Tenth International Conference on Computational Fluid Dynamics (ICCFD10), Barcelona, Spain, July 9-13, 2018

ICCFD10-237

Jet Noise Prediction for Chevron Nozzles using a Direct-Hybrid CFD/CAA Method

A. Niemöller[*], M. Schlottke-Lakemper[**], M. Meinke[*,**] and W. Schröder[*,**]
Corresponding author: a.niemoeller@aia.rwth-aachen.de

[*] Institute of Aerodynamics, RWTH Aachen University, Germany.
[**] JARA - High-Performance Computing, RWTH Aachen University, Germany.

Abstract: High parallel efficiency for large-scale coupled multiphysics simulations requires the computational load to be evenly distributed among all compute cores. However, for complex applications and massively parallel computations, even minor load imbalances can have a severe impact on the overall performance and resource usage. The current direct-hybrid method is an example for a volume-coupled multiphysics simulation, in which a CFD and a CAA simulation are performed concurrently on the same parallel subdomains. Thus, for differing load compositions on each subdomain, accurate computational weights for CFD and CAA cells must be known to determine an efficient domain decomposition. In this study, a dynamic load balancing scheme is presented, which allows to increase the efficiency of complex coupled simulations with non-trivial domain decompositions. A fully-coupled three-dimensional jet simulation with approximately 300 million degrees of freedom demonstrates the effectiveness of the approach to reduce load imbalances. A detailed performance analysis substantiates the necessity of dynamic load balancing. Furthermore, the results of a strong scaling experiment show the benefit of load balancing to be proportional to the degree of parallelism. In addition, it is shown that the approach allows to attenuate imbalances also for parallel computations on heterogeneous computing hardware. Besides the software related results, also the acoustic field of a chevron nozzle will be discussed.

Keywords: direct-hybrid method, coupled multiphysics simulation, dynamic load balancing, parallel efficiency, computational aeroacoustics, jet noise, chevron nozzle.

1 Introduction

Reduction of aircraft noise is important for the airport environment and also for passenger comfort. Engine jet noise is a major issue, which can be reduced by nozzle shape modifications such as chevrons [1]. The optimization of such nozzle modifications requires accurate and efficient noise prediction methods. A well established hybrid computational fluid dynamics (CFD) computational aeroacoustics (CAA) CFD-CAA approach [2] consists of a large-eddy simulation (LES) to compute the unsteady turbulent flow field and a subsequent CAA step, in which the acoustic perturbation equations (APE) are solved. In such a hybrid approach, the acoustic source terms for the CAA step are obtained from the instantaneous flow field. A major drawback of using two disjoint solvers, however, is the necessity to exchange volume source terms via disk I/O. Besides huge disk storage requirements for large-scale problems, the overall parallel scalability is limited by the available I/O bandwidth. Therefore, a new fully-coupled direct-hybrid method has been developed, in which the LES and CAA solvers are coupled in the same simulation framework [3]. This allows the concurrent execution of both solvers with in-memory data exchange of the acoustic source terms. Furthermore, by employing a joint hierarchical Cartesian grid, which is partitioned on a coarse level via a space-filling curve (SFC), both solvers are efficiently coupled and parallelized, and the use of solution adaptive meshes with dynamic load balancing (DLB) is possible. Thus, the direct-hybrid method combines the advantages of the direct and hybrid solution strategies, i.e., minimizes I/O and exploits the different length scales in the acoustic and flow fields, and therefore it is a promising candidate to perform large-scale simulations of three-dimensional turbulent aeroacoustic problems.

SFC-based partitioning is widely used for parallelizing and load balancing scientific computing applications [4, 5]. SFCs provide a linearisation of objects located in a higher-dimensional space, while preserving spatial locality. Thus, determining a domain decomposition reduces to solving a one-dimensional partitioning problem, which is known as the chains-on-chains partitioning (CCP) problem. The CCP problem has been extensively studied in the literature, giving exact solutions [6], parallel heuristics [7], scalable hierarchical algorithms [8], and also an extension to heterogeneous systems [9]. All those methods assume that sufficiently accurate information about the workload distribution is available. This can, for example, be achieved by using code instrumentation to measure computing times on an object basis [10]. However, in case such an approach is impracticable, an alternative is required to estimate weights for different types of objects with dissimilar computational costs. This assumes that each object of a certain type exhibits the same constant workload. Nevertheless, for complex applications this will often yield a suboptimal load balanced domain decomposition, since local workload variations are not captured. Therefore, a DLB method for SFC-based partitioning is required that targets load imbalances more directly.

In this paper, a dynamic load balancing scheme for the coupled solvers in the direct-hybrid method is presented, which allows to increase the efficiency of complex simulations with non-trivial domain decompositions. It comprises a measurement based estimation of computational weights, using the compute time on each parallel process and the current distribution of cells of both solvers in all subdomains. Further, an incremental, diffusive DLB approach is employed to determine a new partitioning of the SFC into parallel subdomains. Additionally, the applicability of the direct-hybrid method for aeroacoustic problems is demonstrated.

This study has the following structure. First, the direct-hybrid method is introduced in Sec. 2. Then, in Sec. 3 the dynamic load balancing scheme is described. Simulation results are presented in Sec. 4. Finally, conclusions are drawn in Sec. 5.

2 Direct-hybrid CFD-CAA method

Standard hybrid methods for CAA are based on a two-step approach, since in general two simulation tools are used to compute the flow and subsequently the acoustic field [11, 12]. Thus, the exchange of volume coupling source terms is done via disk I/O, which becomes the bottleneck in large-scale parallel simulations. The direct-hybrid method circumvents this issue by coupling both solvers in the same simulation framework, enabling the concurrent computation of the flow and the acoustic field by means of an LES and a CAA [3]. Both solvers are based on hierarchical Cartesian meshes with cut-cell formulations. For the prediction of the turbulent flow field, a finite-volume method is employed to solve the Navier-Stokes equations [13]. The acoustic field is determined by a discontinuous Galerkin spectral element method, which is used to solve the APE. Both solvers operate on a joint hierarchical Cartesian grid that is partitioned on a coarse level via a space-filling curve. Full subtrees of the grid are then assigned to the parallel subdomain, which facilitates an efficient spatial coupling without any communication overhead.

In the following, the governing equations describing the flow and the acoustic field are introduced in Sec. 2.1. Then, the numerical methods for the CFD and the CAA simulation are given in Sec. 2.2, while Sec. 2.3 focuses on the mesh topology and the partitioning approach of the grid into parallel subdomains. Finally, the coupling approach employed in the direct-hybrid method is presented in Sec. 2.4.

2.1 Governing equations

The conservative form of the non-dimensional Navier-Stokes equations is given by

$$\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{u}) = 0, \tag{1a}$$

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} + \boldsymbol{\nabla} \cdot \left(\rho \boldsymbol{u}\boldsymbol{u} + p\boldsymbol{I} + \frac{\boldsymbol{\tau}}{\mathrm{Re}_0} \right) = 0, \tag{1b}$$

$$\frac{\partial \rho E}{\partial t} + \boldsymbol{\nabla} \cdot \left((\rho E + p)\boldsymbol{u} + \frac{1}{\mathrm{Re}_0}(\boldsymbol{\tau}\boldsymbol{u} + \boldsymbol{q}) \right) = 0, \tag{1c}$$

in which the conservative variables comprise the density ρ, the velocity vector $\boldsymbol{u}$, and the total specific energy $E = e + \frac{1}{2}\boldsymbol{u}^2$. The internal energy e is linked to the pressure p via the equation of state for an ideal gas to close the equation system

$$p = (\gamma - 1)\,\rho e, \tag{2}$$

with the isentropic exponent $\gamma = 1.4$. The non-dimensionalization is based on stagnation quantities (subscript 0) leading to the Reynolds number

$$\mathrm{Re}_0 = \frac{\rho_0 c_0 L}{\mu_0}, \tag{3}$$

with a reference length L, the speed of sound $c_0 = \sqrt{\gamma p_0/\rho_0}$, and the dynamic viscosity μ_0. The stress tensor for a Newtonian fluid is given by

$$\boldsymbol{\tau} = -\mu\left[\boldsymbol{\nabla}\boldsymbol{u} + (\boldsymbol{\nabla}\boldsymbol{u})^{\intercal}\right] + \frac{2}{3}\mu(\boldsymbol{\nabla}\cdot\boldsymbol{u})\boldsymbol{I}. \tag{4}$$

The dynamic viscosity μ is modelled by Sutherland's law. The heat conduction vector $\boldsymbol{q}$ is computed based on the static temperature T using Fourier's law

$$\boldsymbol{q} = -\frac{k}{\mathrm{Pr}(\gamma - 1)}\boldsymbol{\nabla}T \tag{5}$$

with constant Prandtl number $\mathrm{Pr} = \mu_0 c_p/k_0 = 0.72$, the specific heat at constant pressure c_p, and the thermal conductivity $k(T) = \mu(T)$.

