CEAS Aeronautical Journal (2024) 15:105–123
https://doi.org/10.1007/s13272-023-00638-1

ORIGINAL PAPER

A time-accurate inflow coupling for zonal LES

Marcel P. Blind[1] · Johannes Kleinert[1] · Thorsten Lutz[1] · Andrea Beck[1]

Received: 20 September 2022 / Revised: 2 January 2023 / Accepted: 11 January 2023 / Published online: 9 March 2023

Abstract
Generating turbulent inflow data is a challenging task in zonal large eddy simulation (zLES) and often relies on predefined DNS data to generate synthetic turbulence with the correct statistics. The more accurate, but more involved alternative is to use instantaneous data from a precursor simulation. Using instantaneous data as an inflow condition allows to conduct high fidelity simulations of subdomains of, e.g. an aircraft including all non-stationary or rare events. In this paper, we introduce a toolchain that is capable of interchanging highly resolved spatial and temporal data between flow solvers with different discretization schemes. To accomplish this, we use interpolation algorithms suitable for scattered data in order to interpolate spatially. In time, we use one-dimensional interpolation schemes for each degree of freedom. The results show that we can get stable simulations that map all flow features from the source data into a new target domain. Thus, the coupling is capable of mapping arbitrary data distributions and formats into a new domain while also recovering and conserving turbulent structures and scales. The necessary time and space resolution requirements can be defined knowing the resolution requirements of the used numerical scheme in the target domain.

Keywords DGSEM · Instantaneous inflow condition · Coupling · Zonal LES

1 Introduction

In modern computational fluid dynamics (CFD) research, large eddy simulation (LES) is becoming more popular due to increased computation performance [1]. However, many practical applications still are out of range for a detailed investigation using this technique. Therefore, many simulations run today are based on so-called zonal approaches that often depend on predefined DNS data to generate synthetic turbulence with the correct statistics. By zonal, we mean that only a small subset of a domain is simulated with the LES method in order to decrease computational cost, while the majority of the domain is for example solved by a much cheaper Reynolds-averaged Navier–Stokes (RANS) method, or the subset is equipped with suitable boundary conditions, in particular scale-resolving inflow data. This can be of special interest in the simulation of turbulent boundary layers, where we do not want to simulate the initial transition process but are just interested in the fully developed boundary layer as a starting point. To achieve this, many approaches have been developed, such as the synthetic eddy method [2, 3] or the recycling–rescaling approach [4, 5] which allow for significantly smaller domains. A practical example is the simulation of acoustic noise at the trailing edge of an airfoil where a detailed simulation of only a part of the airfoil is needed [6]. One similarity of the applications just described is their dependency on boundary layer properties and, therefore, reference data from the literature.

Another slightly different example is the investigation of the interaction of the turbulent wake from the wing of an aircraft with the boundary layer of a horizontal tail plane (HTP) of the same aircraft (cf. Fig. 1). The described scenario poses a challenge, since fully scale-resolving codes often cannot afford to compute the whole aircraft and codes that are capable of running a simulation of a whole aircraft are often not able to run high fidelity simulation of parts of it. Therefore, there is the need to map

✉ Marcel P. Blind
blind@iag.uni-stuttgart.de

Johannes Kleinert
kleinert@iag.uni-stuttgart.de

Thorsten Lutz
lutz@iag.uni-stuttgart.de

Andrea Beck
beck@iag.uni-stuttgart.de

[1] Institute of Aerodynamics and Gas Dynamics, University of Stuttgart, Pfaffenwaldring 21, 70569 Stuttgart, Germany

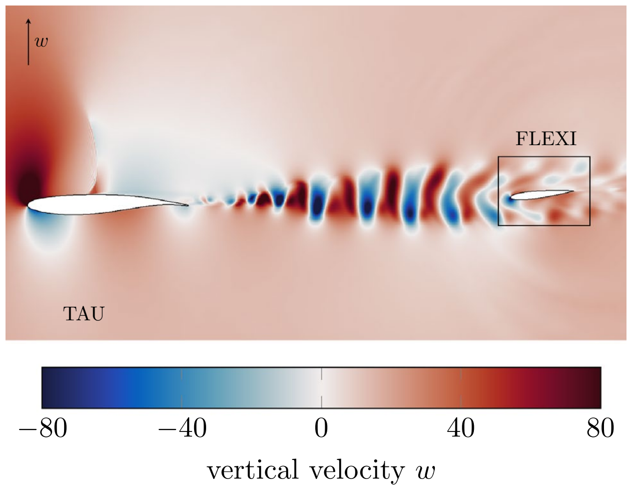


Fig. 1 Visualization of the w-component of the velocity in the tandem wing configuration showing the wing wake interacting with the HTP. The simulation was run using TAU. Region of interest for detailed simulation in FLEXI is highlighted

the results in a time-accurate manner from one simulation to a boundary/initial condition on a detailed simulation with a smaller domain. In this work, the main focus lies on imposing inflow boundary conditions from mapped data. The use of mapped data at the outflow of the smaller domain is in principle possible but was not investigated.

Using the described method allows for refined simulation of areas of special interest. In addition, such a coupling between these codes enables a way to further investigate the interaction between turbulence of different physical scales very efficiently. Therefore, zonal simulations of high Reynolds number flow become feasible.

When trying to couple different numerical codes, we encounter several problems on how to approach this:

1. Is a two-way coupling necessary or is one-way sufficient?
2. Do we couple the codes during runtime?
3. Are the underlying numerics compatible?

The first question is—in context of the scenario described above—easy to answer. Since we only are interested in the effects of an incoming flow to the target domain, a one-way coupling is sufficient. We note that, depending on the equation system, a one-way coupling via a prescribed Dirichlet boundary is prone to errors, since information transport is limited to one direction. However, we can justify this simplification by assuming that we e.g. extract the flow in a wake region with no proximity to a wall in case of the compressible Navier–Stokes equations. In addition, the simplification removes a lot of complexity and thus enables efficient coupling of different solvers and experiments which would not be possible in a two-way coupled way.

Thus, we can directly answer the second question. Having both codes run separately allows us to perform the mapping in a preprocessing step for the zonal simulation and thus removes a bottleneck during runtime. In addition, it avoids the complexities of having to solve the high-performance computing (HPC) problem of having to run two possibly very heterogeneous codes synchronously and establish efficient parallel communication patterns. As mentioned before the coupling is designed to perform detailed simulations of a subdomain, meaning that the area of special interest is also contained in the full domain simulation and, therefore, is assumed to be represented in a sufficient way to capture all the necessary physics. In addition, we have to consider that the incoming physics can be truncated. We thus investigate the influence of different resolution combinations in order to quantify this error.

The third question is harder to answer since we not only have to take the spatial discretization such as finite-volume, finite-difference and finite-element methods into account, but also take care of the temporal discretization, which in most applications is either implicit or explicit. In general, two choices for mapping the solutions between two heterogeneous representations are possible: a projection approach and an interpolation method. While the former is (approximately) conservative, ensuring this property on arbitrary meshes in space and time is cumbersome, expensive (as it requires non-local operations) and not very flexible. The interpolation approach relaxes this condition, yielding a very general mapping process and allowing for extending the algorithms to work with arbitrary (x, y, z) data as an input and map it to a compatible data format. The mapping algorithms thus have to be able to capture resolution differences from both grid spacing and numerical efficiency per DOF, arbitrary points and inconsistent time steps. Hence, interpolation algorithms seem to be a good choice for achieving these properties in space and time.

Another problem we have to tackle is how to deal with large data sets. Although we opt for an offline coupling which avoids having to transfer the data in situ, time-resolved surface or volume data is very memory intensive. Thus, memory management algorithms have to be taken into account, including parallelization, load balancing and data reduction in order to keep compute costs low.

In this paper, we want to show that mapping instantaneous data from the DLR finite-volume code TAU to the high-order spectral element code FLEXI is possible and allows for detailed simulation of subdomains. Thus, we want to answer the following research questions:

1. Is it possible to get a mapping for the data which allows for a numerically stable coupled simulation?

2. Which temporal resolution is needed?
3. How does the difference in spatial resolution (mesh and numerical efficiency per DOF) affect the mapping error?

2 Numerical methods

An important aspect for the mapping algorithm is the knowledge of the underlying numerics and the associated scale-resolving properties which we are going to assess in more detail in the following section.

2.1 Code frameworks

The target numerical scheme for which the data has to be prepared is the open-source discontinuous Galerkin framework FLEXI, developed at the University of Stuttgart [7]. In this scheme, the domain is partitioned into non-overlapping unstructured hexahedral elements. We can choose an arbitrary polynomial degree for the elements. This also means that the solution in each element is represented as a polynomial.

In contrast to the spectral element code FLEXI, the source data are typically point-wise data resulting from an experiment or finite volume code. The presented mapping algorithms are designed and implemented to be generally applicable but are optimized to work with the DLR finite-volume code TAU. In Fig. 1, a typical application of TAU is visualized. TAU uses hybrid RANS/LES methods, e.g. for cases with separated flows, where attached boundary layers are treated in RANS mode and detached wake regions are resolved in LES mode. Thus, the effect of the wing wake on the boundary layer of the HTP is hard to investigate within TAU. FLEXI on the other hand resolves the boundary layer and thus is capable of quantifying the influence of the wing wake. Thus, the region of interest that can be simulated within FLEXI is marked in Fig. 1. TAU will also be used to validate the results in Sect. 4.

2.1.1 TAU

The finite-volume solver TAU is developed by the German Aerospace Center (DLR) and widely used among the aviation industry [8] and universities. It solves the Euler or RANS equations both on structured and unstructured grids. Several one- and two-equation models and Reynolds stress models are implemented for turbulence modeling. In addition, LES or hybrid RANS/LES simulations can also be performed. Hexahedra, tetrahedra, triangular prisms and pyramids are supported elements for the cells of the primary grid. TAU uses the so-called dual mesh approach. Control volumes are constructed around the nodes of the primary mesh, which are used for the spatial discretization. For the computation of the numerical fluxes at the cell boundaries of the dual mesh cells, different upwind schemes and central approximations are available. Both explicit and implicit schemes can be chosen for the integration in time. The resulting linear system is solved with SGS or LUSGS schemes. For convergence acceleration, local time stepping, residual smoothing and multigrid methods are used. Parallelization is achieved by domain decomposition, with communication through the message passing interface (MPI).

2.1.2 FLEXI

FLEXI is a high-performance open-source CFD solver based on the discontinuous Galerkin spectral element method (DGSEM). It utilizes hexahedral tensor product elements with an arbitrary polynomial degree in each element. Since DG methods are hybrid schemes combining finite-element and finite-volume methods, we use a Roe Riemann solver with minimum dissipation for the fluxes between the elements [9]. In addition, we represent the polynomial solution on a non-equispaced Legendre–Gauss or Legendre–Gauss–Lobatto point set. We advance the resulting equations in time by applying an explicit-in-time Runge–Kutta method. Since FLEXI acts as a framework there are multiple equation systems implemented. In this paper, we mainly use the compressible Navier–Stokes equations. For validation, the linear scalar advection system is used. The results in the application section are created using the compressible Navier–Stokes equations, which are implemented as skew-symmetric split form approximations to minimize aliasing instabilities [10]. The boundary conditions generally are imposed weakly. This means that we do not prescribe the state at the corresponding solution point in the element, but rather prescribe the numerical flux. FLEXI is parallelized using MPI and was successfully tested on up to $\mathcal{O}(10^5)$ cores [11].

