

An octree-based sampling algorithm for analyzing big simulation data

Janis Geise¹, Sebastian Spinner², Richard Semaan³, and Andre Weiner¹

¹Technical University of Dresden, Institute of Fluid Mechanics, 01062 Dresden, Germany,

janis.geise@tu-dresden.de, ORCID ID 0009-0009-9537-4294,

andre.weiner@tu-dresden.de, ORCID ID 0000-0001-5617-156,

²DLR, German Aerospace Center, 38108 Braunschweig, Germany,

sebastian.spinner@dlr.de, ORCID ID 0000-0002-3661-6814

³ZEISS Segment Semiconductor Manufacturing Technology

Carl Zeiss SMT GmbH, 73447 Oberkochen, Germany

richard.semaan@zeiss.com, ORCID ID 0000-0002-3219-0545

November 26, 2025

Abstract

As computational resources continue to increase, the storage and analysis of vast amounts of data will inevitably become a bottleneck in computational fluid dynamics (CFD) and related fields. Although compression algorithms and efficient data formats can mitigate this issue, they are often insufficient when post-processing large amounts of volume data. Processing such data may require additional high-performance software and resources, or it may restrict the analysis to shorter time series or smaller regions of interest. The present work proposes an improved version of the existing *Sparse Spatial Sampling* algorithm (S^3) to reduce the data from time-dependent flow simulations. The S^3 algorithm iteratively generates a time-invariant octree grid based on a user-defined metric, efficiently down-sampling the data while aiming to preserve as much of the metric as possible. Using the sampled grid allows for more efficient post-processing and enables memory-intensive tasks, such as computing the modal decomposition of flow snapshots. The enhanced version of S^3 is tested and evaluated on the scale-resolving simulations of the flow past a tandem configuration of airfoils in the transonic regime, the incompressible turbulent flow past a circular cylinder, and the flow around an aircraft half-model at high Reynolds and Mach numbers. S^3 significantly reduces the number of mesh cells by 35% to 98% for all test cases while accurately preserving the dominant flow dynamics, enabling post-processing of CFD data on a local workstation rather than HPC resources for many cases.

Keywords— big simulation data, mesh sampling, post-processing, modal analysis

Nomenclature

Abbreviations

AZDES	Automated Zonal Detached Eddy Simulation
CFD	Computational Fluid Dynamics
DNS	Direct Numerical Simulation
HPC	High Performance Computing
IDDES	Improved Delayed Detached Eddy Simulation
MAC	Mean Aerodynamic Chord
S^3	Sparse Spatial Sampling
SVD	Singular Value Decomposition
TKE	Turbulent Kinetic Energy

Greek symbols

ν	kinematic viscosity
-------	---------------------

ν_t	eddy viscosity field
σ_i	i^{th} singular value
Σ	matrix of singular values
τ	dimensionless time

Latin symbols

c	chord length
d	diameter
D	number of physical dimensions
\mathbf{f}	generic flow field
\mathbf{k}	turbulent kinetic energy field
K	number of neighbors
l	cell level
l_0	edge length of the initial cell
M	number of mesh cells
Ma_∞	free stream Mach number
\mathbf{Ma}	Mach number field
$\widehat{\mathbf{Ma}}$	interpolated Mach number field
N	number of snapshots
\mathbf{p}_i	spatial coordinates of a point i
$\widehat{\mathbf{p}}_i$	spatial coordinates of a query point i
Re	Reynolds number
Sr	Strouhal number
t	time
\mathbf{u}_i	i^{th} component of the velocity field
\mathbf{U}	matrix of left-singular vectors
\mathbf{v}_i	i^{th} right-singular vector
V_l	area (2D) / volume (3D) of a cell of level l
\mathbf{V}	matrix of right-singular vectors
w	inverse distance weight
x	coordinate in x -direction
\mathbf{x}_n	snapshot at time step n
$\hat{\mathbf{x}}_n$	interpolated snapshot at time step n
\mathbf{X}	data matrix
y	coordinate in y -direction
z	coordinate in z -direction

Symbols

\mathcal{G}_l	gain of a cell at level l
\mathcal{M}_i	metric of cell i
$\widehat{\mathcal{M}}_i$	interpolated metric of cell i
\mathcal{M}	metric field

$\widehat{\mathcal{M}}$	interpolated metric field
$\mathcal{M}_{\text{approx}}$	metric approximation quality defined as $\ \widehat{\mathcal{M}}\ /\ \mathcal{M}\ $
\mathcal{M}_{\min}	threshold value for minimum metric approximation quality
Indices	
a	adaptive
c	child cell
g	geometry
ℓ	leaf cell
rn	renumber
uni	uniform

1 Introduction

Increasing computational resources enable the usage of more complex models and scale-resolving simulations in computational fluid dynamics (CFD). Moreover, the decreasing cost of high-performance computing (HPC) resources and improved accessibility motivate performing more simulations, e.g., for parameter studies and optimization [4]. However, high-fidelity simulations such as large eddy simulations (LES) or detached eddy simulations (DES) yield massive amounts of data that need to be stored and processed. The progressive usage of machine learning techniques for flow analysis and modeling, e.g., via modal decompositions for dimensionality reduction, requires further availability of highly sampled flow field sequences [2, 23]. While computational resources were the main limiting factor for CFD for a long time, it is foreseeable that now the available storage capacity and data analysis will become the bottleneck for large-scale simulations [4]. Post-processing such amounts of data nowadays often requires HPC resources. In 2022, data centers already contributed 2% to the worldwide energy consumption, and a doubling is expected in their energy consumption by 2026 [10], emphasizing the need for efficient data management and data reduction.

Existing data reduction techniques can be classified as lossless or lossy compression algorithms. While lossless compression algorithms aim to remove redundancies in the data, the achievable compression ratio is limited [4]. Lossless compression is generally applied whenever the exact reconstruction of the data is crucial, e.g., when compressing text files such as source code. Although lossless compression can reduce the required disk space, it does not reduce the required computational resources for post-processing. A widely used alternative is lossy compression. A variety of algorithms already exists that utilize digit rounding, quantization, or similar methods [4]. Although lossy compression can reach a significantly higher compression ratio than lossless compression, generally applicable algorithms do not leverage fundamental physical phenomena inherent to fluid flows.

Data reduction for post-processing CFD data using flow physics can be accomplished by decreasing the number of snapshots, i.e., the temporal resolution of output files, or by reducing the mesh size of the simulation, i.e., the spatial resolution. Decreasing the number of snapshots is rarely an option for tasks such as modal analysis, as a sufficiently high sample frequency is required to avoid undesired effects like aliasing. Consequently, most of the work conducted so far focuses on utilizing coarse grids to accelerate the simulation in conjunction with coarse-grid corrections applied to improve the obtained solution [9, 11, 19].

One possible reduction approach is to optimize the grid topology by removing nodes that are not necessary to represent spatial or temporal variation. This idea was investigated by Klein et al. [12], aiming to decrease the number of nodes for the surface representation of geometries in medical applications. More recently, Lorsung and Farimani [17] utilized deep reinforcement learning to identify mesh nodes that can be removed, accelerating simulations while minimizing the associated accuracy loss. While the latter approach focused on reducing the runtime of the simulation, Lee et al. [15] leveraged autoencoders to compress spatiotemporal CFD data, achieving compression by two orders of magnitude.

It is important to note that lossless and lossy approaches can be combined to maximize data reduction while minimizing information loss. For instance, using binary formats such as HDF5 combined with lower precision - e.g., single precision floating point numbers - will lead to a significant decrease in the computational resources required and may already be sufficient in many cases. If a higher compression ratio is required, lossy algorithms such as mesh reduction algorithms or truncation can be applied additionally.

The concept of removing unnecessary grid nodes and associated snapshot data is also the core idea of *Sparse Spatial Sampling* (S^3), introduced by Fernex et al. [6]. S^3 consists of three steps. Initially, a point cloud is generated from the vertices of the original mesh. Then, a subset of the point cloud is selected based on high values of a user-defined metric. A sensible metric to investigate temporal fluctuations of turbulent flows is the turbulent kinetic energy (TKE). Next, a coarse mesh is generated

from the reduced point cloud by employing 3D Delaunay triangulation. One or more Laplacian smoothing iterations are run to improve the mesh quality. Lastly, the mesh vertices of the coarse mesh are snapped to the closest vertices of the original mesh. The snapping avoids the need for more sophisticated interpolation methods to map the field data between the meshes. The new mesh is static (invariant in time). While demonstrating the potential of mesh reduction based on flow physics, the original S^3 algorithm has the following shortcomings:

- Point cloud reduction: selecting points based on high metric values, e.g., the temporal variance of a field, leads to large empty regions in some parts of the domain, while most points are clustered in other parts. The derived mesh can display poor quality, and the reduction can be suboptimal in regions where the metric is uniformly high.
- Cell topology: the derived mesh comprises tetrahedral cells. The reduction is less effective if the original mesh comprises hexahedral or polyhedral cells.
- Boundary treatment: the algorithm does not handle spurious cells outside the actual domain, which can be created during the remeshing.
- Missing interpolation: while snapping points from the derived mesh to the closest ones of the original mesh avoids additional interpolation, the field values of small original mesh cells might be copied to large cells of the derived mesh, potentially introducing a significant interpolation error.

In the fully revised version of S^3 presented in this contribution, the mesh generation starts with a single cubic (3D) or quadrilateral (2D) cell placed around the numerical domain, which is then refined iteratively based on a user-defined metric field, aiming to approximate the metric field as closely as possible. The refinement process stops once a user-controlled stopping criterion is reached. Finally, field values are interpolated to the resulting quadtree/octree mesh.

The remainder of this article is structured as follows: section 2 presents the updated S^3 version. Section 3 evaluates S^3 on three unsteady flow simulations, namely, the transonic flow past an airfoil tandem configuration (section 3.1), the fully-resolved turbulent flow past a circular cylinder (section 3.2), and the transonic flow around an aircraft half-model (section 3.3). We employ a singular value decomposition (SVD) to compare original and reduced data. The S^3 source code and the examples are publicly available on GitHub [7].

2 Sparse Spatial Sampling

The new version of S^3 follows the same principle idea as the original version presented in [6], namely generating a time-invariant coarse grid based on a user-defined metric field. This coarse grid can then be used to post-process CFD data more efficiently. Rather than successively coarsening the original grid, the revised version of S^3 starts by creating a single large cell, which is refined iteratively. This procedure enables the use of efficient recursive trees to store and update the mesh with each iteration.

As already mentioned in section (1), S^3 consists of three main steps: (i) the computation of a metric field, (ii) the adaptive, metric-based grid generation, and (iii) the interpolation of the original data onto the new grid. This process is illustrated in fig. (1) and will be discussed thoroughly below.