Since acoustic fields of compressible flows are analyzed, the APE-4 formulation is used in this study [2]

$$\frac{\partial \boldsymbol{u}'}{\partial t} + \boldsymbol{\nabla}\left(\bar{\boldsymbol{u}} \cdot \boldsymbol{u}' + \frac{p'}{\bar{\rho}} \right) = \boldsymbol{q}_m, \tag{6a}$$

$$\frac{\partial p'}{\partial t} + \bar{c}^2 \boldsymbol{\nabla} \cdot \left(\bar{\rho}\boldsymbol{u}' + \bar{\boldsymbol{u}}\frac{p'}{\bar{c}^2} \right) = 0, \tag{6b}$$

to determine the acoustic pressure and velocity fluctuations p' and $\boldsymbol{u}'$. Time averaged quantities denoted by an overbar are computed by the compressible flow simulation. The right-hand side source term $\boldsymbol{q}_m$ consists of the perturbed Lamb vector $\boldsymbol{L}'$

$$\boldsymbol{q}_m = -\boldsymbol{L}' = -\left(\boldsymbol{\omega} \times \boldsymbol{u}\right)', \tag{7}$$

with the vorticity vector $\boldsymbol{\omega} = \nabla \times \boldsymbol{u}$. Note that $\boldsymbol{L}'$ is the dominant acoustic source for cold jet noise [14]. The source terms are determined by the LES of the turbulent flow.

2.2 Numerical methods

The turbulent flow field is computed by a large-eddy simulation (LES) on an unstructured hierarchical Cartesian grid using a finite-volume discretization [15]. A monotonic upstream-centered scheme for conservation laws (MUSCL) is employed to determine the state variables on the cell surfaces [16]. The convective terms are approximated by a modified low-dissipation advection upstream splitting method (AUSM) [17], while the viscous terms are computed by central differences [18]. Time integration is performed using a five-stage explicit Runge-Kutta method. The boundaries are resolved by a strictly conservative cut-cell approach [13, 19]. Overall, the scheme is second-order accurate in space and time. Further details and applications to various flow problems can be found in [20, 21, 22].

The acoustic perturbation equations are discretized by a nodal discontinuous Galerkin spectral element method (DGSEM) [23, 24] using hexahedral mesh elements on an unstructured hierarchical Cartesian grid. In each DG element, the solution state is approximated by Lagrange polynomials of arbitrary degree employing a tensor product ansatz for each spatial direction. Legendre-Gauss nodes constitute the nodal basis, which is used for interpolation and integration by Gauss quadrature. The numerical flux on element interfaces is approximated using the local Lax-Friedrichs flux. Non-conforming elements occurring in locally refined Cartesian grids are treated using a mortar element method [25]. An explicit five-stage low-storage Runge-Kutta method [26] is employed for the time integration of the semi-discrete DG operator. A detailed description of the methodology can be found in [3].

2.3 Mesh topology and partitioning approach

The CFD and CAA solvers operate on a hierarchical Cartesian grid, in which cells are organized in an octree structure with parent, child, and neighbor relationships [21]. The grid generation process starts at the 0^{th} level with an initial cube-shaped cell enclosing the whole computational domain, which is then recursively subdivided in three dimensions into eight identical child cells. First, a uniform grid with the specified base refinement level is created, which is called the partition level. Afterwards, each cell on this coarse level is recursively subdivided until grid resolution requirements are fulfilled (Fig. 1). Thus, each coarse partition

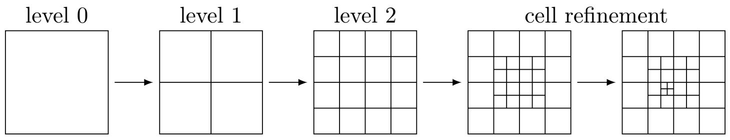
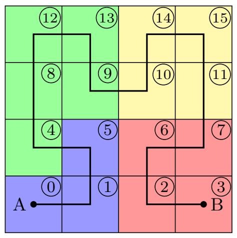


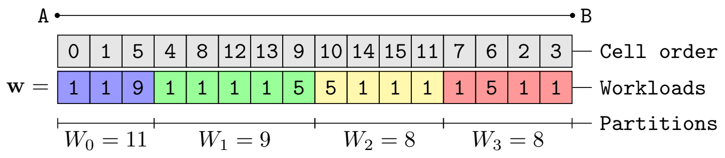
Figure 1: Creation of a uniform grid of level 2 and additional cell refinement in a 2D example.

cell becomes the root cell of a subtree of the whole grid. Two-dimensional grids are created analogously. Details on the grid generation process can be found in [27]. The partitioning of the computational grid takes place on the partition level, where a Hilbert space-filling curve is used to obtain a one-dimensional ordering of all partition cells (Fig. 2a). Each cell of the grid is assigned a computational workload depending on its use by either one or both solvers. By traversing the subtrees of the grid, the accumulated workload for each coarse partition cell is then determined. Thus, the parallelization is reduced to a so-called chains-on-chains partitioning (CCP) problem [6]. That is, a chain of weighted computational tasks is contiguously mapped onto a chain of parallel processes. Consequently, a domain decomposition can be obtained by splitting the one-dimensional workload distribution into partitions of similar total workload. The approach is illustrated in Fig. 2b using the linearization of cells along the Hilbert SFC connecting point **A** and **B** (see Fig. 2a). The list of workloads $\mathbf{w}$ comprises the partition-cell workloads w_k. Solving the CCP problem, for which $\overline{W} = 9$ is the average workload per process, the partitioning into subdomains is obtained. The domain offsets o_j are given by the splitting positions, which correspond to the first partition cell of each domain. With W_i the workload of the i^{th} domain, the efficiency of the partitioning is given by the maximum domain workload. Accordingly, the partition quality [28] is assessed by

$$P = \frac{\overline{W}}{\max_i W_i}, \tag{8}$$



(a) 2D Hilbert SFC at refinement level 2.



0	1	5	4	8	12	13	9	10	14	15	11	7	6	2	3
1	1	9	1	1	1	1	5	5	1	1	1	1	5	1	1

(b) SFC linearization, partition-cell workloads and total workloads of the resulting partitions (solution of CCP problem, average workload $\overline{W} = 9$).

Figure 2: Domain decomposition for four processes in 2D (color-coded by parallel subdomain).

which is $P = \frac{9}{11} \approx 82\%$ in the given example. Each partition, i.e., a continuous range of partition cells with the corresponding subtrees of the grid, is alloted to a dedicated parallel process. To prevent coarse-grained partitionings, the partition level can locally be shifted to a higher refinement level, in case the partition cell workload becomes too large. Intra-process spatial compactness of cells is implicitly ensured up to a certain degree by the locality property of the Hilbert SFC, which reduces the communication cost during the simulation.

