SPLISS – LINEAR ALGEBRA FOR IMPLICIT CFD METHODS

A Sparse Linear System Solver for Transparent Integration of Emerging HPC Technologies into CFD Solvers

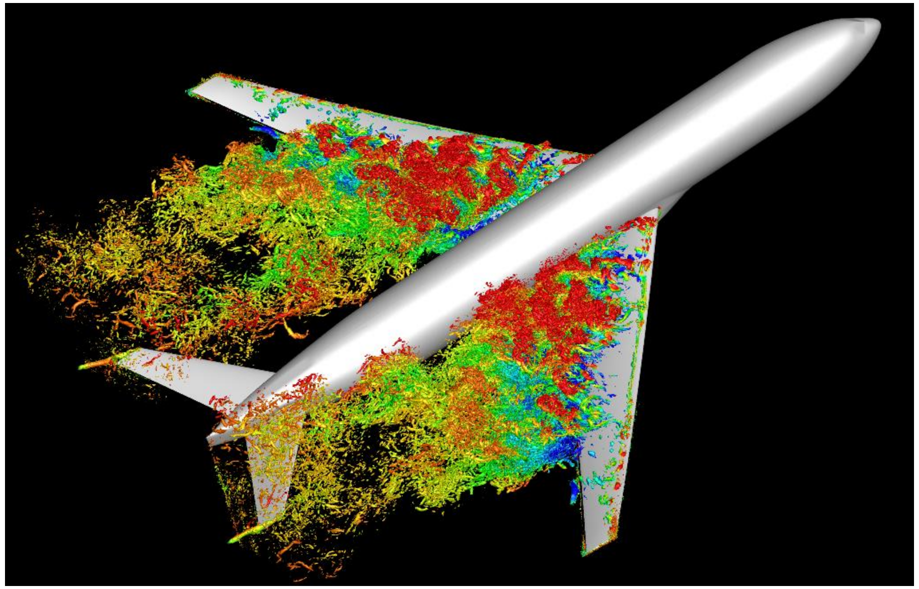
October 15th, 2025, 40th WSSP, Tohoku University, Sendai, Japan

Arne Rempke, Olaf Krzikalla, Jasmin Mohnke, Johannes Wendler, Michael Wagner, Marco Cristofaro
Institute of Software Methods for Product Virtualization, High Performance Computing, German Aerospace Center (DLR)

WHAT'S THE CONTEXT

What we (DLR aerospace) do

- Wind tunnel experiments
- Flight tests
- …
- Computational Fluid Dynamics
  - Numerically solving nonlinear partial differential equations
  - For implicit schemes the most expensive part is solving linear equation systems
  - Industrial relevant cases require efficient use of HPC (turbulence is difficult)



Our challenges/chances:

- Try to make use of current (and be ready for future) hardware technology, but codes are often complex, large, calibrated to physical measurements and quality assured, so it is not so easy to adopt fast
- Due to recent changes in hardware technology (Many-core, SIMD, GPU, …), we have worked on new implementations

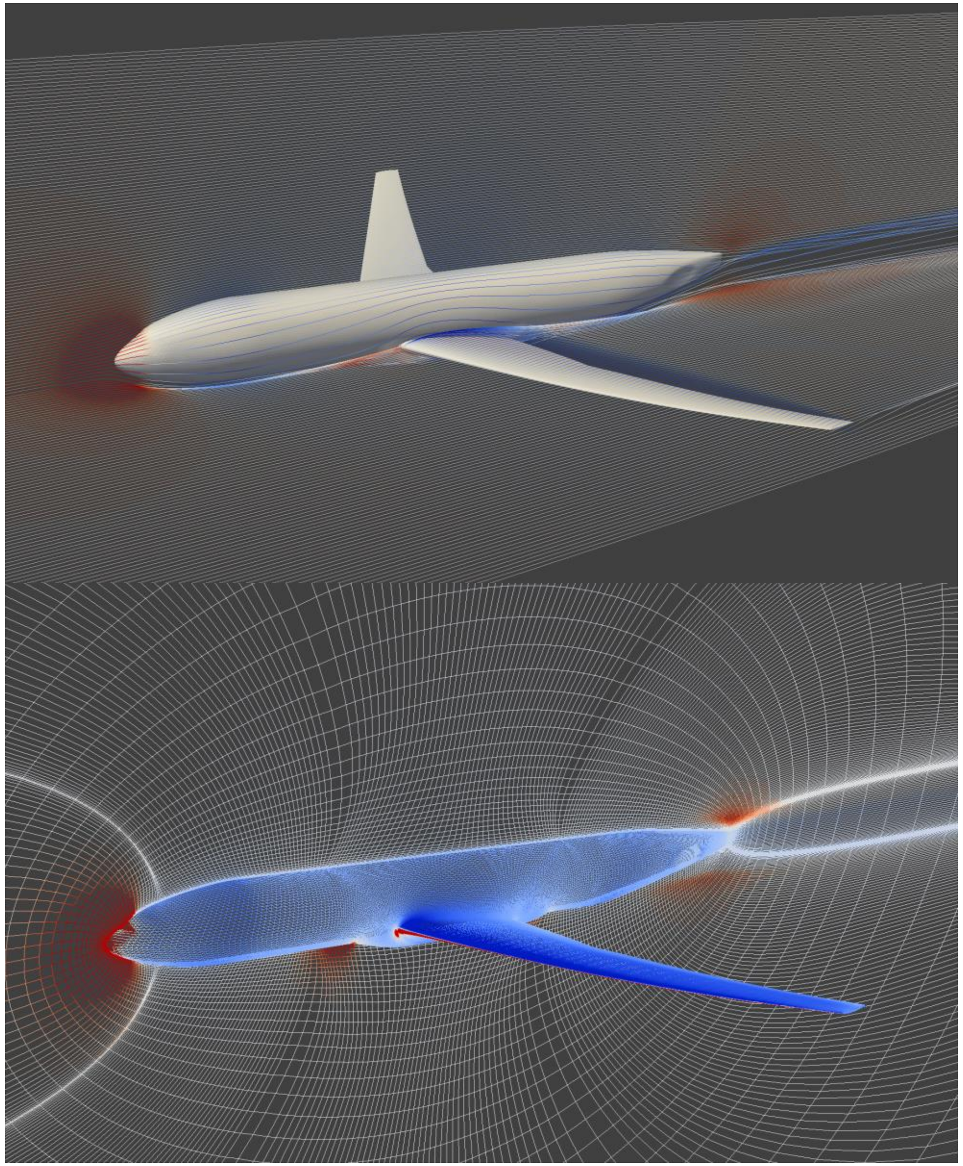
Software Approach to tackle these challenges

- Different CFD solvers for specific flow characteristics
  - TRACE for turbomachinery
  - CODA for aerodynamics
  - …
  - Contain physical modeling, handling of boundary conditions, nonlinear relations, wind-tunnel calibration, transsonic/hypersonic/… flow regime, …

- 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

SPLISS OVERVIEW

Key features of a linear solver for aeronautical CFD



Sparse 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, …

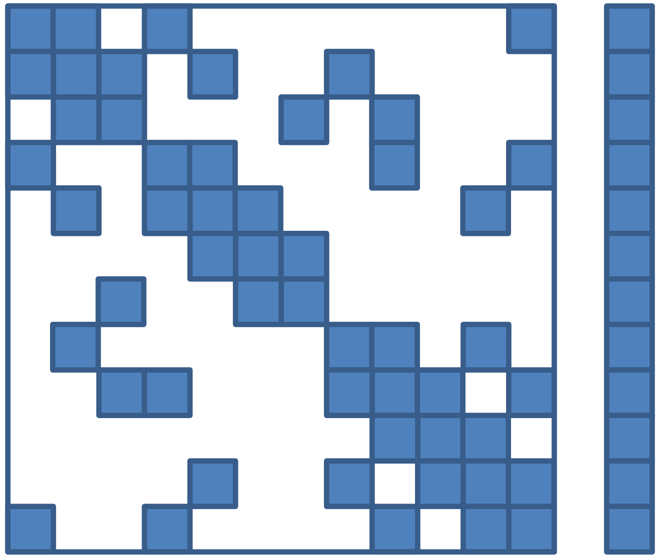
Efficient parallelization for HPC

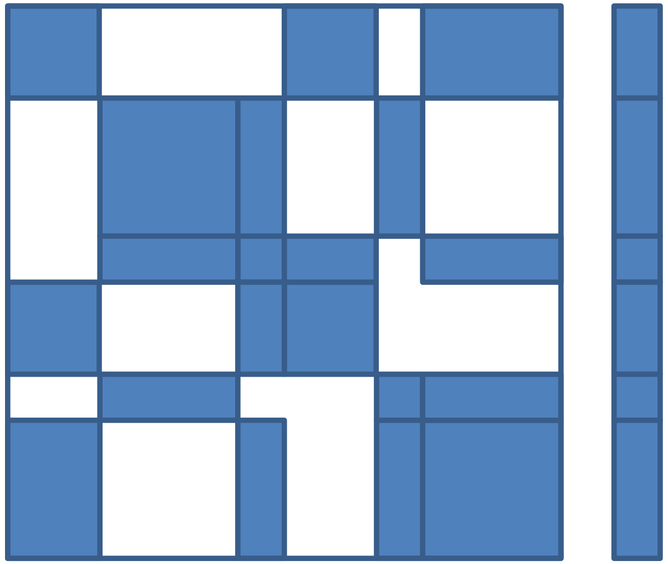
- Distributed memory (GASPI, MPI)
- Shared memory (Threading)
- GPU support
- Vector instructions (SIMD)

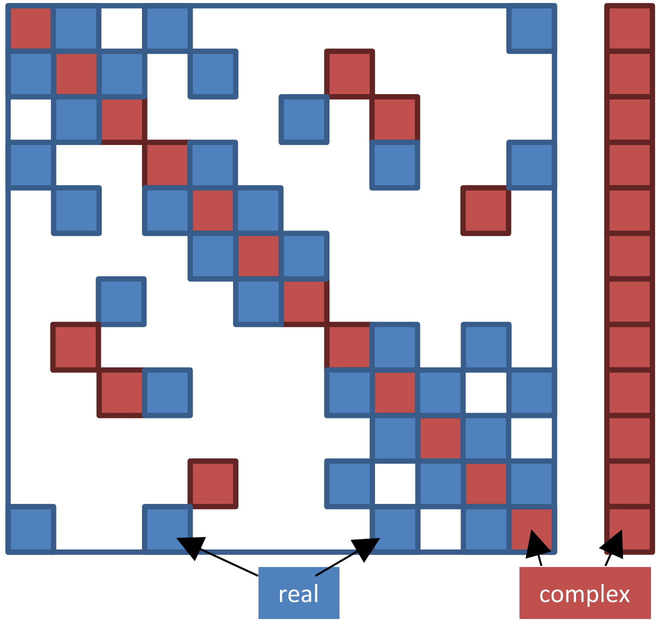
Matrix Structure

Sparse matrices with dense blocks

- Blocks of fixed size (e.g. 5x5, 7x7, 12x12 for all blocks within a single sparse matrix) (finite-volume Euler or RANS method)
- Blocks of variable sizes within one sparse matrix (e.g. 12x12, 48x48, 120x120 and 240x240 in one sparse matrix) (mixed-order Discontinuous-Galerkin method)
- Mixed data types: e.g. some entries are **complex**, others **real** (time-spectral/harmonic balance method)