Suppose that scalar field values are given at M spatial points $\mathbf{p}_i = [x_i, y_i, z_i]^T$ of the original mesh, with $i \in \{1, 2, \dots, M\}$. For a co-located finite volume mesh, these points correspond to the cell centers, so M is equivalent to the number of cells composing the mesh. Since the example data used in section (3) are generated by finite volume codes, we stick to this interpretation of M for the remainder of this manuscript. However, in principle, S^3 does not rely on the mesh topology or the underlying numerical method. The first step is the calculation of a metric field $\mathbf{M} \in \mathbb{R}^M$ based on the time series of N snapshots at discrete times t_n of one or more scalar fields $\mathbf{f}_n \in \mathbb{R}^M$ with $n \in \{1, 2, \dots, N\}$. The metric is a user-defined scalar field that should express the importance of each location \mathbf{p}_i . For example, a simple but effective metric to accurately capture the shock oscillations of transonic shock buffet is the temporal standard deviation of the local Mach number, i.e., $\mathbf{M} = \text{std}(\mathbf{Ma})$ with

$$\text{std}(\mathbf{f})^2 = \frac{1}{N-1} \sum_{n=1}^N (\mathbf{f}_n - \bar{\mathbf{f}})^2, \quad \bar{\mathbf{f}} = \frac{1}{N} \sum_{n=1}^N \mathbf{f}_n. \quad (1)$$

There is virtually no constraint on \mathbf{M} other than it has to contain a scalar value for each point \mathbf{p}_i . Users can easily tailor it to their specific target application and embed physical knowledge, e.g., to focus the refinement on regions with strong temporal fluctuations or production of turbulent kinetic energy. The target-specific metric enables higher compression rates of the data while maintaining flexibility.

As shown in fig. (1), the grid generation consists of three steps. First, a uniform background grid is created (step 2.1), followed by an adaptive, metric-based refinement (step 2.2), and an optional geometry refinement (step 2.3). The mesh structure is a quadtree for 2D or an octree for 3D cases. Each cell has the same edge length in all coordinate directions. The uniformly refined background mesh serves as a starting point for the metric-based refinement, which accelerates the overall process because the adaptive refinement is more computationally expensive. The mesh generation starts by creating an initial cell with edge length l_0 that contains all points \mathbf{p}_i (not shown in fig (1)). Although the uniform refinement is straightforward, it serves here to explain the terminology. Depending on the number of spatial dimensions, $D \in \{2, 3\}$, each cell marked for refinement is split into four or eight child-cells for 2D and 3D cases, respectively. The cell that has been split

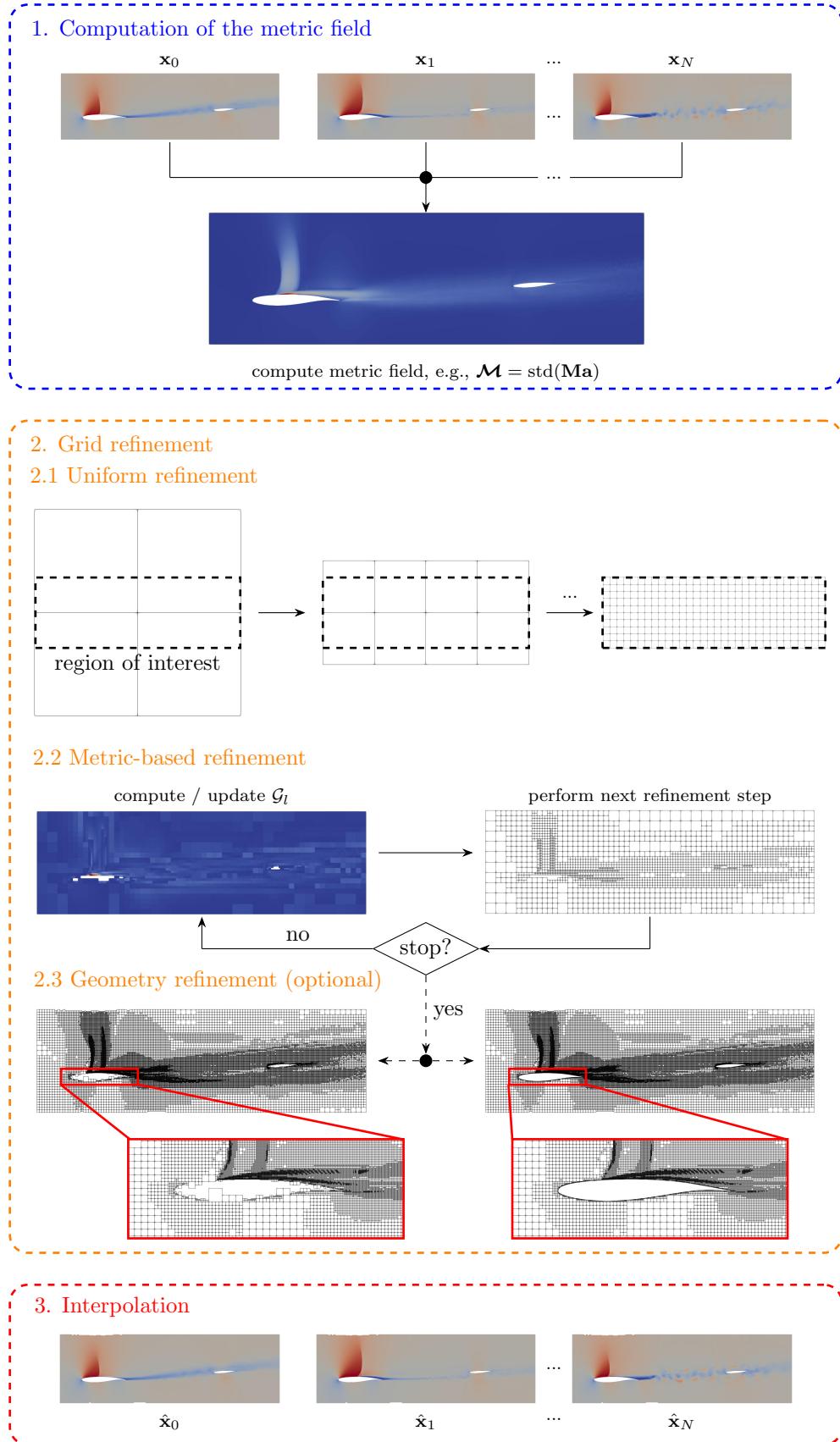


Figure 1: The three main steps of S^3 . The stopping criterion is either the maximum number of leaf cells $N_{\ell,\max}$ or the minimum percentage of the original metric that must be captured. The depicted test case represents a generic tandem configuration of an *ONERA OAT15A* airfoil (front) and a *NACA64A110* airfoil (rear).

is referred to as the root or parent cell. All cells without child cells are called leaf cells and represent the current grid. During each uniform refinement iteration, all available leaf cells are marked for refinement. Since the original numerical domain is seldom a cube, an increasing refinement level inevitably leads to child cells being located outside the original numerical domain. As shown in fig. (1), these invalid cells are removed from the tree to avoid refining cells in regions that are not part of the domain. A child cell is only considered outside the original domain if all 2^D corner points are located outside. The uniform refinement stops after a predefined number of iterations.

Next, the metric-based refinement (step 2.2) follows, where in each refinement iteration, only a subset of the leaf cells is split. Suppose the cell centers of a leaf cell are denoted as $\hat{\mathbf{p}}_j$, $j \in \{1, 2, \dots, 2^D\}$. The cell volume of a cell that results from l subdivisions of the initial cube is:

$$V_l = \frac{1}{2^D} \left(\frac{l_0}{2^{l-1}} \right)^D. \quad (2)$$

The adaptive refinement process requires an estimate of the metric at the cell centers $\hat{\mathbf{p}}_j$. We employ an off-the-shelf K-nearest neighbors (KNN) algorithm to interpolate between the original point cloud and the octree mesh (*scikit-learn* library [18, 3]). Let $\{(\mathbf{p}_k, \mathcal{M}_k)\}_{k=1}^K$ be the set of the K nearest neighbor points of a query point $\hat{\mathbf{p}}_j$ and their corresponding metric values. Then the metric value at the query point can be approximated as:

$$\widehat{\mathcal{M}}_j = \frac{\sum_{k=1}^K \mathcal{M}_k w_k}{\sum_{k=1}^K w_k}, \quad w_k = \frac{1}{\|\mathbf{p}_k - \hat{\mathbf{p}}_j\|}, \quad (3)$$

where we employ inverse distance weighting by default. Motivated by the topology of Cartesian grids, we set $K = 8$ and $K = 26$ for 2D and 3D cases, respectively. Since the original mesh is typically much denser than the sub-sampled mesh, the precise value of K played no significant role in the performed numerical tests.

In contrast to the original S^3 algorithm, the metric value is not used directly to mark leaf cells for refinement. Instead, we estimate by how much the approximation of the original metric field improves when splitting a leaf cell. The improvement of the approximation when splitting a leaf cell at refinement level l is denoted as gain \mathcal{G}_l and defined as:

$$\mathcal{G}_l = \frac{V_{l+1}}{\mathcal{G}_0} \sum_{j=1}^{2^D} |\widehat{\mathcal{M}}_l - \widehat{\mathcal{M}}_{l+1,j}|, \quad (4)$$

Algorithm 1 Improved Sparse Spatial Sampling pseudo algorithm

Require: Metric field \mathcal{M} , points \mathbf{p}_i , tolerance $\mathcal{M}_{\min} \vee$ max. number of leaf cells $N_{\ell,\max}$

- 1: Initialize KNN, sampling tree
- 2: Create root cell covering the computational domain
- 3: Compute reference norm $\|\mathcal{M}\|$
- 4: Perform N_{uni} uniform refinement cycles ▷ creates uniform background mesh
- 5: Evaluate gain \mathcal{G} for all leaf cells using eq. (4)
- 6: **while** $\|\widehat{\mathcal{M}}\| / \|\mathcal{M}\| \leq \mathcal{M}_{\min} \vee N_\ell \leq N_{\ell,\max}$ **do**
- 7: Select N_c cells with largest \mathcal{G} ▷ avoid too large updates of $\widehat{\mathcal{M}}$
- 8: Include neighboring cells with $\Delta l > 1$ ▷ optional constraint
- 9: Refine selected cells
- 10: Update gain \mathcal{G} for all new leaf cells using eq. (4)
- 11: Remove invalid cells from tree
- 12: Update $\|\widehat{\mathcal{M}}\|$ and N_ℓ
- 13: **end while**
- 14: **if** geometry refinement is enabled **then**
- 15: **for** each geometry g **do**
- 16: Identify cells intersecting or near g
- 17: Refine cells until prescribed target level l_{\max} is reached ▷ no re-evaluation of $\mathcal{M}_{\text{approx}}$
- 18: **end for**
- 19: **end if**
- 20: Re-number leaf cells for export and interpolation
- 21: **Output** Generated grid

where we normalize by the gain of the initial cell, \mathcal{G}_0 , for numerical stability. The summation in eq. (4) measures the error reduction resulting from a potential split of the leaf cell and promotes refinement in regions where the metric field undergoes strong spatial variations. In contrast to the original version of S^3 , regions with uniformly high metric are not excessively refined. To quantify how much the local approximation error contributes to the global approximation error in terms of a volume integral over the domain, the error is weighted by the child cells' volume V_{l+1} . The volume weighting also avoids large jumps in the refinement level between neighboring leaf cells, which in turn tends to improve interpolation errors when mapping fields between original and octree mesh. Optionally, the implementation allows limiting the difference in refinement level between neighboring cells to one, which is useful, e.g., when computing gradients on the octree mesh.

Refining only the leaf cell with the highest gain value in each adaptive refinement results in the most efficient octree mesh, but it also leads to an impractically high iteration count. Therefore, the current leaf cells are first sorted from the highest to the lowest gain value, and then the first N_c cells are marked for refinement. N_c may be varied linearly between $N_{c,\text{start}}$ and $N_{c,\text{end}}$, with $N_{c,\text{start}} \geq N_{c,\text{end}}$, to promote a more fine-grained refinement towards the end. By default, we set $N_c = N_{c,\text{start}} = N_{c,\text{end}} = 0.01 M$, which appeared as a reasonable compromise between effective cell placement and computational speed in a variety of numerical tests. As in the uniform refinement, leaf cells that fall outside the domain are removed at the end of each iteration.