2.4 Coupling approach

In the direct-hybrid method, a joint hierarchical Cartesian grid is used for the CFD and the CAA solver [3]. Refinement constraints for each physical system are taken into account during the grid generation process. All cells are tagged according to their use by either solver. Due to different domain sizes, regions can exist in which only a single solver is active. As described in the previous section, the domain decomposition is performed on the coarse partition level via the Hilbert SFC, incorporating the computational workloads of both solvers. Full subtrees of the grid are then alloted to the parallel subdomains yielding varying load compositions on each process depending on the local amount of cells used by the two solvers. For instance, in the domain decomposition shown in Fig. 3, with in total 28 and 22 active cells for the two domains, the local share of CFD cells is $\frac{17}{28} \approx 61\%$ and $\frac{8}{22} \approx 36\%$. This partitioning strategy facilitates an efficient spatial coupling, since all CFD and CAA cells contained in the same volume of the three-dimensional domain are assigned to the same process allowing in-memory exchange of the acoustic source term data. Spatial interpolation from the CFD to the CAA solution representation is performed by local Galerkin projection [29], which can handle arbitrary spatial mappings while requiring only data that is locally available. In this study, the same time step is used in the CFD and the CAA simulation. Thus, temporal interpolation of source term data, e.g., by quadratic interpolation [30], is not required. The parallel coupled algorithm alternately advances the Runge-Kutta time integration stages of both solvers. This interleaved execution pattern prevents any overhead due to differing load compositions and allows to overlap communication with computations. A detailed description of the coupling approach can be found in [3].

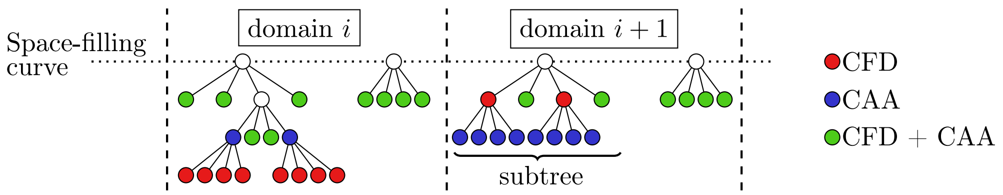


Figure 3: Domain decomposition of a quadtree grid with cells used for CFD, CAA, or both.

3 Dynamic load balancing

The ultimate goal of dynamic load balancing is to maximize the overall performance of a parallel computation by redistributing the workload among processes, such that all available computing power is used to full capacity [5]. Load balancing in highly parallel applications is key for an efficient resource usage [31], since even minor imbalances can have a severe impact on the performance and scalability of the computation. Numerical simulations typically involve frequent communication between MPI subdomains, such that an aggravation of imbalances will occur in every iteration or stage of a time step due to synchronization [32]. Furthermore, the performance penalty is sensitive to the specific imbalance pattern. A single overloaded process will significantly degrade performance. The reason for this is that all other processes wait and sit idle until the communication among neighbouring domains can proceed. On the other hand, a few underloaded processes will only have a minor impact on the overall performance. Finally, imbalances can be classified as dynamic or static [32]. Dynamic imbalances arise if the workload distribution varies as a function of time, which occurs, for example, when solution adaptive meshes are used. Constant workload distributions will lead to static imbalances.

Load imbalances and the resulting performance impact can be quantified by different metrics. Here, the imbalance percentage [33] defined by

$$I_{\%} = \frac{t_{max} - t_{avg}}{t_{max}} \cdot \frac{N}{N-1}, \tag{9}$$

which expresses the severity of an imbalance, is used. The quantities t_{max} and t_{avg} represent the maximum and average time needed to process a given section of code and N is the number of parallel processes. A value of $I_{\%} = 0\%$ corresponds to a perfectly balanced load distribution, while $I_{\%} = 100\%$ is associated with a code section executed only on a single process. Thus, the metric indicates the amount of wasted resources. This assumes that all but the slowest process sit idle at the end of the corresponding code section, which can be related to the completion of a single time step or stage of a time integration scheme in a numerical simulation. The potential run time savings are expressed by the imbalance time $I_t = t_{max} - t_{avg}$ assuming that perfect load balance is attainable [33]. Furthermore, the allocation-time impact is estimated by $I_T = N \cdot I_t$, which is an upper bound on the total amount of wasted resources [32].

Different strategies exist for DLB in scientific computing applications [4, 5, 34]. The current numerical method, the grid type, and its partitioning approach determine which DLB algorithms are applicable. In general, such an algorithm needs to identify load imbalances, determine the amount of workload to be transferred, select corresponding objects to migrate, and finally perform the relocation [35, 36].

Dynamic load balancing of large-scale coupled multiphysics simulations is inherently complex and poses various challenges. For instance, multi-stage calculations and interleaved computations with inherent communication barriers may prevent load balancing [37, 38]. The direct-hybrid method is an example for such a coupled multiphysics simulation and even for static load distributions, a parallel computation might exhibit load imbalances. Without load balancing, the parallel efficiency of the overall computation is determined by the initial domain decomposition, which is based on estimates regarding the different computational costs for CFD and CAA cells. These estimates can be determined by relating measured run times obtained from independent CFD and CAA simulations. However, this approach does not take into account the idle times of all parallel subdomains or for example boundary conditions that might be computationally more expensive. Consequently, the predicted load might deviate significantly from the actual load on each parallel subdomain during the coupled simulation [39].

This performance impact is exacerbated when considering large-scale simulations with relatively small domain workloads or when using heterogeneous computing hardware [40]. A DLB method is then required that automatically redistributes the computational workload to reduce imbalances during the simulation. In the direct-hybrid method, load balancing requires the redistribution of mesh cells used for the fully-coupled CFD/CAA solver, implicitly ensuring locality of the coupling acoustic sources. Furthermore, if the flow and the acoustics domain are of different size, a change of the grid partitioning can change the list of used parallel processes for each solver. In the following, a simulation framework that implements the direct-hybrid method is extended by a DLB algorithm that satisfies the stated constraints. Based on measurements of the computing time on each parallel process and the current distribution of cells of both solvers in all subdomains, new computational weights are computed (Sec. 3.1). These weights are then used to determine a new domain decomposition (Sec. 3.2).

3.1 Estimation of computational weights

Key to partitioning and load balancing of coupled multiphysics simulations is the estimation of computational costs for different objects present in the simulation [10]. By objects the constituent parts of the numerical setup, such as cells of the grid, are referred to. The standard approach for single-solver frameworks aims at distributing objects evenly among all parallel subdomains, assuming a homogeneous computing environment and that each object has the same constant computational cost. However, this poses difficulties if multiple types of objects with dissimilar computational costs exist. This problem can be reduced by using a-priori determined computational weights for each object type, which for the considered case includes CFD and CAA cells. However, for complex applications this will often yield a suboptimal load balanced domain decomposition, due to inaccurate computational weights, which do not take into account the computational effort of all involved algorithmic elements or inhomogeneous computing hardware. Code instrumentation to measure the actual computing time of each object constitutes a possible countermeasure [10, 39, 41]. However, this approach is often infeasible for scientific applications, because of extensive implementation efforts and additional overhead caused by the instrumentation. Thus, a minimally intrusive and reliable method is required for the estimation of computational weights, which in turn can be used for load balancing.

