FIRST ASSESSMENTS REGARDING THE USE OF SPLISS IN VARIOUS CFD-RELATED APPLICATIONS

September 25th, 2023

HPCN Workshop Göttingen

Arne Rempke[1], Olaf Krzikalla[1], Jan Backhaus[2], Alexander Bleh[2], Jasmin Mohnke[1], Johannes Wendler[1], Michael Wagner[1], Marco Cristofaro[1], Ralf Hartmann[3], Wojciech Laskowski[3]
[1]Institute of Software Methods for Product Virtualization, [2]Institute of Propulsion Technology, [3]Institute of Aerodynamics and Flow Technology,
German Aerospace Center (DLR)

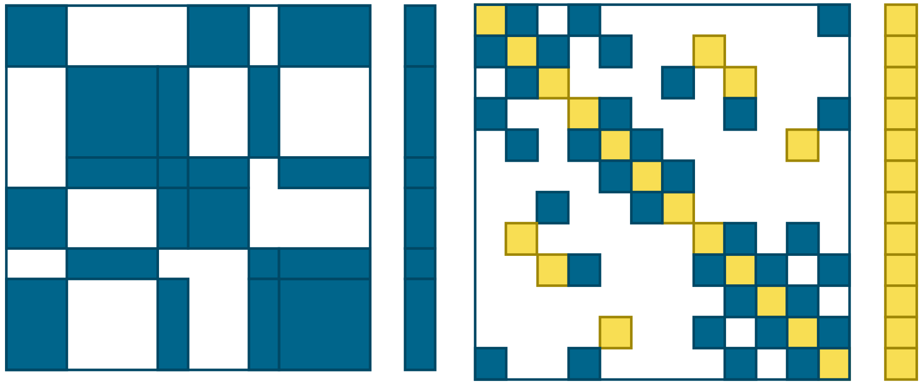
DLR

Intro: CFD and related simulations at DLR

- High Fidelity Computational Fluid Dynamics requires the use of **large computational resources in parallel**
- At least implicit methods require to (approximately) **solve large linear equation systems**
- There are different CFD codes (even within DLR) for different flow regimes, which can benefit from a shared development:

- Common library for (approximatively) solving a linear equation system with characteristics from aeronautical CFD
  - More focus on low-level performance and hardware technologies
  - May adapt to specific technologies more easily due to its comparably limited functional range

Key features of a linear solver for aeronautical CFD

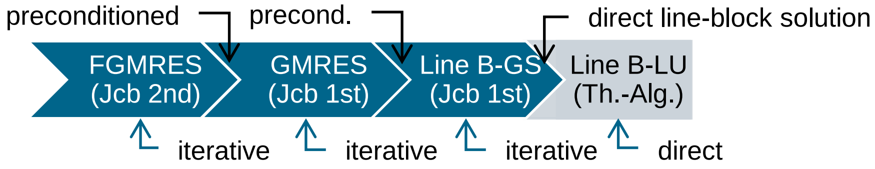


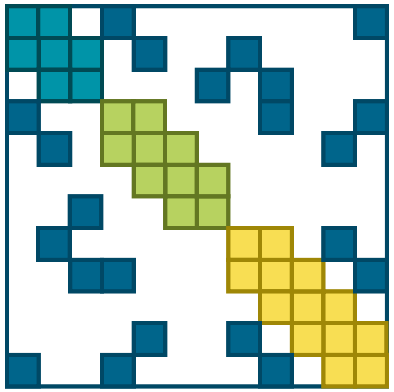
Sparse block matrices

- Dense blocks with a fixed block size or variable block sizes
- Mixed data types: e.g. some entries are complex, others real, some multiscalars

Solver

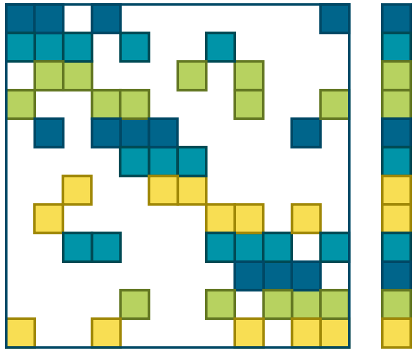
- Different components should be combinable (as preconditioner)
- Robust methods for stiff CFD problems:
  - Direct inversion of (generalized) diagonal blocks (LU/Thomas-Algorithm)
  - Jacobi, Gauss-Seidel, GMRES, linear multigrid, …





Efficient parallelization for HPC

- Distributed memory (e.g. MPI)
- Shared memory (Threading)
- GPU support
- Vector instructions (SIMD)
- Reduced memory footprint



ALGORITHMICAL FEATURES

Coloring & permutation

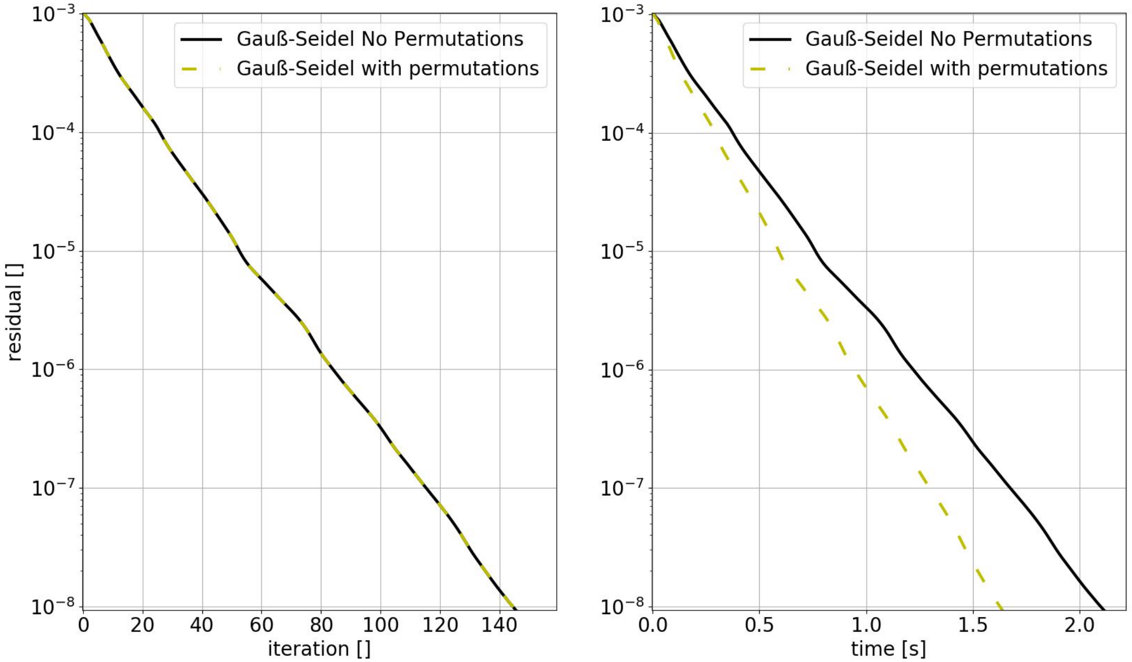
Coloring

- Used for (otherwise sequential) solver components like Gauss-Seidel or ILU
- Allows **parallelism** while keeping results **independent from partitioning**