Before starting the next iteration, the user-defined stopping criterion is evaluated. The stopping criterion is either the maximum number of leaf cells $N_{\ell,\text{max}}$ or the minimum percentage of the original metric that must be captured. While the first criterion is straightforward to evaluate, the latter is not directly available, since the current number of leaf cells differs from the number of cells in the original mesh. Suppose that $\widehat{\mathcal{M}} \in \mathbb{R}^{N_\ell}$ is a vector holding the approximated metric of all N_ℓ leaf cells. To stop the refinement, we measure the captured metric as the ratio of the vector norms and compare it to a user-defined threshold value, i.e., stop if $\mathcal{M}_{\text{approx}} = \|\widehat{\mathcal{M}}\|/\|\mathcal{M}\| \geq \mathcal{M}_{\min}$. Note that approximation errors in $\widehat{\mathcal{M}}$ may cancel out when comparing the norms. However, we found the impact of this cancellation to be negligible.

Once the metric-based refinement is completed, an optional mesh refinement can be performed near geometries to improve their representation (step 2.3). This additional refinement is advantageous if the metric has only small gradients near geometries, but further post-processing requires a good approximation of the geometry.

Once the refinement is complete, the final leaf cells are extracted from the tree and converted into a suitable CFD mesh description. We denote the extraction as the renumbering (rn) step in the remainder of this manuscript.

Interpolating the original data onto the generated grid marks the final step (step 3). We employ the same KNN interpolation approach as for the metric field. Although not observed in the investigated test cases, the lack of exception handling for missing neighbors near domain boundaries or thin geometries, such as the trailing edge of an airfoil, may lead to spurious values and may necessitate a more careful neighbor selection. Depending on the size of the snapshots and the available memory, the snapshots are either loaded and interpolated all at once, in batches, or snapshot by snapshot. The interpolated snapshots are exported to an HDF5 binary file format. A summary of all steps is shown in pseudo-algorithm (1).

It was generally found that the resulting grid topology is only slightly influenced by the hyperparameters of S^3 , provided they are chosen within reasonable bounds; therefore, results of conducted hyperparameter studies are not included here. A summary of all hyperparameters and their default values can be found in table (1).

Table 1: Hyperparameters and associated default values for S^3 . The parameters $N_{c,\text{start}}$ and $N_{c,\text{end}}$ denote the number of cells selected for refinement at the beginning and at the end of the adaptive refinement, respectively, while M represents the number of cells of the original grid. The default stopping criterion is activated when the current approximation of the metric field $\|\widehat{\mathcal{M}}\|$ relative to the original metric field $\|\mathcal{M}\|$ exceeds the given threshold \mathcal{M}_{\min} .

	N_{uni}	$N_{c,\text{start}}$	$N_{c,\text{end}}$	stopping criterion
default value	5	$0.01 M$	$N_{c,\text{start}}$	$\ \widehat{\mathcal{M}}\ /\ \mathcal{M}\ \geq \mathcal{M}_{\min}$

3 Results

3.1 Airfoil tandem configuration

The first test case is a tandem airfoil configuration in high-speed stall conditions. The data are kindly provided by research partners within the research unit FOR 2895 [21]. A detailed description of the numerical setup is presented in [13, 14]. The leading airfoil within the tandem configuration is a supercritical *ONERA OAT15A* with chord length $c_{\text{front}} = 0.15 \text{ m}$ at an angle of attack of 5° , whereas the rear airfoil is a *NACA64A110* with chord length $c_{\text{rear}} = 0.075 \text{ m}$ placed at a horizontal distance of $2c_{\text{front}}$ behind the leading airfoil's trailing edge. The inflow Mach number is $Ma_\infty = 0.72$, leading to a freestream Reynolds number of $Re_\infty = 2 \cdot 10^6$ based on the chord length of the leading airfoil. Under these conditions, the leading airfoil exhibits periodic shock-boundary layer interactions known as shock buffet. Vortices are generated at the leading airfoil's

trailing edge and in the separating boundary layer. These vortices are propagated through the wake and impact on the rear airfoil.

The original finite volume mesh consists of $16 \cdot 10^6$ grid points in total. The simulation was executed with the DLR TAU code, employing automated zonal detached eddy simulation and an SSG/LRR- ω Reynolds stress model. In contrast to [13], only a two-dimensional plane at a fixed spanwise location is analyzed here. The plane consists of $2.68 \cdot 10^5$ cells. Since the majority of the mesh cells are accumulated in the vicinity of the two airfoils, the region of interest is restricted to $x/c_{\text{front}} \in [-0.5, 4.5]$, $z/c_{\text{front}} \in [-0.5, 1.0]$, where x is the coordinate in streamwise direction, and z is the second plane coordinate normal to x . The origin is located at the leading airfoil's nose; refer to fig. (2b). The region of interest contains $2.456 \cdot 10^5$ cells in total. The dataset comprises 559 snapshots taken at an equidistant time interval of $\Delta t = 4.4 \cdot 10^{-5}$ s. Storing the time sequence of a scalar field occupies approximately 550 MB of disk space (single-precision floating-point numbers). Since shock oscillations and periodic vortex shedding characterize the shock buffet, the metric field for this test case is defined as the temporal standard deviation of the local Mach number computed over all snapshots, $\mathbf{M} = \text{std}(\mathbf{Ma})$.

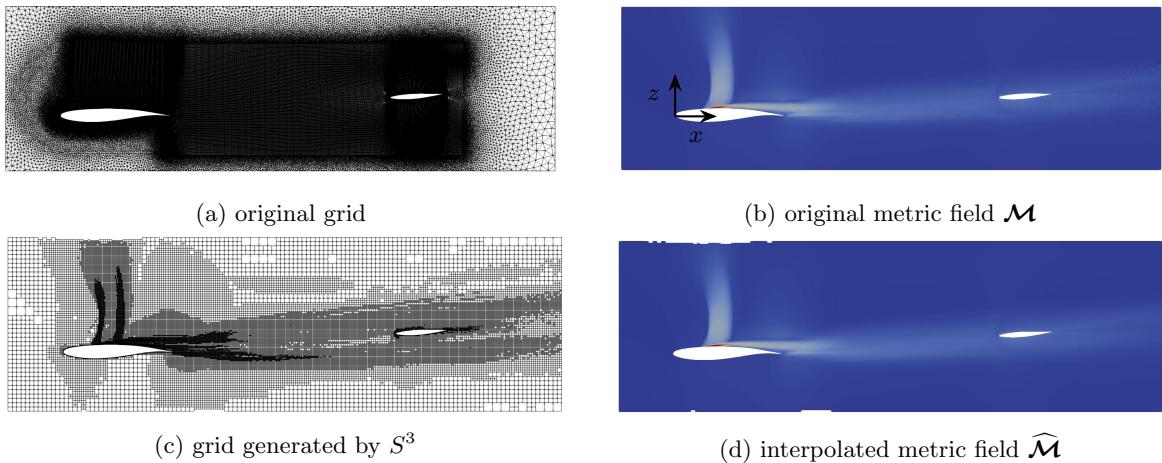


Figure 2: The left column shows a comparison of the original grid (2a) and the grid generated by S^3 for a captured metric of $\mathcal{M}_{\text{approx}} = \|\widehat{\mathbf{M}}\|/\|\mathbf{M}\| = 0.75$ (2c). The right column depicts the metric on the original grid (2b) and the metric interpolated onto the S^3 -grid (2d). Both contour plots are scaled from zero to $\|\mathbf{M}\|_\infty$.

Fig. (2a) shows the original grid within the defined region of interest, along with the metric depicted in fig. (2b). The increased metric in the shock region and the wake of the leading airfoil are clearly visible. The metric reaches its maximum in the region where the boundary layer detaches. The stopping criterion is set to a minimum approximation of the metric, $\mathcal{M}_{\min} = 0.75$. Due to the refinement of multiple cells within a single iteration, discussed in section (2), the mesh generated by S^3 captures $\mathcal{M}_{\text{approx}} = 0.752$. The final mesh, shown in fig. (2c), consists of $5.83 \cdot 10^4$ cells corresponding to a reduction of 76.22%. The progression of $\mathcal{M}_{\text{approx}}$ with respect to the adaptive refinement iterations is shown in the appendix, section A.3 (fig. (14)). We also report detailed timings for the individual refinement steps in section (3.4). It is important to note that for this test case, a subsequent geometry refinement is employed to demonstrate the ability of S^3 to resolve complex geometries properly, which is not included in the captured metric. However, this geometry refinement is not significant, e.g., when computing the SVD. By omitting the geometry refinement, a total reduction of 81.54% can be achieved without further loss of information.

To assess the quality of the interpolated Mach number snapshots $\widehat{\mathbf{Ma}}$, we map the interpolated field back to the original grid using KNN interpolation, denoted as \mathbf{Ma}^* , and compare to the original field \mathbf{Ma} . The overall local interpolation error (original grid $\rightarrow S^3 \rightarrow$ original grid) of the n th snapshot is $\Delta \mathbf{Ma}_n = |\mathbf{Ma}_n^* - \mathbf{Ma}_n|$. The temporal mean and standard deviation of the interpolation error over all snapshots are shown in fig. (3). Both the normalized mean error and the standard deviation are very small, $O(10^{-5})$, with their maximum values located in the wake region aft of the rear airfoil. As we show next, the approximation error is sufficiently small for most post-processing and analysis tasks.

The SVD is a matrix factorization that provides an optimal low-rank representation of data arranged in matrix form. If the sequence of flow states is arranged into a data matrix as a sequence of column vectors, the matrix of left-singular vectors, \mathbf{U} , represents spatial structures (equivalent to the so-called POD modes), while the matrix of right-singular vectors, \mathbf{V} , represents temporal importance. Both sets of vectors are sorted according to the singular values, Σ , from most to least important. We provide a compact mathematical description in section (A.1). Comparing the SVDs computed with the original and reduced data matrices poses a challenging test that validates the suitability of S^3 data for analyzing dominant coherent structures. For the tandem configuration, the data matrix is constructed based on the local Mach number. We tested a variety of threshold values in the range $0.25 \leq \mathcal{M}_{\min} \leq 0.75$ and found a strong robustness of the modal decomposition with respect to the stopping criterion. For brevity, we only present POD modes (left-singular vectors) for the lowest threshold value. Fig. (4)

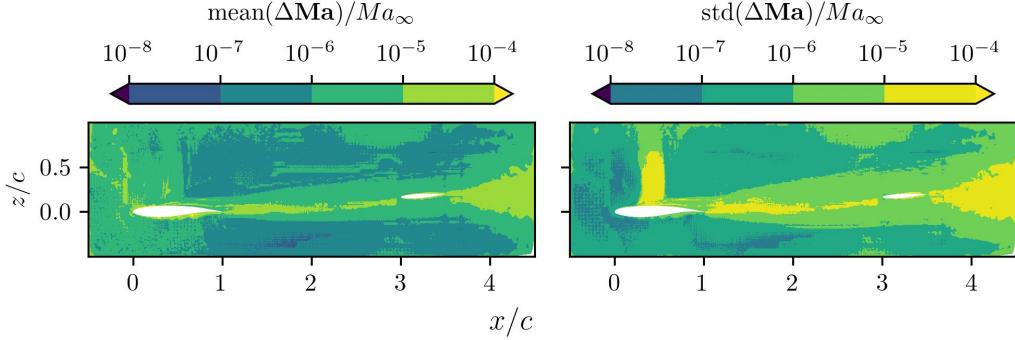


Figure 3: Temporal mean (left) and standard deviation (right) of the absolute spatial error $\Delta\mathbf{Ma}_n = |\mathbf{Ma}_n^* - \mathbf{Ma}_n|$ scaled with the free stream Mach number Ma_∞ for $\mathcal{M}_{\text{approx}} = 0.75$.

compares every second of the first 12 modes and the corresponding singular values σ_i for $\mathcal{M}_{\text{approx}} = 0.27$ (note that $\mathcal{M}_{\text{approx}}$ can be slightly larger than \mathcal{M}_{\min}). We show every second mode, since spatially periodic flow structures give rise to pairs of similar modes. Modes one to four encode the main shock motion and the associated boundary layer separation typical for buffet. Modes four and five represent periodic vortex shedding at the leading airfoil's trailing edge. Higher-order modes show a mixture of these two phenomena. For more details on the physical interpretation, we refer to the literature [23, 13]. Despite the low threshold value, the modes are visually nearly indistinguishable, and the first six singular values differ only in the fourth significant digit. More precisely, the error is around 0.1% for σ_1 and increases to 1.7% for σ_{11} . The low error in the modes and singular values is also reflected in the mode coefficients \mathbf{v}_i (right-singular vectors), shown in fig. (5). The time is non-dimensionalized with the free stream velocity U_∞ and the chord length of the leading airfoil, yielding $\tau = tU_\infty/c_{\text{front}}$. Both phase and amplitude are in good agreement for the tested threshold values, even for higher-order modes (not depicted here).