2.1.3 Comparison of the code frameworks

Discontinuous Galerkin methods are commonly used high-order schemes. Finite-volume methods in contrast are for unstructured meshes often limited to second order. It is well known that for the same number of degrees of freedom high-order methods can achieve lower error and need fewer solution points to resolve the same structures. This is known as numbers per wavelength n_{PPW} criteria [1, 12]. High-order methods can achieve $n_{\mathrm{PPW}} > 4$, while second-order finite-volume method is often limited to $n_{\mathrm{PPW}} \approx 20$. This means that for resolving a structure of a given wavelength, high-order methods would need up to a factor of 5 fewer resolution points. However, this property heavily depends on the used polynomial degree N as shown in Fig. 2. Gassner et al. [12] provide reference values for n_{PPW} depending on

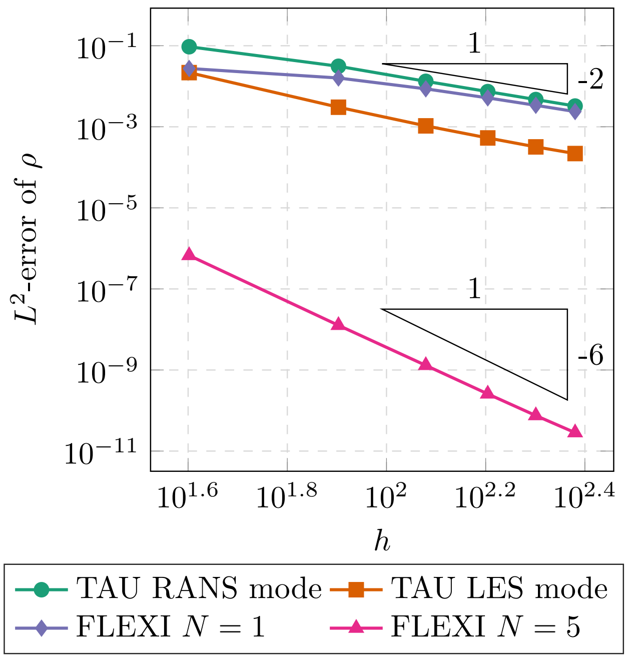


Fig. 2 Comparison of the convergence behavior of TAU and FLEXI for different settings and meshes

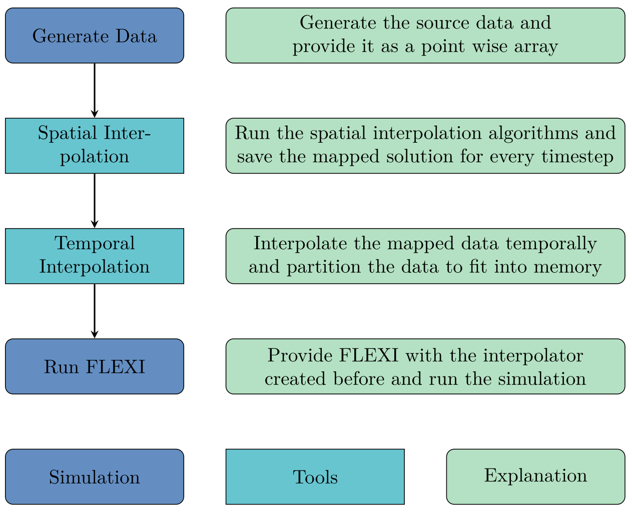

Generate Data
Generate the source data and provide it as a point wise array
Spatial Interpolation
Run the spatial interpolation algorithms and save the mapped solution for every timestep
Temporal Interpolation
Interpolate the mapped data temporally and partition the data to fit into memory
Run FLEXI
Provide FLEXI with the interpolator created before and run the simulation
Simulation
Tools
Explanation

Fig. 3 Workflow of imposing a time-resolved boundary condition

the polynomial degree. In this paper, we use $N \in (1, 3, 5, 7)$ and thus need $n_{\mathrm{PPW}}(N = 1) \approx 20$, $n_{\mathrm{PPW}}(N = 3) \approx 9$, $n_{\mathrm{PPW}}(N = 5) \approx 7$ and $n_{\mathrm{PPW}}(N = 7) \approx 6$.

Thus, on the same mesh, the simulation with FLEXI $N = 5$ not only has a faster decreasing error but also has the smaller error for a given number of degrees of freedom. This becomes obvious since FLEXI $N = 1$ or TAU on the $h = 10^{2.4}$ grid correspond in terms of amount of degree of freedom with FLEXI $N = 5$ at $h = 10^{1.6}$. The Shu-vortex test case [13] is utilized to conduct this study. All simulations are run independently and thus without mapping.

The results denoted as “TAU RANS mode” are obtained with numerical settings commonly used for the RANS zones of hybrid RANS/LES simulations in TAU. A second-order central flux approximation stabilized by artificial dissipation is used, derived from the scheme after Jameson, Schmidt and Turkel [14]. Applying a skew-symmetric scheme with matrix dissipation [15] already reduces the dissipation level compared to the TAU-default average of flux scheme with scalar dissipation. The simulations denoted with “TAU LES mode” additionally utilize a reconstruction of the convective fluxes using a linear gradient extrapolation at the cell faces, in a way to reduce the numerical dispersion of the skew-symmetric scheme [16]. Moreover, the coefficient of artificial dissipation is lowered by a factor of 16. These settings are suitable for the LES zones of a hybrid simulation. In the FLEXI runs, a Roe Riemann solver is used [9]. The FLEXI simulation for $N = 1$ shows a result similar to the TAU runs with the same order of convergence. However, $N = 1$ is typically not used in practical application, since the advantages of high-order schemes are not visible for such low polynomial degree. The runs with $N = 5$ represent a more realistic scenario and show the advantage of high-order polynomials.

2.2 Workflow

Before presenting the mapping routines, we first discuss the general workflow of how to run a simulation with time-resolved input data. The general workflow is visualized in Fig. 3.

First, the source data have to be provided. Generally, this can be in the form of point-wise scattered data. Since in this paper we focus on the procedure for mapping TAU data to FLEXI, we assume to get either volume snapshots or surface data from TAU.

Second, we process the data by choosing an appropriate spatial mapping mechanism. In addition, we have to decide if we only want to map surface data for an instantaneous boundary condition, or if we also want to get the volume information to e.g. generate a restart file for FLEXI. The tool creates an interpolated file for each input file. The results are saved in a HDF5® format that uses a polynomial structure closely related to FLEXI. Thus, it contains an array with the interpolated values stored as coefficients of a two- or three-dimensional polynomial for each element depending on whether surface or volume interpolation is used. To ensure compatibility with FLEXI, the output polynomial degree is identical to the degree of the simulation we want to run afterwards.

Higher order representations are prone to aliasing and oscillations in general and the quality of the results heavily relies on the used point set and polynomial degree we perform the interpolation on. Since the output degree and point set is defined by the simulation we want to perform eventually, we cannot use these parameters for mitigating errors. Thus, we super-sample the target point representation which helps avoiding errors due to oscillations resulting from the non-polynomial character of the source points. We then map the source data to the super-sampled target data points. The super-sampled points are constructed using the same set of points (e.g. Legendre–Gauss–Lobatto) as the original target point set. We found that using $M \approx [1.5, 2] \cdot (N + 1)$ points for super-sampling yields good agreement and mitigates oscillations significantly. This is in line with the literature values for overintegration of turbulent data, which is commonly used in the DG community for dealiasing [17]. After interpolating the source data to super-sampled target points, we project the solution to the original basis $N < M$ which removes the high modal information that is especially affected by aliasing.

Third, we interpolate the results from the second step temporally. The mapped volume or surface files created in the second step are converted into a dataset containing the temporal interpolator for each solution point. The interpolator consists of the coefficients of the polynomial, which are dependent on the evaluation time. In addition, we partition the data into a user-defined number of subsets to limit the amount of data of each interpolator and avoid memory overflow during FLEXI runtime.

Fourth, we provide FLEXI with the resulting file. FLEXI then evaluates the interpolant in each time step and sets the according boundary condition to the interpolated values.

2.2.1 Some remarks on surface data

The mapping algorithms we present can be applied to volume as well as surface data. Depending on the provided data, the spatial mapping algorithms will either use the volume solution to extract the target boundary or use the provided surface plane directly.

TAU is able to write 2D data from a user-defined plane, onto which the flow variables are interpolated internally using algorithms of the chimera technique [18]. This interpolation will also of course introduce an error that leads to a mismatch between the volume solution of TAU and the 2D data on the plane. The resulting chimera plane can be read in separately. Hence, there can be made significant simplifications in term of area reduction which reduces the overall cost of the mapping algorithms.

2.3 Spatial interpolation

An important step to achieve the coupling of TAU with FLEXI is the spatial mapping. Since ultimately we want to create an instantaneous boundary condition, we have to map surface data only. To keep the interpolation more general, we implement a three-dimensional method to also allow volume interpolation and arbitrary-oriented surface planes in the source domain.

A major challenge in creating a mapping between TAU and FLEXI is the difference in spatial discretization. Industrial finite-volume codes rely often on tetrahedral meshes. Therefore, a direct interpolation is difficult since mesh data structures for hexahedral only codes are dissimilar. In contrast to FLEXI, TAU stores its solution as low-order point-wise data. This results in arrays containing the coordinates (x, y, z) and the flow solution in primitive variables $\vec{U} = (\rho, u, v, w, p)$ [19]. FLEXI, however, stores its solution as polynomial data for each element [7, 20]. The difficulty now is to consistently map the point-wise TAU data to the polynomial coefficients needed for the solution polynomials in FLEXI. Since FLEXI only uses unstructured hexahedral elements and TAU is in contrast also based on tetrahedrals, we cannot use the TAU mesh information natively. An example of this incompatibility including a schematic of the interpolated solution is visualized in Fig. 4.

The interpolation algorithms thus have to work with scattered data. This means we have a cloud of points with a submerged target mesh. Therefore, interpolation from the TAU source data to the FLEXI target data has to be done using unstructured interpolation algorithms. There are several algorithms available that are suitable for such tasks.

Scattered data interpolation generally can achieve good accuracy and performance but is highly dependent on the distribution of the source points [21]. On the other hand, implementing scattered data interpolation routines enable us to gain a more universal interface, since other codes and solution formats can be implemented easily. In Sect. 2.3.1, we provide an overview over the scattered data interpolation methods used for spatial mapping.

Since the solution data of the source solver is provided beforehand, we do not have to map the data during run time, which saves a lot of computation time. In addition, the meshes used by both codes are known a priori (and remain constant during the computation). Still, good performance is crucial. Therefore, we implement the interface framework with MPI parallelization.

This is done by reading in the mesh and distributing the elements equally between each processor using MPI. The elements are sorted along a Hilbert curve [20] which minimizes the communication interfaces between the individual processors cf. red curve in Fig. 5a. This approach, however, can only be done for the target FLEXI mesh because we need the mesh information to distribute the data directly. For the source data, we only read in data points. Hence, a simplification and distribution of the load between the processors is not possible in a first step. Thus, we read in the source data

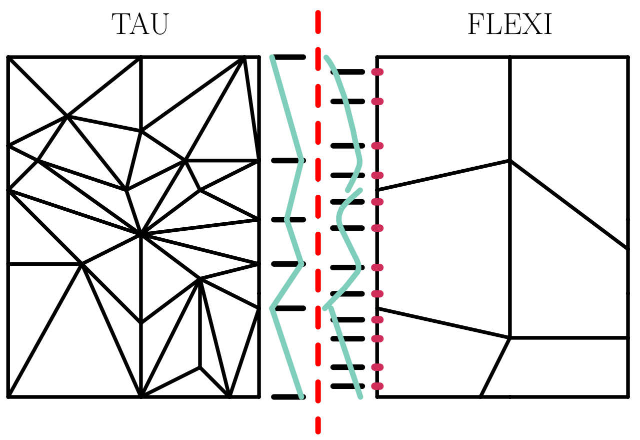


Fig. 4 Visualization of the TAU-FLEXI interface including different point sets. Light green curve shows an approximate solution after interpolating the TAU solution to the FLEXI point set

into shared memory [11, 22]. Each processor then accesses the shared memory source data and simplifies it by only selecting the data which is necessary to spatially interpolate in its individual domain. With this method, we can save a lot of computational time and decrease our memory footprint without sacrificing accuracy.

