down the simulation significantly
    - Incremental porting possible
- Link to paper pre-print for more details

GitLab Repository:	https://gitlab.hpc.cineca.it/exafoam/spuma (GPL3 License)

- **Developed by:** Simone Bnà and team at CINECA High Performance Computing Department



- **Status:**
  - Current release 0.3-v2512:
    - Most common solvers GPU ready: *simpleFoam, pimpleFoam + compressible variants + buoyant variants*
    - Standard turbulence models available
    - Most common linear solvers, preconditioners and smoothers available
    - Thermophysical modelling & AMI interpolation
    - For more info see: https://gitlab.hpc.cineca.it/exafoam/spuma/-/wikis/GPU-support
  - Currently working on features for rotating machinery (e.g. mesh-to-mesh interface)

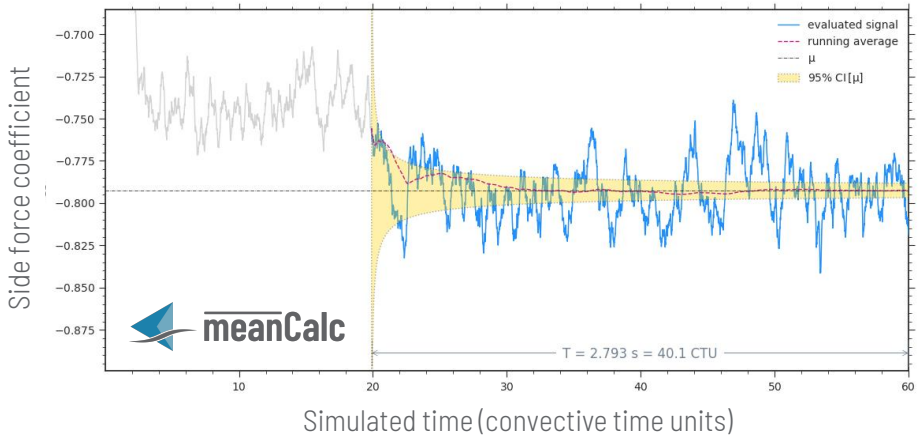
Simulation setup for CPU and GPU runs

- CPU solver: Keysight-OpenCFD's OpenFOAM v2412
- GPU solver: CINECA's SPUMA 0.2-v2412

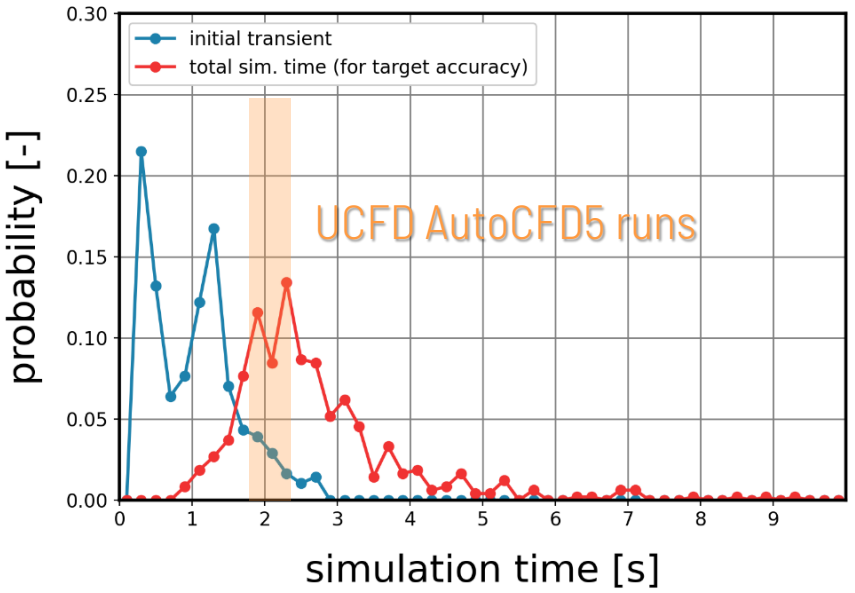
- Solver & simulation settings:
  - Customised version of incompressible, pressure-based solver *pimpleFoam* (SIMPLE algorithm) → enhanced Rhie-Chow interpolation for consistent pressure-velocity coupling and lower dissipation
  - Both codes were run in double precision
  - Meancalc run-time control active to ensure that all simulations converge to the same statistical error interval (i.e. 1.5 drag counts)
- Turbulence modelling:
  - Grey-area enhanced SA-σ-DDES model (2019) with improved boundary layer protection function from Deck & Renard (2020)
  - All y^+ wall function based on Spalding law of the wall
- Discretisation schemes:
  - Modified hybrid convection scheme of Travin et al. (2000) → blending between 2nd order CDS and 2nd order upwind
  - 2nd order implicit Euler time integration scheme, $\Delta t = 10^{-4}$ s