Permutation

- For cache optimization, it can be beneficial to permute the matrix entries

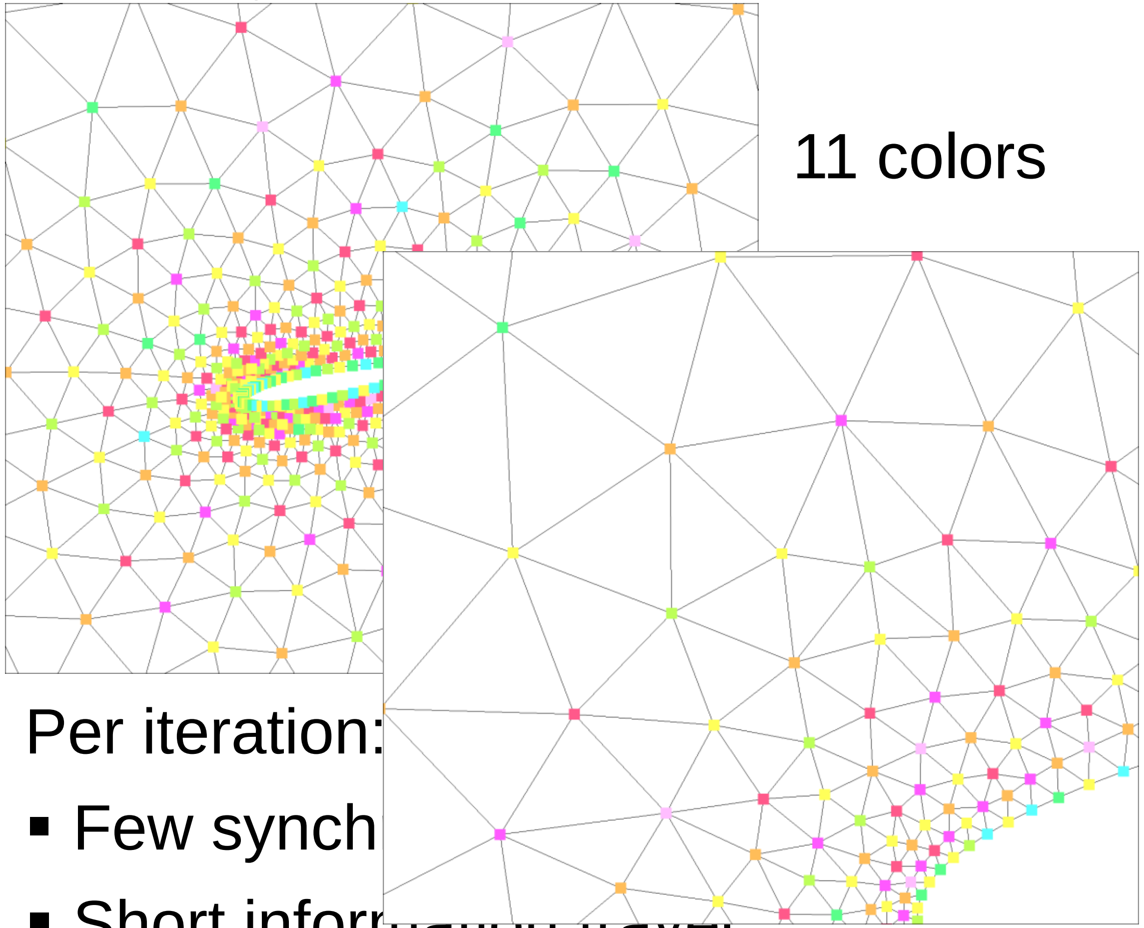
➔ Use a permutation according to colors, when multi-colored algorithms are used



TRACE matrix, results from Alexander Bleh

Comparison of different color selections for multi-color algorithms

- Greedy minimal number of colors

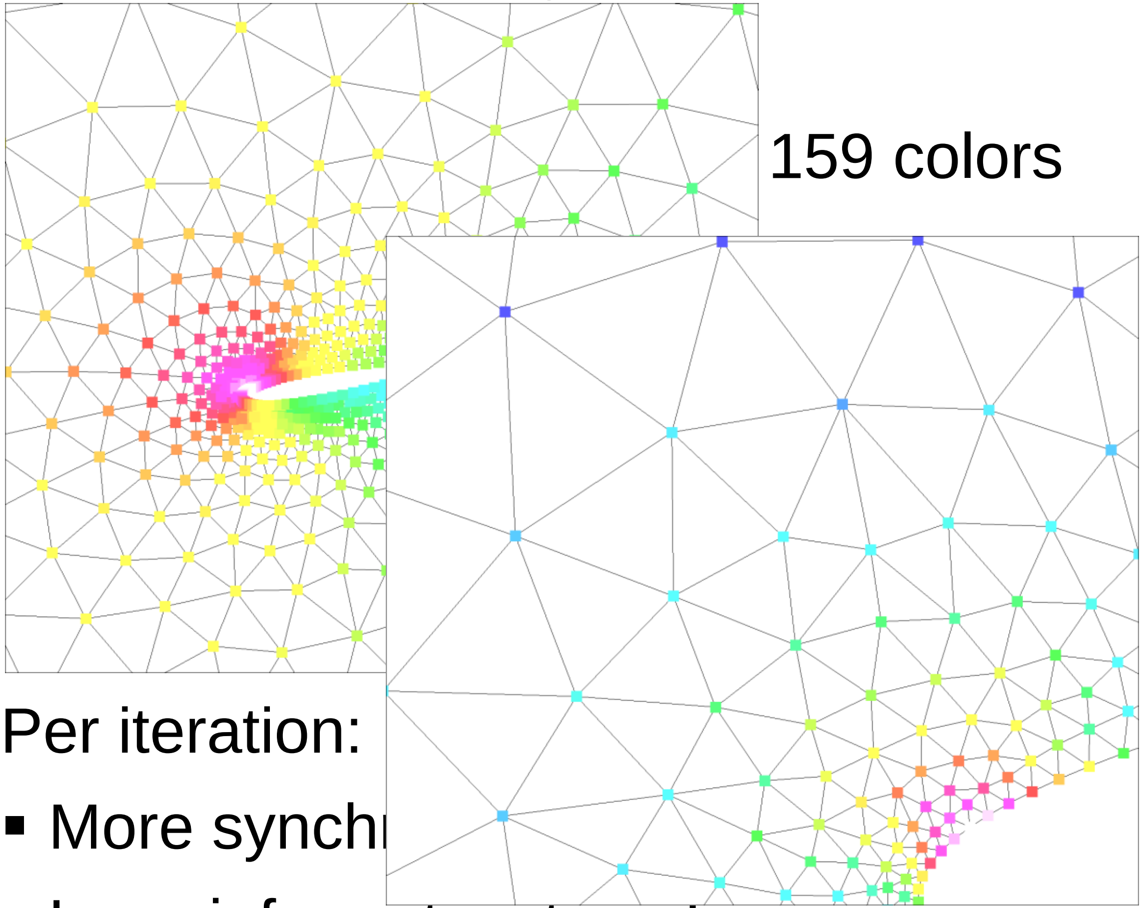


11 colors

Per iteration:

- Few synch
- Short information travel

- Reverse marching front

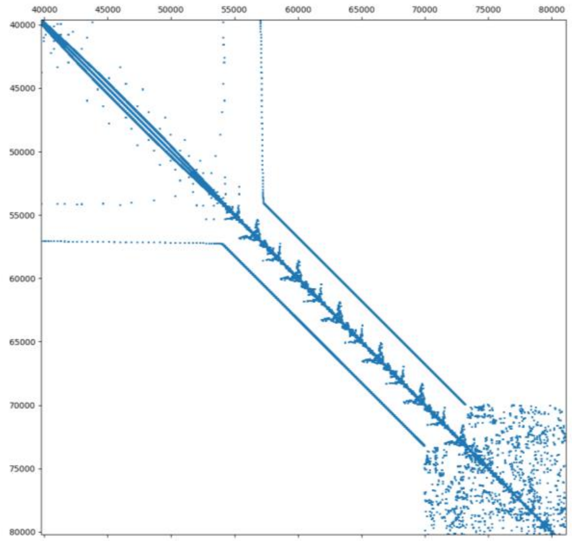


159 colors

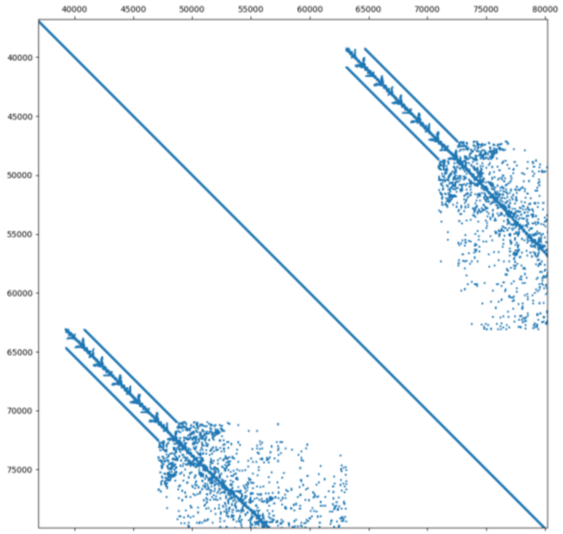
Per iteration:

- More synch
- Long information travel

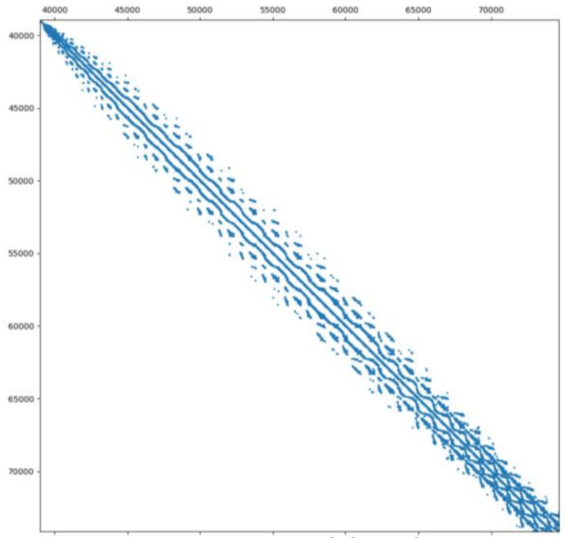
TRACE: Different colors for symmetrical Gauss-Seidel