The DLB algorithm proposed in this study estimates computational weights based on measurements of the computing time on each parallel subdomain. Furthermore, the current distribution of objects of the distinct solvers among all subdomains is incorporated. These types of objects, which can be regarded as different load types, give rise to varying load compositions. During the simulation, the collective compute time of all solvers is measured locally for each time step. On each subdomain i, an average value r_i is then determined as the 25% truncated mean to filter out program-external influences such as system noise [32, 42]. With the global average compute time among all N parallel processes given by

$$\overline{r} = \frac{1}{N} \sum_{i=0}^{N-1} r_i, \tag{10}$$

the local computational load l_i is determined as

$$l_i = \frac{r_i}{\overline{r}}. \tag{11}$$

Subsequently, the computational weights $\boldsymbol{c}$ for the different load types can be estimated by solving the least squares problem $\boldsymbol{Ac} = \boldsymbol{l}$, with the right-hand side given by the load vector $\boldsymbol{l}$ and the left-hand side matrix $\boldsymbol{A}$ representing the current workload distribution among all subdomains. This assumes that on average the load can be expressed as a linear combination of the individual workload contributions. The linear least-squares problem with unique minimum-norm solution can be written as

$$||\boldsymbol{Ac} - \boldsymbol{l}||_2 = \min_{\boldsymbol{v}} ||\boldsymbol{Av} - \boldsymbol{l}||_2 \quad \text{and} \quad ||\boldsymbol{v}||_2 \text{ is minimal.} \tag{12}$$

In the present work, the DGELSD routine of LAPACK [43] is used to solve the least-squares problem given in Eq. 12. An example for this procedure with two load types distributed among $N = 4$ parallel subdomains is given in Eq. 13. Solving the overdetermined system of linear equations in the least-squares sense yields a computational weight ratio of 2.61 between the two load types. Thus, the average compute time for the second object type is estimated to be 2.6-times higher than that of the first one.

$$N \left\{ \begin{bmatrix} 10 & 7 \\ 13 & 4 \\ 12 & 2 \\ 5 & 8 \end{bmatrix} \right. \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} 1.2 \\ 0.9 \\ 0.8 \\ 1.1 \end{bmatrix} \Leftrightarrow \boldsymbol{Ac} = \boldsymbol{l} \quad \overset{\substack{\text{Least-} \\ \text{squares}}}{\underset{\text{solution}}{\Longrightarrow}} \quad \boldsymbol{c} = \begin{bmatrix} 0.0420 \\ 0.1097 \end{bmatrix}, \quad \frac{c_1}{c_0} = 2.61. \tag{13}$$

3.2 Partitioning approach

In general, computing a domain decomposition based on SFCs reduces to solving the resulting CCP problem. A straightforward extension for dynamic load balancing is given by estimating the computational weights for different load types during the simulation to determine new workloads as an input to the 1D partitioning problem. However, since local workload variations are not captured, the partitioning might be suboptimal in terms of performance, since any chosen CCP algorithm aims at an optimal partition quality (see Eq. 8). Therefore, to alleviate load imbalances more rigorously, a different procedure is proposed to determine new domain offsets for the SFC-based partitioning. First, the initial partitioning is improved by using the aforementioned method to obtain a reasonably good starting point for the succeeding steps. In the following repartitioning steps, the individual domain offsets are refined iteratively based on the measured load imbalance and the computed computational weights. As a starting point, the cumulative load imbalances given by

$$s_j = \sum_{i=0}^{j-1} l_i - 1.0 \quad \forall j \in \{1, \ldots, N\}, \quad \text{with} \quad s_0 = 0, \tag{14}$$

are determined. Each value quantifies for the corresponding domain offset o_j the overall load imbalance of domains left and right of the splitting position. Thus, an optimal local position requires shifting the offset o_j along the SFC to minimize the cumulative imbalance. The general assumption of this approach is that by optimizing each offset individually global load balance can be obtained. This procedure can be understood as an incremental, diffusive DLB method [38] based on the SFC partitioning, which might be better suited for large-scale simulations [44]. The approach is depicted in Fig. 4, with a list of partition cells split into four partitions. With the computational load l_i of each parallel subdomain, the cumulative imbalances s_j are computed according to Eq. 14. Thus, the load discrepancy between all processes on the left and right of each splitting position is quantified. This implies considering the value of $s_2 = 0.45$ in the example, that the compute load of the first two processes is significantly higher than that of the two remaining ones. Therefore, by shifting the offset to the left, workload is moved from overloaded to underloaded processes. Accordingly, global load imbalances can be reduced by individually shifting each domain offset in the direction given by

$$d_j = -\operatorname{sign}(s_j). \tag{15}$$

The load share $\widetilde{w}_k$ of the partition cell k on domain i is computed as

$$\widetilde{w}_k = l_i \cdot \overline{w}_k, \quad \text{with} \quad \overline{w}_k = \frac{w_k}{W_i}, \tag{16}$$

which can be interpreted as an allocation of the load l_i onto the local partition cells. By counterbalancing the cumulative imbalance s_j with the traversed partition cell load shares

$$s_j^k = s_j^{k-1} + d_j \cdot f_{penalty} \cdot \widetilde{w}_{m(k)} \quad \text{for} \quad k \geq 1, \quad \text{with} \quad m(k) = o_j + d_j k - \frac{1}{2}(d_j + 1) \quad \text{and} \quad s_j^0 = s_j, \tag{17}$$

the necessary displacement of each offset o_j can be assessed by the sequence index k for which $s_j^k \approx 0$ holds. The additional penalization factor $f_{penalty} \geq 1$ allows to limit the displacements and prevent overshooting. Thus, the DLB algorithm refines the partitioning during the simulation. This approach is illustrated in Fig. 4 and Eq. 18 for the second domain offset o_2, using a penalty factor of $f_{penalty} = 1.25$. According to Eq. 17, the initial cumulative imbalance of $s_2 = 0.45$ is minimized by shifting the offset by two partition cells, which yields the new domain offset.

$$s_2^0 := 0.45 \quad \Rightarrow \quad s_2^1 := 0.45 - 1.25 \cdot 1.2 \cdot 0.1 = 0.3 \quad \Rightarrow \quad s_2^2 := 0.3 - 1.25 \cdot 1.2 \cdot 0.15 = 0.075 \tag{18}$$

Furthermore, by introducing performance factors estimating the processing speed on differing compute nodes, the approach can be extended to target also computations on heterogeneous computing hardware. The relative processing capacity can be defined by relating the local load to the inverse of the relative domain workload

$$p_i := l_i \cdot \frac{\overline{W}}{W_i}, \tag{19}$$

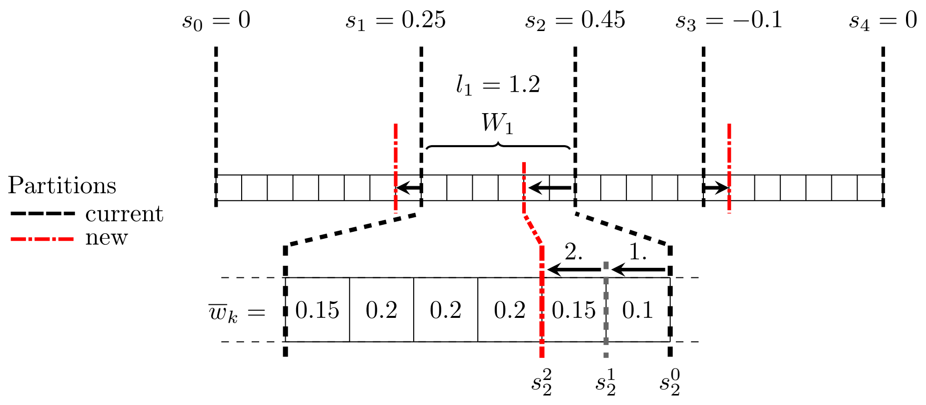


Figure 4: Basic concept of the DLB approach, with a list of partition cells split into four parallel subdomains. Combining the computational loads l_i, the cumulative imbalance s_j at each domain offset is determined. The new partitioning is then obtained by shifting each offset until according to Eq. 17, the imbalance is predicted to be counterbalanced. The cut-out illustrates this procedure for the second domain offset, given the corresponding partition cell workload shares and a measured overload of 20% on the second domain.

which is similar to the power weight for quantifying relative computing speeds [45]. By employing related processing capacities as an additional factor in the iteration procedure given in Eq. 17, more workload can be transferred from slower to faster processes, and vice versa. For example, given a high and a low processing capacity of $p_{i-1} = 0.8$ and $p_i = 1.2$, the workload shifted from the faster to the slower process is weighted by a factor of $p_i/p_{i-1} = 1.5$. In general, this DLB approach is suitable for applications with static or dynamic load imbalances. In the latter case, however, the cost of frequent redistributions needs to be balanced against the potential performance gains [10].