3.2 Flow past a circular cylinder

A direct numerical simulation (DNS) of the flow past a circular cylinder, in the following legends abbreviated as *cylinder*, was chosen to demonstrate the data reduction abilities of S^3 on a more challenging 3D test case. The simulation setup is based on the work of Lehmkuhl et al. [16]. The Reynolds number based on the cylinder diameter $d = 0.1\text{ m}$, the uniform inflow velocity $U_\infty = 39\text{ m/s}$, and the kinematic viscosity $\nu = 10^{-3}\text{ m}^2/\text{s}$ is $Re = 3900$. The extent of the numerical domain along the streamwise, lateral, and axial directions is $x/d \in [0, 24]$, $y/d \in [0, 20]$, and $z/d \in [0, \pi]$, respectively. The cylinder is placed at $x/d = 8$ and $y/d = 10$. The block-structured mesh consists of $9.416 \cdot 10^6$ hexahedral cells. The incompressible simulation is executed using *OpenFOAM-v2206* [5]. For more information on the numerical setup and the flow physics, we refer to [16]. The numerical setup is publicly available on GitHub [1].

The time interval between two subsequent snapshots is set to $\Delta t = 2.5 \cdot 10^{-4}\text{ s}$, corresponding to $\Delta\tau = \Delta t U_\infty/d = 0.0975$ convective time units (CTU), and a total number of 500 snapshots in the range $75 \leq \tau \leq 124$ is collected, corresponding to roughly 50 CTU. The selected time interval only extends over 11 vortex shedding cycles. This limitation stems from the available memory on a single node of our HPC cluster when computing the SVD of the original CFD data, which is needed for validation. We define the state vector using the instantaneous velocity fluctuations in all three directions. The full data matrix occupies 120 GB on the hard drive when stored in double-precision binary format.

For this test case, we use the TKE as the metric field, which is defined as:

$$\mathbf{k} = \frac{1}{2} \left(\overline{(\mathbf{u}'_x)^2} + \overline{(\mathbf{u}'_y)^2} + \overline{(\mathbf{u}'_z)^2} \right), \quad \overline{(\mathbf{u}'_i)^2} = \overline{(\mathbf{u}_i - \bar{\mathbf{u}}_i)^2}, \quad (5)$$

where the overbar denotes temporal averaging and \mathbf{u}_i is the i th component of the velocity vector field.

Fig. (6a) shows the original grid in the x - y -plane at $z/d = \pi/2$ along with \mathbf{M} in fig. (6b). It is clearly visible that the major part of the TKE is located in the wake of the cylinder, while its magnitude decreases in a streamwise direction. The generated coarse grid consists of $2.188 \cdot 10^5$ cells, corresponding to a reduction rate of 97.68% when using $\mathcal{M}_{\min} = 0.25$ as the stopping criterion ($\mathcal{M}_{\text{approx}} = 0.27$). A slice through the resulting grid is depicted in fig. (6c). Clearly, S^3 strongly refines the near wake regions where gradients of the target metric are high, while creating a coarse grid in regions with low TKE. Regions with uniformly high metric values are not excessively refined, e.g., at a distance of one to three diameters downstream of the cylinder. As expected, the new grid is roughly symmetric with respect to the horizontal dividing line through the cylinder. Small asymmetries may occur because of the limited accuracy of the sampled TKE and the empirically chosen number of leave cells marked for refinement in each iteration. S^3 is able to capture the TKE without a qualitative deterioration of its

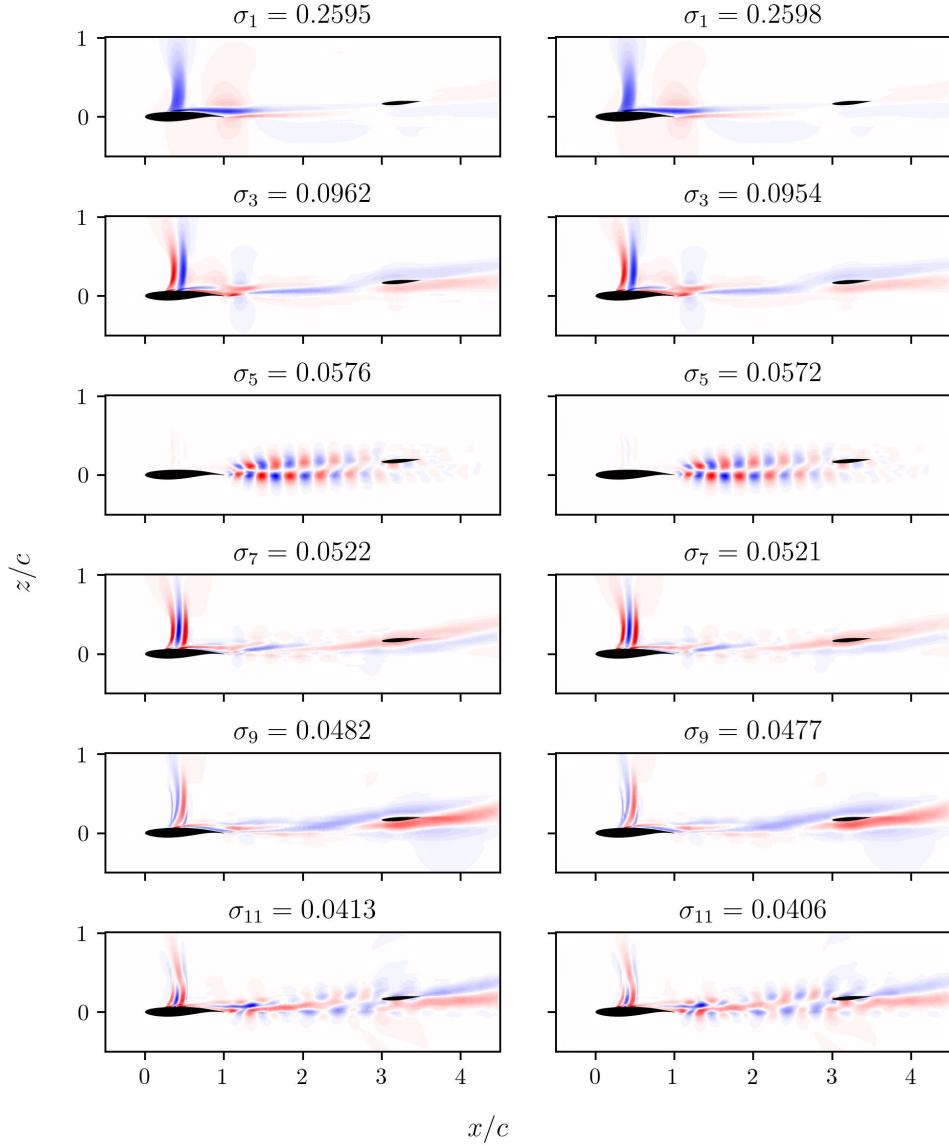


Figure 4: Comparison of the leading POD modes and the associated singular values for the tandem configuration. The left column shows the POD modes on the original grid, while the right column depicts the POD modes on the grid generated with S^3 and $M_{\text{approx}} = 0.27$. The colorscale is identical for all contours and bounded by $\pm \|\mathbf{U}\|_\infty$.

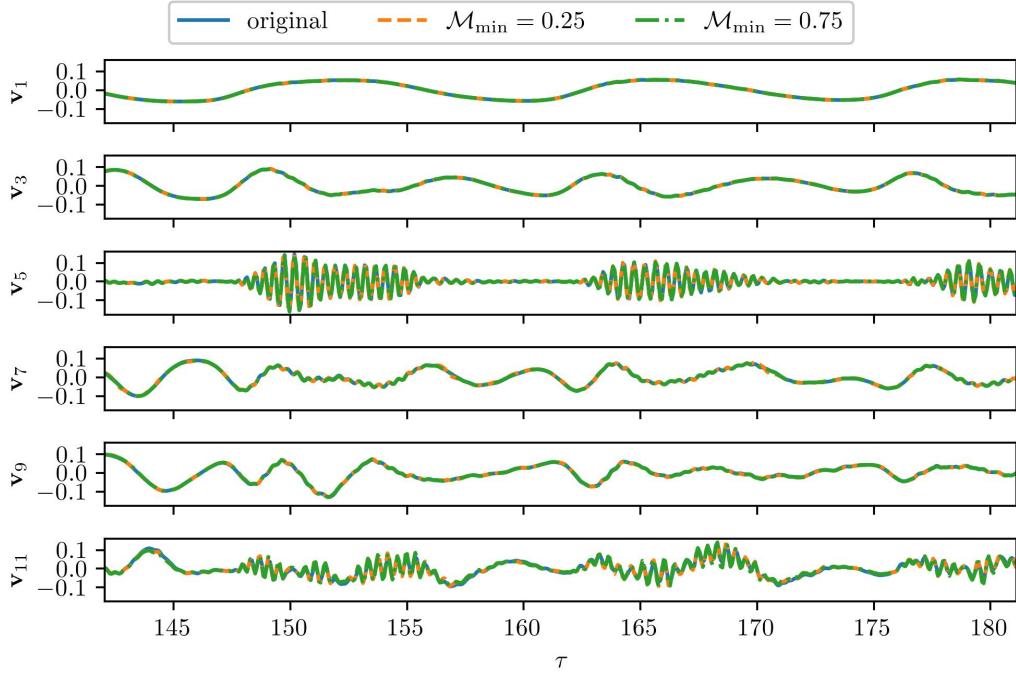


Figure 5: Comparison of the leading right-singular vectors for the tandem configuration. As for the modes, only every second mode coefficient is shown.

approximation, as shown in fig. (6d). Fig. (7c) shows a second x - z -slice at $y/d = 8$. The grid is approximately homogeneous in z -direction, and the TKE is captured well on the coarse grid. It is also visible that the domain boundaries in the axial direction are not captured uniformly well due to differences in the refinement level. This behavior could be avoided by an additional geometry refinement step for the outer boundary, which we did include here. The same applies to the cylinder boundary, which is only captured roughly. Comparable results for $\mathcal{M}_{\min} = 0.75$ are available in section (A.3).