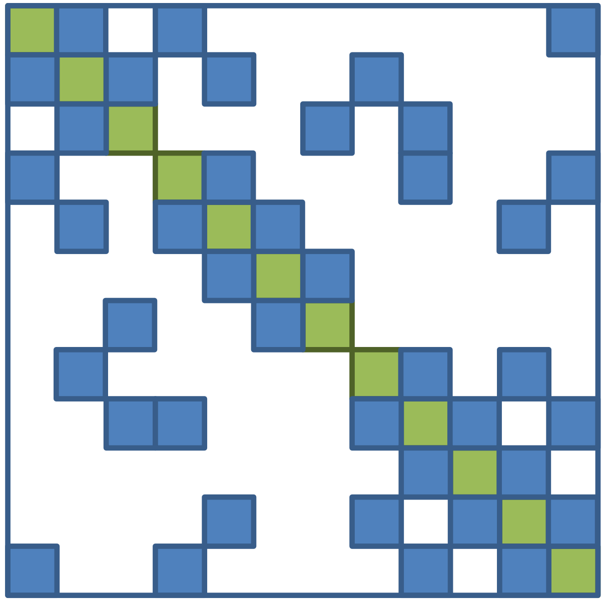


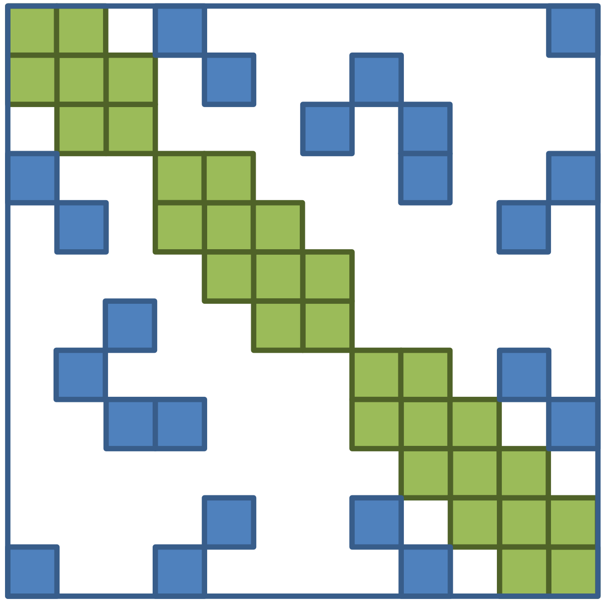


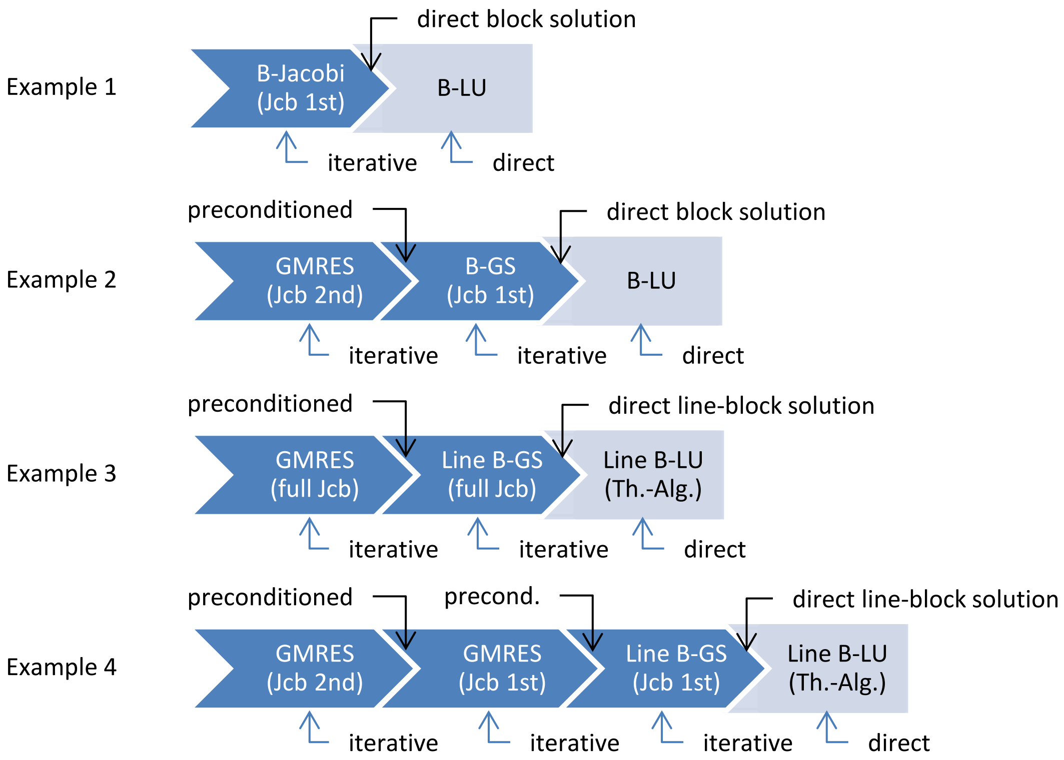
Solver Structure

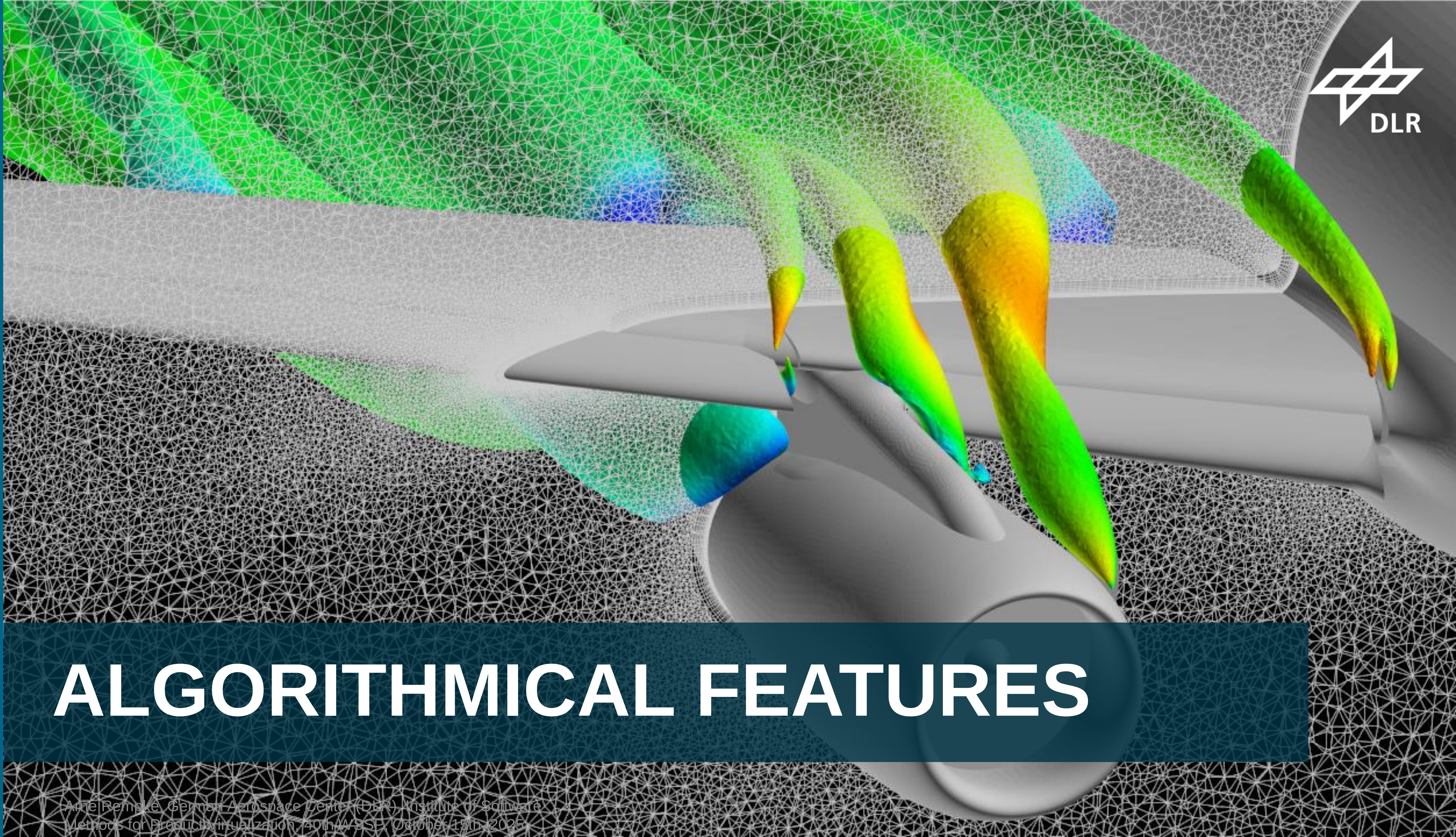
Robust methods for stiff CFD problems:

- Block- and line-implicit methods relying on a direct solution of diagonal blocks (LU) or tridiagonal blocks (=lines, Thomas-Algorithm)
- Jacobi, Gauss-Seidel, GMRES, linear multigrid, …





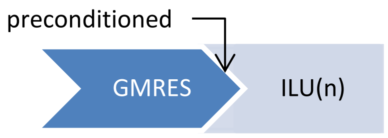




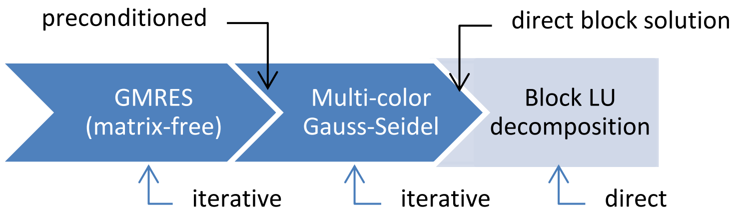
ALGORITHMICAL FEATURES

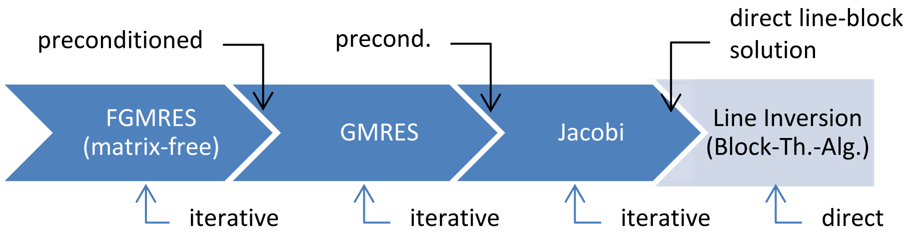
Flexible solver components

- Standard linear algebra packages provide solver/preconditioner combination:



- Spliss supports to chain multiple solver components, even with different linear operators:





Solver chaining

right preconditioned

direct diagonal solution

GMRes(A) → Jacobi(A) → LU(A_{Diagonal})