In this study, a sequential version of the proposed DLB algorithm is employed, which requires gathering the partition cell workloads on a single domain. To reduce communication overhead and memory requirements, an extension to a hierarchical parallel version is feasible and will be studied in the future. Nevertheless, with only a single value per partition cell, the communication volume is rather low, compared to, e.g., the high amount of load information reported in [46].

4 Results

The CFD/CAA simulation of a three-dimensional turbulent jet is considered to demonstrate the effectiveness of the presented dynamic load balancing scheme for large-scale coupled multiphysics simulations. Performance evaluations and a detailed analysis are given in Sec. 4.1. Furthermore, jet noise predictions obtained by the direct-hybrid method are presented in Sec. 4.2.

4.1 Dynamic load balancing

The dynamic load balancing scheme is employed in the direct-hybrid simulation of the three-dimensional turbulent jet with a Mach number $M = 0.9$ and a Reynolds number $Re_D = 400{,}000$ based on the jet diameter D. The flow field is computed on a domain with an extent of $36D$ in the downstream and $16D$ in the sideline direction. The CFD solver uses highly refined cells in the acoustic source term region to accurately predict the turbulent motion of the flow. Farther away, the resolution requirements are lowered such that the grid cells employed in the solver become gradually coarser. In total, about 57.3 million cells are used in the CFD computation. The CAA domain encloses the CFD domain with an extent of $41D$ and $30D$ in the downstream and sideline direction. The level difference in the acoustic source region between CAA and CFD cells is two, i.e., CAA cells have a four-times lower spatial resolution. In the near far field grid cells with a uniform size are employed to capture the acoustic propagation. Overall, the CAA simulation uses about 3.71 million cells. A polynomial degree of $p = 3$ is chosen for the DG solution representation, yielding approximately 237.4

million degrees of freedom. The initial mesh partitioning is based on an a-priori estimated computational weight ratio of $w_{CAA} = 8.5$ between CAA and CFD cells. This ratio was determined by relating measured run times for CFD and CAA simulations, which were obtained from standalone simulations of the considered case without coupling of the solvers on 3072 cores.

A strong scaling of the described direct-hybrid simulation is carried out from 192 to 6144 compute cores. The performance experiments were conducted using the Cray XC 40 “Hazel Hen” system of the High Performance Computing Center Stuttgart, Germany (HLRS). Each of the 7712 compute nodes consists of two Intel Haswell E5-2680v3 12-core processors with base and maximum processing frequencies of 2.5GHz and 3.3GHz, respectively. Simulations with DLB and the corresponding references using the initial mesh partitioning were performed alternately on the same compute nodes. All measurements were repeated at least three times from which the result with the minimum run time per time step was selected for comparison.

In the following, Sec. 4.1.1 gives a detailed performance analysis for the simulations on 3072 compute cores. Then, the results of the strong scaling experiment are presented in Sec. 4.1.2. Finally, it is shown in Sec. 4.1.3 how the proposed DLB approach can improve the efficient utilization in heterogeneous computing environments.

4.1.1 Parallel efficiency analysis

First, the run time composition for the jet simulation on 3072 compute cores without and with load balancing is shown in Fig. 5. The average wall time per simulation time step is split into computation and communication among all parallel subdomains. In addition to the CFD and CAA compute loads, the cost for the spatial interpolation of the acoustic source terms is presented. The CAA/CFD communication times correspond to the average time waiting for the completion of an inter-rank data exchange. For clarity, the compute cores are sorted by the CFD load in decreasing order. The reference simulation evidences that the initial domain partitioning without load balancing is suboptimal regarding the workload distribution. A few domains exhibit an increased compute load, thus blocking all other parallel processes until the synchronization point is reached and communication of data takes place. In total, the communication overhead accounts for 24% of the total accumulated run time. This value is reduced to 11% after load balancing. For both cases, the CFD simulation requires the major share of the compute load with 72% of the total time, while 27% are used for the CAA. The spatial interpolation of acoustic source terms requires only 1% of the total compute time.

With DLB, the total compute load is more evenly spread among the parallel subdomains, while simultaneously more ranks participate in the CFD computation. The imbalance percentage $I_{\%}$ is diminished from 23% to 10%, corresponding to a significant perfomance improvement with run time savings of 12.5%. As indicated by this imbalance metric, the potential savings are even higher assuming perfect load balance. However, regarding the complexity of the fully-coupled simulation with drastically varying load compositions, the already quite good initial partitioning is improved considerably.

The automatic and minimally intrusive DLB algorithm operates with minimum knowledge about the

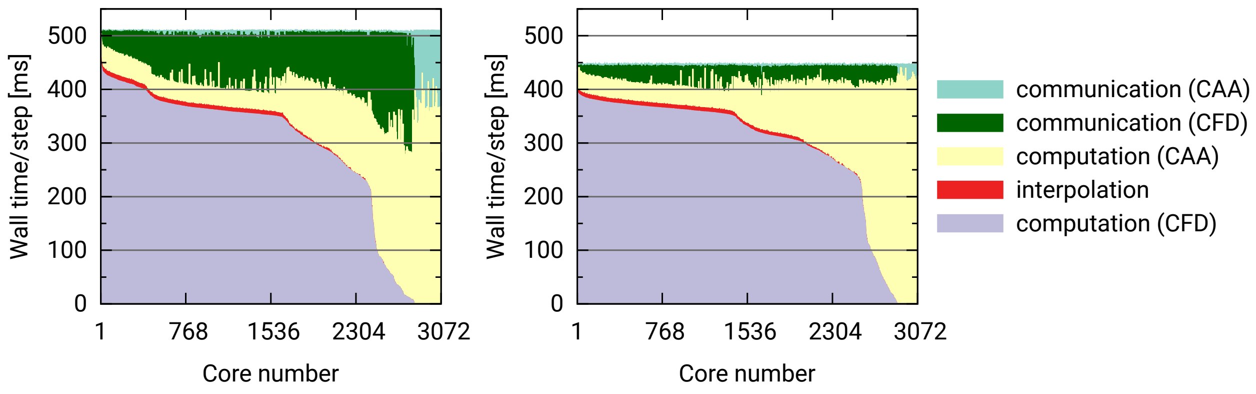
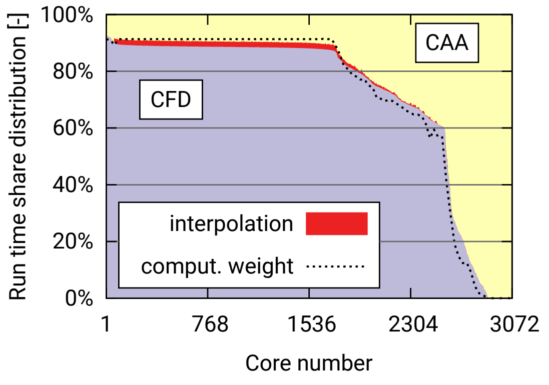
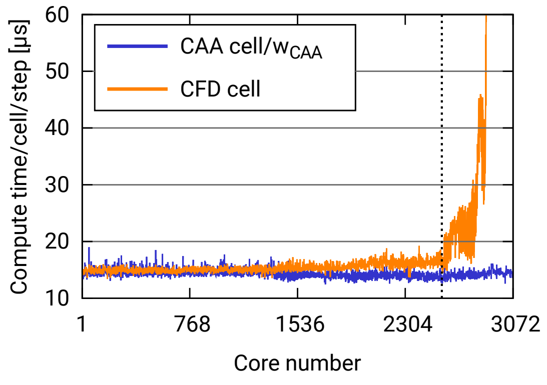


Figure 5: Run time distribution for the jet simulation on 3072 cores without (left) and with load balancing (right). Compute cores are sorted by the CFD compute load in decreasing order.



(a) Relative compute load distribution and estimated computational weight distribution. Compute cores are sorted by the relative CFD compute load in decreasing order.