If we use surface data as an input to the spatial interpolation algorithms, we have to consider load balancing in more detail. The distribution of volume elements that has just been described does not take surfaces into account. Thus, for the surface mapping, we cannot ensure that every part of the decomposed domain contains surface elements that have to be mapped (cf. Fig. 5a). Therefore, especially for small domains with many processors, it is possible that not all processors are working on the task resulting in an inefficient mapping. Hence, we have to redistribute the load between the processors according to the number of surface elements (cf. Fig. 5b). This can be done by assigning each surface element a high weight for domain decomposition. This weight is chosen by counting the number of boundary sides that have to be mapped per element and it generally reduces the computational load on MPI ranks that contain such a coupling interface. To do so, we first have to read in the mesh file normally, then apply the surface weighting and finally reinitialize the mesh reading process [23]. With this approach, we can gain performance improvements while sacrificing a few seconds in the initialization process due to the necessary reinitialization of the mesh. The overall cost of surface interpolation will be lower than using volume data. The difference between volume and surface distribution is visualized in Fig. 5.

In addition, we have to ensure to provide a buffer region around every individual MPI domain in order to establish the interpolation stencils for each point. The buffer area is estimated by taking the size of the largest element in the complete domain of the source points into account. Since the largest element is not known directly, we take the distances between the scattered points into account and use the largest distance for that matter.

2.3.1 Nearest neighbor interpolation

The nearest neighbor interpolation checks the source data coordinates and finds the closest point to the desired FLEXI target point by point-wise comparison. The values U of the source data are then directly stored as a nodal coefficient in FLEXI. This type of interpolation is piece-wise constant between source data points. Another requirement is an evenly distributed source mesh. If these requirements cannot be met, there is risk of bad results. This does not automatically mean that the results are not physical, but rather that the resulting interpolation polynomial inside a DG cell is ill-conditioned and can, due to the massive jumps, result in an unnaturally oscillating mapped solution. Another phenomenon one can observe is the possibility to get jumps on the element boundaries of the target mesh, if the boundary nodes are not included. On the other hand, this interpolation technique yields fast and good results if the source and target data are well aligned or if the meshes coincide at the interface.

2.3.2 Inverse distance weighting

A more general approach is available using inverse distance weighting [24]. The target solution is calculated using a weighted average of the source value

$$u(\vec{x})_{\text{target}} = \frac{\sum_{i=1}^{N_{\text{source}}} \omega_i(\vec{x}) u_{i,\text{source}}}{\sum_{i=1}^{N_{\text{source}}} \omega_i(\vec{x})} \tag{1}$$

with N_{source} denoting the number of source points in the whole domain. In contrast to the nearest neighbor approach, we not only take one point into account, but all in the source area. The weights $\omega(\vec{x})$ are depending on the distance between the source points and the target solution point

$$\omega_i = \frac{1}{\left(\|\vec{x} - \vec{x}_i\|_{L_2}\right)^p} \tag{2}$$

and a weighting exponent p. For $p \Rightarrow \infty$, the inverse distance weighting approach resembles the nearest neighbor method. A modification to the general inverse distance weighting was introduced by Shepard, who proposed to only take the source points into account that are within a predefined radius R around the target point [25]. This reads as

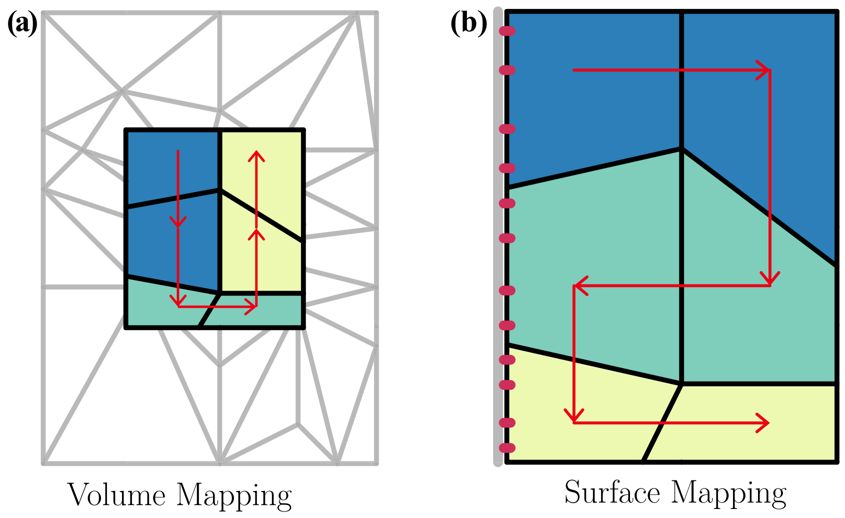


Fig. 5 Differences between MPI domain decomposition for volume and surface mapping on three MPI processors. Same colors correspond with the same MPI domain. Space filling curve is visualized in red

$$\omega_i = \left(\frac{\max(0, R - \|\vec{x} - \vec{x}_i\|_{L_2})}{R\|\vec{x} - \vec{x}_i\|_2} \right)^2 \tag{3}$$

with R denoting a predefined search radius.

2.3.3 Radial basis functions

A third option to consider for unstructured interpolation are radial basis functions φ [26, 27]. These methods allow for high-order accurate interpolants s of unstructured data. The interpolant consists of the weighted sum of radial basis functions. In contrast to the other methods introduced earlier, we have to solve a linear equation system to invert the Vandermonde and to determine the weights ω satisfying

$$s(\vec{x}) = \sum_{i=1}^{N_{\text{source}}} \omega_i \varphi(\|\vec{x} - \vec{x}_i\|_{L_2}) \tag{4}$$

and, therefore,

$$u_{j,\text{source}} = \sum_{i=1}^{N_{\text{source}}} \omega_i \varphi(r_{ji}) \tag{5}$$

with $r_{ji} = \left\|\vec{x}_j - \vec{x}_i\right\|_{L_2}$. We rewrite the interpolation condition in matrix notation

$$\begin{bmatrix} \varphi(r_{11}) & \varphi(r_{12}) & \cdots & \varphi(r_{1N}) \\ \varphi(r_{21}) & \varphi(r_{22}) & \cdots & \varphi(r_{2N}) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi(r_{N1}) & \varphi(r_{N2}) & \cdots & \varphi(r_{NN}) \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_N \end{bmatrix} = \begin{bmatrix} u_{1,\text{source}} \\ u_{2,\text{source}} \\ \vdots \\ u_{N,\text{source}} \end{bmatrix}. \tag{6}$$

This can be rewritten in matrix form as $\mathbf{\Phi}_{ij}\vec{\omega}_i = \vec{u}_{j,\text{source}}$ using index notation. Since we have to invert the matrix $\mathbf{\Phi}$ for interpolation, the radial basis approach is the most expensive of the introduced methods. To solve the equation system, one can use algebraic approaches or iterative equation system solvers. The size of the equation system can be limited by introducing a limit to the support radius (cf. Eq. (3)) and thus not taking all source points into account.

We evaluate the interpolant

$$u(\vec{x}) \approx \sum_{i=1}^{N_{\text{source}}} \omega_i \varphi(\|\vec{x} - \vec{x}_i\|_{L_2}) \tag{7}$$

and get the value at an arbitrary point in the computational domain.

Typical radial basis functions for interpolation are multiquadric $\varphi(r) = \sqrt{1 + (\varepsilon r)^2}$, inverse multiquadric $\varphi(r) = \frac{1}{\sqrt{1+(\varepsilon r)^2}}$, Gaussian $\varphi(r) = e^{-(\varepsilon r)^2}$ and thin plate spline $\varphi(r) = r^2 \ln(r)$ functions with $r = \left\|\vec{x}_j - \vec{x}_i\right\|_{L_2}$. The parameter ε defines the shape of the function and is used for scaling. The multiquadric and the thin plate spline have shown to be the most reliable radial basis functions for this use case. Since the thin plate spline does not require any additional user parameter ε, we use this function for all further investigations.

During implementation of the algorithms above some observations were made. First, none of the scattered interpolation method is designed in a way to be conservative. Thus, we interpolate the primitive variables and, for consistency reasons, convert to conservative variables after mapping.

2.3.4 Comparison of the spatial interpolation methods

Before assessing the performance of the spatial interpolation routines in context of the mapping routines, we investigate the convergence behavior in an isolated test case. Thus, we calculate the L_2-error of a simple one-dimensional interpolation of a sine function $f(x) = \sin(2\pi x)$.

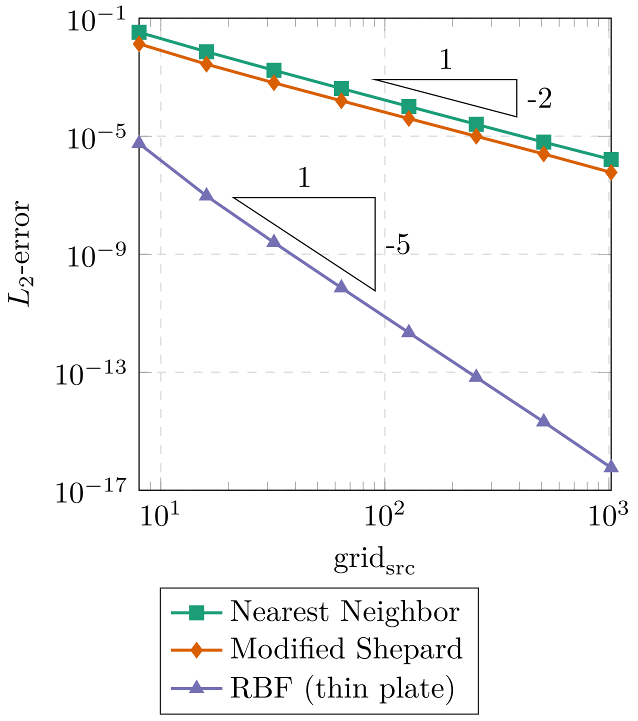


Fig. 6 Convergence of the L_2-error of an interpolated one-dimensional sine function for different interpolation methods

We plot the error in Fig. 6 over the sampling resolution of the source data. We can see that radial basis function interpolation clearly yields the best results with lower errors and a better convergence rate EOC = 5 than nearest neighbor and inverse distance weighting interpolation with EOC = 2. We assess the accuracy and the differences between the spatial interpolation schemes in more detail in Sect. 3.

2.4 Temporal interpolation

In addition to the spatial interpolation, we also have to interpolate temporally in order to account for the different time stepping schemes in the source and target codes. FLEXI, e.g. uses an explicit low storage Runge–Kutta method to advance the equation systems in time. TAU on the other hand uses an implicit time discretization to accomplish that. However, in addition to the different time stepping schemes, the time step and output rate of the simulation data can change between different simulations. For using the data as an instantaneous boundary condition, we have to ensure that we can provide the target solver with the correct inflow data at an arbitrary point of time. Thus, it is crucial to interpolate the results of the spatial interpolation in time to get a continuous temporal interpolator.

In contrast to the volume and surface mapping the temporal interpolation consists of purely one-dimensional problems. For one-dimensional data, there are vast numbers of different interpolation techniques. In this work, we use polynomial interpolation in combination with a Lagrange basis and spline interpolation.

We use the Lagrange interpolation basis since coefficient and solution values coincide [28]. The Lagrange interpolation uses a uniform point set, as the time sampling is usually regular, and multiple time steps are combined to a polynomial, without constraining the time derivative at each polynomial interval.

Furthermore, two different variants of spline interpolation are implemented. A common open spline as well as the Akima spline [29]. In contrast to a typical spline an Akima spline does not take the second derivative into account. This leads to a more evenly distributed solution and fewer overshoots compared to the open spline. The problem of overshoots can also be found in polynomial interpolation of degree $p \geq 2$. This becomes especially important if an implicit source method is paired with an explicit target solver. In this case, the temporal interpolation has to come up for a huge number of time steps since the time step in an explicit scheme is typically much smaller than implicit time steps. Thus, overshoots can play a significant role for the overall mapping quality. If the time steps of source and target method are similar, the effect of the temporal interpolation becomes smaller. However, it should be noted that even in this case, overshoots can be generated. Generally, for these reasons, it is recommended to either use linear interpolation or the Akima interpolation for the most reliable results. We will show this in Sect. 3.