For a quantitative assessment of the reduced data, we build the data matrix using the velocity fluctuations in all three directions and compute the SVD. Fig. (8) shows the first four uneven left-singular vectors and the associated singular values. The modes' z -component is omitted for compactness, but the agreement is equally good as for the other two components. The difference in the singular values is slightly larger than for the tandem configuration, but the deviation for the depicted modes remains below 1%.

The good agreement between the original and reduced data is also reflected in the right-singular vectors in fig. (9). While minor deviations are visible for $\mathcal{M}_{\min} = 0.25$, the results are visually indistinguishable when capturing roughly two-thirds of the metric. The higher threshold value still leads to an 85% reduction in the cell count. The influence of the threshold value on the cell count and the associated computational cost is discussed in section (3.4). Overall, it is fair to say that the same conclusions can be drawn from the SVDs of the original and reduced data.

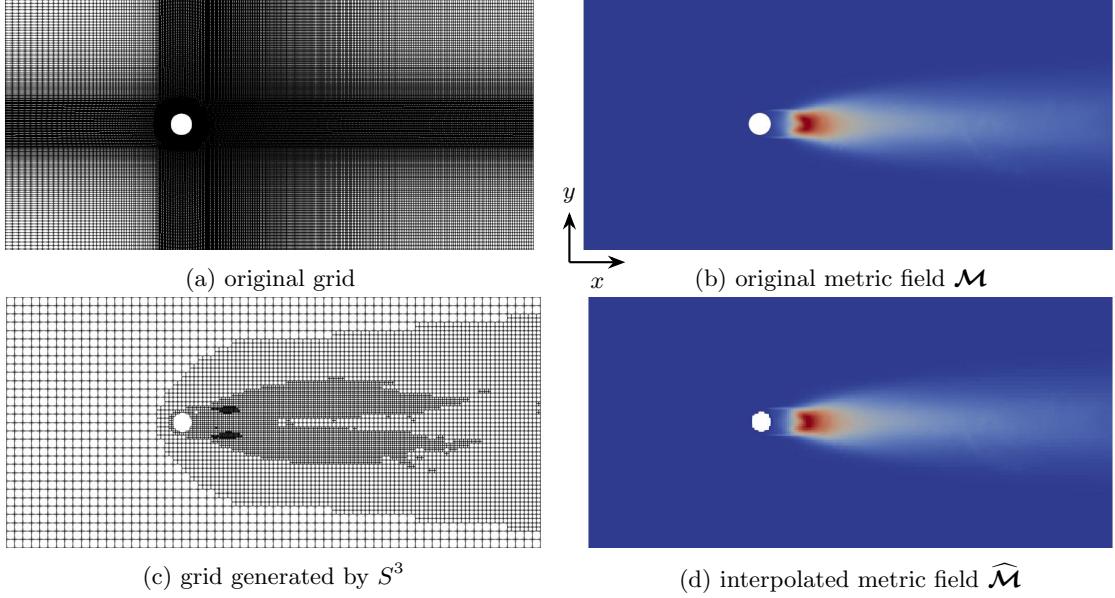


Figure 6: The left column shows a comparison of the original grid (6a) with the grid generated by S^3 for a captured metric of $\mathcal{M}_{\text{approx}} = 0.27$ (6c). The right column depicts the metric on the original grid (6b), and the metric interpolated onto the grid created by S^3 (6d). All figures depict the x - y -plane at $z/d = \pi/2$. Both contour plots are scaled from zero to $\|\mathcal{M}\|_\infty$.

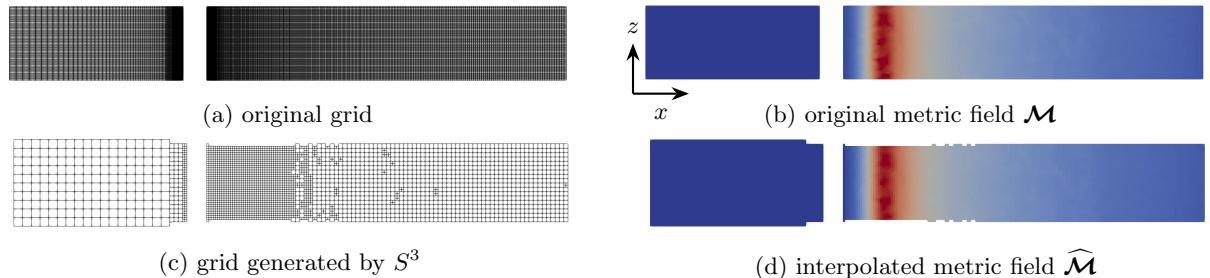


Figure 7: The left column shows a comparison of the original grid (7a) with the grid generated by S^3 for a captured metric of $\mathcal{M}_{\text{approx}} = 0.27$ (7c). The right column depicts the metric on the original grid (7a), and the metric interpolated onto the grid created by S^3 (7d). All figures depict the x - z -plane at $y/d = 8$. Both contour plots are scaled from zero to $\|\mathcal{M}\|_\infty$.

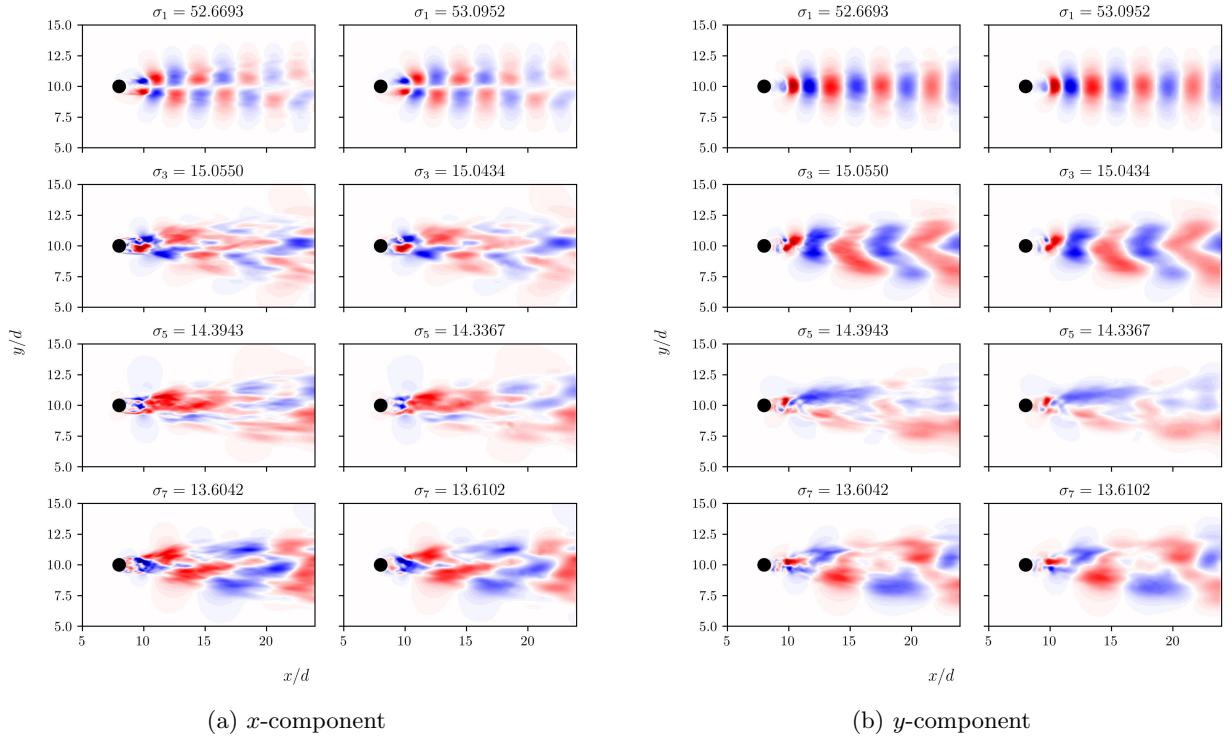


Figure 8: Comparison of the first four uneven POD modes and associated singular values for the original data (left columns) and the S^3 -interpolated data with $\mathcal{M}_{\text{approx}} = 0.27$ (right columns) in the x - y -plane at $z/d = \pi/2$ for the cylinder test case. The colorscale is identical for all contours and bounded by $\pm \|\mathbf{U}\|_\infty$.

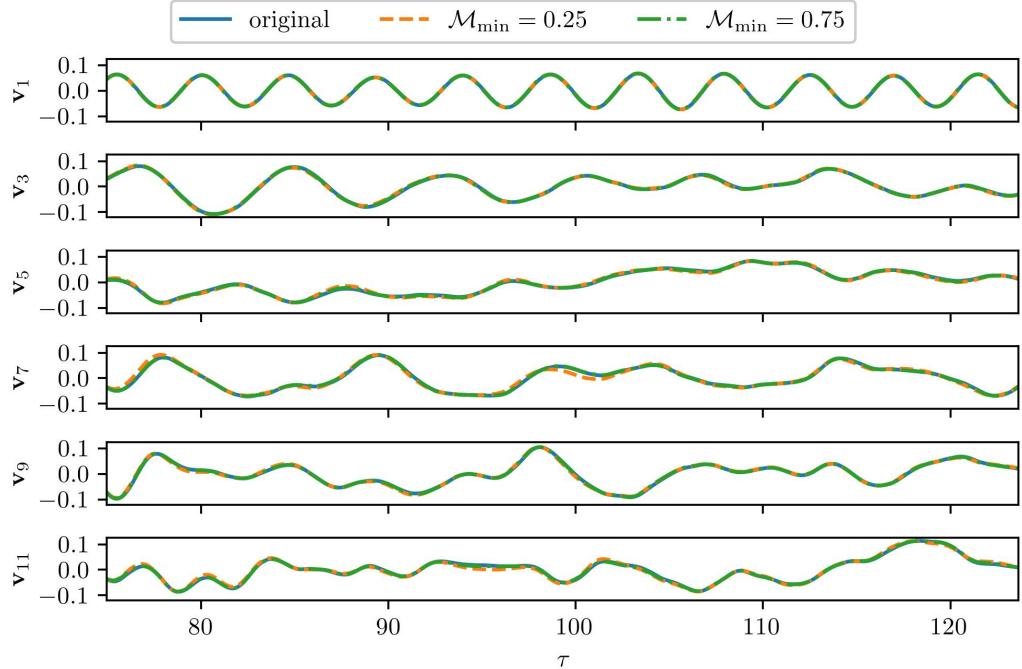


Figure 9: Comparison of the first four uneven right-singular vectors for the cylinder test case.