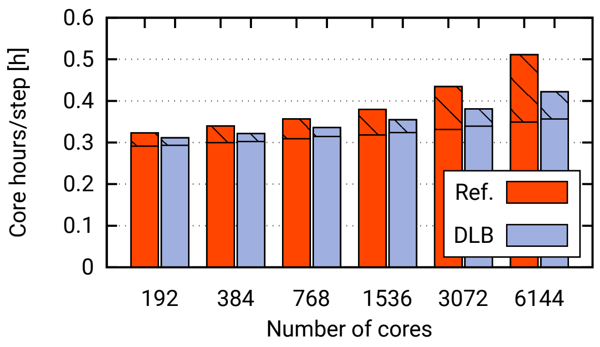
(b) Compute time per cell and time step on each subdomain. The time per CAA cell is scaled with the estimated computational weight. Compute cores are sorted by the number of CFD cells in decreasing order. The dashed line separates domains with less than 50% of the maximum number of CFD cells.

Figure 6: Load distribution and performance evaluation for the jet simulation on 3072 cores with DLB.

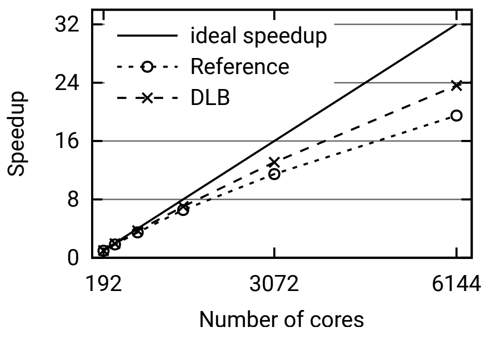
simulation, while the complexity of the numerical schemes is concealed in the solvers. This is further illustrated by the run time share distribution pictured in Fig. 6a with the compute cores sorted by the relative CFD compute load in decreasing order. The computation is dominated by the CFD load on more than 80% of all domains, while the CFD run time share is near 90% on more than half of all ranks. Additionally, the computational weight distribution is displayed, which is based on the weight ratio between CAA and CFD cells of $w_{CAA} = 6.09$ estimated during the simulation run. Overall, the curve follows the trend of the run time share distribution. However, the CFD compute load is overestimated for compute cores with a significant dominance of the CFD computation, while it is underestimated for domains with a higher CAA share. This indicates that even if accurate computational weights are employed the resulting domain decomposition will likely not be optimal due to differing load compositions exhibiting dissimilar compute times. Moreover, this issue is highlighted in Fig. 6b by the local average compute time per cell and time step. The compute cores are sorted by the number of CFD cells in decreasing order, while the computational time for CAA cells is scaled by the estimated weight ratio. Considering the first half of the sorted compute cores, the simulation time per CFD cell and step is nearly constant and matches with the scaled value for a CAA cell. On the contrary, for domains with less CFD cells the computational effort per cell increases slightly before becoming significantly higher for domains with less than half the local maximum number of CFD cells. An explanation for this performance variation of the CFD solver is that for larger number of cells the efficiency increases due to a higher benefit from code vectorization and improved memory accesses. Besides some smaller deviations in the compute time, this behavior is not observed for the acoustics solver.

4.1.2 Strong scaling experiment

The total computational resources required per simulation time step for a strong scaling experiment from 192 to 6144 compute cores without and with DLB are depicted in Fig. 7a. Additionally, the average idle times are shown for each configuration. For lower numbers of compute cores, the parallel computations exhibit only small imbalances. Due to the low degree of parallelism, each domain is assigned a huge compute load that keeps imbalances caused by, e.g., inaccurate computational weights, relatively small. Nevertheless, for DLB small improvements regarding the overall parallel performance are observed. During the scaling, the load imbalances in the reference simulations start with an imbalance percentage of $I_{\%} = 9.6\%$ for 192 cores and increase to $I_{\%} = 31.2\%$ for 6144 cores. Simultaneously, the total amount of computational resources required to perform the computations in the simulation rises by nearly 20%, due to decreasing local problem sizes. With load balancing, the imbalances are reduced to about $I_{\%} = 2.6\%$ and $I_{\%} = 14.6\%$ for the simulations



(a) Total computational resources required per simulation time step. Hatched parts correspond to the average idle time.



(b) Scalability of the simulation using the DLB measurement for 192 cores as a reference.

Figure 7: Strong scaling for a turbulent jet simulation without (Ref.) and with dynamic load balancing (DLB) from 192 to 6144 compute cores.

on 192 and 6144 compute cores. Thus, even though there is still potential to achieve further improvements with DLB, the benefit grows with the degree of parallelism, saving 17.5% of computational resources when using 6144 cores.

The speedup of the strong scaling experiment is depicted in Fig. 7b, with the DLB measurement for 192 cores as a reference for the scalability analysis. The parallel efficiency of the reference simulation declines to about 61% during the scaling, whereas with load balancing an efficiency of 74% is obtained with the 32-fold increase in the number of compute cores.

The trend of the estimated computational weight ratios between CAA and CFD cells in Fig. 8 gives further insight regarding the inferior performance of the reference simulation for large number of cores. For low degrees of parallelism, the computational weight ratio is close to that estimated a-priori, which is used for the initial partitioning. However, for larger numbers of compute cores this ratio is reduced considerably. Analogously to the observations based on the compute time per cell and time step in Fig. 6b, this behavior can be explained by the efficiency decline of the CFD solver when the problem size on each parallel subdomain is reduced. Due to the increasing discrepancy with respect to the prescribed weight ratio, the quality of the domain decomposition degrades gradually and causes significant load imbalances.

In general, multiphysics simulations defined by multiply coupled solvers will most probably exhibit similar effects due to differing scalabilities of the numerical codes. Thus, considering the ever increasing degree of parallelism, accompanied by decreasing local problem sizes [31], the use of static a-priori determined computational weights will significantly degrade the parallel performance of future large-scale applications.

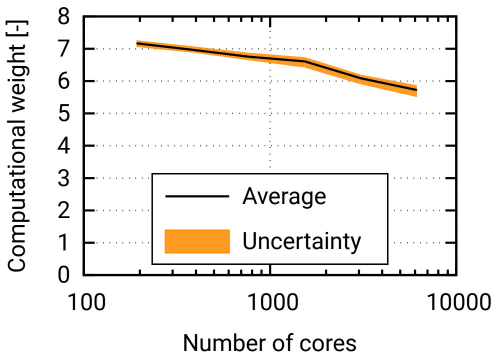


Figure 8: Computational weight ratio w_{CAA} between CAA and CFD cells estimated during the fully-coupled simulation for varying number of compute cores. An average value of various estimates and the region of uncertainty containing all values are shown.

4.1.3 Heterogeneous systems

A reduced version of the previously considered direct-hybrid jet simulation shows the ability of the presented DLB approach to diminish load imbalances for computations on heterogeneous computing architectures. The setup consists of about 2.4 million CFD cells, and 10.9 million degrees of freedom in the CAA simulation. The a-priori determined computational weight ratio is again used for the initial domain partitioning. In total, 40 parallel processes are used in a heterogeneous computing environment combining two Intel Xeon E5-2695v3 and two Intel Xeon X5650 processors both of which have 14 and 6 cores and processing base frequencies of 2.3GHz and 2.66GHz. With the X5650 belonging to a considerably older processor generation, a significantly lower performance per compute core is available, causing load imbalances in the simulation. The single core performance measured for either of the two nodes shows an increased compute time per step of 20% and 34% for CFD and CAA simulation for the X5650 processor.

The computational loads defined in Eq. 11 for the simulations without and with DLB are shown in Fig. 9. In the reference simulation, the first 28 ranks located on the faster processors are mostly assigned too small a compute load and the remaining domains exhibit severe overload degrading the parallel performance significantly. Thus, with an imbalance percentage of about $I_{\%} = 45\%$ there is no benefit to use the additional processors in the computation. With load balancing the compute load is evenly distributed among all ranks in the heterogeneous computation. The amount of wasted resources as indicated by the imbalance metric is less than 5%. This confirms the ability of the proposed DLB method to eliminate load imbalances when using inhomogeneous computing hardware. Nevertheless, large-scale heterogeneous simulations for significantly more processors are required to substantiate the analysis. An additional field of application is the efficient concurrent utilization of CPUs and GPUs [47], which requires suitable DLB algorithms. This necessity grows since heterogeneity of high performance computing systems will increase in the future [31].