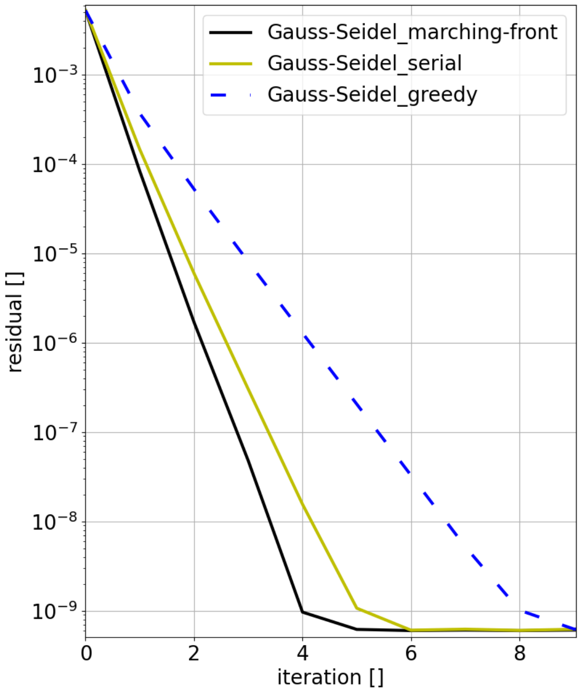
Serial

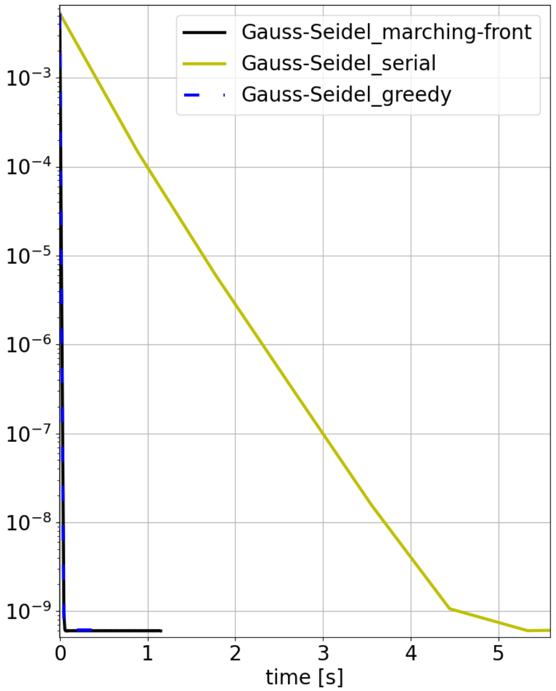


Greedy



Reverse marching front



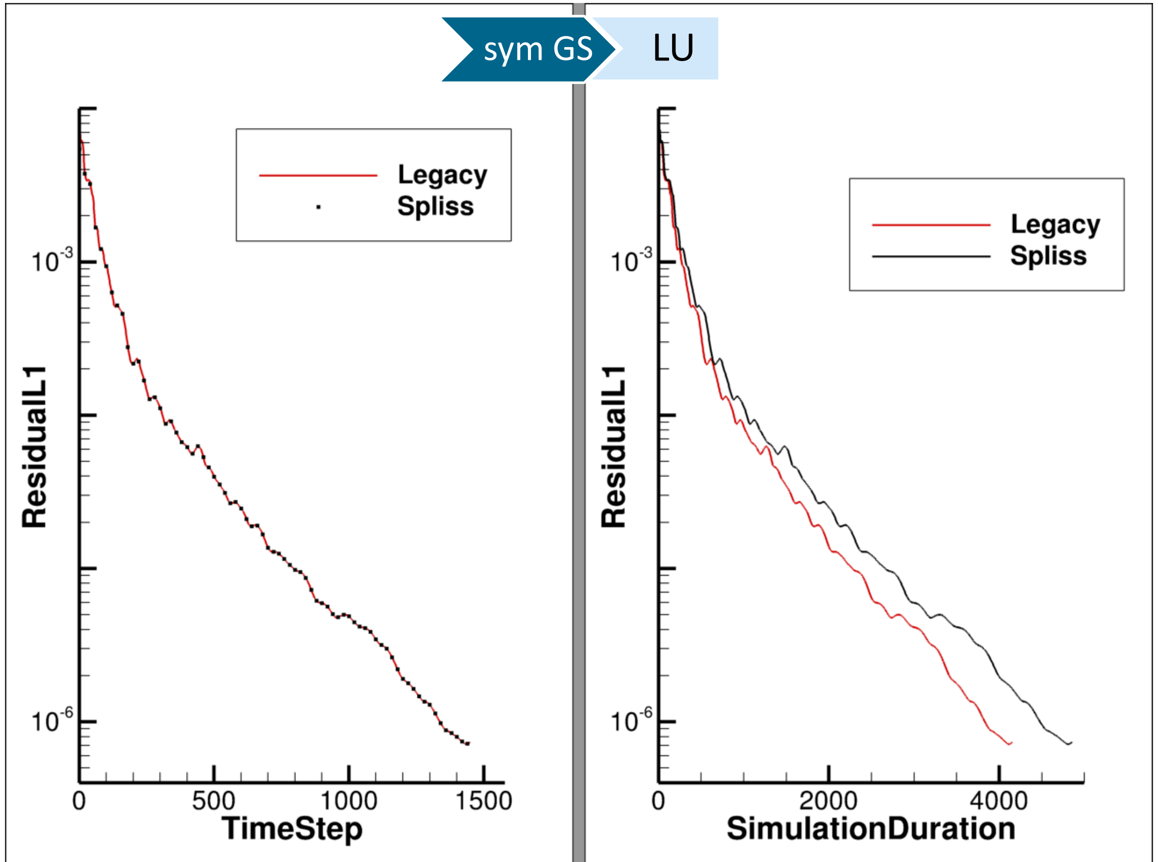


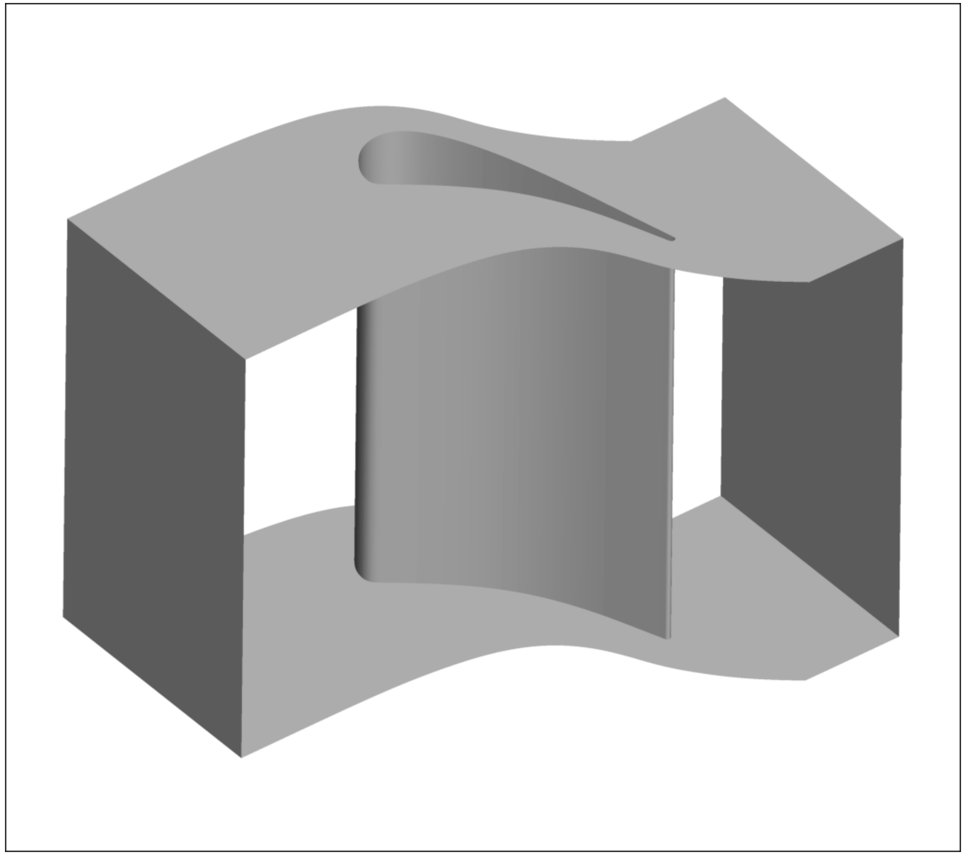
- TRACE case of transsonic bump
- hybrid unstructured/structured mesh
- 13e3 hexahedrons, prisms & tetraeder
- Euler 2nd order FV
- „Serial“ implementation uses a huge amount of colors