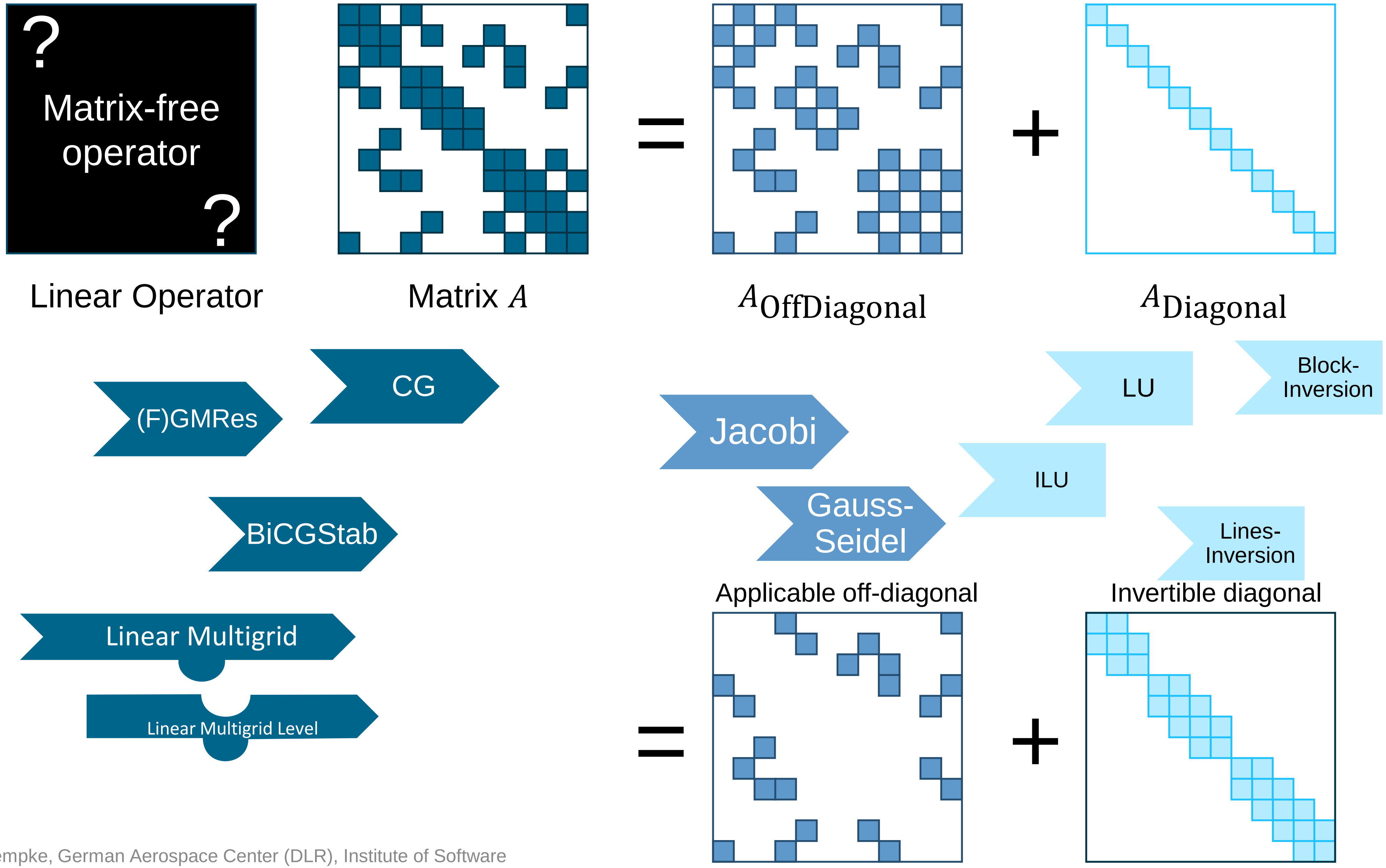
```
x = GMRes(A).Apply(b):
  v_0 = b − A x
  for i = 0, …, maxIts:
    w = A(Successor(v_i))
    v_{i+1} = Orthonormalize(w)
    Update(H, γ)
  Solve H y = γ
  w = ∑_i y_i v_i
  x += Successor(w)
```

```
x = Jacobi(A).Apply(b):
  if (A − A_OffDiagonal = Successor_Matrix):
    for i = 0, …, maxIts:
      r = b − A_OffDiagonal x
      x = (1 − λ)x + λ Successor(r)
  else:
    for i = 0, …, maxIts:
      r = b − A x
      x += λ Successor(r)
```

```
x = LU(D).Apply(b):
  x = D^{-1} b
```

Note that in case the matrices for different solver components match really well, an optimized version is applied

Featured Solver Components



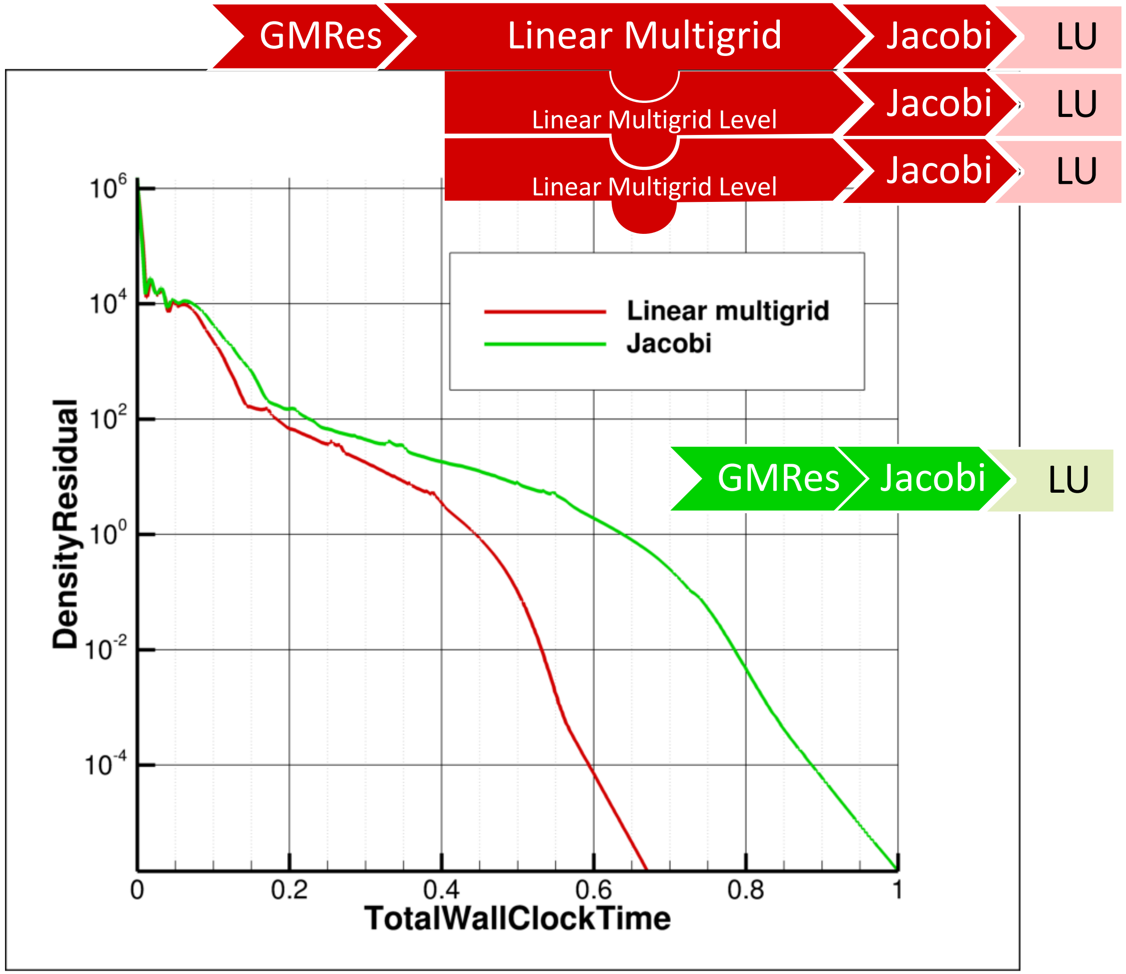
Multigrid Solver Component

Flexible integration

- Each level can use its own smoother
- Transfer operators can be user-provided

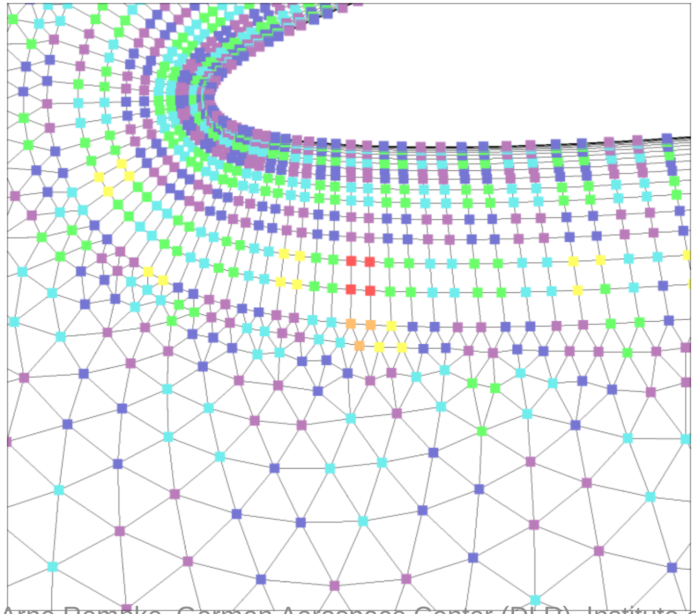
RAE2822 65k elements, CODA

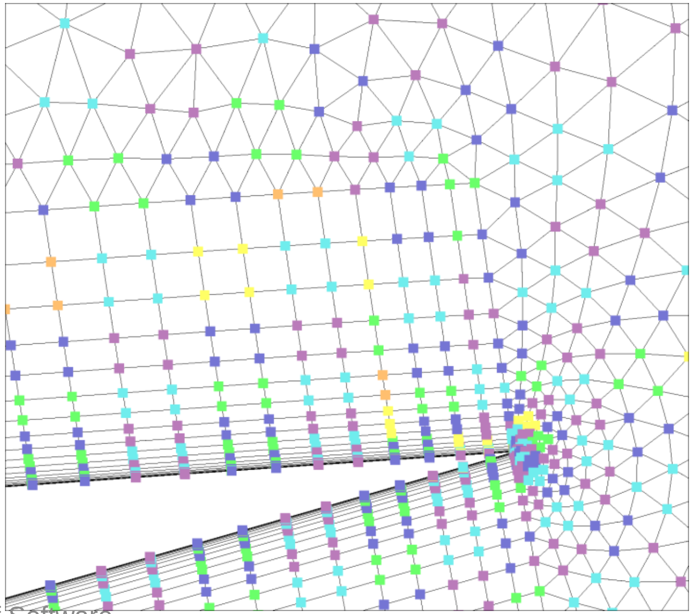
- Reduction of time to solution by 1/3 already for very small test case