Hence, the resulting quality of the interpolation depends on multiple factors. First, on the chosen interpolation method. Second, on the sample rate of the provided state or boundary source files. Thus, a general prediction of the error resulting from temporal interpolation is difficult.

In Fig. 7, a schematic overview of the temporal interpolation methods is visualized. The vertical dashed lines mark the sampling positions of the black reference data. For polynomial interpolation, the polynomials of degree P are constructed using $P+1$ points, resulting in a piece wise polynomial representation of the reference data. For spline and Akima interpolation, we use the whole one-dimensional time series to build up the spline or Akima representation, respectively.

The interpolation is done in a separate tool and is not only limited to surface data but can also be done with restart files of any FLEXI simulation. The result is processed and saved in an HDF5® file which includes the coefficients for every polynomial at every temporal sample point.

The resulting files of the temporal interpolation routine can either be directly used in FLEXI for evaluation of the

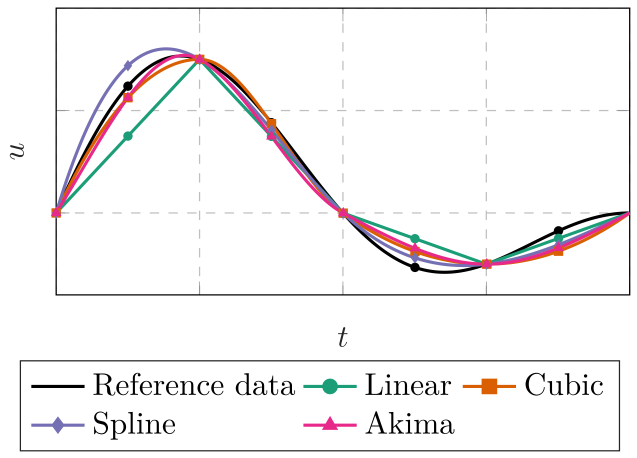


Fig. 7 Overview over the temporal interpolation methods on a one-dimensional time series of an arbitrary variable u

interpolant or even be used to generate a restart file to continue a simulation at an arbitrary point of time.

The temporal interpolator generated contains the resulting polynomial or spline at each degree of freedom. Thus, the overall size of the interpolator array has two more dimensions (polynomial coefficients and time) than the solution array. With increasing dimensionality, the memory requirements of the array also increase. Depending on the amount of source data available, it might be necessary to partition the resulting temporal interpolant in order to avoid memory overflow during simulation of the target domain. Therefore, a maximal size for the interpolant array has to be provided by the user. The interpolation algorithm will then partition the data into equally sized datasets, each containing a period of time which results from the user parameter. FLEXI then only reads in the dataset that contains the temporal information of the current FLEXI time step. Thus, the read in interpolant allows for obtaining an interpolated solution at an arbitrary point of time within its temporal partition. During runtime of FLEXI, the interpolant is evaluated in every Runge–Kutta stage of the timestep.

3 Validation of the interface

In this section, we start validating the mapping algorithms. We chose a gradual approach and start by showing a proof of concept, followed by the temporal algorithms and in the end assess the convergence behavior of the spatial interpolation algorithms of the interface.

We start to evaluate the algorithms by applying them to very simple test cases. Thus, we chose the linear scalar advection equation

$$u_t + \vec{a} \cdot \nabla u = 0 \quad \text{with} \quad \vec{a} = (a_1, a_2, a_3)^{\mathrm{T}} \in \mathbb{R} \quad \text{and} \quad \vec{x} = (x, y, z)^{\mathrm{T}} \tag{8}$$

due to its simplicity and a priori known exact solution for given initial conditions. The transport velocity is set to $\vec{a} = (1, 0, 0)^{\mathrm{T}}$. We vary the initial conditions between the tested scenarios and describe them in the corresponding sections in more detail. The source domain $\Omega \in [-1, 1]^3$ and the target domain $\Omega \in [1, 3] \times [-1, 1]^2$ are designed to have a shared interface at $x = 1$. The source data as well as target data for these test cases are fully generated using FLEXI. Triple-periodic boundary conditions are used for the source mesh and the target mesh is designed to have periodic boundary conditions in y and z. In x-direction, we have the instantaneous interface condition at $x = 1$ and a outflow at $x = 3$.

3.1 Proof of concept

We start the validation of the interface by applying it to a very basic sine test case with

$$u(\vec{x}, t) = \sin(\pi(\vec{x} - \vec{a}t)). \tag{9}$$

The initial conditions are set to $u_0(\vec{x}, t = 0)$. The exact solution $u(\vec{x}, t)$ is purely x dependent and thus the values at the interface plane $u(x = 1, t)$ do not vary in y and z. The target domain is initialized with a constant solution $u_0 = \text{const.} = 0$.

For this first test, we match the surface elements at the interface and thus can use nearest neighbor interpolation without sacrificing accuracy (source and target points coincide). In this case, the nearest neighbor algorithm will just copy the data from the source to the target domain. Source and target mesh are only offset in x-direction by the length of the domain. In Fig. 8, the initial condition is depicted in light green. One can also see the solution of Eq. (8) after $t = 0.8$ and $t = 1.6$. The vertical gray dashed grid lines depict the mesh of the simulation grid and the red dotted line visualizes the interface between source and target domain. The graphs are extracted from the center line in x-direction of the equispaced Cartesian cubes, which each have a resolution of $16 \times 16 \times 16$ using $N = 4$ polynomials in each element. Legendre–Gauss–Lobatto points are used for the source and target simulation. In addition, we avoid temporal interpolation by sampling the interface data at every physical time step $dt \approx 0.0125$. However, due to the utilized explicit Runge–Kutta scheme, we note that we still have to use interpolation in order to recover the boundary values in each Runge–Kutta stage.

In Fig. 8, we can see that the general workflow presented performs as expected and the information gets propagated over the interface with $a = 1$. Since we do not interpolate the

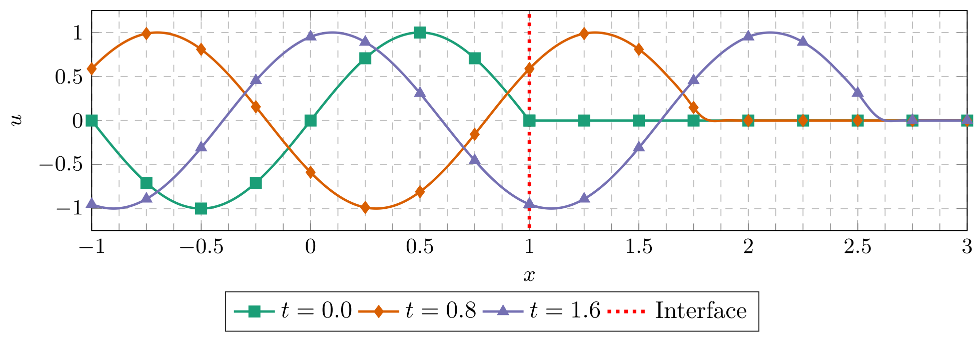


Fig. 8 Overview of the spatial mapping process for a traveling sine wave

data in any way in this test case, we expect the overall errors between source sine and target sine wave to be comparable. In the source domain, we have an L_2-error of 9.6506E−07 after $t = 2$. The L_2-error in the target domain after $t = 2$ is evaluated in the same way and is 9.6859E−07. This successfully proves that the workflow is capable of mapping the data without any information loss. We stress that the full framework is working as if we were coupling between two heterogeneous solvers, with the exception that the source and target points coincide.

Note that we used continuous initial conditions between source and target domain. We found that one should avoid having jumps or large discrepancies of the initial conditions between the source and target domain due to nonphysical disturbances created at the inlet of the target domain which are further propagated. This, however, is not due to the interface mapping algorithms but rather due to the nature of the high-order scheme. In practical applications, especially for transient simulations, this does not pose a problem since all structures starting from free stream will be mapped into the target domain.

3.2 Assessing the temporal interpolation and sampling

Next, we evaluate the effect of temporal interpolation/sampling on the interface mapping process. Thus, we investigate the effects of different temporal interpolation schemes and sampling rates on the incoming solution, which we map via the instantaneous boundary condition. For this test case, we chose different exact solution and initial conditions for the linear advection equation (8). To evaluate the effect of the sampling, we chose an initial condition that includes a discontinuity in order to visualize the information loss. Thus, we use

$$u(x,t) = \begin{cases} 1. & \text{if} \quad -0.5 < x - a_1 t < 0.5 \\ 0. & \text{else} \end{cases} . \tag{10}$$

We use Legendre–Gauss–Lobatto nodes with $N = 4$ on a $256 \times 1 \times 1$ source and target domain. The surface elements at the interface are again matched. Thus, we can use nearest neighbor interpolation to interpolate the surface data in space, without sacrificing accuracy (copy values from source to target). Figure 11a depicts the simulation at two discrete points in time evaluated with different Δt-interpolants. The interface is located at $x = 1$ and the discontinuities travel into the target domain on the right side of the red dotted interface with $a = 1$.

Already in the overview graph in Fig. 11a, we can see substantial differences between the two interpolated jumps. For this test, we chose in total three sampling rates. A fine sampling rate at $\Delta t \approx 10dt$ which is approximately ten times the explicit FLEXI time step $dt \approx 0.01$ and two coarser sampling rates at $\Delta t \approx 50dt$ and $\Delta t \approx 100dt$. In Fig. 11a, only the finest and the coarsest sampling rate are visualized. Figure 11b shows the jumps of all three sampling rates at the same evaluation time t in more detail. Since the x-axis is scaled identically, one can see that the influence of the temporal mapping on the target FLEXI simulation is very high. There are two main effects visible:

1. The temporal distance between two samples affects the slope of the jump and
2. the temporal interpolation method has an effect on the quality of the jump representation.

The first observation has to only result from the temporal interpolation since the slope of the shock has been steeper in the source domain. In addition, we see that lowering Δt increases the slope again. Thus, Δt has to be chosen in a way that the steepest gradient in data can be represented sufficiently. This, however, is very much problem dependent and requires knowledge of the data. In Sect. 4, we assess an approach on how to determine this in the context of turbulent eddies.

For the second point, a more general statement can be made, since this observation is nearly independent of Δt and only becomes more prominent if Δt becomes sufficiently large. Higher order polynomial approximations tend to oscillate, especially for equispaced point distribution which is the case for temporal sampling. Therefore, in Fig. 9, polynomial interpolation is only shown up to second degree. In addition, the well-known Spline interpolation tends to oscillations for high Δt. Most favorable thus is linear or Akima interpolation, which both represent the vertical jump best and recover the steepest gradients. Higher order polynomial and spline interpolation in this case fail mainly because the physical time steps dt at which we sample are roughly equispaced. Thus, we see the so-called Runge's phenomenon for interpolation using an equispaced point set in time. This is a crucial point since the TAU output frequency is only determined by its implicit timestep. The target solver FLEXI thus has to recover the data in every explicit time step. However, having a fixed source sampling rate and decreasing the target time step, will not further increase the error since the interpolant is only determined by the sampling rate of the source data and is only evaluated during FLEXI runtime.

Since Akima interpolation yields slightly smoother results in combination with steeper gradients, we use Akima interpolation for all following test cases.

3.3 Convergence of the spatial mapping

Another important aspect one has to consider is the convergence behavior of the mapping process. To measure the effect of the interpolation routines, we decided to calculate the error norm of the whole mapping and simulation process. Thus, the error includes spatial and temporal interpolation errors as well as the error associated with imposing the instantaneous boundary condition in FLEXI (e.g. discretization error).