Results from Jan Backhaus

TRACE VKI-LS89 Comparison to Legacy

- VKI-LS89 single blade configuration
- RANS-kω, Ma=0.92, Re=2.1e6
- Grid with 2e6 elements, FV scheme





- Focus on **replicating the algorithm** from Legacy to get equal results
- Currently Spliss is 10% slower to TRACE Legacy

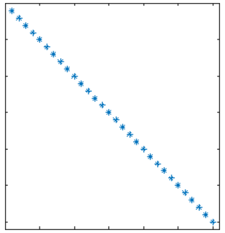
Computation & results from Alexander Bleh & Jan Backhaus

CFD SPECIFIC ALGORITHMS

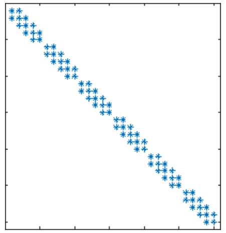
Lines Inversion / Thomas Algorithm

- Jacobi-method uses a diagonal inversion:

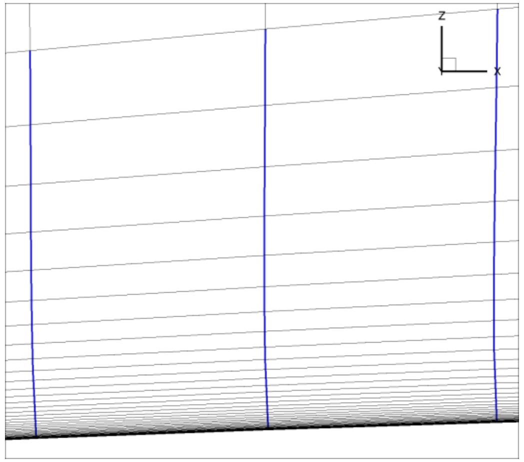
$$x^{(i+1)} := x^{(i)} + D^{-1}\left(b - Ax^{(i)}\right)$$



where



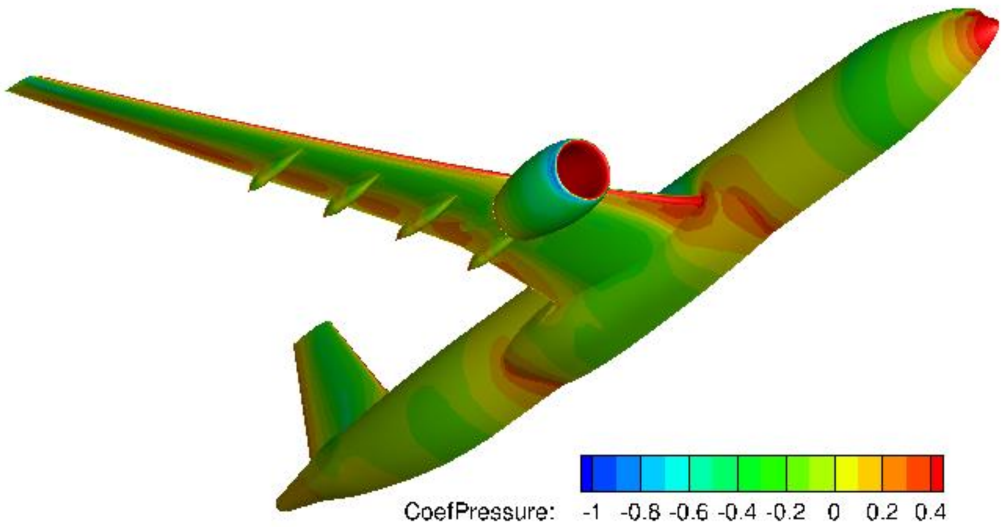
- $D := \mathrm{diag}(A)$ (point-implicit) or
- $D := \mathrm{tridiag}(A)$ (lines-implicit)

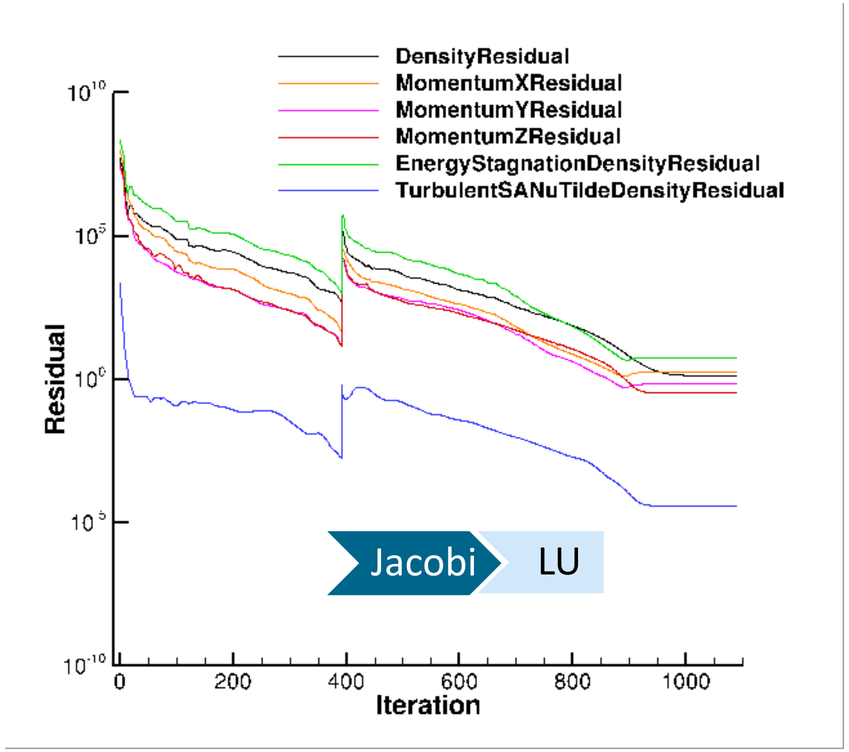


- Especially favourable/needed when mesh has very anisotropic cells, aspect ratios ≥5000:1

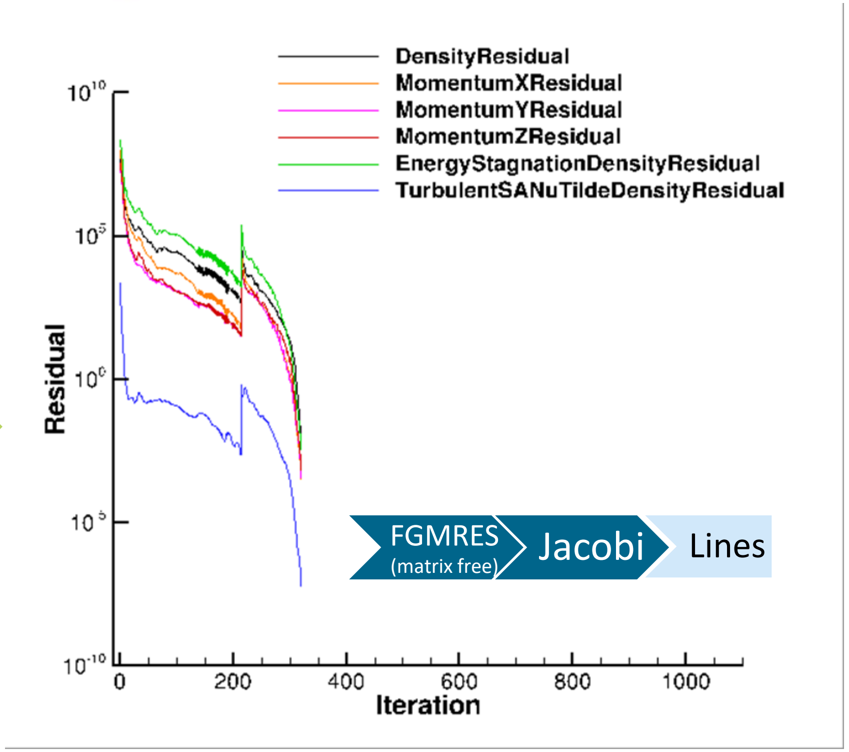
CODA XRF1-V4 FV

- Airbus XRF1-V4 flow-through-nacelle (power off) configuration
- RANS-SAneg, Ma=0.86, Re=25e6
- Grid with 32e6 elements, FV scheme