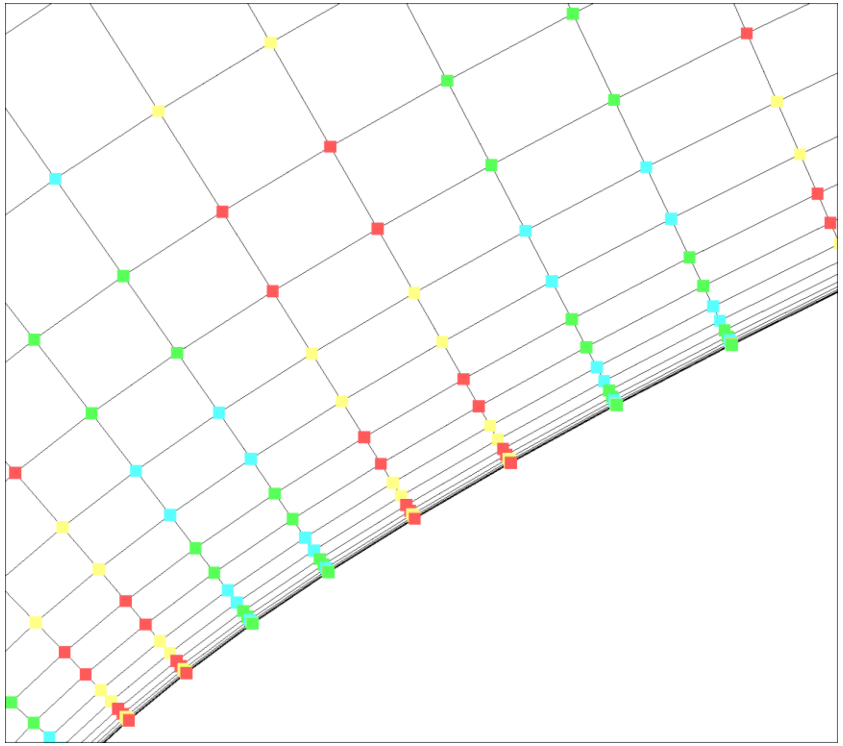


Algebraic agglomerations visualized

- 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



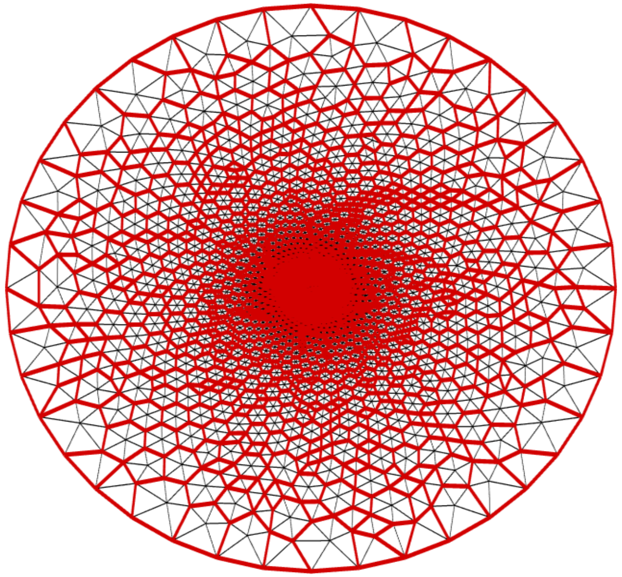


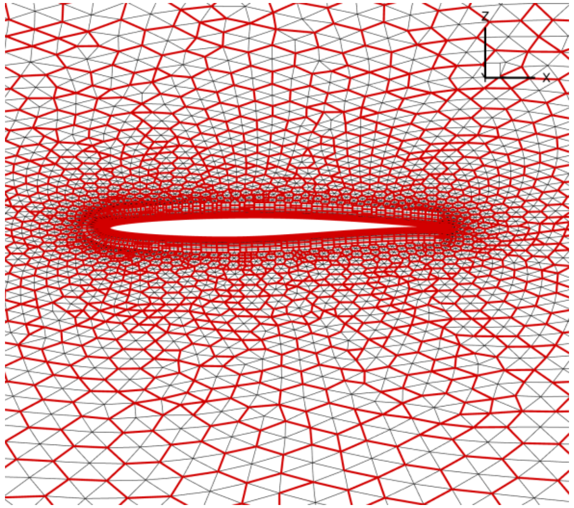


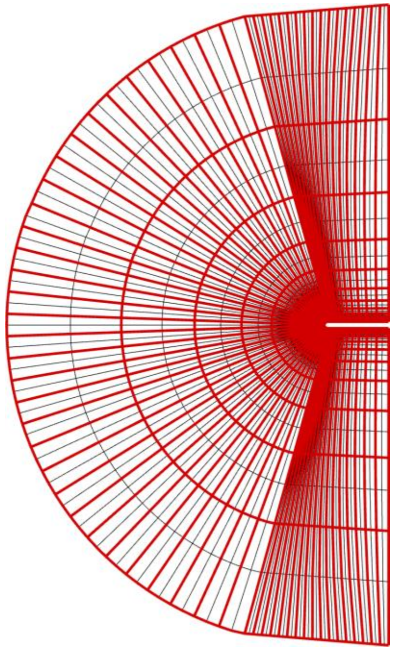
- First level agglomerates for a vertex-based discretization

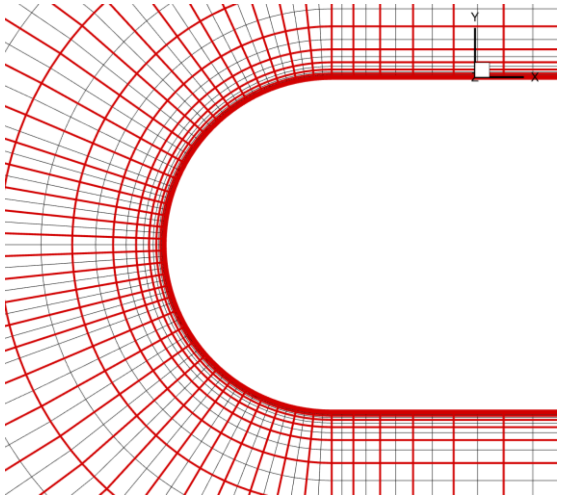
Algebraic agglomerations visualized

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







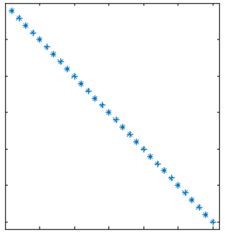


CODA integration and images by Wojciech Laskowski

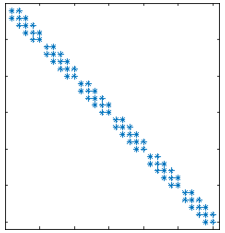
Lines Inversion / Thomas Algorithm

- Jacobi-method uses a diagonal inversion:

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



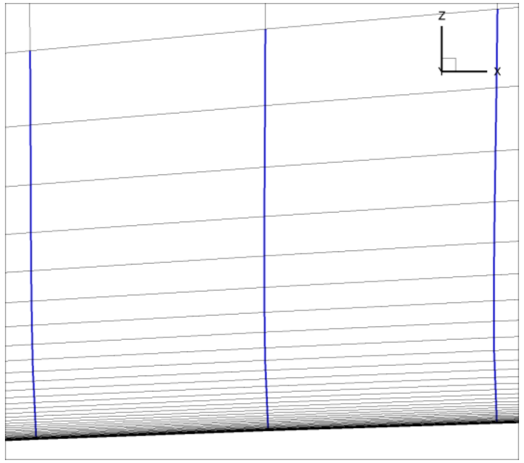
where



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

or

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

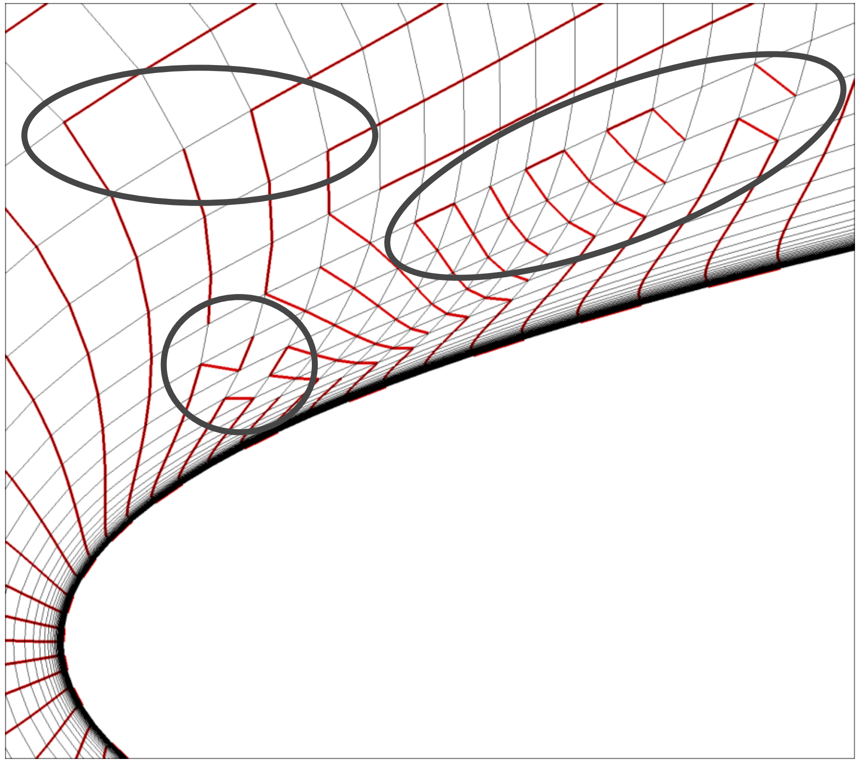


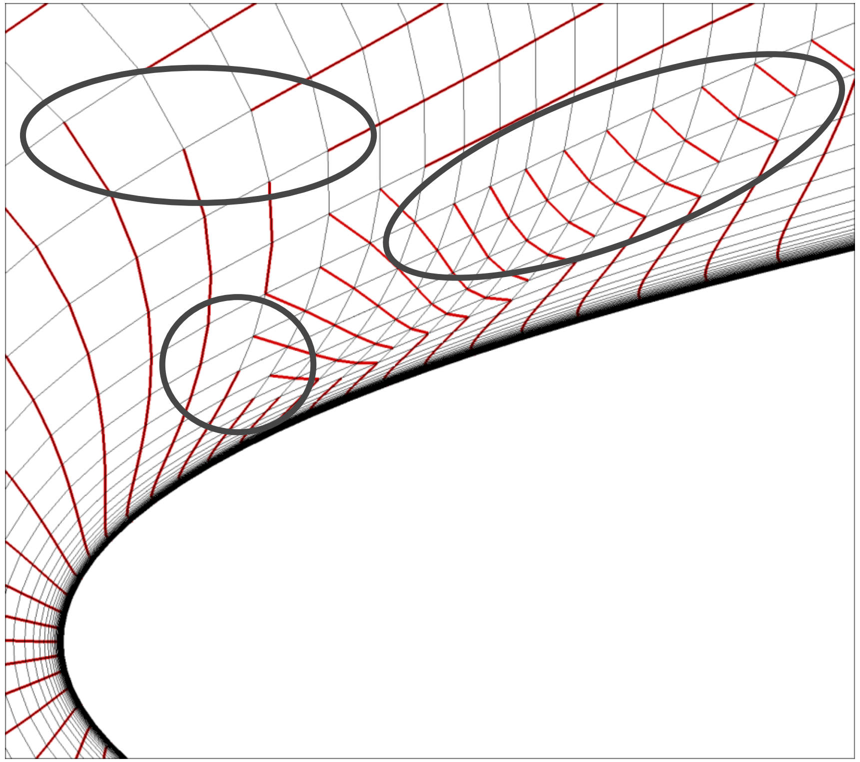
- Especially favourable/needed when mesh has very anisotropic cells, aspect ratios ≥5000:1