To run the convergence test, we modify the initial conditions from the one-dimensional sine in Fig. 8. We add a y and z dependency to the exact solution u in order to have varying u values on the interface plane. Thus, we get

$$u(x, y, z, t) = \sin(\pi(x - a_1 t)) + sin(\pi y) + sin(\pi z). \tag{11}$$

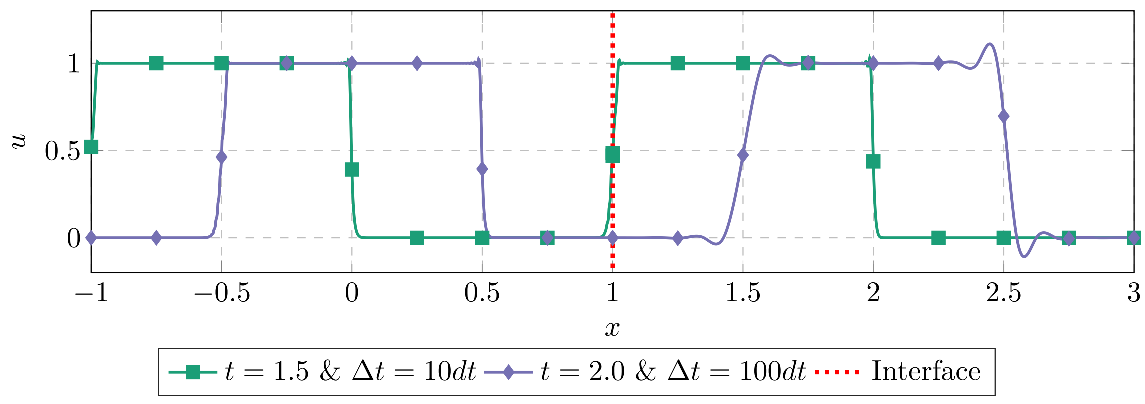
For the sine wave and the linear transport, we have seen earlier that we can recover the exact solution on the target domain and that the information is propagated correctly via the instantaneous boundary condition if there is no spatial and temporal interpolation involved (just copy the values from source to target). Thus, we want to investigate the effect of different non-matching interfaces (point sets and resolution) on the error of the simulation and, therefore, have to combine spatial and temporal interpolation techniques for the first time. We again use Legendre–Gauss–Lobatto points with $N = 5$ in the source and target domain. In addition, we use super-sampling with $M = 8$. For the linear transport, this is not necessary, since in contrast to the Navier–Stokes equations, we do not see aliasing here. However, we want to assess the convergence as close to the later application as possible and additionally avoid matching all the degrees of freedom in any case ($N_{\text{src}} = 5 \neq 8 = M_{\text{tar}}$).

In Fig. 10, we see two different testing scenarios. The first in Fig. 10a shows the L_2-error for increasing target resolution and a fixed source mesh. The second scenario in Fig. 10b depicts the error for an increasing source resolution and a fixed target mesh. With "grid", we mean the number of elements in each spatial direction of the Cartesian cube. The sampling timestep is defined by the physical timestep dt of the source data. Thus, for, e.g. the source grid "1", we extract the interface data at every physical time step and use Akima interpolation to interpolate it to the physical time step of the "32" target grid that is 32 times smaller.

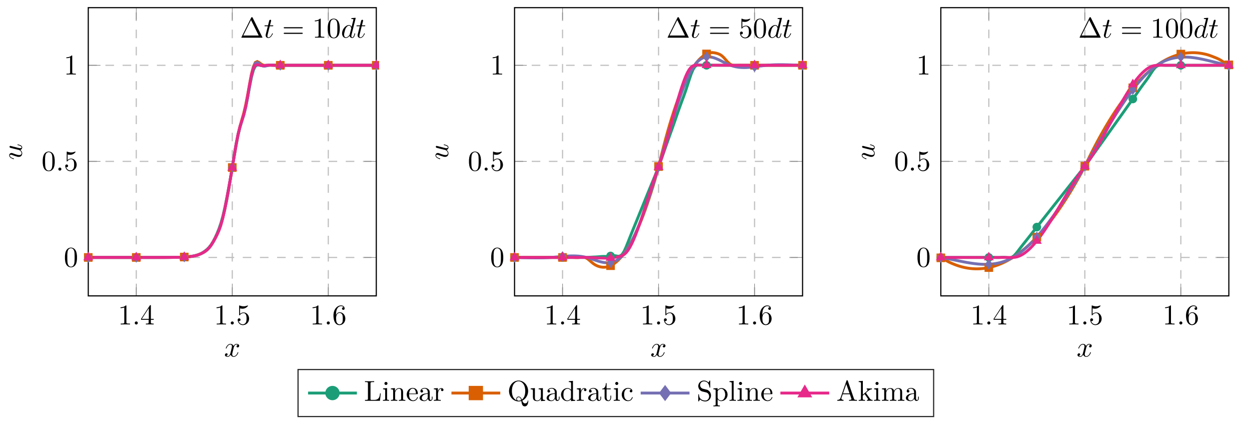
In Fig. 10a, we assess the effect of varying target mesh resolutions on the overall error. The resolution of the source mesh is fixed at $8 \times 8 \times 8$ and $32 \times 32 \times 32$ respectively. We expect the overall error to converge, since the error cannot be mitigated any further if it is dominated by the source data. Thus, we can see the influence of the source data on the target domain. For the $8 \times 8 \times 8$ source mesh, we can observe this behavior very well. Starting at $\text{grid}_{\text{tar}} \approx 4$, we see that for all interpolation algorithms, there is no further improvement. For the finer source mesh, we can observe a similar behavior, however, the overall error is lower and the error is converged later. Due to the error introduced with the spatial interpolation, we cannot see a decreasing error until the source mesh resolution.

In Fig. 10b, we investigate the effect of a varying source mesh on a fixed target mesh. This can be interpreted as increased input quality for the mapping for a given target domain. In this test case, we again assessed the influence for two fixed target resolutions at $8 \times 8 \times 8$ and $32 \times 32 \times 32$. Nearest neighbor, Shepard as well as RBF interpolation show decreasing errors for increasing source grid resolution. This time nearly linear decaying errors can be seen up to the resolution of the target mesh. However, especially for the $\text{grid}_{\text{tar}} = 8 \times 8 \times 8$ case, we can see that RBF interpolation is capable of recovering information from source grids with finer resolution than the target mesh. Shepard and nearest neighbor show clearly weaker performance here and have changing slopes of the error in this source grid regime.

Overall, we can rank the performance of the three tested spatial interpolation techniques. Nearest neighbor interpolation shows as expected the weakest performance with an experimental order of convergence of EOC ≈ 1.2 in Fig. 10b. Shepard interpolation shows overall lower errors at roughly the same order of convergence EOC ≈ 1.5. However, Shepard interpolation is capable of retaining the error even for source resolutions higher than the target resolution in Fig. 10b where nearest neighbor interpolation shows inconsistent results. Finally, radial basis function interpolation clearly yields the best results with an order of convergence of EOC ≈ 2.8. Thus, using RBF interpolation is recommended. Overall, the results in Fig. 10b underline the



(a) Overview over the simulation domain of the jump test case using spline interpolation. The jumps are shown in more detail in Fig. 9b.



(b) Detailed view of the jumps containing the interpolation techniques visualized in 9a.

Fig. 9 Overview and detailed plots of the jump test case for different Δt and temporal interpolation algorithms

observations made in Fig. 6. However, the convergence rates in Fig. 10b are lower for all interpolation methods. The qualitative observations, however, are identical and the losses in EOC are equivalent for all interpolation techniques. One should note that the test case in Fig. 10b has an increased complexity, since it is two-dimensional and we evaluate the error over the whole mapping process compared to an isolated one-dimensional interpolation test in Fig. 6.

For large source datasets, one should keep in mind that solving the equation system necessary to the get the interpolation coefficients for the radial basis functions gets very expensive and RBF interpolation even in this simple test case was noticeably (approx. up to an order of magnitude) slower than nearest neighbor and Shepard interpolation.

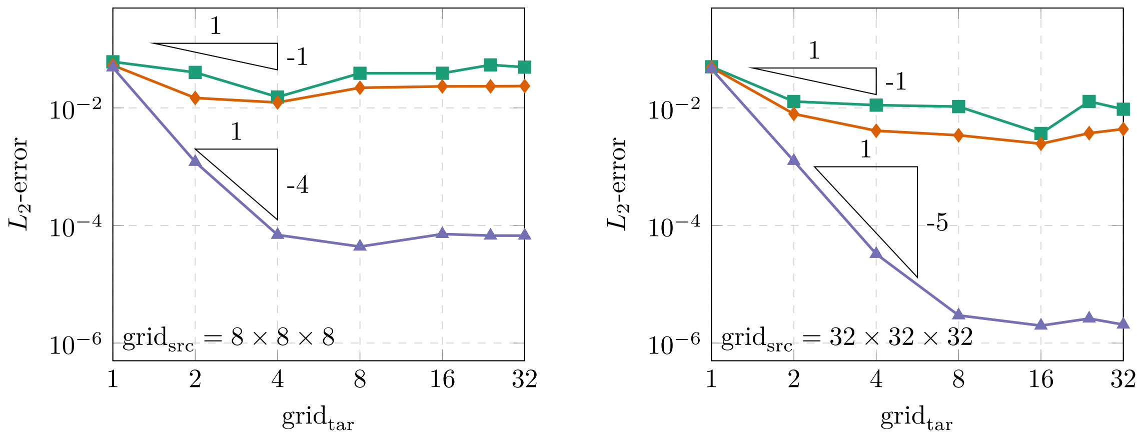
4 Results: cylinder flow

In this section, we investigate the flow around a cylinder at a Reynolds number of $\mathrm{Re}_c = 3900$ [30, 31]. The diameter of the cylinder is defined as c and is used as the characteristic length in this investigation. The domain has a spanwise extension of c. For the first time, we now map actual TAU surface data into a FLEXI domain.

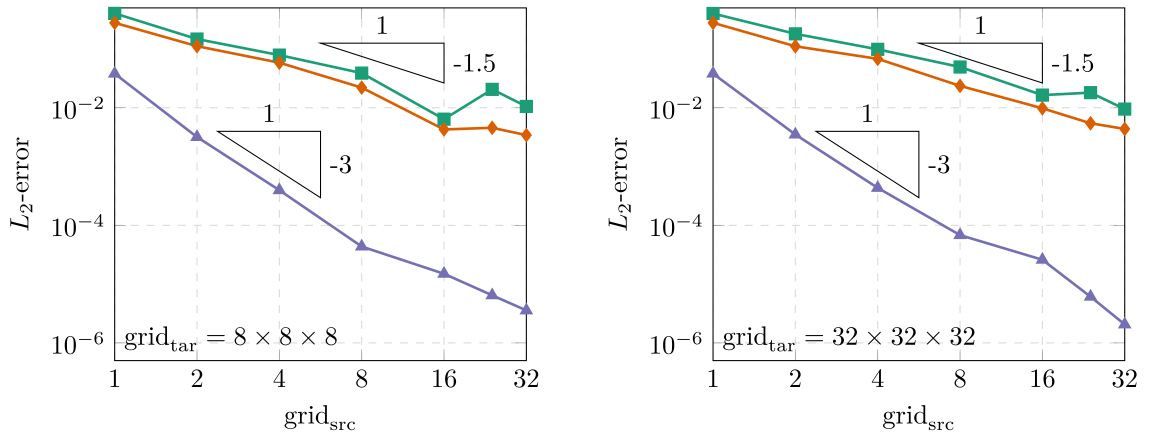
4.1 Generating the reference data

The main reason we chose the cylinder flow as the main test case is the fact, that we can afford to run the whole domain in FLEXI and in TAU. Thus, we not only can compare the mapped results against the TAU solution but also against the reference DNS created with FLEXI. In addition, the cylinder is a well-known geometry in the fluid mechanics community and has been investigated in detail before.