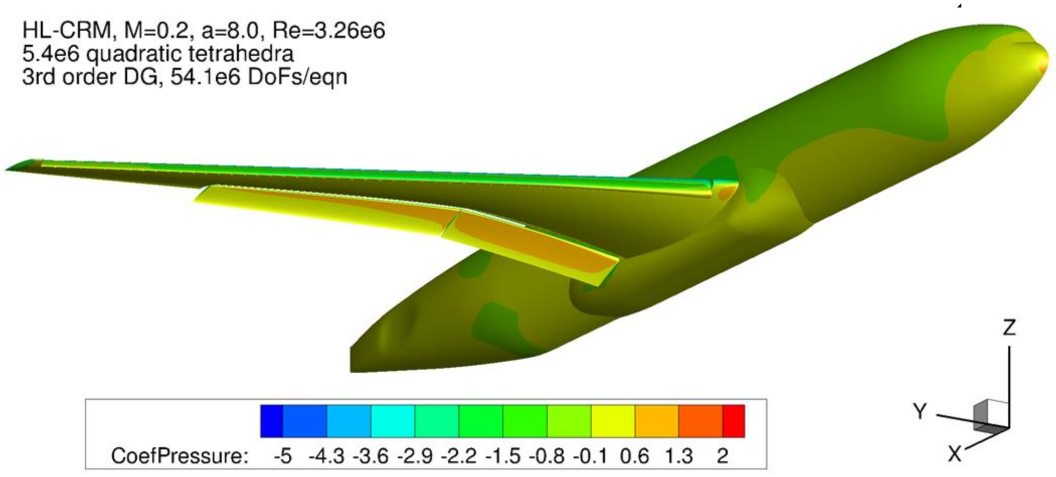


Reduction of time to solution by **73%** on 256 cores

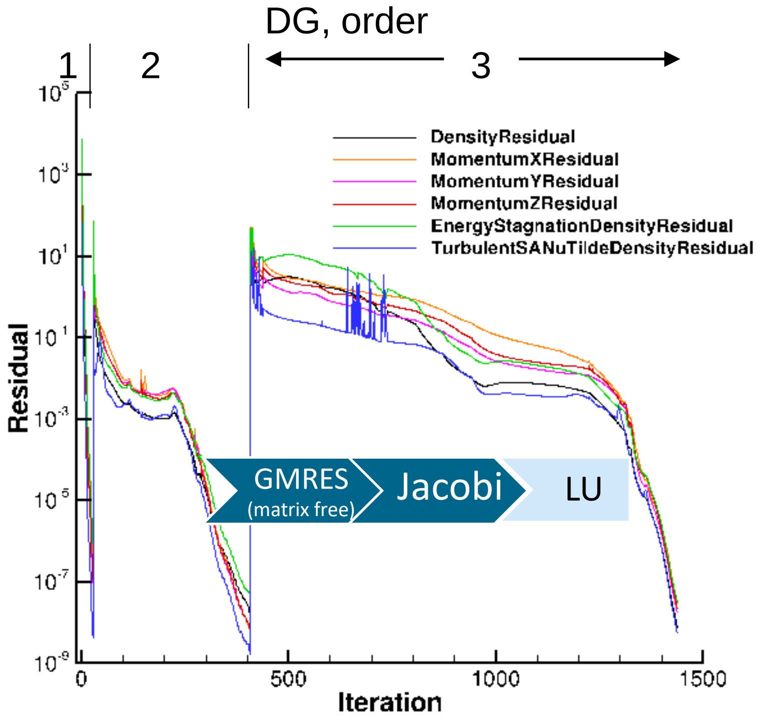


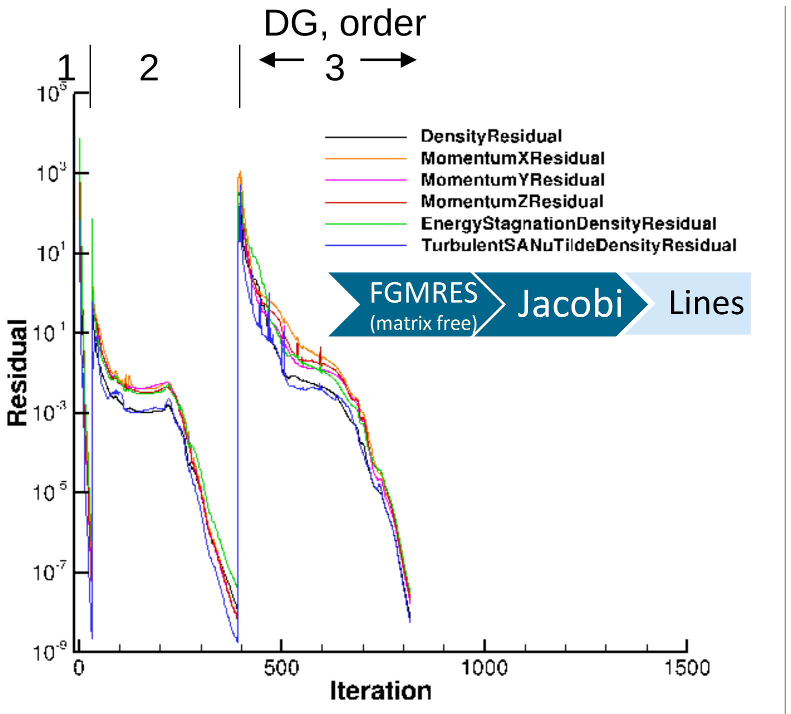
Computation & results from Ralf Hartmann

CODA High-Lift CRM DG



- Geometry from High Lift Prediction Workshop 3
- RANS-SAneg, Ma=0.2, Re=3.26e6
- Curved Grid with 5.4e6 quadratic tetrahedra, DG scheme (3rd order: 54.1e6 DoFs/eqn)