Simulation run-time control via Meancalc

- For all our CPU and GPU simulations, the simulation run-time control was automatically controlled via Meancalc statistics tool:
  - Simulation is automatically stopped once Meancalc says that the initial transient is bridged, a minimum sample size is simulated and a user-defined target accuracy is reached

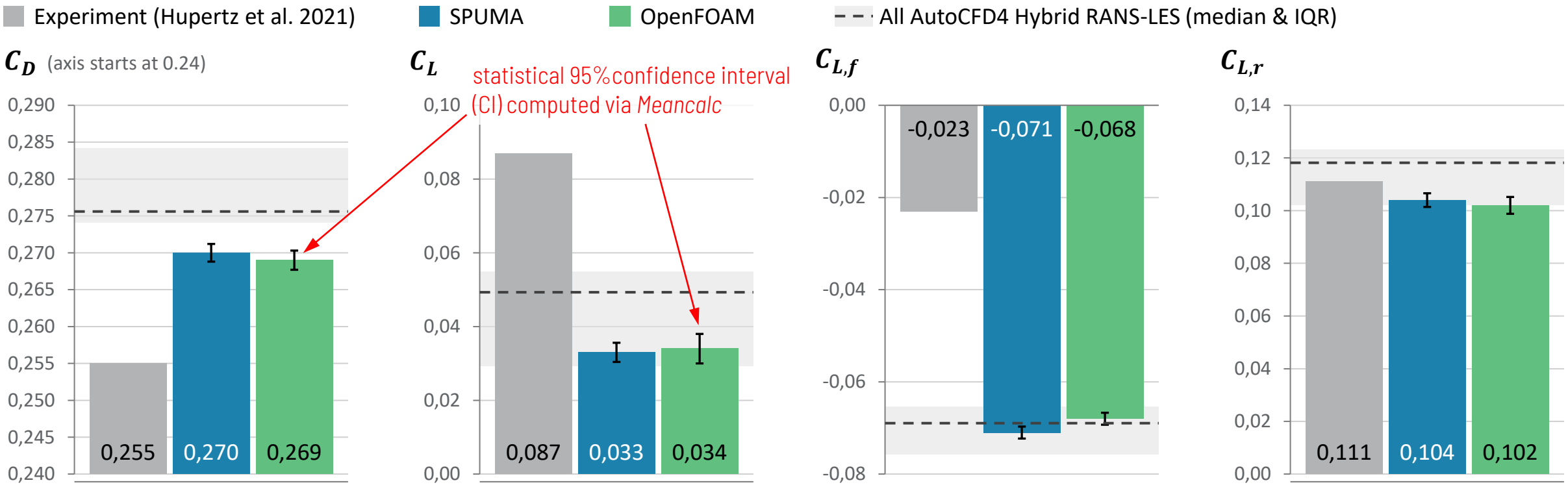


- Probability density function for **initial transient** and **required simulation length** to reach target accuracy of DrivAerML study (500 simulations, 16 geometric design variables) by Ashton et al. (2024):
  - Large scatter seen → a fixed run length either wastes computational resources (if too long) or exceeds the target accuracy (if too short)
  - Our AutoCFD5 runs (SPUMA & OpenFOAM) stopped between 1.8s and 2.3s simulated time using a target 95% CI of 1.5 drag counts