3.3 Flow around an aircraft

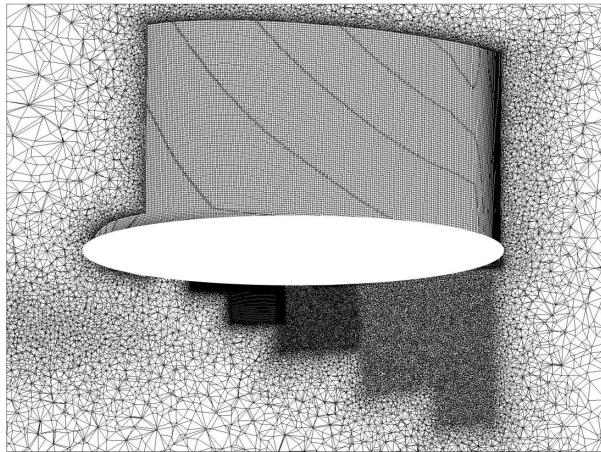
We finally consider the flow around an aircraft half-model to demonstrate the potential of S^3 to efficiently reduce extremely large amounts of data. An improved delayed detached eddy simulation (IDDES) of the XRF-1 wind tunnel model was carried out by Spinner et al. [20] at $Re = 3.3 \cdot 10^6$, $Ma_\infty = 0.84$, and an angle of attack of $\alpha = -4^\circ$. The simulation was executed using the DLR TAU code with an SSG/LRR $\ln\omega$ turbulence model acting as background RANS model in the hybrid RANS-LES simulation. A detailed description of the numerical setup and methods is provided in [20].

A total number of 64 snapshots are used for the analysis, representing approximately 9 CTU with $CTU = MAC/U_\infty$ (MAC - mean aerodynamic chord). The computational mesh consists of $M = 7.97 \cdot 10^8$ cells, occupying 5.9 GB of disk space per snapshot and field, and leading to a size of roughly 380 GB for the data matrix based on the local Mach number. The metric field is composed of the temporal standard deviation of the Mach number field and eddy viscosity, respectively, and is computed using all of the 64 snapshots. Both parts are weighted, leading to a final metric field of $\mathcal{M} = 0.6 \cdot \text{std}(\mathbf{Ma}) + 0.4 \cdot \text{std}(\boldsymbol{\nu}_t)$. As for the tandem configuration, the Mach number variation ensures a high resolution in regions of shock motion. The addition of the eddy viscosity to the metric field accounts for large separated, turbulent structures convected into the far field, where temporal changes of the local Mach number are small. The weighting factor expresses a mild prioritization of shock over far-field unsteadiness. We did not study the sensitivity of the resulting mesh to the weighting factor, but we assume its influence to be minor. The region of interest for the analysis was reduced to $x/MAC \in [-2.5445, 12.7226]$, $y/MAC \in [0.0509, 5.089]$ and $z/MAC \in [-2.5445, 2.5445]$ with $MAC = 0.1965 \text{ m}$ containing a total number of $M = 7.87 \cdot 10^8$ cells. The nose of the aircraft is located at $x/MAC \approx 0.9$, $y/MAC = 0$ and $z/MAC = 0$.

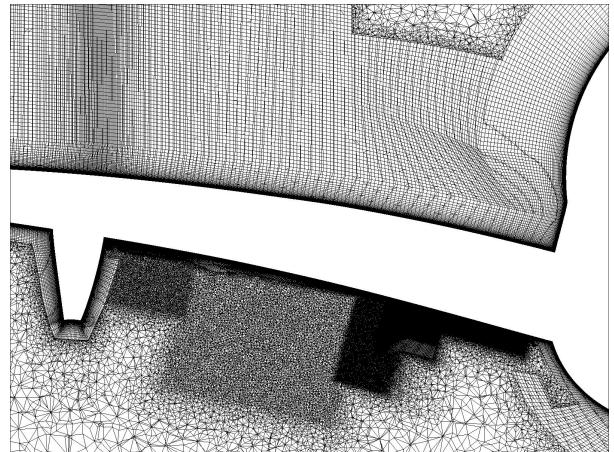
Fig. (10a) shows an enlarged view of the original grid in the x - z -plane at $y/MAC = 0.9274$ while fig. (10b) depicts the original grid in the y - z -plane at $x/MAC \approx 4.9$. Due to the enormous snapshot size, we computed the metric field incrementally using Welford's algorithm [25].

To achieve a high compression ratio while preserving the dominant flow dynamics, a maximum number of $50 \cdot 10^6$ cells is set as a stopping criterion for S^3 . Due to the refinement of multiple cells per iteration, the final grid generated has $N_\ell = 50.988 \cdot 10^6$. The cell count is reduced by 93.52% while $\mathcal{M}_{\text{approx}} = 0.138$ of the metric are captured.

The resulting grid is shown in fig. (11a) and fig. (11b) in the same planes as for the original grid, along with the corresponding metric field in fig. (11c) and fig. (11d). It is clearly visible that the grid is refined in regions where the spatial change of the metric is high, as presented in section (2). Further, the approximation of the geometry is rather poor compared to the previously shown test cases. No subsequent geometry refinement was employed to keep the number of cells to a minimum. Since the first modes typically capture large, dominant flow structures, refining the grid near the aircraft model has a minor impact on the SVD results and can therefore be omitted here. Fig. (11a) further shows cells located inside the geometry, which should have been marked as invalid and removed during the refinement process. A possible reason why these cells are not removed is a compression step of the STL file representing the aircraft half-model, carried out before starting the refinement process. This step was performed to decrease the number of triangles within the STL file to accelerate the refinement process, but it may lead to cavities and poor representation of the original geometry in regions with high curvature. Therefore, cells in these regions, although invalid when compared to the original geometry, may be considered valid when using the coarsened geometry.

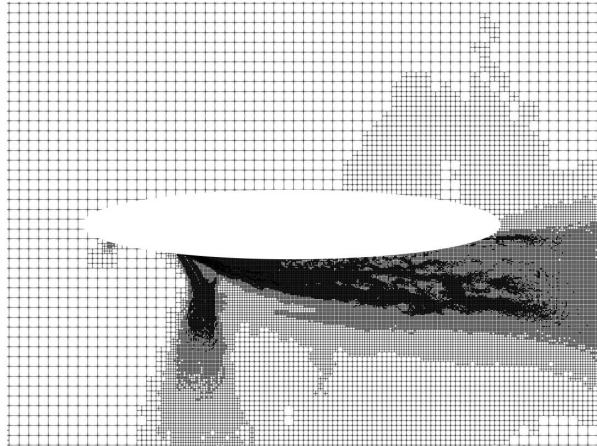


(a) original grid in the x - z -plane

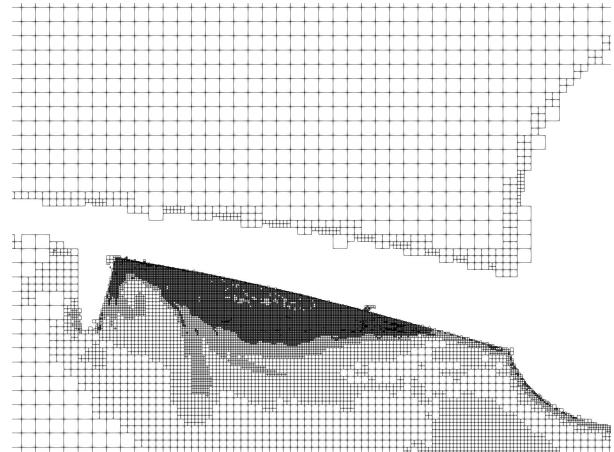


(b) original grid in the y - z -plane

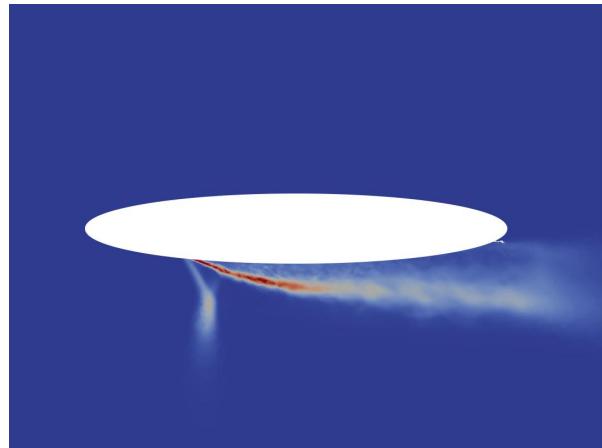
Figure 10: Original grid in the x - z -plane at $x/MAC \approx 4.9$ (10a) and in the y - z -plane at $y/MAC \approx 0.9$ (10b). The airfoil geometry is proprietary to Airbus and therefore redacted in fig. (10a).



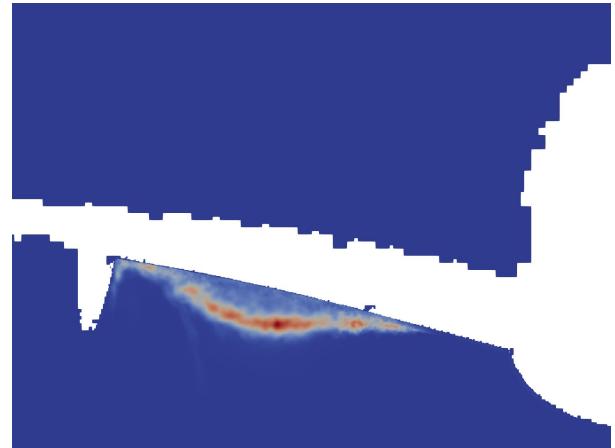
(a) grid generated by S^3 in the x - z -plane



(b) grid generated by S^3 in the y - z -plane



(c) interpolated metric field in the x - z -plane



(d) interpolated metric field in the y - z -plane

Figure 11: Grid generated by S^3 in the x - z -plane at $x/\text{MAC} \approx 4.9$ (11a) and the y - z -plane at $y/\text{MAC} \approx 0.9$ (11b). The lower row shows the interpolated metric field in the same planes. The airfoil geometry is proprietary to Airbus and therefore redacted in fig. (11a) and (11c).

Reducing the grid size enabled the SVD computation using the volume data, which is not feasible for the original data without a dedicated distributed-memory parallel algorithm. Hence, only SVD results obtained on the reduced data are presented. Fig. (12) shows isocontour plots of the first and second POD modes. The modes clearly show the connection between shock oscillation and flow separation downstream of the shock.

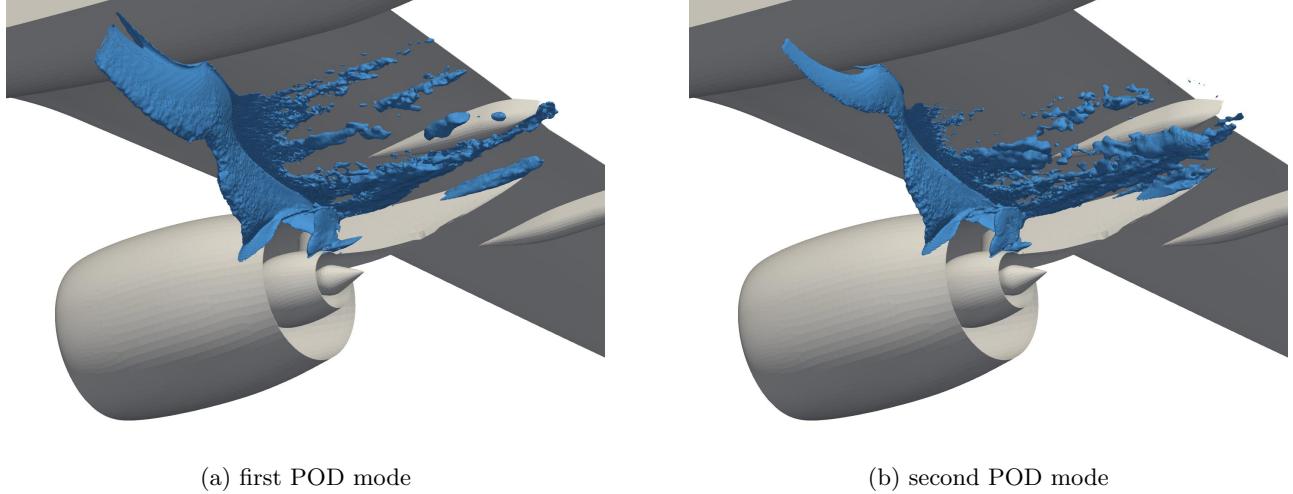


Figure 12: Isocontours of the first two POD modes based on the Mach number field for the grid generated by S^3 . The threshold values for the isocountours are ± 30 .