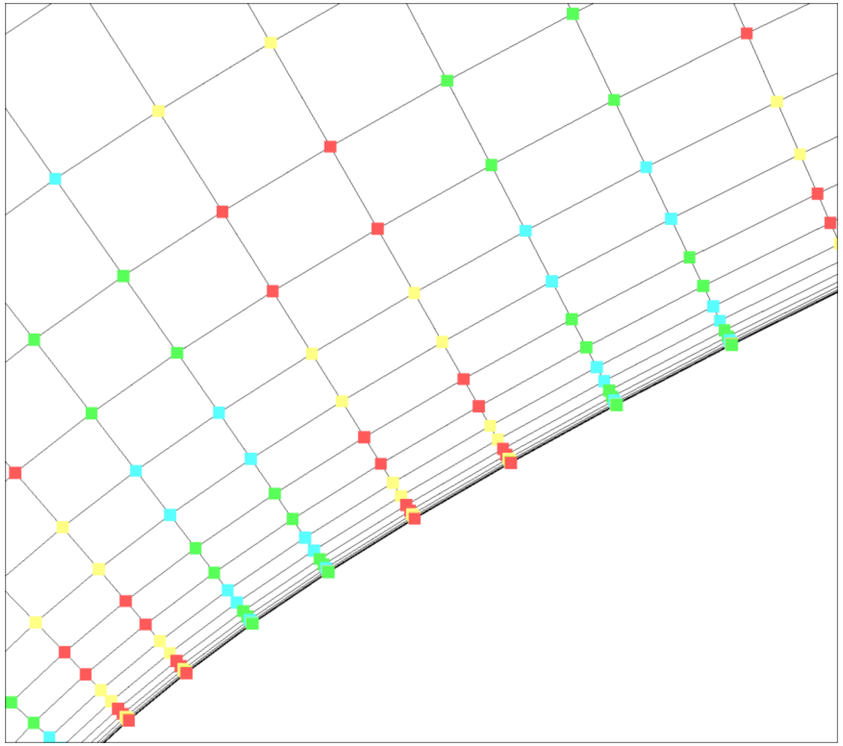


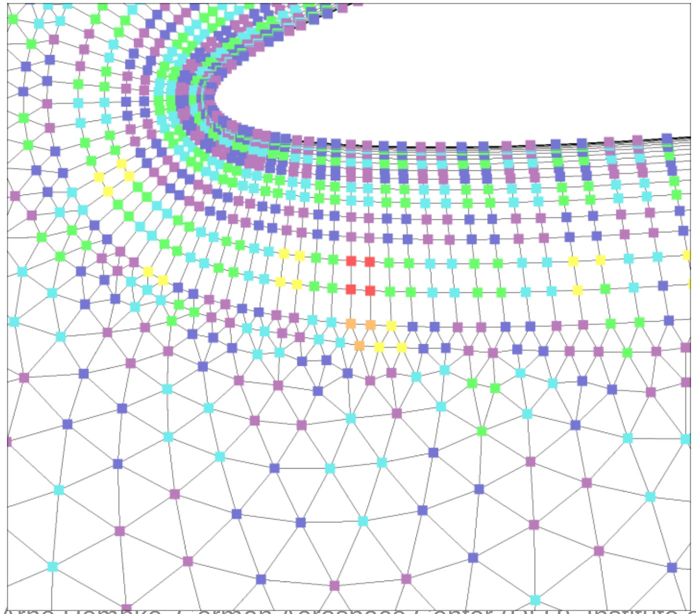


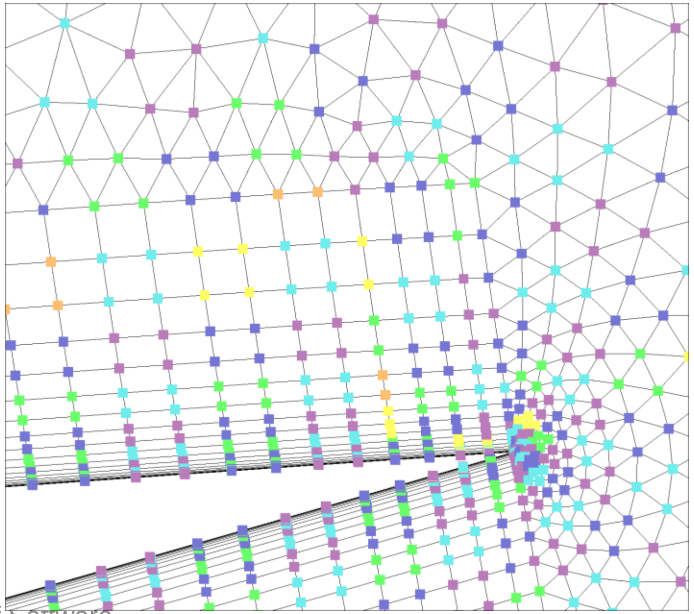
Computation & results from Ralf Hartmann

Algebraic agglomerations visualized

- Agglomerations are computed simply by inspecting the matrix connectivity, not the values
- When the matrix blocks correspond to geometrical elements/vertices, the agglomerates can be visualized in the original mesh



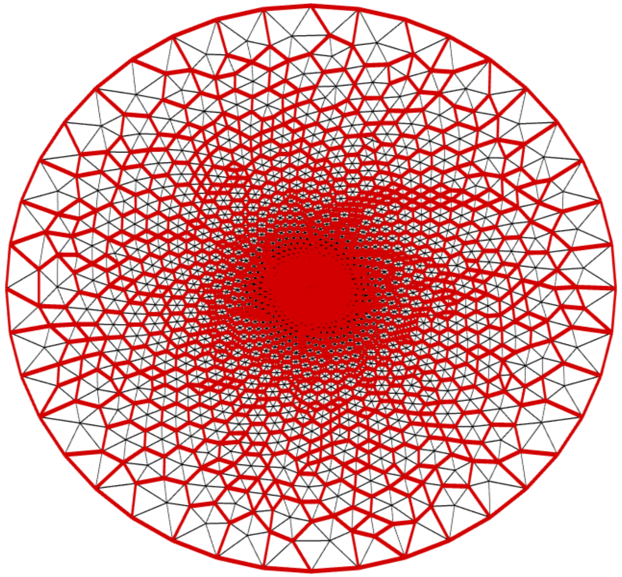


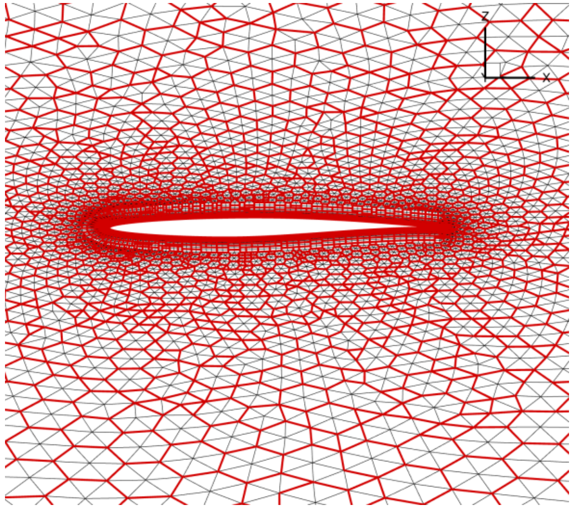


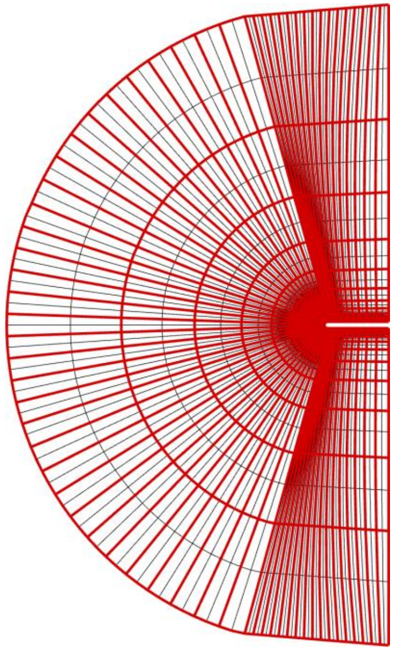
- First level agglomerates for a vertex-based discretization

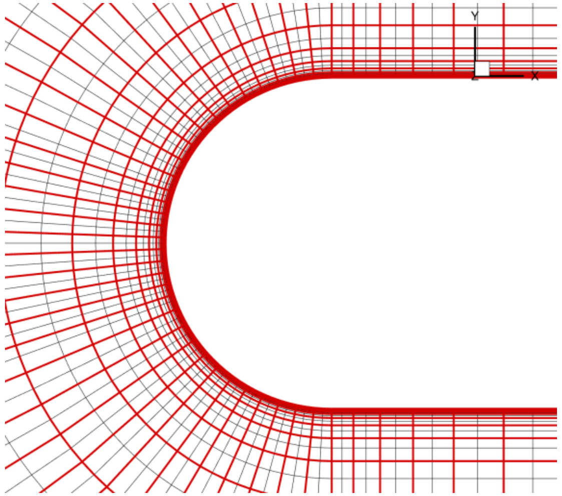
Algebraic agglomerations visualized

- First level agglomerates for a volume-based discretization (CODA FV)