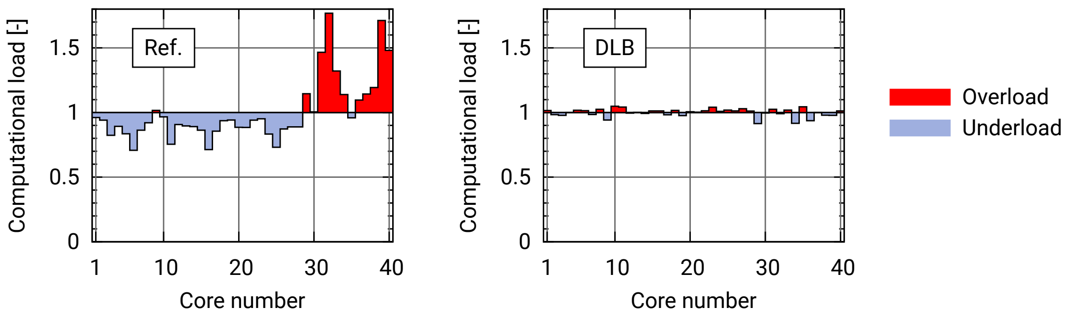


Figure 9: Computational load distribution for the reduced jet simulation without (left) and with load balancing (right) using 40 processes in a heterogeneous computing environment in which two different processor types are combined.

4.2 Jet noise prediction

In the following, jet noise predictions obtained with the direct-hybrid method are presented. First, results for a nozzle-free jet are given in Sec. 4.2.1, which serves as a validation case. Then, a jet emanating from a chevron nozzle is considered in Sec. 4.2.2.

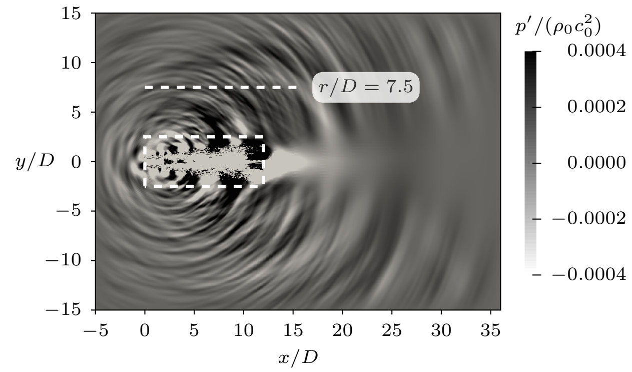
4.2.1 Nozzle-free jet simulation

The acoustic field of an isothermal circular jet is simulated by the direct-hybrid method. The numerical setup corresponds to that in Sec. 4.1. The flow conditions are chosen according to Bogey and Bailly [48], with a hyperbolic-tangent velocity profile with diameter D prescribed at the jet inlet boundary. A vortex ring forcing in the shear layer accelerates the transition to turbulence. The acoustic source region in which the CFD and CAA methods are coupled, has an extent of $12D$ in the downstream and $5D$ in the sideline direction. A detailed description of the numerical setup can be found in [29].

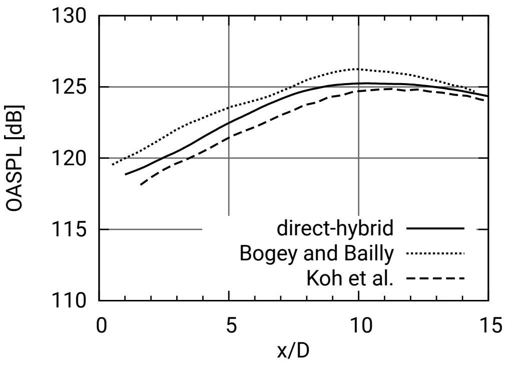
A snapshot of the predicted acoustic pressure field is shown in Fig. 10a. The pressure signal is recorded on circles with radius $r/D = 7.5$ along the jet centerline. The overall sound pressure level (OASPL) measured in decibel (dB) is

$$\mathrm{OASPL} = 20 \log_{10} \left(\frac{p'_{\mathrm{rms}}}{p_{\mathrm{ref}}} \right) \tag{20}$$

with the root-mean-square pressure value p'_{rms}, and the reference pressure $p_{\mathrm{ref}} = 2{\cdot}10^{-5}\mathrm{Pa}$. The OASPL along the axial direction with the p'_{rms} value averaged over each circle is shown in Fig. 10b. The comparison with reference data from the literature shows that the direct-hybrid method convincingly captures the directivity pattern of the noise generated by the turbulent jet.



(a) Acoustic pressure field in the x-y-plane, with acoustic source region (dashed rectangle) and microphone positions (dashed line).



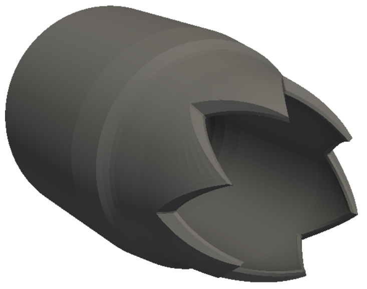
(b) Overall sound pressure level with reference data from Bogey and Bailly [48] and Koh et al. [30].

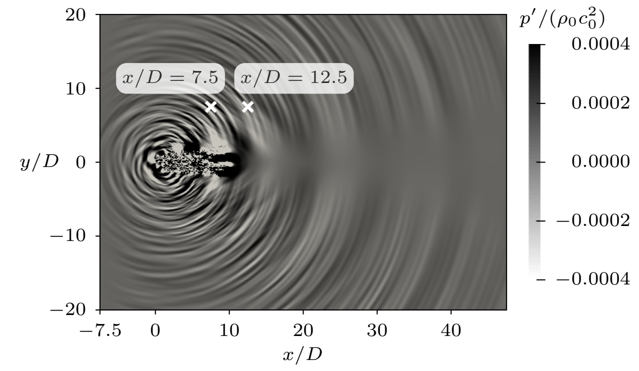
Figure 10: Noise prediction for a nozzle free turbulent jet.

4.2.2 Chevron jet noise

In the following, the cold jet flow exhausting from the NASA Glenn nozzle SMC006 [1] is considered. This chevron nozzle shown in Fig. 11a consists of six symmetric lobes, which have the same penetration angle 18.2°. The flow conditions are chosen according to the test point 7 of Tanna [49], with the jet temperature related to the ambient conditions $T_j/T_\infty = 0.84$ and an acoustic Mach number based on the jet exit velocity of $Ma_{ac} = U_j/c_\infty = 0.9$. The nozzle exit diameter based Reynolds number is $\mathrm{Re}_D = 400{,}000$. The numerical setup for the CFD simulation is similar to that employed in [50], such that the full chevron nozzle with about 170 million grid cells is considered. The highest grid resolution covers the acoustic source region. It has an extent of $10D$ in the downstream and $5.5D$ in the sideline directions. In the CAA simulation, cells in the source region have a four-times lower spatial resolution, while the DG solution representation uses a polynomial degree of $p = 3$, yielding about 350 million degrees of freedom. A snapshot of the predicted acoustic field is shown in Fig. 11b. The power spectral density (PSD) of the acoustic pressure fluctuations is evaluated at the radial distance $r/D = 7.5$ for the two axial positions $x/D = 7.5$ and $x/D = 12.5$.