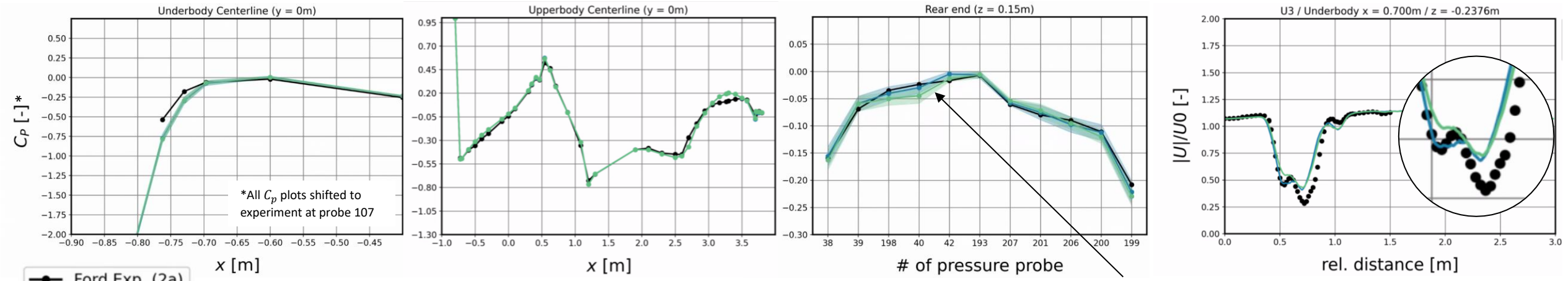


Simulation quality: Force coefficients (DrivAer 2a)

- SPUMA and OpenFOAM are statistically indistinguishable for all 4 coefficients (i.e. within 95% CI)
- Both codes overpredict drag by ~0.015 and significantly underpredict the measured lift, mostly on the front axle → in line with most CFD results presented at AutoCFD4

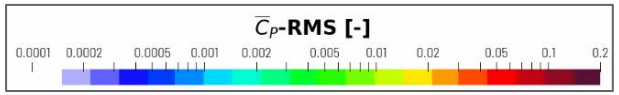


Simulation quality: Surface pressure distribution (DrivAer 2a)

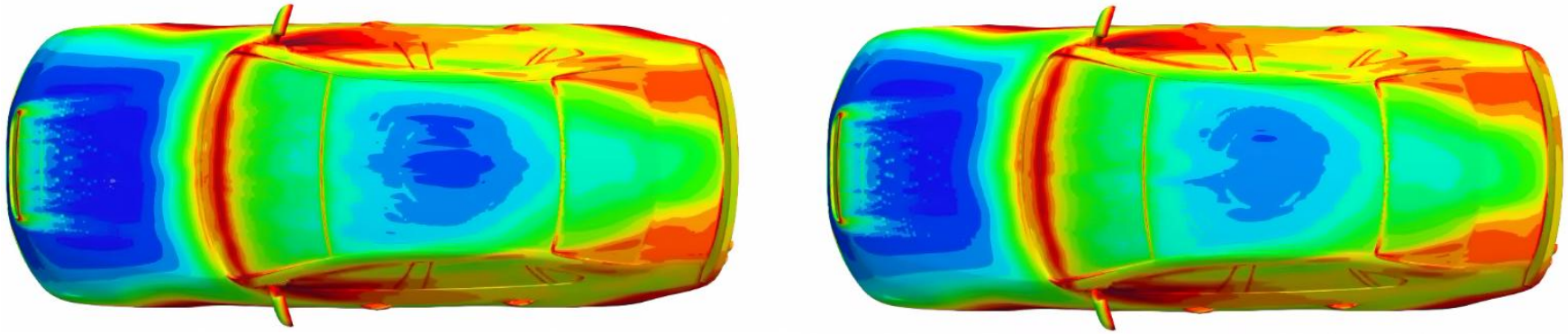


surface pressure agrees in majority of locations and even more sensitive RMS levels are very similar

differences observed in a few sensitive areas potentially due to statistics



SPUMA



OpenFOAM

Performance: SPUMA vs. OpenFOAM

upstream CFD

Solver (hardware)	Hardware	Nodes	Wall-clock time*	Costs (AWS)*
OpenFOAM (CPU)	**912x** Intel Xeon Platinum 8368 cores (76/node)	12	44.1 h	$1497
OpenFOAM (CPU)	**3800x** Intel Xeon Platinum 8368 cores (76/node)	50	11.8 h	$1668
SPUMA (GPU)	**48x** NVIDIA A100-40GB GPUs (4/node, 2.5M cells/GPU)	12	11.5 h	$647