Due to the large number of cells generated, this test case could not be executed on a local machine. Instead, both the grid generation and SVD were performed on an HPC cluster using 128 CPU cores. The overall execution time for S^3 is $t_{\text{tot}} = 6.43 h$, which is composed of the uniform refinement ($t_{\text{uni}} = 202 s$), metric-based refinement ($t_a = 6.22 h$) and renumbering of the leaf cells to create the final grid ($t_{\text{rn}} = 539 s$).

3.4 Benchmark and comparison of the test cases

In this section, we present detailed timing for individual steps of the mesh generation process and how these timings depend on the metric threshold value. Due to the size of the half-model simulation, we conducted these tests only for the tandem configuration and the cylinder flow. All benchmarks are performed on a laptop equipped with an *Intel® Core™ i7-11800H* with 8 physical cores and 32 GB of RAM. We did not repeat the tests multiple times, but made sure that the impact of other processes remained marginal.

Fig. (13) depicts the number of cells as well as the composition of the execution times for varying $\mathcal{M}_{\text{approx}}$. Although the number of cells increases with an improved approximation of the metric, reduction rates of 65% for the tandem configuration and 76% for the cylinder are achievable even with $\mathcal{M}_{\text{approx}} = 1$. It should be noted that the choice of the metric significantly influences the overall mesh reduction and must be done carefully.

The total execution time (t_{tot}) is composed of the uniform refinement (t_{uni}), metric-based refinement (t_a), geometry refinement (t_g), and the final mesh renumbering (t_{rn}). From fig. (13), it can be seen that adaptive refinement significantly dominates the overall execution time for larger mesh sizes. The geometry refinement is the second most expensive part of the algorithm and depends on the geometry itself and the maximum refinement level. For the cylinder flow, the relative contribution to the overall meshing time is relatively constant. For the more complex airfoil geometry, the relative contribution varies in the range 10 – 30%. We note once more that the geometry refinement is optional and not required in many cases. The mesh renumbering at the end of the refinement process scales approximately linearly with the number of leaf cells, while the influence of the uniform refinement is negligible. Since the latter quantities contribute only a minor part to the overall execution time, they are summarized in fig. (13) as *other*. The physical execution times for $\mathcal{M}_{\text{approx}} = 1$ remained in the order of one minute for the tandem test case, whereas the grid generation of the cylinder test required up to 23 minutes on the local workstation. Both overall execution times refer to benchmarks employing a subsequent geometry refinement. The mapping between original and new meshes is not included in the timing. The interpolation and export of the flow fields highly depend on the available RAM, which limits the maximum number of snapshots that can be processed in parallel. However, loading, interpolation onto the generated grid, and export to HDF5 require approximately the same time as the mesh generation on a local machine for the investigated cases. These timings emphasize that incorporating S^3 into an existing post-processing pipeline results in minimal additional computational overhead relative to the overall execution time of the CFD simulation. Using coarse meshes from S^3 enables additional post-processing steps to be executed on a local machine, thus avoiding the need for HPC resources in some cases.

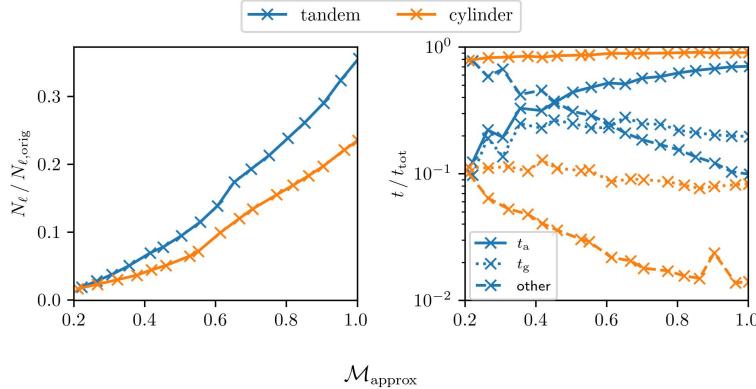


Figure 13: Cell count progression with respect to the captured metric (left) and composition of the meshing times (right) for the tandem configuration and the cylinder flow. The timing distinguishes between adaptive refinement (t_a), geometry refinement (t_g), and the remaining steps, i.e., uniform refinement and renumbering (*other*).

4 Conclusion

The present article introduced an improved version of S^3 for efficiently downsampling CFD data for post-processing. This new version was tested successfully on the scale-resolving simulation of the flow past a tandem configuration of airfoils in the transonic regime, the incompressible turbulent flow past a circular cylinder, and the flow around an aircraft half-model. S^3 efficiently decreased the mesh size for all cases, while preserving a user-provided metric. The SVD performed to investigate the information loss caused by S^3 revealed negligible differences for both test cases. Parameter studies further showed achievable overall mesh reductions between 35%...98%, depending on the metric and stopping criteria, providing the possibility to execute post-processing steps on a local machine rather than using HPC resources.

Although we presented S^3 as a post-processing tool, it can also be applied directly during the runtime of a simulation. One possible approach would be to generate the coarse grid after the initial transient or warm-up phase of the simulation. Once the S^3 mesh is available, new snapshots created during the productive simulation run can be directly interpolated onto the coarse grid, enabling, e.g., smaller write intervals.

There are still some limitations for S^3 when dealing with large amounts of data. The most significant limitation is the lack of support for distributed parallelism across computing nodes on an HPC system. The limited memory per node can become an issue when generating large meshes in the order of $\mathcal{O}(10^8)$ or larger. One possible solution to this problem is to remove unused nodes from the tree when generating the grid. Currently, the unused nodes are only discarded when exporting the final leaf cells. As presented in section (2), S^3 utilizes a recursive tree to store and address the mesh cells internally. An intermediate pruning of the tree would decrease the memory requirements to the point where support for distributed parallelism may be redundant.

Since S^3 is designed to compress large amounts of volume data, it is unsuited for surface data applications. In this case, the octree mesh cannot accurately represent the surface contour, leading to high interpolation errors when exporting the flow fields onto the grid created by S^3 . These errors are caused by the KNN interpolation and may be pronounced near thin geometries, such as the trailing edge of an airfoil. However, for the test cases shown here, the error remains within acceptable bounds.

Acknowledgment

The authors gratefully acknowledge the Deutsche Forschungsgemeinschaft DFG (German Research Foundation) for funding this work in the framework of the research unit FOR 2895 under the grant number 406435057. The authors also gratefully acknowledge the computing time made available to them on the high-performance computer at the NHR Center of TU Dresden. This center is jointly supported by the Federal Ministry of Education and Research and the state governments participating in the NHR (www.nhr-verein.de/unsere-partner).

References

- [1] Andre Weiner and Janis Geise. *Common flow problems for modal analysis implemented in OpenFOAM*. URL: https://github.com/AndreWeiner/flow_data.

- [2] P. J. Baddoo, B. Herrmann, B. J. McKeon, and S. L. Brunton. “Kernel Learning for Robust Dynamic Mode Decomposition: Linear and Nonlinear Disambiguation Optimization (LANDO)”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 478.2260 (2022), p. 20210830. DOI: 10.1098/rspa.2021.0830.
- [3] P. Cunningham and S. J. Delany. “k-Nearest Neighbour Classifiers: 2nd Edition (with Python examples)”. In: *ACM Computing Surveys* 54.6 (2022), pp. 1–25. DOI: 10.1145/3459665.
- [4] K. Duwe, J. Lüttgau, G. Mania, J. Squar, A. Fuchs, M. Kuhn, E. Betke, and T. Ludwig. “State of the Art and Future Trends in Data Reduction for High-Performance Computing”. In: *Supercomputing Frontiers and Innovations* 7.1 (2020). Number: 1, pp. 4–36. DOI: 10.14529/jsfi200101.
- [5] ESI Group. *OpenFOAM version 2206*. 2206. URL: <https://www.openfoam.com/news/main-news/openfoam-v2206>.
- [6] D. Fernex, A. Weiner, B. Noack, and R. Semaan. “Sparse Spatial Sampling: A mesh sampling algorithm for efficient processing of big simulation data”. In: *AIAA Scitech 2021 Forum*. AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, 2021. DOI: 10.2514/6.2021-1484.
- [7] J. Geise and A. Weiner. *Git repository accompanying the article*. 2024. URL: <https://github.com/JanisGeise/sparseSpatialSampling>.
- [8] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.
- [9] B. N. Hanna, N. T. Dinh, R. W. Youngblood, and I. A. Bolotnov. “Machine-learning based error prediction approach for coarse-grid Computational Fluid Dynamics (CG-CFD)”. In: *Progress in Nuclear Energy* 118 (2020), p. 103140. DOI: 10.1016/j.pnucene.2019.103140.
- [10] International Energy Agency. *Electricity 2024 - Analysis and forecast to 2026*. IEA. 2024.
- [11] A. Kiener, S. Langer, and P. Bekemeyer. “Data-driven correction of coarse grid CFD simulations”. In: *Computers & Fluids* 264 (2023), p. 105971. DOI: 10.1016/j.compfluid.2023.105971.
- [12] R. Klein, G. Liebich, and W. Strasser. “Mesh reduction with error control”. In: *Proceedings of Seventh Annual IEEE Visualization ’96*. Proceedings of Seventh Annual IEEE Visualization ’96. 1996, pp. 311–318. DOI: 10.1109/VISUAL.1996.568124.
- [13] J. Kleinert, M. Ehrle, A. Waldmann, and T. Lutz. *Wake Tail Plane Interactions for a Tandem Wing Configuration in High-Speed Stall Conditions*. 2023. DOI: 10.1007/s13272-023-00670-1.
- [14] J. Kleinert, J. Stober, and T. Lutz. “Numerical simulation of wake interactions on a tandem wing configuration in high-speed stall conditions”. In: *CEAS Aeronautical Journal* 14.1 (2023), pp. 171–186. DOI: 10.1007/s13272-022-00634-x.
- [15] J. Lee, K. S. Jung, Q. Gong, X. Li, S. Klasky, J. Chen, A. Rangarajan, and S. Ranka. *Machine Learning Techniques for Data Reduction of CFD Applications*. 2024. DOI: 10.48550/arXiv.2404.18063.
- [16] O. Lehmkühl, I. Rodríguez, R. Borrell, and A. Oliva. “Low-frequency unsteadiness in the vortex formation region of a circular cylinder”. In: *Physics of Fluids* 25.8 (2013), p. 085109. DOI: 10.1063/1.4818641.
- [17] C. Lorsung and A. Barati Farimani. “Mesh deep Q network: A deep reinforcement learning framework for improving meshes in computational fluid dynamics”. In: *AIP Advances* 13.1 (2023), p. 015026. DOI: 10.1063/5.0138039.
- [18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, and G. Louppe. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2012).
- [19] S. Shumilin, A. Ryabov, N. Yavich, E. Burnaev, and V. Vanovskiy. “Self-Supervised Coarsening of Unstructured Grid with Automatic Differentiation”. In: Forty-first International Conference on Machine Learning. 2024.
- [20] S. Spinner, R. Rudnik, M. Herr, A. Probst, and R. Radespiel. “Scale Resolving Simulation of Buffet Effects Induced by Ultrahigh Bypass Ratio Nacelle Installation”. In: *Journal of Aircraft* 62.3 (2025). Publisher: American Institute of Aeronautics and Astronautics .eprint: <https://doi.org/10.2514/1.C038119>, pp. 551–563. DOI: 10.2514/1.C038119.
- [21] *Unsteady flow and interaction phenomena at High Speed Stall conditions, research unit FOR 2895*. URL: <https://www.for2895.uni-stuttgart.de/>.