Improvements in line detection algorithm

Improvements especially for

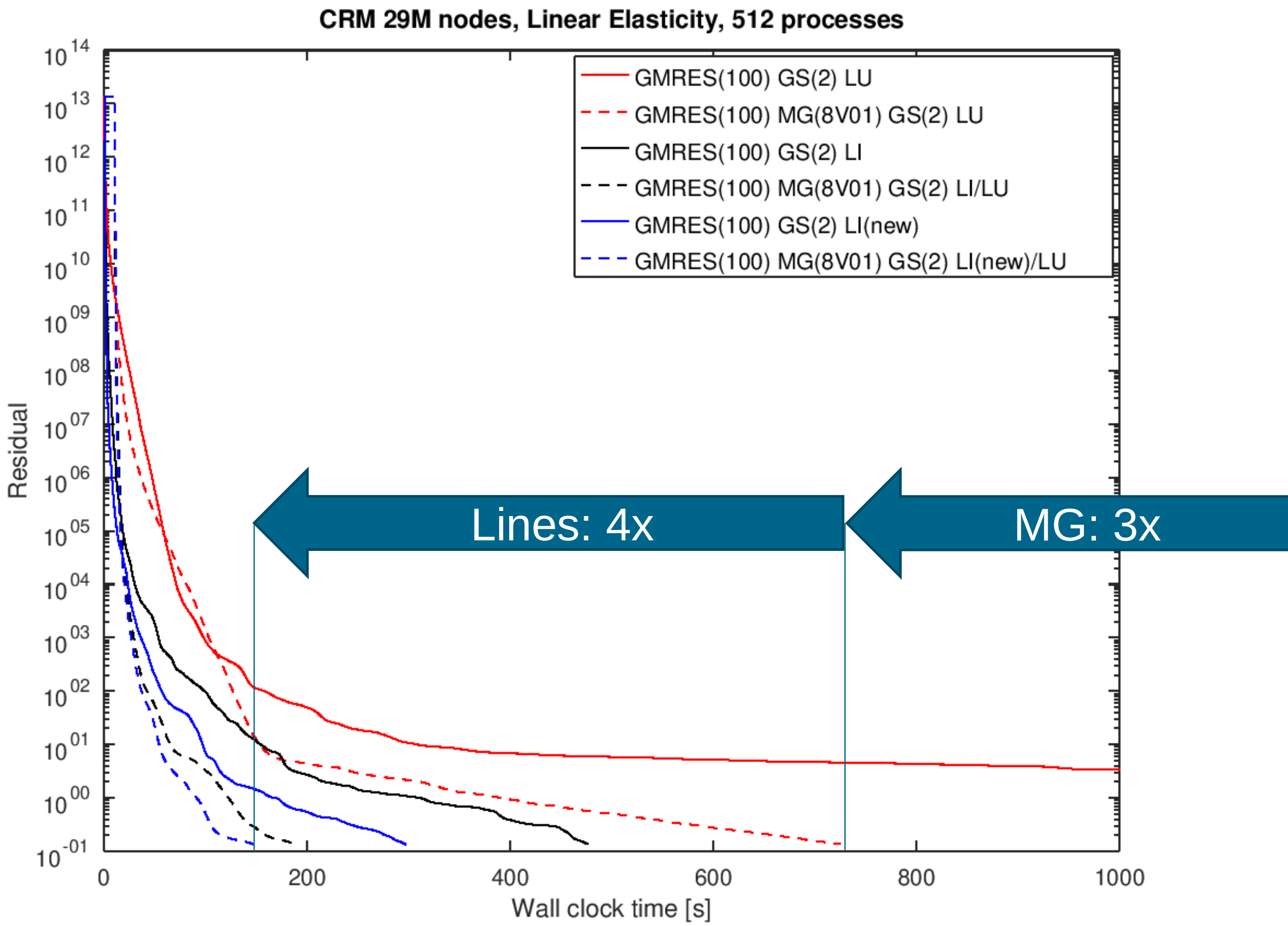
- high proportion of elements in lines
- vertex-based discretizations





Efficiency of the tailored solver components

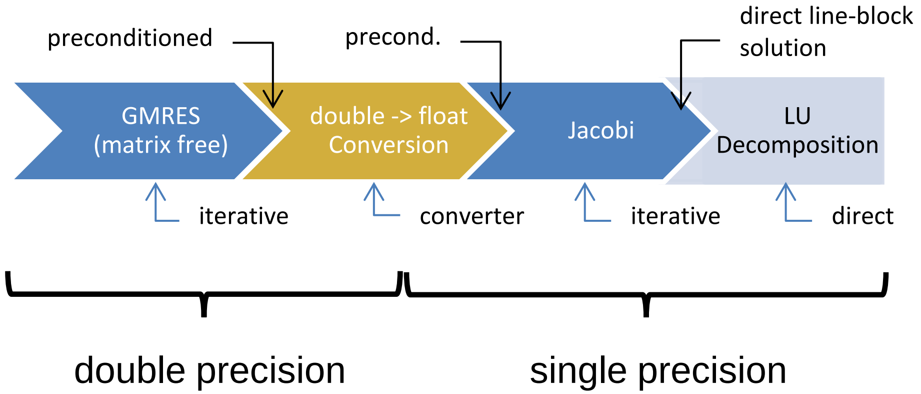
- Red solid curve is a „standard linear solver“
- Multigrid gives speedup of 2-3 (dashed)
- LinesInversion gives additional speedup of 3-4 (black/blue)

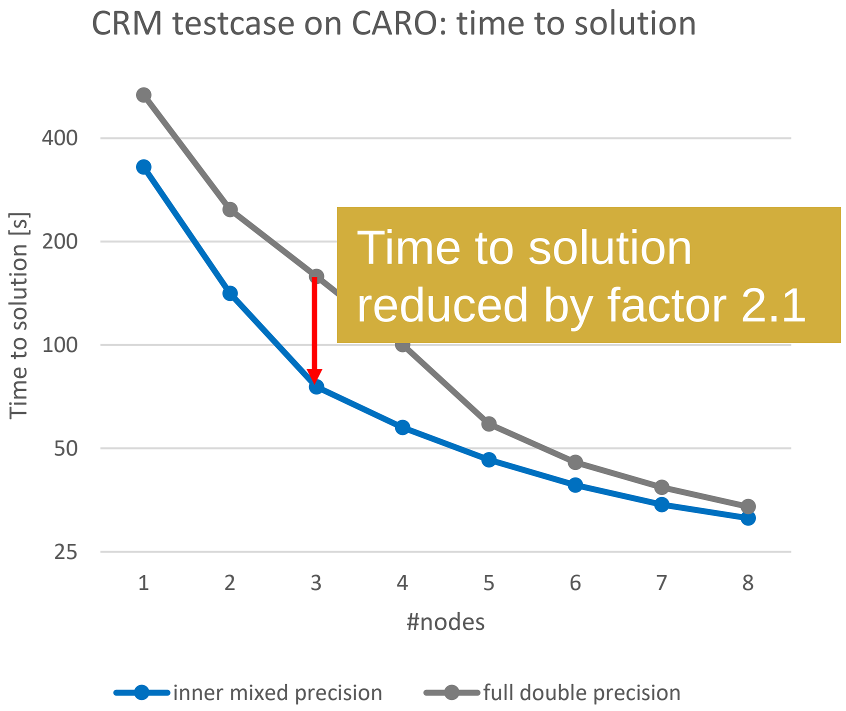


INCREASING PERFORMANCE: MIXED PRECISION

Mixed precision

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

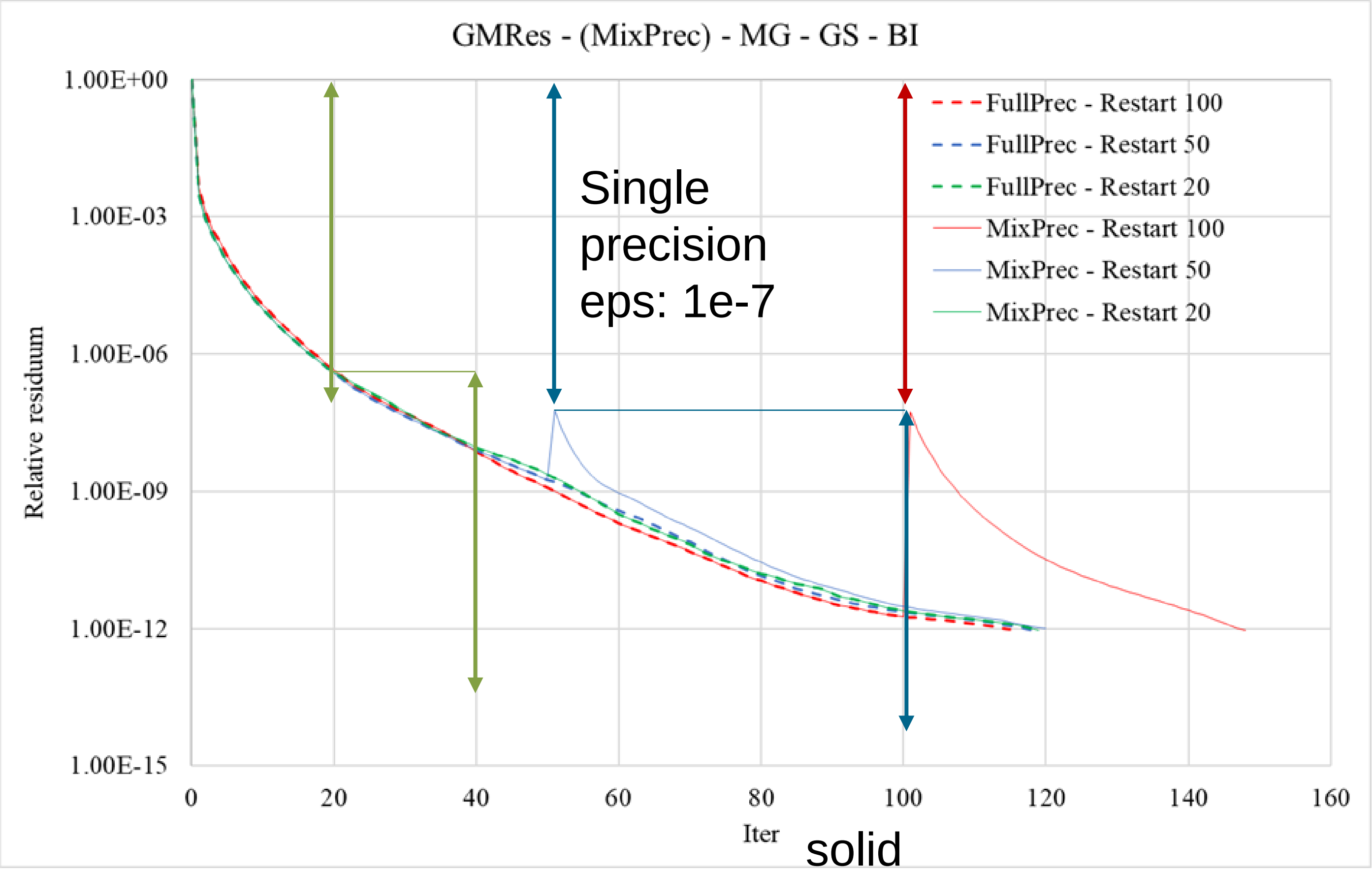




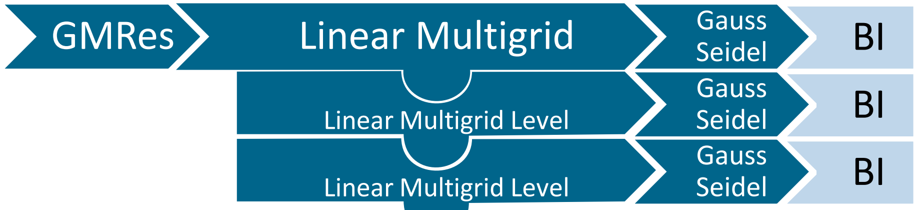
- User still provides matrix / input vectors and receives solution vector in double precision
- Inner Spliss solver components operate in float precision