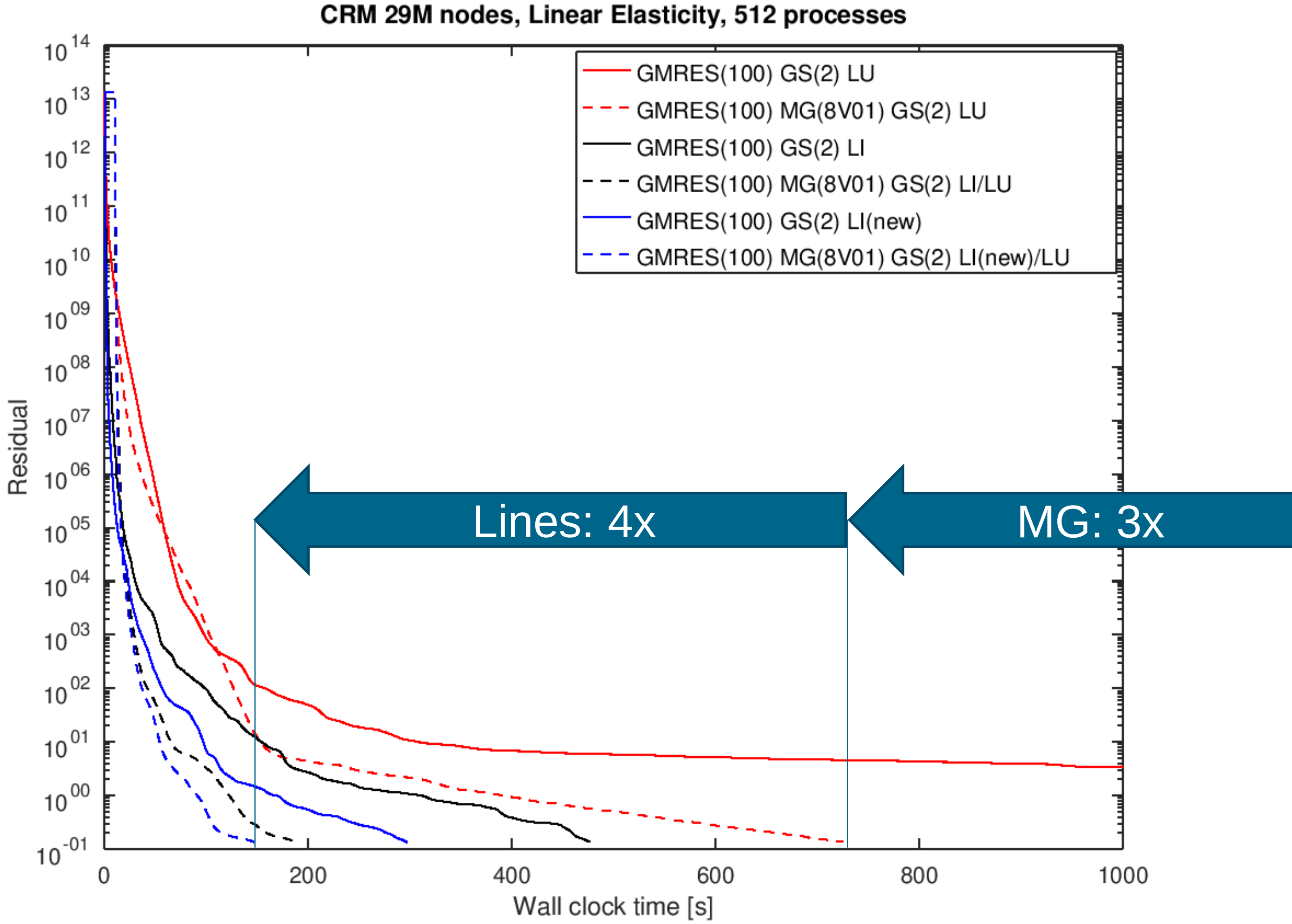




CODA integration and images by Wojciech Laskowski

FSMeshDeformation: Efficiency of the tailored solver components

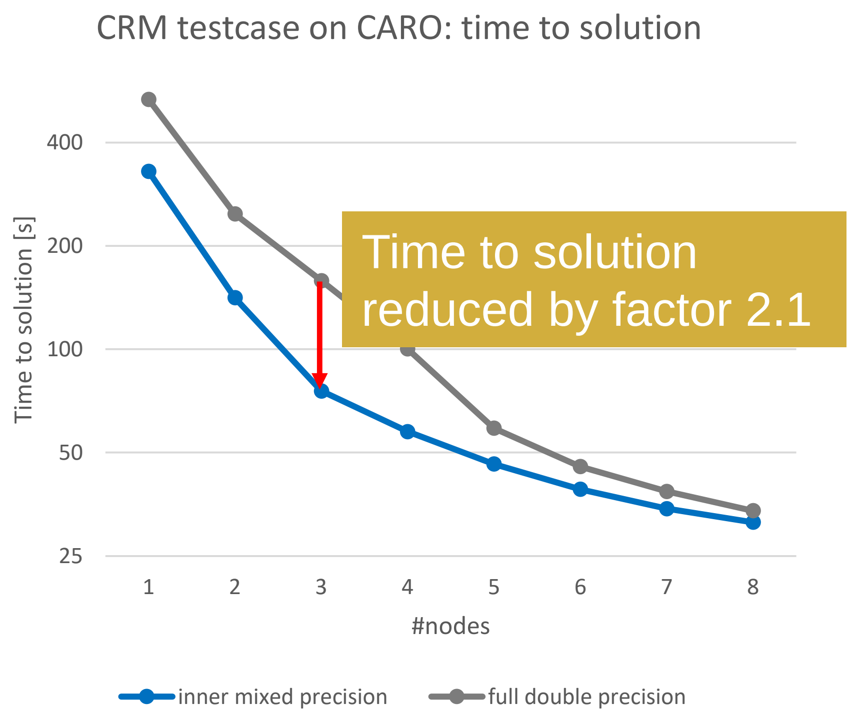
- Red solid curve is a „standard linear solver“
- Multigrid gives speedup of 2-3 (dashed)
- LinesInversion gives additional speedup of 3-4 (black/blue)

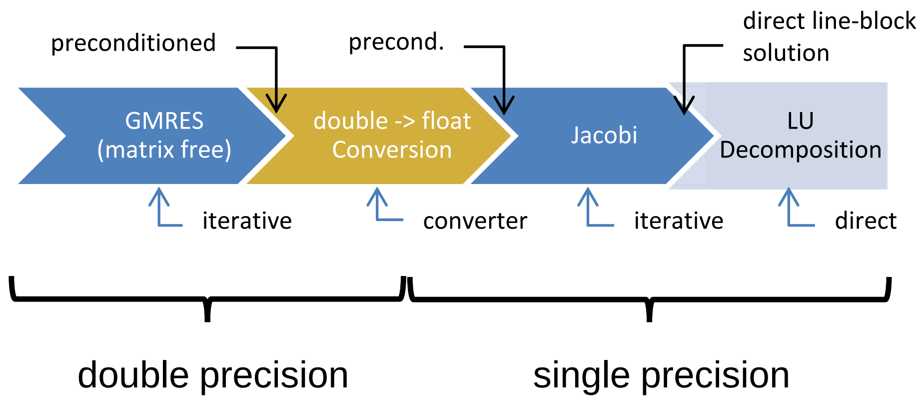


PERFORMANCE, SCALING & ACCELERATORS

Mixed precision

- Idea: Reduce memory footprint of inner hot loops since performance is memory bound

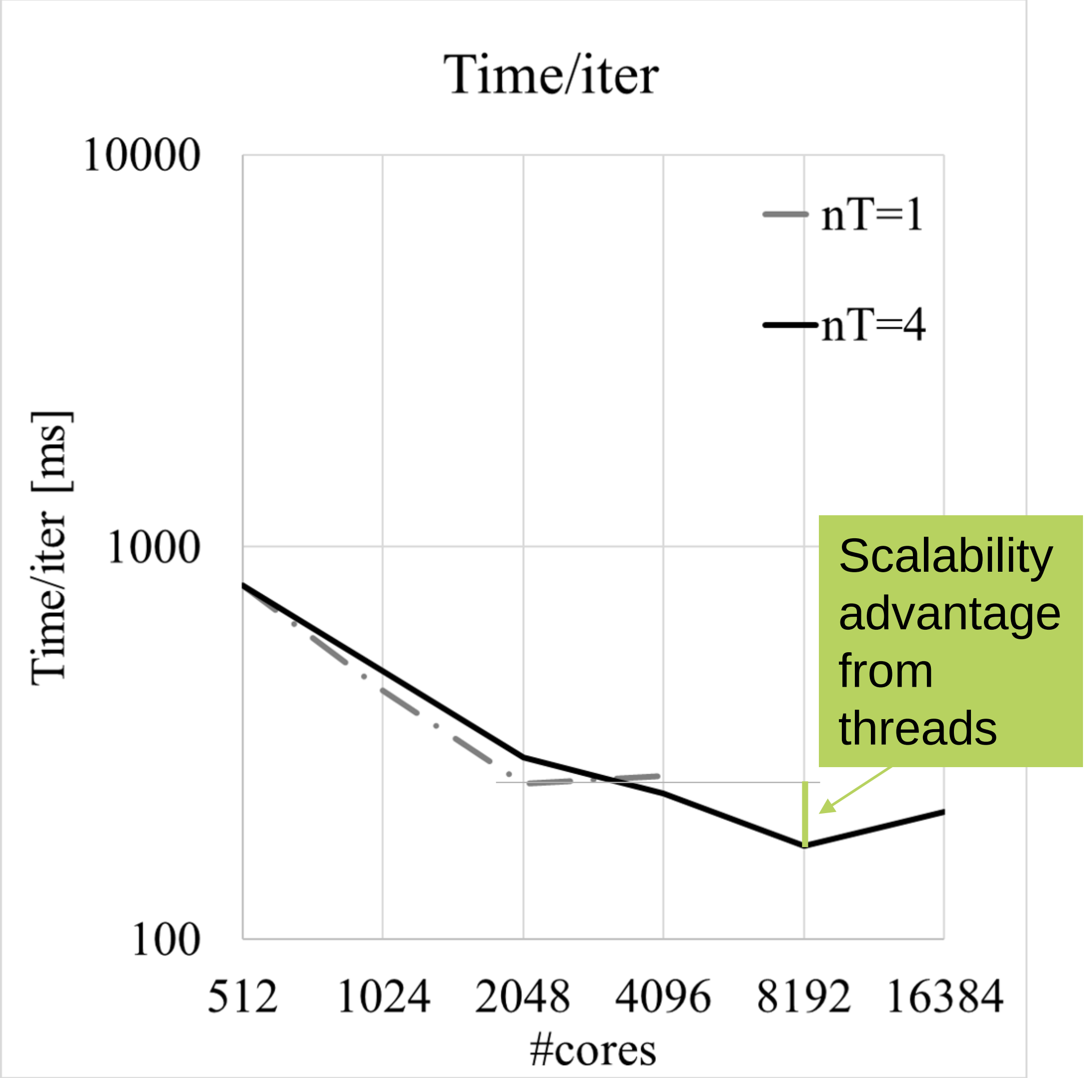




- User still provides matrix / input vectors and receives solution vector in double precision
- Inner Spliss solver components operate in float precision