Node-to-node: same 12 nodes

"If I equip my CPU node with GPUs" (rows 1 and 3)

3.8x faster

2.3x cheaper

1.3x less energy

OEM production scale: same turnaround

3800 CPU cores vs. 48 GPUs (rows 2 and 3)

~12 h same turnaround

2.6x cheaper

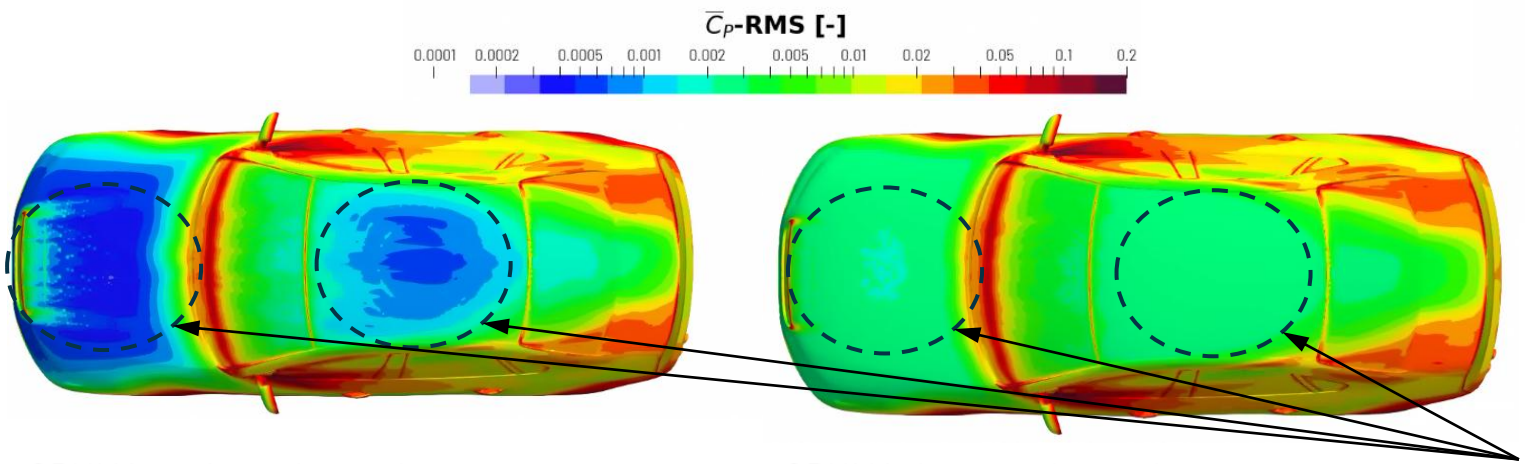
4x fewer nodes (12 GPU nodes vs. 50 CPU nodes)

At 2.5M cells/GPU the GPUs run below peak efficiency, traded for turnaround

*Wall-clock time and costs for 2.1 s simulated time (≈29 CTU, median of the meancalc runtime-control runs). Costs from public cloud HPC prices on AWS, energy from TDP.