We use the same number of degrees of freedom in the TAU source and the appended FLEXI target mesh. Thus, the target mesh resolution including the interface has to be divided by a factor of eight in order to accommodate for the higher polynomial degree of FLEXI $N = 7$. With this approach, we minimize the resulting errors (cf. Fig. 10a). We will show later that this resolution is sufficient to map all physical structures occurring in this test case.



(a) Convergence behavior for the linear scalar advection equation system for two different source meshes.



(b) Convergence behavior for the linear scalar advection equation system for two different target meshes.

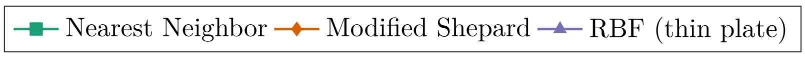


Fig. 10 Comparison of the convergence behavior of the entire mapping procedure including spatial, temporal mapping and the instantaneous boundary condition

4.1.1 Simulation setup

Next, we discuss the simulation setup of the cylinder test case. We describe the setup for FLEXI as well as TAU.

We use the same base mesh setup for the TAU and FLEXI DNS reference simulation. The only difference is again that the FLEXI case is coarser by a factor of eight to consider the high-order polynomials that are used in each element. Thus, if FLEXI is run with $N = 7$, we have the same number of degrees of freedom as TAU. We also investigate the solution quality of lower order polynomials later. For the simulation in FLEXI, we use $N \in [3, 5, 7]$ polynomials on the same grid.

Due to the limitation of FLEXI to hexahedral elements, we compare the amount of DOFs in the structured boundary layer area and the refined wake area since the amount of DOFs in the free stream area differs due to the different element types. For FLEXI with $N = 7$ and TAU, we solve the governing equations at approx. 31 Mio. points. The overall mesh has a radius of $100c$ using periodic boundary conditions in spanwise direction and far field boundary conditions on the outer edge of the mesh. The cylinder itself is modeled using an adiabatic no-slip wall.

For the FLEXI simulation, we apply the BR1 lifting procedure [32] to take the parabolic terms into account. We use kinetic-energy-preserving skew-symmetric splitting of the advective terms of the compressible Navier–Stokes equations according to Pirozzoli [33] to mitigate aliasing instabilities. The Roe numerical flux is used to solve the Riemann problems at the cell interfaces [9]. We use the Vreman model for explicit sub grid scale modeling [34] with $C = 0.11$. The resulting semi-discrete system is advanced in time by applying an explicit-in-time low storage Runge–Kutta method with optimized stability region [35].

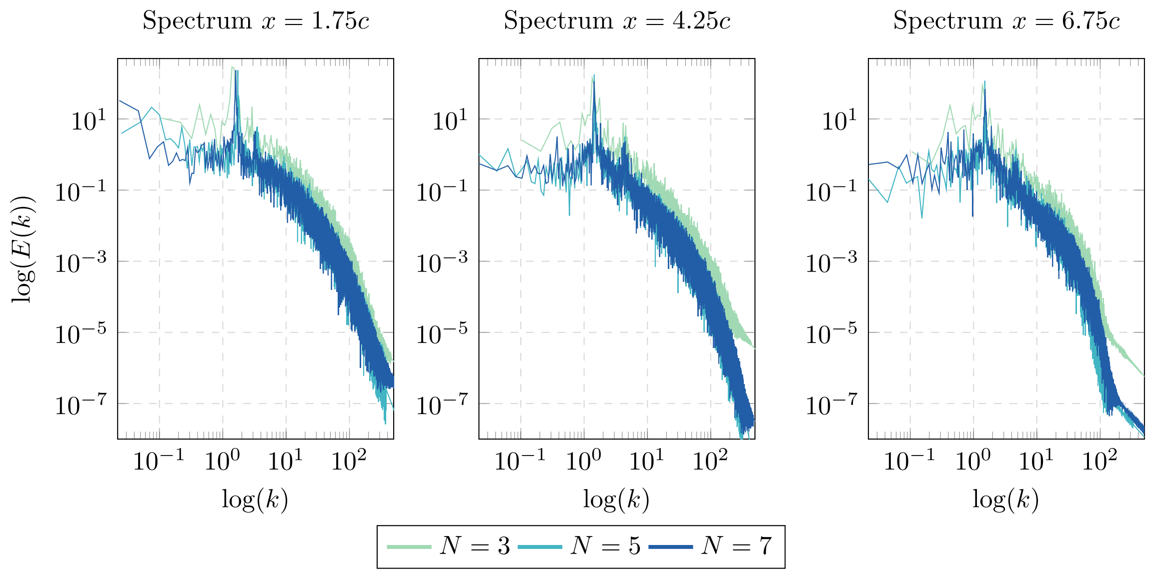


Fig. 11 Comparison of the turbulent kinetic energy of different polynomial degrees N. Taken from the FLEXI simulation of the full domain

We also performed a hybrid RANS/LES simulation with TAU using the zonal AZDES (automated zonal detached eddy simulation) method, which was developed at the Institute of Aerodynamics and Gas Dynamics (IAG) [36, 37]. Within this method, an integral turbulent length scale in the flow field is calculated from a precursor unsteady RANS simulation based on the modeled turbulent quantities, which reaches high values in the areas of separated flow. All areas where this turbulent length scale is greater than a cutoff value chosen by the user, which is based on the mesh resolution, are then marked as DDES [38] zones for the following actual hybrid simulation, as the mesh is fine enough here to resolve the larger turbulent scales. The rest of the flow field is marked for (unsteady) RANS computation, which includes attached boundary layers. In addition, the shielding of the boundary layer can be fine-tuned by enforcing RANS for all positions closer to the airfoil or body surface than a chosen distance. Similarly, LES mode can be forced for all distances greater than a set limit. We use a second-order central flux computation for the convective terms of the RANS equations, stabilized by artificial dissipation (matrix dissipation), applying a low-dissipative skew-symmetric scheme according to Kok [39]. For the turbulence model equations, a second-order upwind scheme according to Roe is employed. For the temporal discretization, a second-order dual time stepping method [40] is used which is based on the implicit BDF2 (backward differentiation formula) scheme. The final physical time step is chosen to 1/150 of the convective time scale, which leads to a CFL number of one in the wake area of the cylinder to reach sufficient temporal resolution for LES. The turbulence is modeled by the SSG/LRR-ω Reynolds stress model [41], which acts as sub grid scale model in the LES zones.

4.1.2 Sensitivity on resolution

First, we assess the sensitivity of the test case on the resolution. We conduct this study in FLEXI since we are interested if the chosen resolution is sufficient for a DNS. This test case was specifically chosen since it allows to conduct a fully resolved simulation in FLEXI and in TAU. For typical applications of the inflow condition, this will not be possible.

In Fig. 11, the spectra of the turbulent kinetic energy are shown at three discrete points in the wake of the cylinder. Each figure contains the spectrum for three simulations using different polynomial degrees on the same grid.

The $N = 3$ spectrum in Fig. 11 shows a deviation from the $N = 5$ and $N = 7$ curves at all evaluation locations. Thus, we can assume that the resolution for $N = 3$ is not sufficient for a DNS and does not yield enough dissipation. Since the turbulent kinetic energy spectra for $N = 5$ and $N = 7$ coincide, we can assume that we are converged at this resolution and thus $N = 5$ is sufficient for running a DNS. The expected Strouhal frequency of the cylinder is clearly visible as a distinct peak in the spectrum [31] for all polynomial degrees.

The simulation with $N = 7$ (same number of degrees of freedom as TAU mesh) is too fine for a typical LES/DNS since the mesh was originally created to be suitable for a hybrid RANS/LES and a FV code. Evaluating the viscous wall spacing in FLEXI yields $y^{+} \approx 0.01$ which is more than sufficient, even for a DNS. This, however, underlines the benefits of a high-order scheme when resolving turbulent eddies.

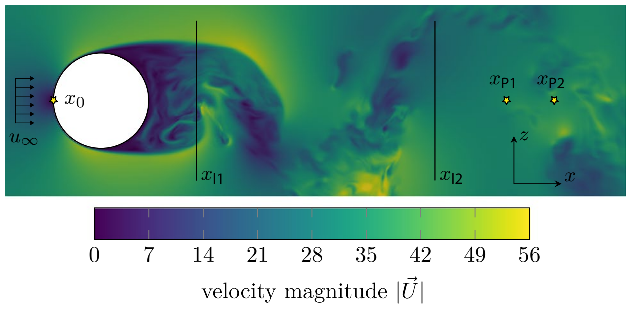


Fig. 12 Cylinder at $Re_c = 3900$ test case definition containing interface planes x_I for the coupled simulation and probe points for evaluation x_P

As already shown in Fig. 2 using the same amount of DOFs in FLEXI and in TAU with a high polynomial degree in FLEXI shows the strength of the high-order scheme, since this resolution is sufficient in FLEXI to run a DNS.

4.2 Coupled simulation

We now continue assessing the FLEXI simulation using the TAU data as an inflow condition. The target mesh in this case is a simple box that has the same y and z dimensions as the interface and a sufficiently long x extension for the turbulent wake to develop and travel. In $\pm y$, the mesh has periodic and in $\pm z$ Euler wall boundary conditions. The inflow is realized using the mapped TAU data. At $+x$, a subsonic outflow condition is used. The computational domain of the target FLEXI mesh has an extension of

$$\Delta_x = 10c, \quad \Delta_y = c \quad \text{and} \quad \Delta_z = 4c.$$

In Fig. 12, the simulation setup is depicted. Note that the size of the interface planes in Fig. 12 at x_I does not match the size in the actual simulation. In the setup, the interface planes are designed in a way that all the turbulent wake structures are captured by the planes and all vortical structures of the wake are fully contained in the interface planes.

The most important aspects for assessing the performance of the interface is to define the interface locations and to define record points (also known as probe points). We decide to place two interface planes at position $x_I \in [1.5c, 4c]$ in the wake as well as two record points $x_P \in [4.75c, 5.25c]$.

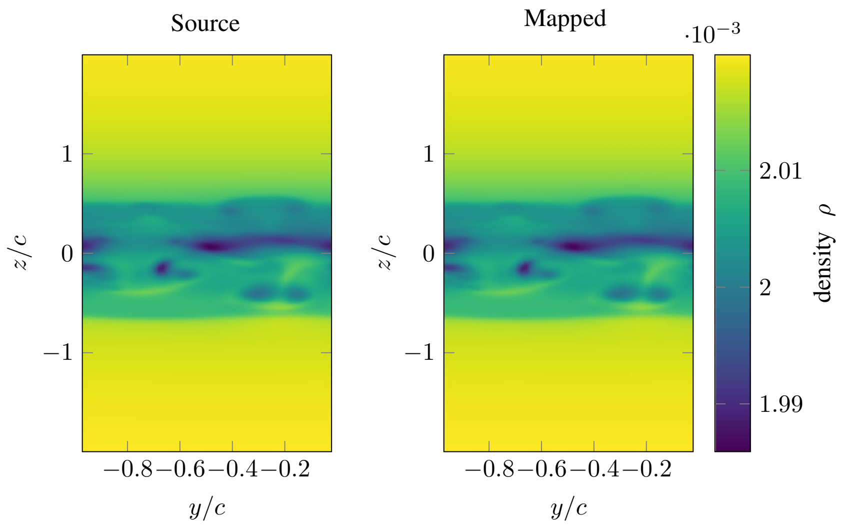
4.3 Interpolation error

Next, we assess the interpolation error resulting from interpolating a wake plane ($\text{grid}_{\text{src}} = 573 \times 64$) as defined in Fig. 12 onto the FLEXI boundary condition ($\text{grid}_{\text{tar}} = 32 \times 8$ with $N = 7$) using the modified Shepard method. We investigate the error for plane x_{I1}, which is the one that is located closest to the cylinder. The error is assessed using the TAU source data as reference data. To get the same point set for TAU and FLEXI, we evaluate the polynomials of the mapped FLEXI solution on the TAU solution points. The results are visualized in Fig. 13.