Investigations on numerical influence of mixed precision

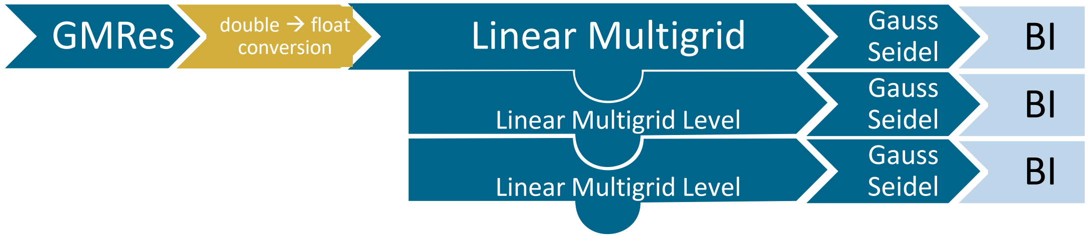
- When the reduced precision is used too much, numerical errors can occur:
  - Obvious for updates below single precision accuracy
  - But: also relevant for right-preconditioned GMRES (operating in high precision!) since the final update within the Krylov subspace is only computed in low precision



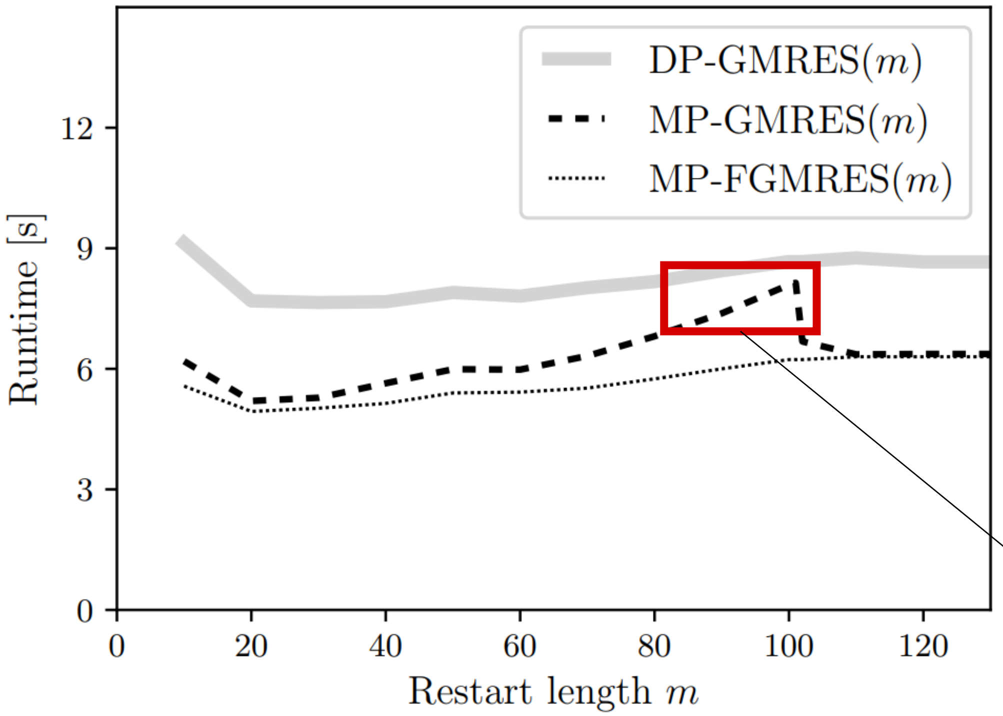
dashed

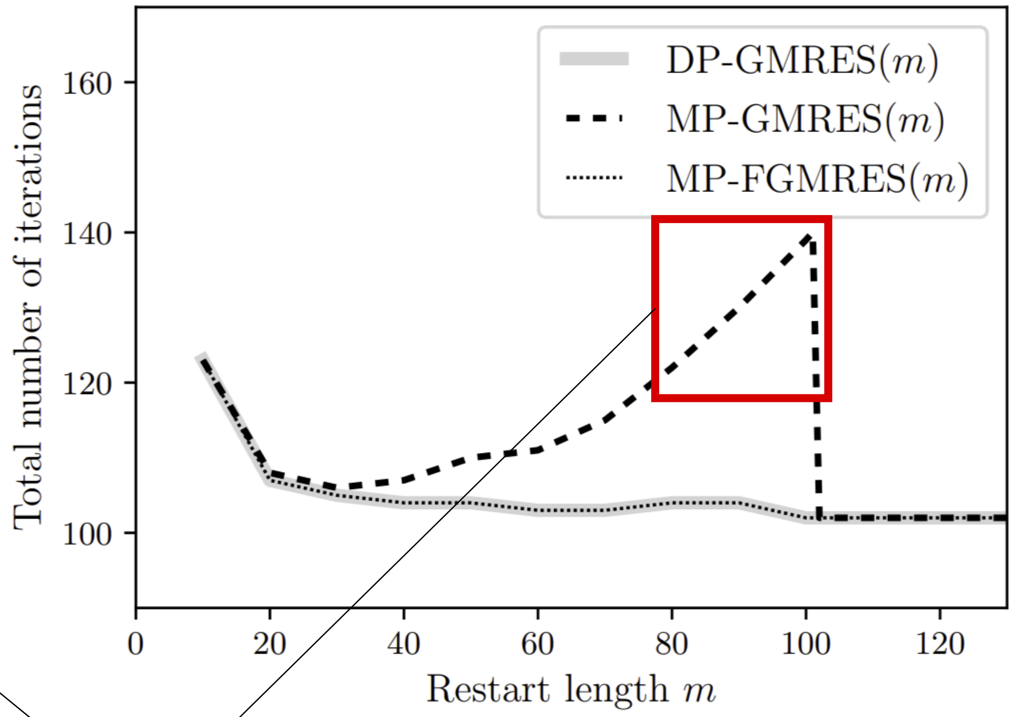


solid



GMRES(m) – variation of m





Almost entire MP runtime benefit lost due to more iterations

GMRES – algorithm

Algorithm 1 GMRES with Right Preconditioner

Require: Matrix A, single precision right preconditioner inverse P, right-hand side vector b, initial guess x

$r \leftarrow b - Ax$ ▷ Initial residual
$v_0 \leftarrow r/\|r\|$ ▷ Normalize residual
while $\|r\| > \text{tol}$ **do** ▷ Main iteration loop
  for $j = 0, 1, 2, \dots, n$ **do**
    $z_j \leftarrow \boxed{P}(v_j)$ ▷ Apply right preconditioner
    $w_j \leftarrow Az_j$ ▷ Matrix-vector product
    Perform Arnoldi process (classical Gram-Schmidt) to orthogonalize w_j
    Update Hessenberg matrix H
    $v_{j+1} \leftarrow w_j/\|w_j\|$ ▷ Normalize new Krylov vector
  end for
  Solve $Hy = \gamma$ ▷ Least-squares solve for coefficients
  $w \leftarrow \boxed{P}\left(\sum_{j=0}^{n} y_j v_j\right)$ ▷ Sum updates
  $x \leftarrow x + w$ ▷ Update solution
  $r \leftarrow b - Ax$ ▷ Recompute residual
end while
return x

GMRES – source of problem: solution update

Classical MGS-GMRES update (error prone):

$$w \leftarrow P\left(\sum_{j=0}^{n} y_j v_j\right) \neq \sum_{j=0}^{n} y_j P\left(v_j\right)$$

Applies preconditioner
to the sum of the weighted directions
to save computational cost

However casting to SP is not a linear operation, thus:

$$\overset{\text{nonlinear}}{P}\left(y_j v_j\right) \neq y_j \overset{\text{nonlinear}}{P}\left(v_j\right)$$

Possible solutions:

- Restart when single precision accuracy reached — problem specific setting or implementation of a custom restart criterion
- F-GMRES — stores preconditioned Krylov vectors at the cost of additional memory
- Matrix only mixed precision — vectors are kept in double precision; no full advantage from mixed precision

PARALLELIZATION AND SCALABILITY

Arne Rempke, German Aerospace Center (DLR), Institute of Software
Methods for Product Virtualization, 40th WSSP, October 15th, 2025

Main Operation during Solving: d = A · s

With:

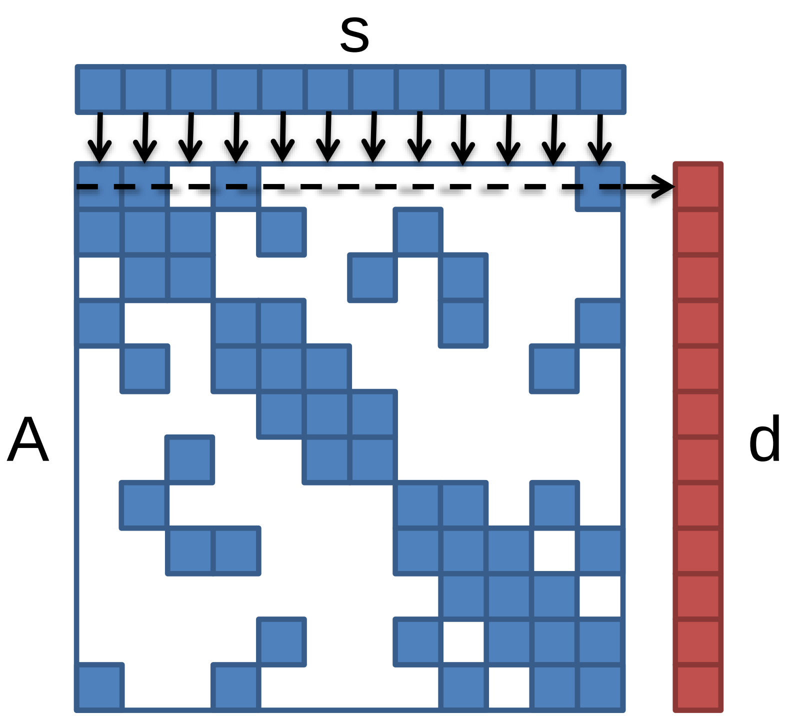
s: Source Vector

d: Destination Vector

A: Matrix

Formula: $\boldsymbol{d_i} = \sum_{j=0}^{N} \boldsymbol{A_{ij}} \cdot \boldsymbol{s_j}$