- [22] A. Weiner and R. Semaan. “flowTorch - a Python library for analysis and reduced-order modeling of fluid flows”. In: *Journal of Open Source Software* 6.68 (2021), p. 3860. DOI: [10.21105/joss.03860](https://doi.org/10.21105/joss.03860).
- [23] A. Weiner and R. Semaan. “Robust Dynamic Mode Decomposition Methodology for an Airfoil Undergoing Transonic Shock Buffet”. In: *AIAA Journal* 61.10 (2023). Publisher: American Institute of Aeronautics and Astronautics, pp. 4456–4467. DOI: [10.2514/1.J062546](https://doi.org/10.2514/1.J062546).
- [24] J. Weiss. “A Tutorial on the Proper Orthogonal Decomposition”. In: *AIAA Aviation 2019 Forum*. AIAA AVIATION Forum. American Institute of Aeronautics and Astronautics, 2019. DOI: [10.2514/6.2019-3333](https://doi.org/10.2514/6.2019-3333).
- [25] B. P. Welford. “Note on a Method for Calculating Corrected Sums of Squares and Products”. In: *Technometrics* 4.3 (1962). Publisher: ASA Website, pp. 419–420. DOI: [10.1080/00401706.1962.10490022](https://doi.org/10.1080/00401706.1962.10490022).

A Appendix

A.1 Singular value decomposition

This section summarizes the fundamentals of the singular value decomposition, which is used in section 3 to compare the results produced by S^3 to the original data. First, the state vectors \mathbf{x}_n at time steps $n = 1, \dots, N$ are organized in a data matrix $\mathbf{X} \in \mathbb{R}^{\tilde{M}N_{\text{var}} \times N}$:

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_n \\ | & | & & | \end{bmatrix}. \quad (6)$$

The state vector is typically composed of N_{var} flow variables known at \tilde{M} discrete locations, where $\tilde{M} = M$ for the original mesh and $\tilde{M} = N_\ell$ for the S^3 mesh. To reduce the mesh dependency of the SVD computation, it is common practice to weigh each entry in the state vector with the square root of the corresponding cell volume [23]. Moreover, the temporal (rowwise) mean is subtracted from each column. The economy SVD of \mathbf{X} reads:

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T, \quad (7)$$

with the matrix of left-singular vectors $\mathbf{U} \in \mathbb{R}^{\tilde{M}N_{\text{var}} \times N}$, the matrix of right-singular vectors $\mathbf{V} \in \mathbb{R}^{N \times N}$, and the diagonal matrix containing the singular values $\Sigma \in \mathbb{R}^{N \times N}$. All triples of left-singular vector, singular value, and right-singular vector are ordered according to the singular values in descending order. The left-singular vectors contain spatial patterns (modes), while the right-singular vectors denote the contribution of each mode over time. For the mode visualization, the volume weighting is reversed. For a more comprehensive overview of SVD fundamentals, we refer to [24, 8]. The SVD is computed with the open-source flow analysis package *flowTorch* [22].

A.2 OAT tandem configuration

Fig. (14) shows the approximation of $\mathcal{M}_{\text{approx}}$ over the course of the metric-based refinement for both the tandem configuration and the cylinder flow. The approximation improves significantly within the first few iterations, since the dominant large-scale structures present in both test cases can be represented with relatively few cells. When the refinement process continues, the improvement with respect to the approximation of $\mathcal{M}_{\text{approx}}$ converges to a linear behavior. Since $M_{\text{tandem}} \ll M_{\text{cylinder}}$ results in a smaller $N_{\text{c,start}}$ for the tandem configuration, more iteration are required to reach the same target value of $\mathcal{M}_{\min} = 0.75$.

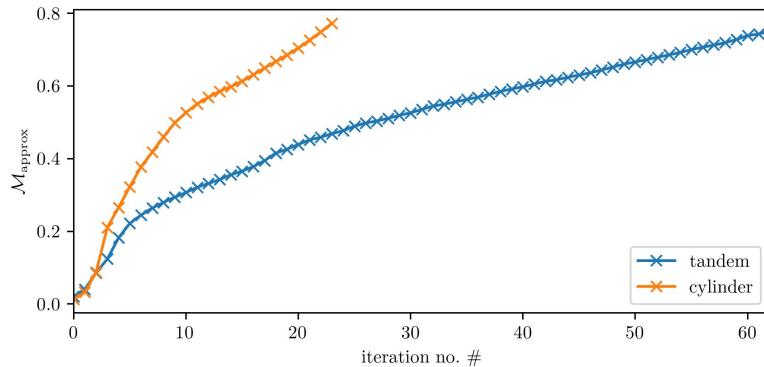


Figure 14: Convergence behavior of the metric-based refinement for the tandem configuration and the cylinder flow using $\mathcal{M}_{\min} = 0.75$.

A.3 Flow past a circular cylinder

Fig. (15a) and fig. (15b) shows slices through the grid generated by S^3 for $\mathcal{M}_{\text{approx}} = 0.53$ and $\mathcal{M}_{\text{approx}} = 0.77$, respectively. The grid remains unchanged in the outer regions of the domain, whereas the refinement level of the wake increases with increasing approximation of the metric. The overall symmetrical shape of the refinement regions in the wake of the cylinder is preserved for larger threshold values of \mathcal{M}_{\min} .

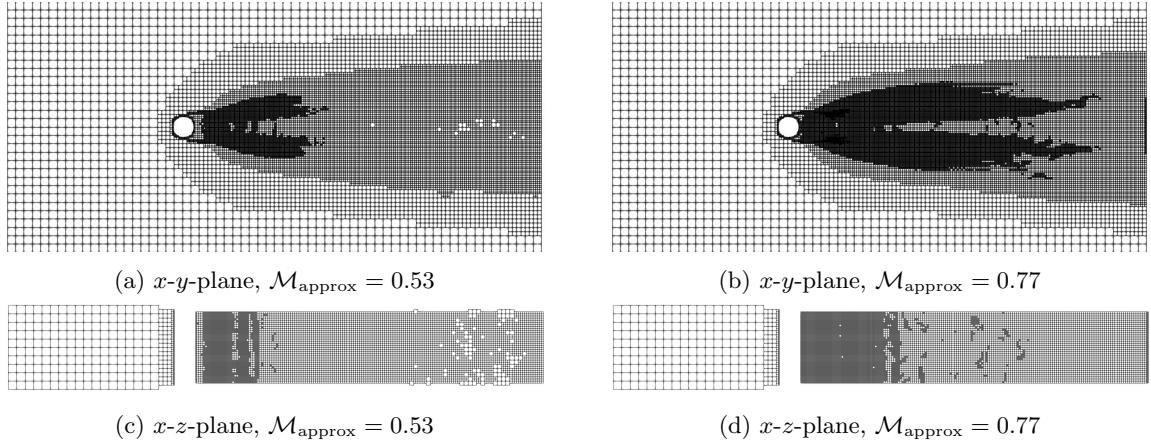
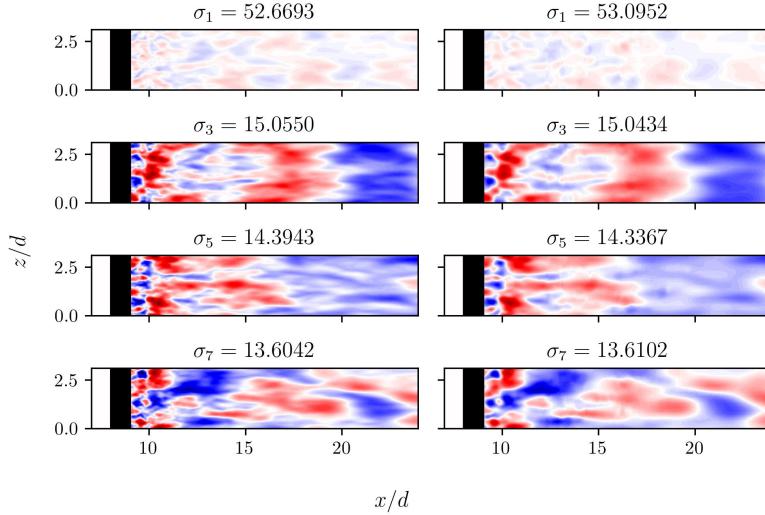
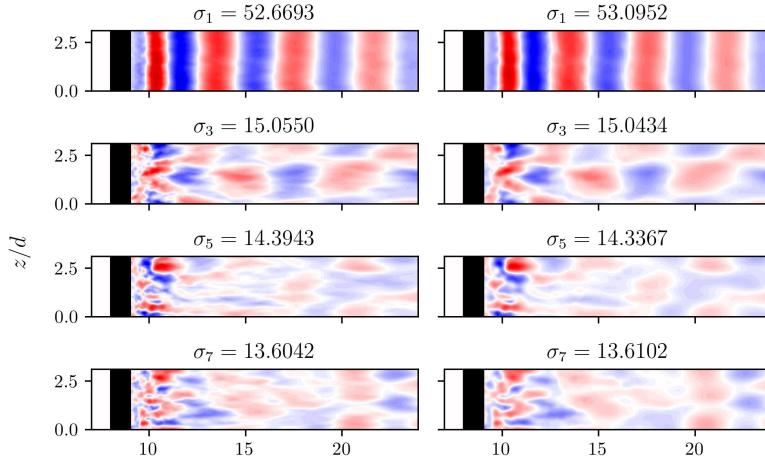


Figure 15: The left column shows the grid generated by S^3 for a captured metric of $\mathcal{M}_{\text{approx}} = 0.53$ in the x - y -plane at $z/d = \pi/2$ (15a) and x - z -plane (15c) at $y/d = 8$. The right column depicts the metric interpolated onto the grid created by S^3 for $\mathcal{M}_{\text{approx}} = 0.77$ in the x - y -plane at $z/d = \pi/2$ (15b) and x - z -plane (15d) at $y/d = 8$.

Fig. (16) shows the first four uneven left-singular vectors and the associated singular values in the x - z -plane for $\mathcal{M}_{\text{approx}} = 0.27$. As in section 3.2, the z -component of the modes is omitted here for compactness. The modes on the reduced grid show no visible difference to the modes on the original mesh from CFD, which is consistent with the results of the x - y -plane. The results for $M_{\text{approx}} = 0.53$ and $M_{\text{approx}} = 0.77$ are not shown here, since the error between the SVD results of the original and reduced data is already within acceptable bounds for $M_{\text{approx}} = 0.27$. However, the error reduces with increasing approximation of the metric.



(a) x -component



(b) y -component

Figure 16: Comparison of the first four uneven POD modes and associated singular values for the original data (left columns) and the S^3 -interpolated data with $\mathcal{M}_{\text{approx}} = 0.27$ (right columns) in the x - z -plane at $z/d = 8$ for the cylinder test case. The colorscale is identical for all contours and bounded by $\pm \|\mathbf{U}\|_\infty$.