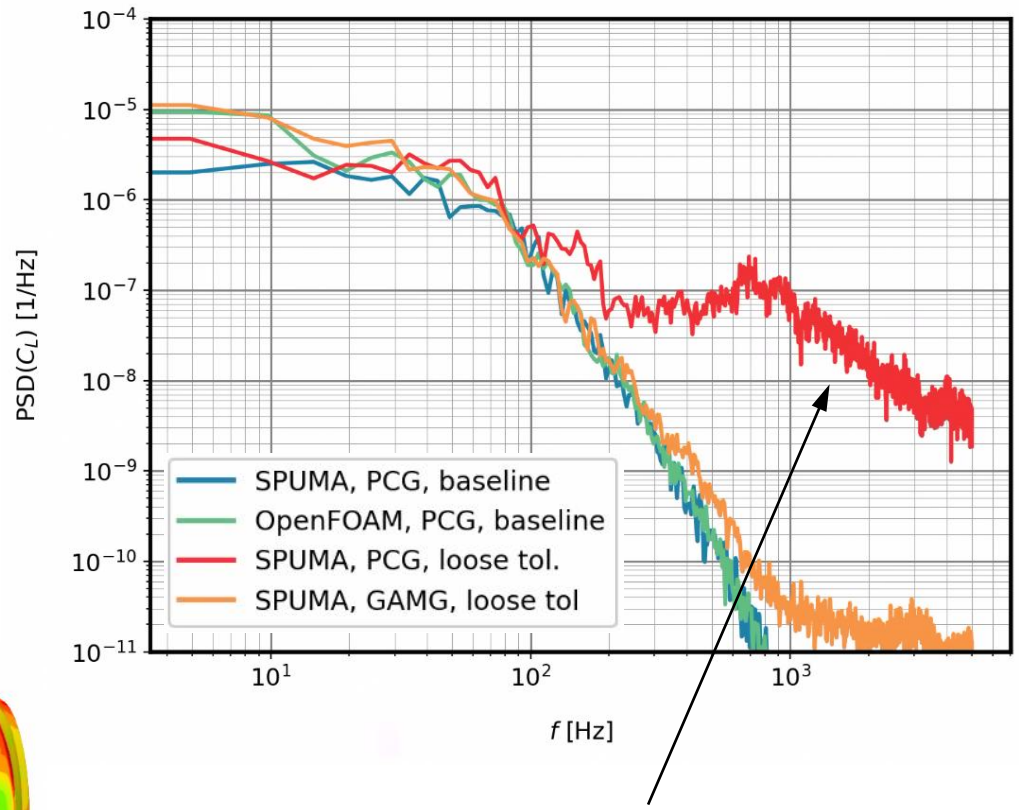
Tuning of solver settings: Performance vs. simulation quality

Solver	Lin. pressure solver	Lin. solver tolerances	Wall-clock time*	Costs (AWS)*
SPUMA (baseline)	Conjugate gradient (PCG)	strict	11.5 h	$647
SPUMA	Conjugate gradient (PCG)	loose	8.6 h	$481
SPUMA	Multigrid (GAMG)	loose	6.1 h	$341



SPUMA, strict solver tolerances

SPUMA, loose solver tolerances



spurious high-frequency oscillations

spurious pressure fluctuations in (quiet) RANS flow regions

Conclusions

- SPUMA GPU code produces same results as OpenFOAM CPU code (within statistical uncertainty)

- What is the performance benefit of running SPUMA:
  - Node-to-node comparison: 3.8x faster, 2.3x cheaper, 1.3x less energy
  - OEM scale: Same time-to-solution, but 2.6x cheaper on 4x fewer nodes

- Best practices for GPU runs:
  - Fill GPU memory as much as possible (less GPU cards), add only GPUs to reduce turnaround time
  - Multigrid solver seems to be more efficient than CG and more effectively damps solution oscillations
  - Tuning of solver settings to gain performance should be done with care → always checks for RMS values in addition to means

- SPUMA is the only GPU-capable OpenFOAM derivative today which is (close to) production ready for automotive applications:
  - Mesh-to-mesh interface for simulating rotating wheels is currently under development

upstreamCFD

This work has been co-funded by the EXASIM* and BUZZKATZ** projects

*The EXASIM project was funded by:

Federal Ministry of Education and Research

Funded by the European Union
NextGenerationEU

**The BUZZKATZ project is funded by:

Supported by:
Federal Ministry for Economic Affairs and Energy
on the basis of a decision by the German Bundestag

We gratefully acknowledge the computing resources provided by the HoreKa supercomputer at KIT.

HoreKa