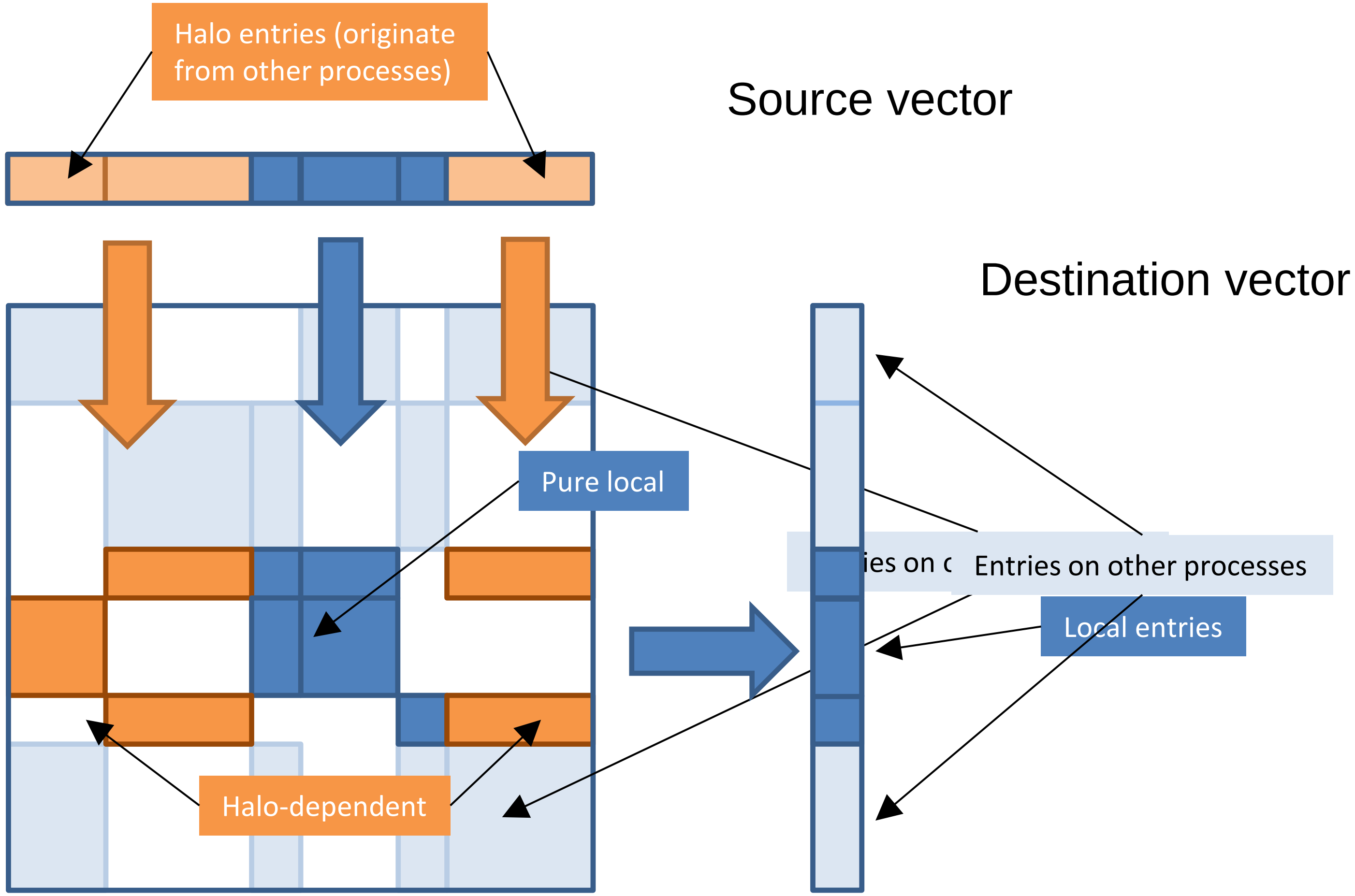
→ All rows can be computed independently.



Distributed Memory Parallelization

Program run:

1. Start sending halo data
2. Computation of pure local parts
3. Computation of halo parts

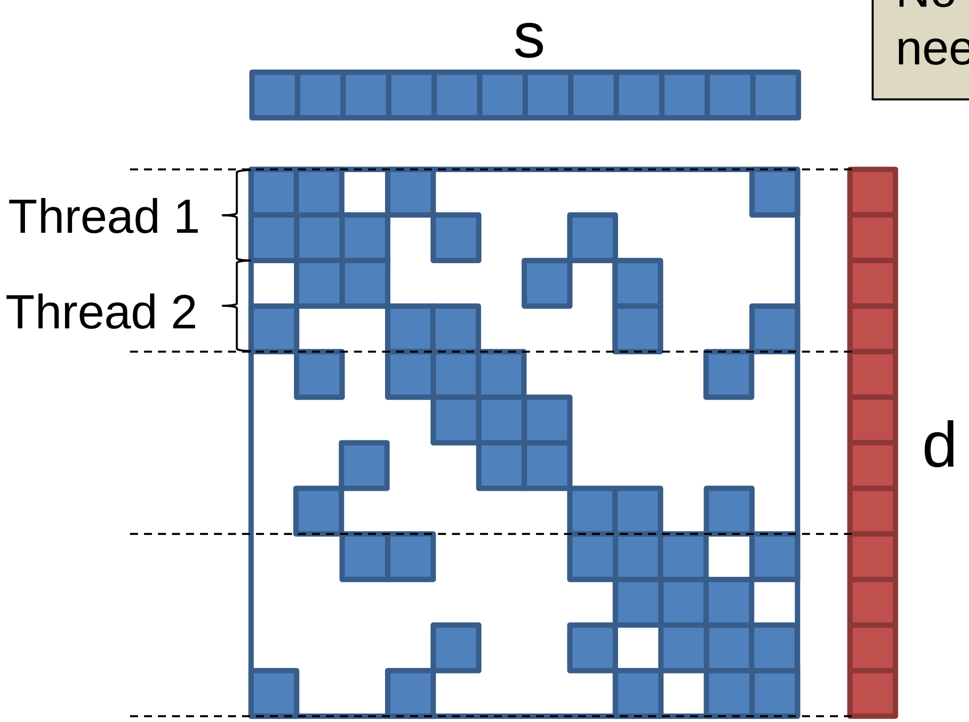


Shared Memory Parallelization

Straightforward derived from cluster level parallelization

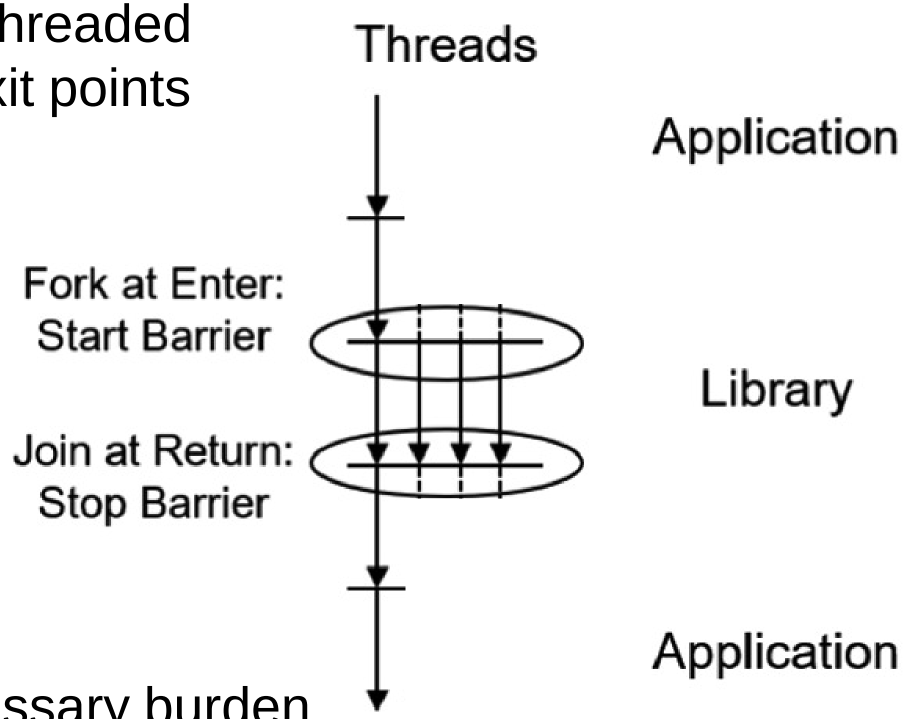
- Every thread computes some rows
- Same strategy on CPU and GPU

No synchronization needed for write



Threading model

- Typical design of a library
  - Single threaded entry/exit points



  - Unnecessary burden if the user code also uses threads

- Spliss design
  - Allows to enter/exit with all threads

GPU Parallelization

- Similar as for Multithreading
- Using alpaka* allows us to write a single Kernel to be executed on CPU or GPU
- Spliss hides the CUDA backend/compiler/… from user code:
  - Explicit template instantiations of CUDA-dependent classes on Spliss compilation
  - No necessity to use nvcc for user code
- Since Spliss is a C++ template library, user calls to small functions, e.g.

```
A[row][col] += myContribution;
```

can still be inlined, allowing a seamless integration while capsulating the actual memory layout

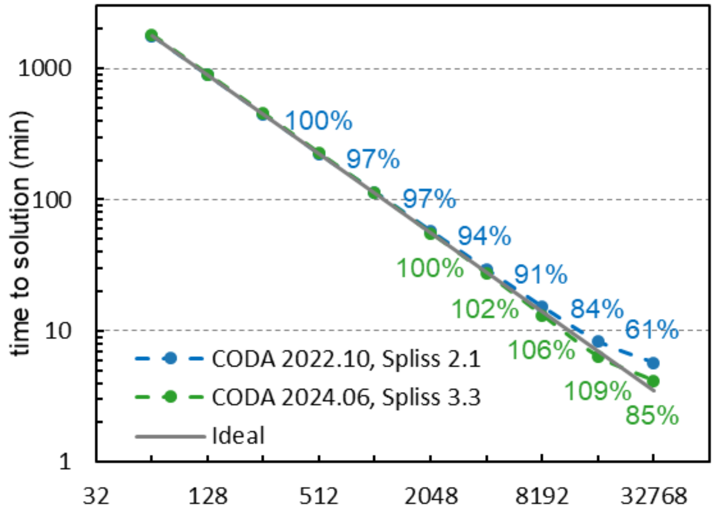
* https://github.com/alpaka-group/alpaka

Efficiency: Scalability on Current Systems (CARA)

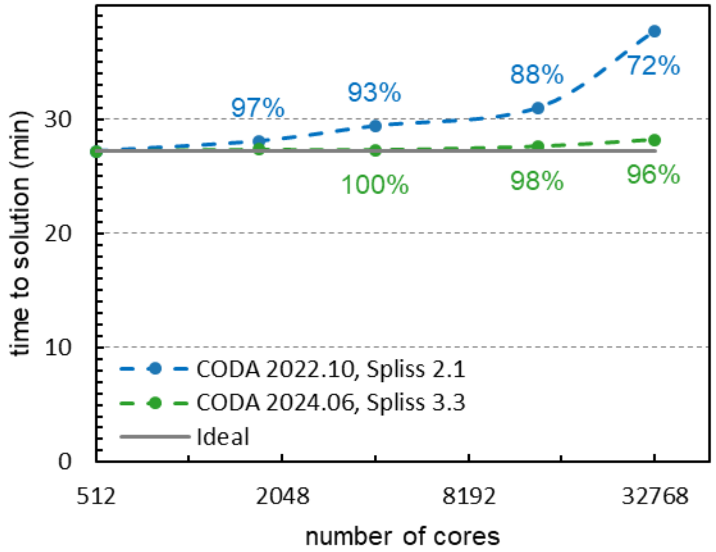
Scalability assessment on DLR's production system CARA

- Strong scaling (CRM, fixed problem size, 24M elements):
  - Scaling from 1 – 512 nodes (largest available partition)
  - Reduce runtime from 1.2 days to 4.2 minutes
  - **Small mesh**: just 730 elements/core @ 32,768 cores
  - Scaling 64 – **32,768 cores: 85% strong scaling efficiency**
  - Small super-linear speedup
- Weak scaling (CRM, fixed workload per core, 3M – 192M elements):
  - Scaling 512 – 32,768 cores: 96% weak scaling efficiency