Qulitatively, the results in Fig. 13 look very convincing. The structures of the source data are all represented in the mapped solution. Taking a closer look, one can see small overshoots of the mapped solution at the element boundaries. This effect has already been mitigated using a supersampling as dealiasing technique. To quantify the error, we look at the difference between the source and the mapped data visualized in Fig. 14.

Thus, we evaluate the error of the x-momentum for three resolutions $\text{grid}_{\text{tar}} = 32 \times 8$, $\text{grid}_{\text{tar}} = 16 \times 4$ and $\text{grid}_{\text{tar}} = 8 \times 2$. The plots show the relative deviation of the interpolated target data based on the source data. The x-momentum plot confirms the observations we made in Fig. 13 and shows a very small error in the whole domain. The error increases as expected for coarser resolutions. Calculating the L_2-errors for all

Fig. 13 Comparison of the instantaneous flow fields of a cylinder wake state at the interface plane $x_{I1} = 1.5c$. Left source data (TAU) on 573×64 points and right target grid of 32×8 elements of degree $N = 7$



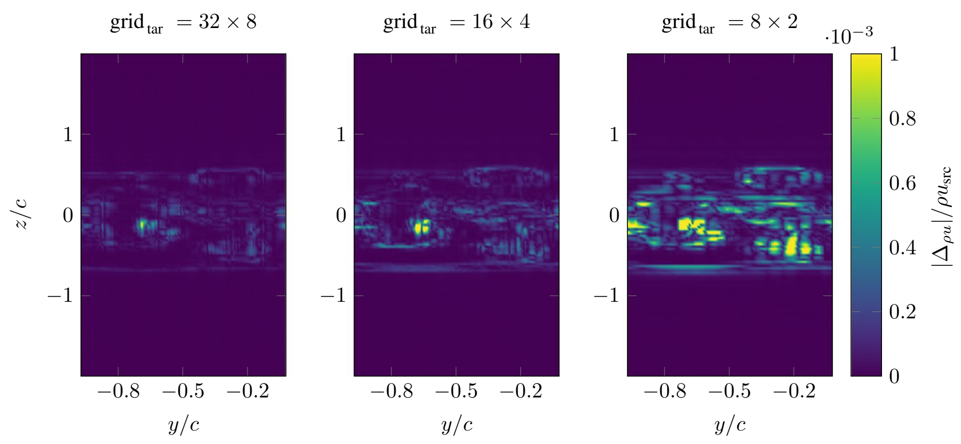


Fig. 14 Relative errors in instantaneous x-momentum in the cylinder wake at $x_{I1} = 1.5c$ plane, for different target grid sizes

three meshes yields L_2-error$_{\rho u}(32 \times 8) = 9.65\text{E}-08$, L_2-error$_{\rho u}(16 \times 4) = 3.03\text{E}-07$ and L_2-error$_{\rho u}(8 \times 2)$ yielding an convergence rate of $EOC = 1.31$ which is in line with our findings from Fig. 10b for Shepard's method. Especially in the part containing eddies in the middle of the interface planes, we see large errors at the eddy boundaries.

In Table 1, we can see the minimal and maximal values of the primitive variables for source and mapped data. In addition, the integral mean value is listed. One can note that the mapping yields very good results for density and pressure. In contrast, especially for the velocities, we see small deviations from source data. This error is based on the fact that the interpolation is not conservative and on the magnitude of the variations in the instantaneous source fields. This gets especially pronounced for the velocity components due to changing signs and the fact that velocity variations are much larger compared to their mean in contrast to density and pressure. However, by applying the conversion of primitive to conservative variables after interpolation, we ensure that—despite the interpolation itself not guaranteeing conservativity—we get consistent conservative variables.

Hence, due to flexibility of the scheme and the generally very small effect on the mapped results, we can neglect the effects of the non-conservativity (cf. Table 1) and directly use the mapped plane as an inflow condition.

4.4 Influence of the sampling rate

Now, we assess the effect of the sampling rate in time of the source data on the quality of the solution in the target domain, which is a very important user parameter that has to be considered when creating a coupled simulation. We do so by investigating the effect on the contribution of the incoming turbulence on the turbulent kinetic energy.

In Fig. 15, the turbulent kinetic energy spectra at two distinct probe points are visualized. The different colors depict different temporal sampling. With N_{Skip}, we mean how many TAU snapshots are skipped in time. $N_{\text{Skip}} = 1$ means that every temporal snapshot is used. The physical TAU sampling rate is ~ 150 snapshots per characteristic time. The characteristic time is defined as the time it takes the fluid to cover the distance of the diameter of the cylinder. For $N_{\text{Skip}} = 2$, we only use every second snapshot. The lighter the color gets the fewer snapshots are used to recover the TAU solution in FLEXI.

Figure 15 shows that the results are heavily dependent on the sampling rate. This seems reasonable since the sampling rate determines which structures are mapped via the instantaneous boundary condition. According to the Nyquist criterion, there is a value for N_{Skip} for which the solution is not represented anymore. In this case for $N_{\text{Skip}} \geq 512$, we

Table 1 Minimum, maximum and integral mean values of the primitive variables resulting from mapping the TAU source data ($\text{grid}_{\text{src}} = 573 \times 64$) onto interface plane $x_{I1} = 1.5c$ ($\text{grid}_{\text{tar}} = 16 \times 4$, $N = 7$)

	ρ	u	v	w	p
Mean					
Source	2.014E−03	2.806E+01	2.001E−01	− 6.915E−01	1.578E+02
Mapped	2.014E−03	2.803E+01	2.013E−01	− 6.888E−01	1.578E+02
Min.					
Source	1.986E−03	− 2.975E+01	− 3.402E+01	− 2.510E+01	1.553E+02
Mapped	1.986E−03	− 2.978E+01	− 3.333E+01	− 2.508E+01	1.553E+02
Max.					
Source	2.020E−03	4.800E+01	3.178E+01	2.967E+01	1.583E+02
Mapped	2.020E−03	4.787E+01	3.153E+01	2.932E+01	1.583E+02

no longer see agreement with the reference solution. For smaller N_{Skip}, there is better agreement with the black reference solution (FLEXI $N = 7$ DNS). Hence, two major observations can be made. First, for high N_{Skip}, the major flow structures cannot be recovered and even the Strouhal frequency is not represented correctly. In addition, after some development in the target domain at $x = 5.25c$, we can see that there is a lot of disagreement even for low k. Second, we can observe that the energy does not adapt and we lose energy in high modes for large N_{Skip}.

From these observations, we can conclude that the sampling frequency is dependent on the structures that have to be mapped to the new domain. Thus, we define a measure to quantify the "eddy size - sampling rate" relation which is closely related to the underlying spatial discretization scheme. From the literature (e.g. [1, 10]), we know that there is a similar criterion for spatial discretization, which uses the parameter numbers per wavelength n_{PPW} to quantify the property of a spatial discretization scheme in resolving multi-scale structures. For DGSEM, it is known that $n_{\text{PPW,DGSEM}} \gtrapprox 6$ for the polynomial degrees used in this paper.

In this case, we take two sizes as reference. First, according to [42], the large structures are of the size of the cylinder which corresponds to $L = 1c$. From the simulation setup and the properties of the DG scheme, we estimate the smallest structures according to

$$l = \frac{L_{\text{domain}}}{\#\text{DOF} \cdot n_{\text{PPW,DGSEM}}} \approx 0.06c. \tag{12}$$

with L_{domain} denoting the size of the domain and #DOF the number of DOFs used to discretize the domain. Taking u_∞ into account, we get an approximation for how long it takes an eddy to be advected over the interface plane, assuming Taylor's hypothesis [43]. Taking the sampling frequency into account, we can estimate that for the smallest structures, we need $N_{\text{Skip}} \approx 4$ and for the large structures $N_{\text{Skip}} \approx 64$ is sufficient. This behavior for $L = c$ is also underlined in Fig. 15. Only using every 64th sample $N_{\text{Skip}} = 64$ still provides us with the main structures and correct amplitudes, while $N_{\text{Skip}} > 64$ shows signs of under-resolution. Using this information, we can approximate a criterion on how many points we need per structure/eddy that has to be transported over the interface. It turns out that for both large and small eddies we need approximately 2.3 samples per eddy. As one would expect, we can conclude that spatial and temporal discretization requirements are similar for the interface.

We repeated this evaluation for both interface planes x_{I1} and x_{I2}. Both showed qualitatively identical results.

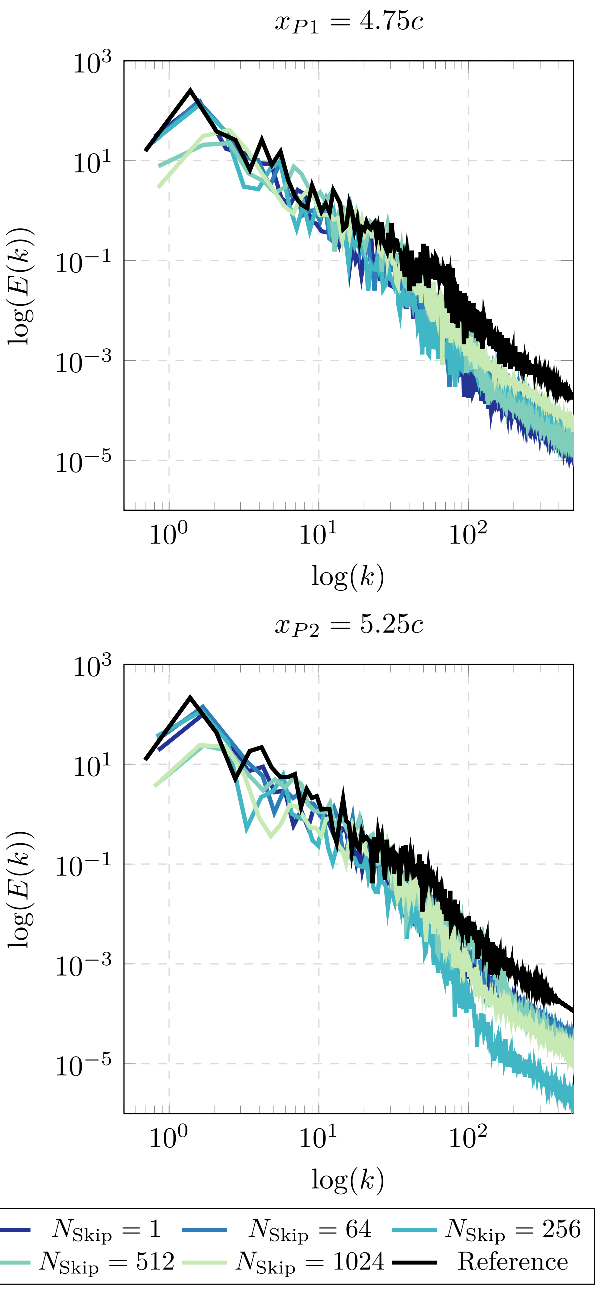


Fig. 15 Turbulent kinetic energy spectra at two distinct probe points, taken from the FLEXI target domain at two distinct probe points x_{P1} and x_{P2} in the wake with varying sampling rate of the TAU inflow data. The black reference is generated calculating the energy spectrum from the FLEXI $N = 7$ DNS on the full mesh at the same points

5 Summary

In this work, we introduced a method to generate an instantaneous boundary condition relying on a precursor simulation. We presented the numerical methods necessary to handle differences in spatial and temporal discretization via interpolation. The scheme is validated for simple test cases and a more complex cylinder wake.

We have shown how to generate numerically stable inflow and initial conditions with the methods described in this paper that are universally applicable also to other solvers than TAU and even experimental data.

The requirements regarding sampling rate are similar to those of the spatial discretization and thus need approximately four sampling points per wavelength, depending on the temporal interpolation scheme used.