FSMeshDeformation: Strong scaling and multi-threading

- XRF1 test case, 31M nodes
- Linear elasticity mesh deformation
- CARO: 2xAMD EPYC 7702 («Rome», 64 cores, 2,0 GHz)
- GMRES Multigrid GaussSeidel configuration
- When using more than 2048 ranks, scaling becomes difficult (very much communication during solving, initialization/partitioner takes very long)
  - But: 2048 ranks can still employ more cores when using threads



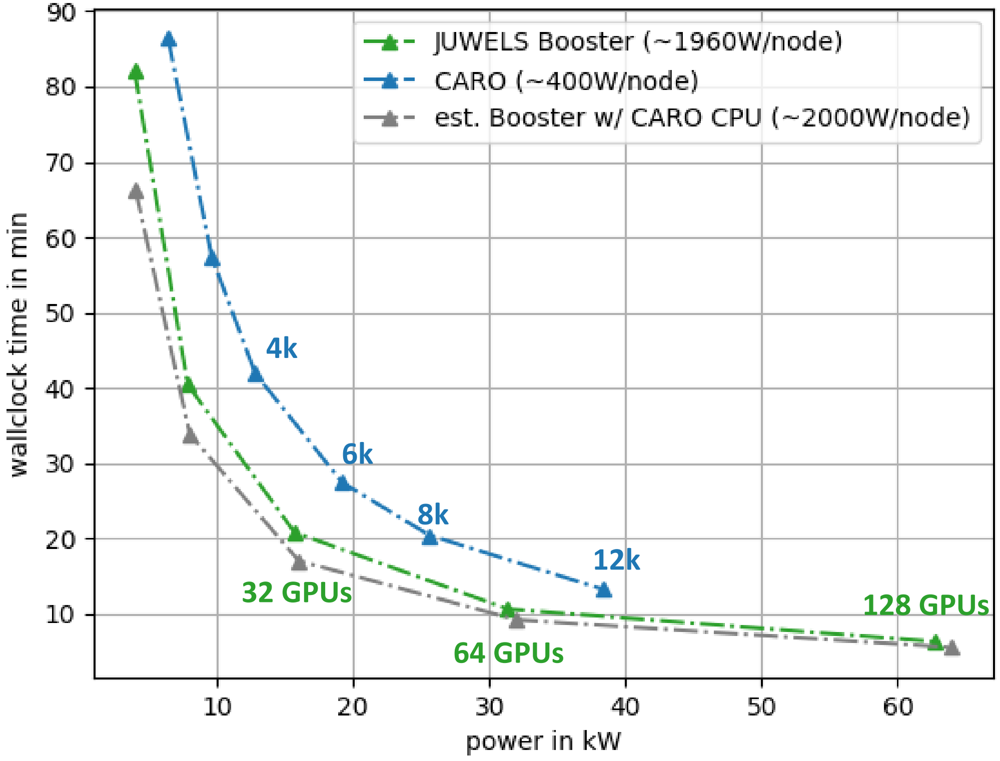
Thanks to Marco Cristofaro for results

GPU Development
Next gen GPUs

Juwels Booster (Jülich)

- 4x Nvidia Tesla A100 per node
- Time to solution: speedup of 8-9 for same number of nodes on Juwels
  - Rather unfair, since on Juwels every process uses a GPU **in addition** to the CPU
- Energy comparison (seconds per used Watt): speedup of 1.6-1.9 on Juwels
- Hypothetical Juwels Booster node with CARO CPU: 1.8-2.3 speedup (energy-wise)

Runtime | CARO (AMD Rome) vs. Juwels (4x Nvidia A100)
M6 wing, 69.2M elements, implicit Euler, Jacobi + Block Inv.



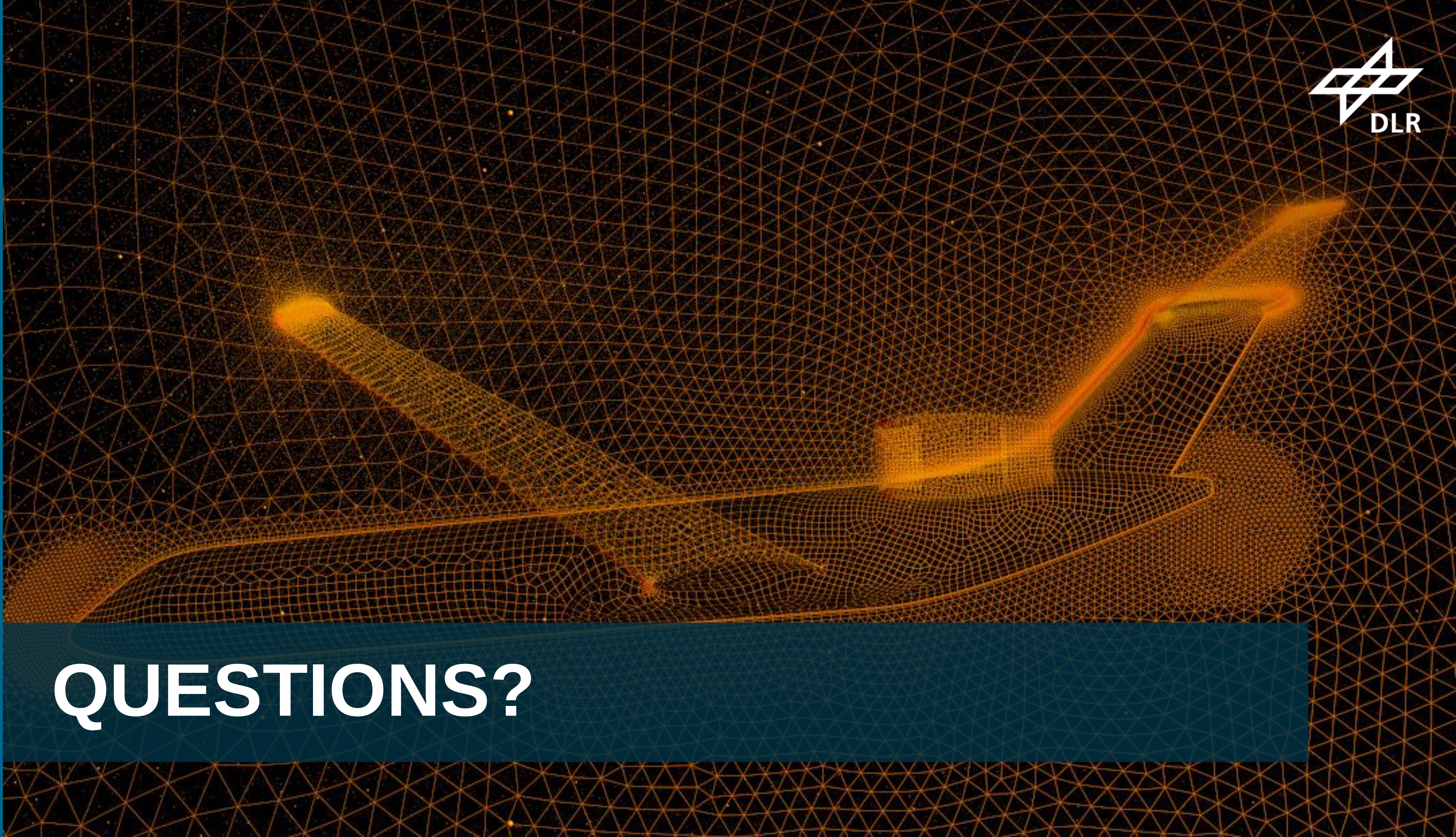
Results from Michael Wagner & Jasmin Mohnke

Conclusion

- Spliss in use in CODA, TRACE, HYDRA, FSMeshDeformation

- Demanded features are supported and successfully demonstrated

- Regular exchange with users in order to still develop further and improve



QUESTIONS?