Runtime | strong scaling 24M elements



Runtime | weak scaling 3M–192M elements



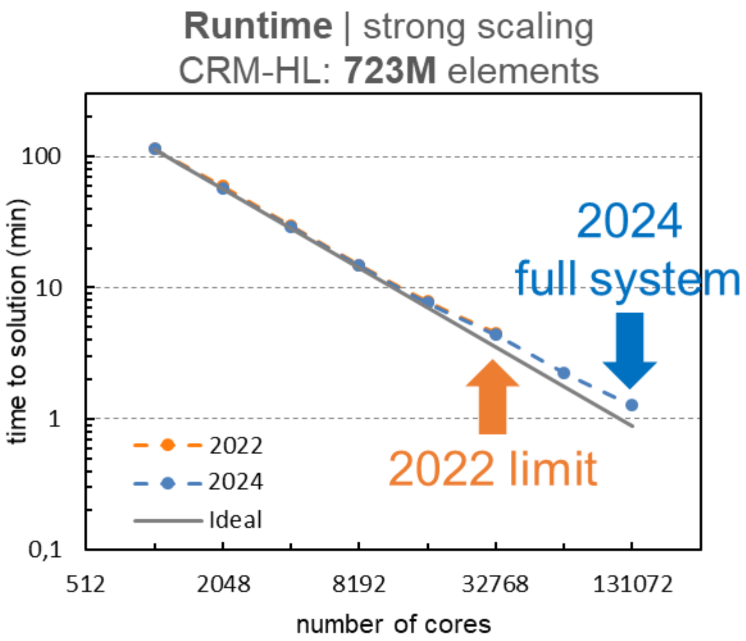
Scalability of the Entire Workflow (FlowSimulator)

Scalability assessment on DLR’s production system CARO

- The entire workflow needs to …
  - support large meshes (>1B elements)
  - support large core counts (>1M cores)

Achievements (so far)

- Several improvements in FlowSimulator to scale to full system
- Improved hierarchical graph partitioning
- Support for meshes >1 billion elements tested
- Efficient scaling to 131,072 cores (full system CARO@DLR)



Heterogeneity: Support for Nvidia GPUs via Spliss

System – Juwels Booster

- AMD Epyc 7401 (2x 24 cores) per node
- 4x Nvidia Tesla A100 per node

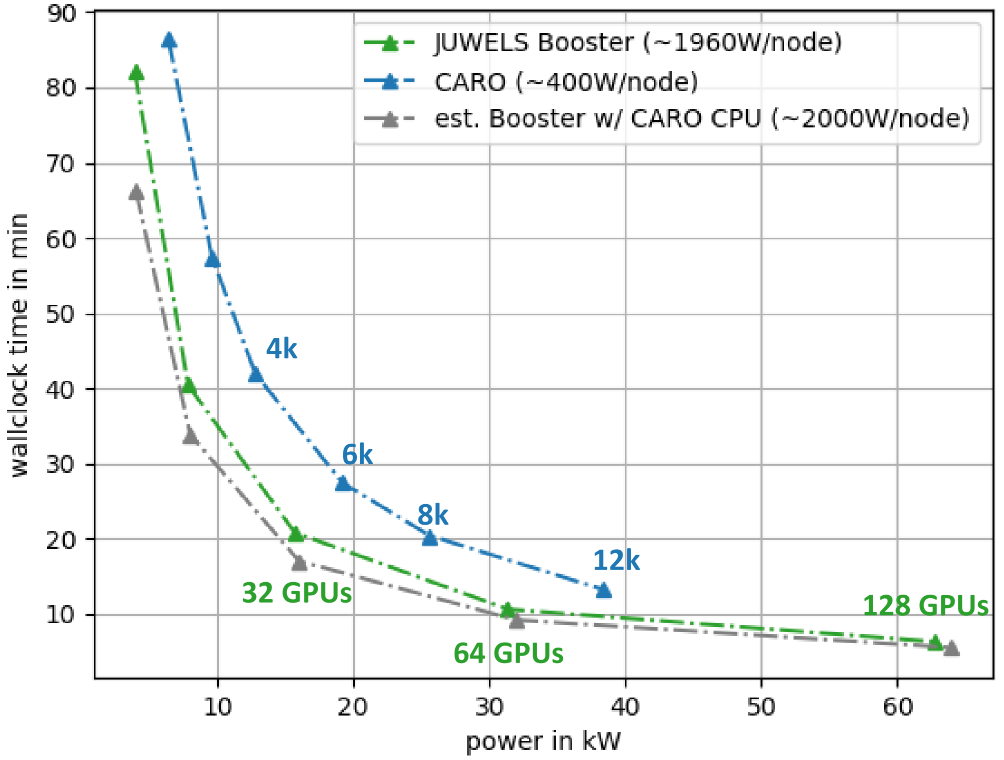
System – CARO

- Nodes with 128 cores (AMD Rome, 2x CARA)
- 8 memory channels @3.2GHz (1.2x CARA)

Observations

- Node-wise comparison (“unfair”): 8-9x speedup
- Energy-wise comparison (“fair”): 1.6-1.9 speedup
- Performance limited by non-linear part on slow CPU
- Hypothetical Juwels Booster node with CARO CPU: 1.8-2.3 speedup (energy-wise)

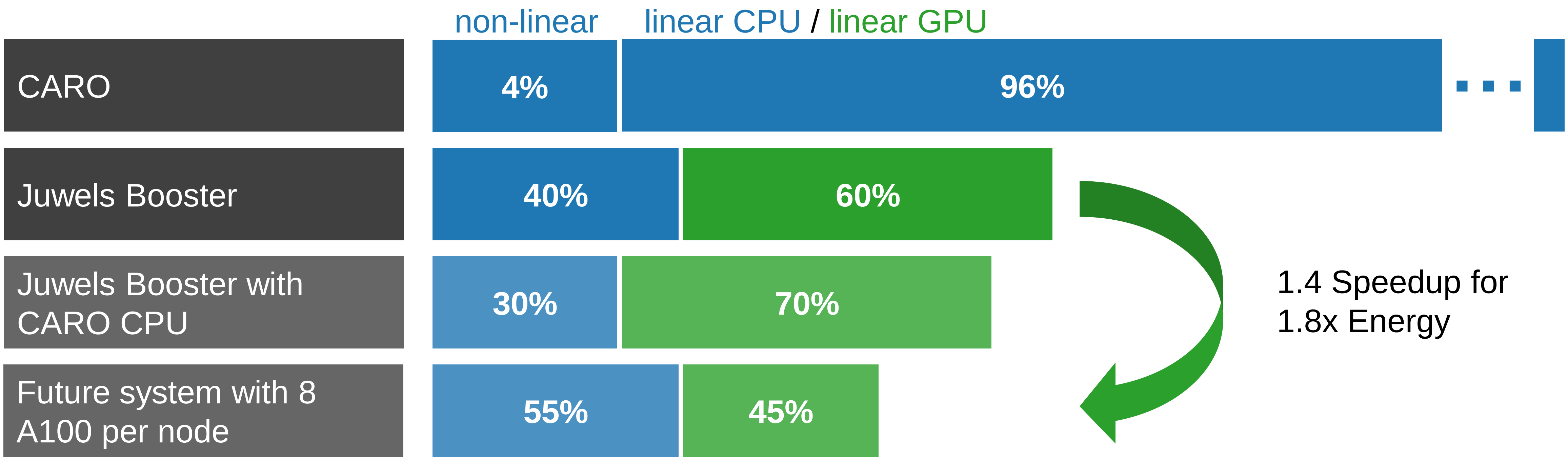
Runtime | CARO (AMD Rome) vs. Juwels (4x Nvidia A100)
M6 wing, 69.2M elements, implicit Euler, Jacobi + Block Inv.

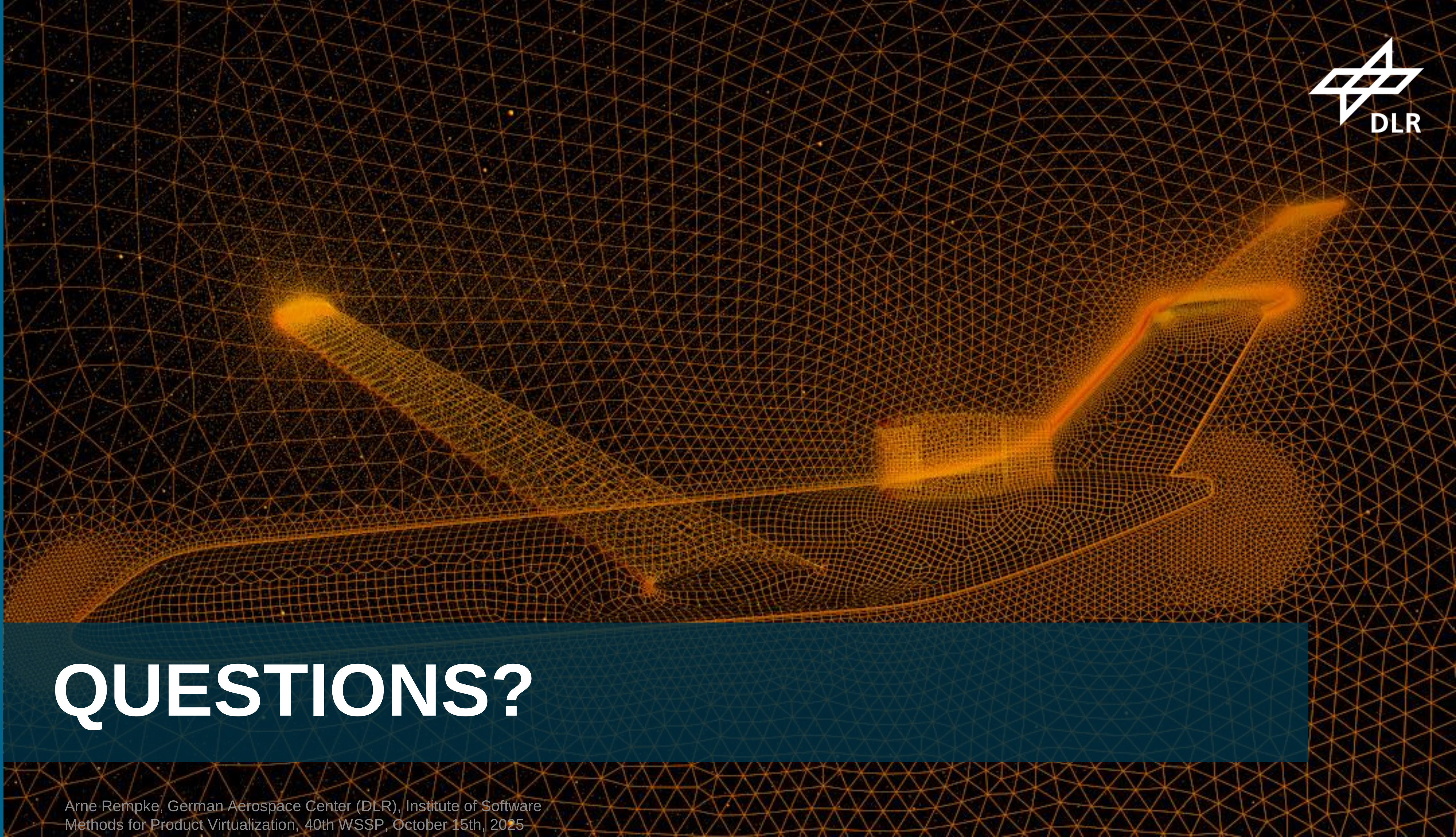


J. Mohnke and M. Wagner: A Look at Performance and Scalability of the GPU Accelerated Sparse Linear System Solver Spliss. In: Euro-Par 2023: Parallel Processing. DOI 10.1007/978-3-031-39698-4_43

GPU Development
Impact of GPU support of Spliss on application wallclock time

- For implicit methods in CODA, linear equation systems are solved via Spliss
- Thus, only the linear part of CODA benefits from GPUs
- For future system with more or more powerful GPUs the non-linear part may become bottleneck





QUESTIONS?