We implemented several mapping techniques and showed the differences in interpolation quality and additionally demonstrated their capabilities of reconstructing scattered source data. In addition, we utilized super-sampling of the interpolation to increase the overall accuracy and to mitigate the errors due to aliasing and numerical incompatibilities.

In terms of spatial resolution difference at the interface, we observed that increasing the resolution of the source data never posed a problem. However, coarsening the data too much can produce large aliasing errors which cause trouble for the high-order scheme. Thus, we recommend at least having the same amount of target sampling points and source points on the interface.

The introduced interface now has to be applied to more complex scenarios. In a next step, we thus plan to apply the coupling between TAU and FLEXI to the tandem wing configuration test case visualized in Fig. 1. That simulation was done using TAU only and provides the capabilities of efficiently using FLEXI for subdomain simulations. In a future work, we hence aim to investigate the effects of the turbulent wake onto the boundary layer of the HTP. The toolchain introduced in this paper is already designed to handle these kind of challenging simulations. Another future application of the interface is the adaption of the framework to more solvers in order to further enhance the capabilities of the toolchain and increase the amount of use cases.

Acknowledgements The authors gratefully acknowledge the Deutsche Forschungsgemeinschaft DFG (German Research Foundation) for funding this work in the framework of the research unit FOR 2895 (grant BE 6100/3-1). We also thank the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project (GCS-lesdg) by providing computing time on the GCS Supercomputer HAWK at Höchstleistungsrechenzentrum Stuttgart (www.hlrs.de).

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Flad, D., Beck, A., Guthke, P.: A large eddy simulation method for DGSEM using non-linearly optimized relaxation filters. J. Comput. Phys. (2020). https://doi.org/10.1016/j.jcp.2020.109303
2. Jarrin, N., Benhamadouche, S., Laurence, D., Prosser, R.: A synthetic-eddy-method for generating inflow conditions for large-eddy simulations. Int. J. Heat Fluid Flow **27**(4), 585–593 (2006). https://doi.org/10.1016/j.ijheatfluidflow.2006.02.006
3. Jarrin, N., Prosser, R., Uribe, J.C., Benhamadouche, S., Laurence, D.: Reconstruction of turbulent fluctuations for hybrid RANS/LES simulations using a synthetic-eddy method. Int. J. Heat Fluid Flow **30**(3), 435–442 (2009). https://doi.org/10.1016/j.ijheatfluidflow.2009.02.016
4. Lund, T.S., Wu, X.H., Squires, K.D.: Generation of turbulent inflow data for spatially-developing boundary layer simulations. J. Comput. Phys. **140**(2), 233–258 (1998). https://doi.org/10.1006/jcph.1998.5882
5. Kuhn, T.: Quantification of Uncertainty in Aeroacoustic Cavity Noise Simulations with a Discontinuous Galerkin Solver. Thesis (2021). ISBN: 978-3-8439-4784-8
6. Kempf, D., Munz, C.-D.: Zonal hybrid computational aeroacoustics simulation of trailing edge noise using a high-order discontinuous Galerkin method (2022). https://doi.org/10.2514/6.2022-3021
7. Krais, N., Beck, A., Bolemann, T., Frank, H., Flad, D., Gassner, G., Hindenlang, F., Hoffmann, M., Kuhn, T., Sonntag, M., Munz, C.-D.: FLEXI: a high order discontinuous Galerkin framework for hyperbolic-parabolic conservation laws. Comput. Math. Appl. **81**, 186–219 (2021). https://doi.org/10.1016/j.camwa.2020.05.004
8. Schwamborn, D., Gardner, A.D., von Geyr, H., Krumbein, A., Lüdecke, H., Stürmer, A.: Development of the TAU-code for aerospace applications. In: 50th NAL International Conference on Aerospace Science and Technology, Bangalore, India (2008). https://elib.dlr.de/55519/
9. Roe, P.L.: Approximate Riemann solvers, parameter vectors, and difference-schemes. J. Comput. Phys. **43**(2), 357–372 (1981). https://doi.org/10.1016/0021-9991(81)90128-5
10. Gassner, G.J., Winters, A.R., Hindenlang, F.J., Kopriva, D.A.: The BR1 scheme is stable for the compressible Navier–Stokes equations. J. Sci. Comput. **77**(1), 154–200 (2018). https://doi.org/10.1007/s10915-018-0702-1
11. Blind, M., Kopper, P., Kempf, D., Kurz, M., Schwarz, A., Beck, A., Munz, C.-D.: Performance improvements for large scale simulations using the discontinuous Galerkin framework FLEXI. In: Accepted by High Performance Computing in Science and Engineering'22. Springer, Cham (2022)
12. Gassner, G., Kopriva, D.A.: A comparison of the dispersion and dissipation errors of Gauss and Gauss–Lobatto Discontinuous Galerkin spectral element methods. SIAM J. Sci. Comput. **33**(5), 2560–2579 (2011). https://doi.org/10.1137/100807211
13. Spiegel, S.C., Huynh, H.T., DeBonis, J.R.: A survey of the isentropic Euler vortex problem using high-order methods (2015). https://doi.org/10.2514/6.2015-2444

14. Jameson, A.: Origins and further development of the Jameson–Schmidt–Turkel scheme. AIAA J. **55**(5), 1487–1510 (2017). https://doi.org/10.2514/1.J055493
15. Swanson, R.C., Turkel, E.: On central-difference and upwind schemes. J. Comput. Phys. **101**(2), 292–306 (1992). https://doi.org/10.1016/0021-9991(92)90007-L
16. Löwe, J., Probst, A., Knopp, T., Kessler, R.: Low-dissipation low-dispersion second-order scheme for unstructured finite volume flow solvers. AIAA J. **54**(10), 2961–2971 (2016). https://doi.org/10.2514/1.J054956
17. Flad, D., Beck, A., Munz, C.-D.: Simulation of underresolved turbulent flows by adaptive filtering using the high order discontinuous Galerkin spectral element method. J. Comput. Phys. **313**, 1–12 (2016). https://doi.org/10.1016/j.jcp.2015.11.064
18. Spiering, F.: Development of a fully automatic chimera hole cutting procedure in the dlr tau code. New Results in Numerical and Experimental Fluid Mechanics X. In: Notes on Numerical Fluid Mechanics and Multidisciplinary Design, vol. 132, pp. 585–595. Springer (2016). https://doi.org/10.1007/978-3-319-27279-5_51
19. NetCDF, Unidata: The NetCDF Classic Format Specification (2022). https://docs.unidata.ucar.edu/netcdf-c/current/file_format_specifications.html
20. Hindenlang, F.: Mesh curving techniques for high order parallel simulations on unstructured meshes. Thesis (2014). https://doi.org/10.18419/opus-3957
21. Laughton, E., Tabor, G., Moxey, D.: A comparison of interpolation techniques for non-conformal high-order discontinuous Galerkin methods (2020). arXiv:2007.15534
22. Kopper, P., Copplestone, S., Pfeiffer, M., Koch, C., Fasoulas, S., Beck, A.: Hybrid parallelization of Euler–Lagrange simulations based on MPI-3 shared memory. Advances in Engineering Software. 174, 09659978 (2022). https://doi.org/10.1016/j.advengsoft.2022.103291
23. Appel, D., Jöns, S., Keim, J., Müller, C., Zeifang, J., Munz, C.-D.: A Narrow Band-Based Dynamic Load Balancing Scheme for the Level-Set Ghost-Fluid Method. Springer, Cham (2022). (**In press**)
24. Mueller, T.G., Pusuluri, N.B., Mathias, K.K., Cornelius, P.L., Barnhisel, R.I., Shearer, S.A.: Map quality for ordinary Kriging and inverse distance weighted interpolation. Soil Sci. Soc. Am. J. **68**(6), 2042–2047 (2004). https://doi.org/10.2136/sssaj2004.2042
25. Shepard, D.: A two-dimensional interpolation function for irregularly-spaced data (1968). https://doi.org/10.1145/800186.810616
26. Hardy, R.L.: Multiquadric equations of topography and other irregular surfaces. J. Geophys. Res. **76**(8), 1905 (1971). https://doi.org/10.1029/JB076i008p01905
27. Hardy, R.L.: Theory and applications of the multiquadric biharmonic method—20 years of discovery 1968–1988. Comput. Math. Appl. **19**(8–9), 163–208 (1990). https://doi.org/10.1016/0898-1221(90)90272-L
28. Kopriva, D.A.: Implementing Spectral Methods for Partial Differential Equations: Algorithms for Scientists and Engineers. Scientific Computation. Springer, Dordrecht (2009). https://doi.org/10.1007/978-90-481-2261-5
29. Akima, H.: A new method of interpolation and smooth curve fitting based on local procedures. J. ACM **17**(4), 589 (1970). https://doi.org/10.1145/321607.321609
30. Kravchenko, A.G., Moin, P.: Numerical studies of flow over a circular cylinder at $Re_D = 3900$. Phys. Fluids **12**(2), 403–417 (2000). https://doi.org/10.1063/1.870318
31. Parnaudeau, P., Carlier, J., Heitz, D., Lamballais, E.: Experimental and numerical studies of the flow over a circular cylinder at Reynolds number 3900. Phys. Fluids (2008). https://doi.org/10.1063/1.2957018
32. Bassi, F., Rebay, S.: A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations. J. Comput. Phys. **131**(2), 267–279 (1997). https://doi.org/10.1006/jcph.1996.5572
33. Pirozzoli, S.: Generalized conservative approximations of split convective derivative operators. J. Comput. Phys. **229**(19), 7180–7190 (2010). https://doi.org/10.1016/j.jcp.2010.06.006
34. Vreman, A.W.: An eddy-viscosity subgrid-scale model for turbulent shear flow: algebraic theory and applications. Phys. Fluids **16**(10), 3670–3681 (2004). https://doi.org/10.1063/1.1785131
35. Niegemann, J., Diehl, R., Busch, K.: Efficient low-storage Runge–Kutta schemes with optimized stability regions. J. Comput. Phys. **231**(2), 364–372 (2012). https://doi.org/10.1016/j.jcp.2011.09.003
36. Ehrle, M., Waldmann, A., Lutz, T., Krämer, E.: An automated zonal detached eddy simulation method for transonic buffet. In: Hoarau, Y., Peng, S.-H., Schwamborn, D., Revell, A., Mockett, C. (eds.) Progress in Hybrid RANS-LES Modelling, pp. 271–281. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-27607-2_22
37. Schulte am Hülse, S.A.: Simulation transsonischen Buffets an Transportflugzeugen mittels hybrider RANS-/LES Verfahren. PhD thesis, Institute of Aerodynamics and Gas Dynamics, University of Stuttgart (2016). ISBN: 978-3-8439-2727-7
38. Spalart, P.R., Deck, S., Shur, M.L., Squires, K.D., Strelets, M.K., Travin, A.: A new version of detached-eddy simulation, resistant to ambiguous grid densities. Theor. Comput. Fluid Dyn. (2006). https://doi.org/10.1007/s00162-006-0015-0
39. Kok, J.C.: A high-order low-dispersion symmetry-preserving finite-volume method for compressible flow on curvilinear grids. J. Comput. Phys. (2009). https://doi.org/10.1016/j.jcp.2009.06.015
40. Jameson, A.: Time dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings. In: 10th Computational Fluid Dynamics Conference, Honolulu, Hawaii (1991). https://doi.org/10.2514/6.1991-1596
41. Eisfeld, B., Brodersen, O.: Advanced turbulence modelling and stress analysis for the DLR-F6 configuration. In: 23rd AIAA Applied Aerodynamics Conference, Toronto, Canada (2005). https://doi.org/10.2514/6.2005-4727
42. Pope, S.B.: Turbulent Flows. Cambridge University Press, Cambridge (2000). https://doi.org/10.1017/CBO9780511840531
43. Moin, P.: Revisiting Taylor's hypothesis. J. Fluid Mech. **640**, 1–4 (2009). https://doi.org/10.1017/S0022112009992126