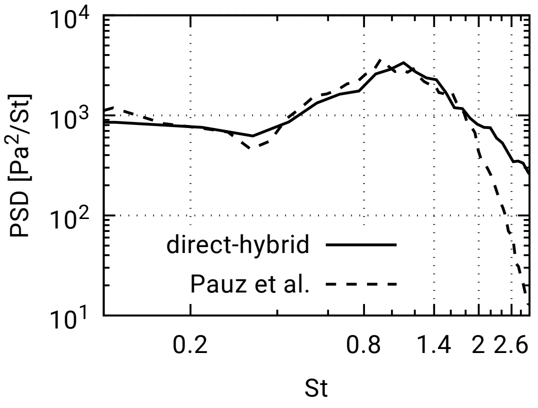
In Fig. 12 the computed power spectra plotted over the Strouhal number $\mathrm{St} = fD/U_j$ are compared with results from a standard hybrid scheme [50]. Overall, the results of the direct-hybrid method are in good agreement with the reference data, especially for Strouhal numbers in the range $0.4 < \mathrm{St} < 2.0$. For lower frequencies, the sound pressure level is underpredicted, which might be due to too short a sampling period. In the high frequency range for the axial position $x/D = 7.5$, the decay of the power spectral density is less pronounced, suggesting that the numerical setup provides a higher spatial and temporal resolution. In summary, the results show the direct-hybrid method to be capable to predict noise from chevron nozzles.


(a) Geometry of the NASA Glenn nozzle SMC006 [1].

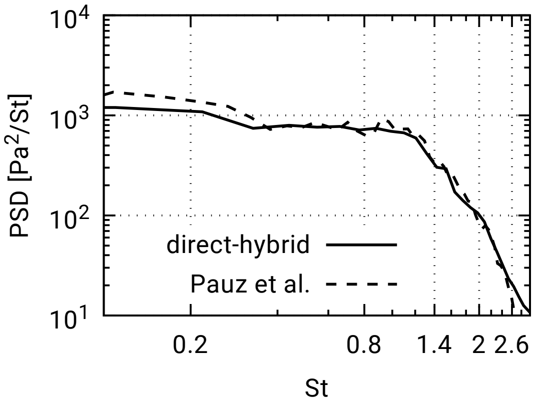


(b) Acoustic pressure field in the x-y-plane and observer locations at a radial distance of $r = 7.5D$.

Figure 11: Noise prediction for a jet emanating from a chevron nozzle.



(a) Sound pressure spectrum at $x/D = 7.5$.



(b) Sound pressure spectrum at $x/D = 12.5$.

Figure 12: Power spectral density (PSD) of the acoustic pressure at the radial distance $r/D = 7.5$ for $x/D = 7.5$ (left) and $x/D = 12.5$ (right) with reference data computed by a standard hybrid scheme from Pauz et al. [50].

5 Conclusions

To ensure high parallel efficiency for large-scale coupled multiphysics simulations is inherently complex since even minor load imbalances can severely impact the overall performance of massively parallel computations. A dynamic load balancing (DLB) scheme for a direct-hybrid multiphysics method was presented, which allows to increase the efficiency of complex simulations with non-trivial domain decompositions. The direct-hybrid scheme couples a CFD and a CAA simulation with distinct numerical methods and differing computational costs per cell, while the partitioning is obtained via a Hilbert space-filling curve (SFC) on a coarse level. A minimum intrusive method for estimating computational weights based on performance measurements during the simulation was proposed. The approach allows to automatically determine suitable weighting parameters that can be used to assess the overall workload distribution. Due to the inability of this estimation procedure to capture local workload variations, the common approach to SFC based load balancing might be suboptimal. Therefore, an incremental, diffusive DLB algorithm based on the SFC partitioning was presented, which allows individual adjustments of the domain decomposition.

A three-dimensional turbulent jet CFD/CAA simulation demonstrates effectiveness of the DLB scheme for large-scale coupled multiphysics simulations. A detailed performance analysis showed the necessity of a DLB method to directly target load imbalances, which are, e.g., caused by the individual efficiency of the computation depending on the local workload composition and the scalability of the individual solvers. Furthermore, a strong scaling experiment showed performance improvements at growing degree of parallelism, when a-priori estimated computational weights for the initial partitioning are used. Additionally, the DLB approach was successfully employed for balancing workload in a heterogeneous computing environment, which is an important requirement for sustained efficient use on existing and emerging high-performance computing systems. However, further research targeting load balancing large-scale simulations on heterogeneous systems is required. Finally, the quality of the physical findings determined by the direct-hybrid method was shown for jet noise.

6 Acknowledgments

The research was funded by the Deutsche Forschungsgemeinschaft (DFG) in the framework of the research group "Noise Reduction through Chevron Nozzles via Multi-Point Optimization". The authors gratefully acknowledge the Gauss Centre for Supercomputing (GCS) for providing computing time for a GCS Large-Scale Project on the GCS share of the supercomputer "Hazel Hen" at HLRS Stuttgart and JUQUEEN [51] at Jülich Supercomputing Centre (JSC). GCS is the alliance of the three national supercomputing centres HLRS (Universität Stuttgart), JSC (Forschungszentrum Jülich), and LRZ (Bayerische Akademie der Wissenschaften), funded by the German Federal Ministry of Education and Research (BMBF) and the German State Ministries for Research of Baden-Württemberg (MWK), Bayern (StMWFK), and Nordrhein-Westfalen (MIWF).

References

[1] J. Bridges and C. Brown, "Parametric testing of chevrons on single flow hot jets," *AIAA Paper 2004–2824*, 2004. DOI: 10.2514/6.2004-2824

[2] R. Ewert and W. Schröder, "Acoustic perturbation equations based on flow decomposition via source filtering," *J. Comput. Phys.*, vol. 188, no. 2, pp. 365–398, Jul. 2003. DOI: 10.1016/S0021-9991(03)00168-2

[3] M. Schlottke-Lakemper, H. Yu, S. Berger, M. Meinke, and W. Schröder, "A fully coupled hybrid computational aeroacoustics method on hierarchical Cartesian meshes," *Comput. Fluids*, vol. 144, pp. 137–153, 2017. DOI: 10.1016/j.compfluid.2016.12.001

[4] J. D. Teresco, K. D. Devine, and J. E. Flaherty, "Partitioning and Dynamic Load Balancing for the Numerical Solution of Partial Differential Equations," ser. Lecture Notes in Computational Science and Engineering, 2006, vol. 51, pp. 55–88. DOI: 10.1007/3-540-31619-1_2

[5] B. Hendrickson and K. Devine, "Dynamic load balancing in computational mechanics," *Comp. Methods in Appl. Mech. Eng.*, vol. 184, no. 2–4, pp. 485–500, Apr. 2000. DOI: 10.1016/S0045-7825(99)00241-8

[6] A. Pinar and C. Aykanat, "Fast optimal load balancing algorithms for 1D partitioning," *J. Parallel Distrib. Comput.*, vol. 64, no. 8, pp. 974–996, Aug. 2004. DOI: 10.1016/j.jpdc.2004.05.003

[7] J. R. Pilkington and S. B. Baden, "Dynamic partitioning of non-uniform structured workloads with spacefilling curves," *IEEE Trans. Parallel Distrib. Syst.*, vol. 7, no. 3, pp. 288–300, Mar. 1996. DOI: 10.1109/71.491582

[8] M. Lieber and W. E. Nagel, "Scalable high-quality 1D partitioning," in *HPCS*, Jul. 2014, pp. 112–119. DOI: 10.1109/HPCSim.2014.6903676

[9] A. Pinar, E. Kartal Tabak, and C. Aykanat, "One-dimensional partitioning for heterogeneous systems: Theory and practice," *J. Parallel Distrib. Comput.*, vol. 68, pp. 1473–1486, Nov. 2008. DOI: 10.1016/j.jpdc.2008.07.005

[10] H. Menon, N. Jain, G. Zheng, and L. Kale, "Automated load balancing invocation based on application characteristics," in *IEEE Cluster Comput.*, 2012, pp. 373–381. DOI: 10.1109/CLUSTER.2012.61

[11] C. Bailly and D. Juve, "Numerical Solution of Acoustic Propagation Problems Using Linearized Euler Equations," *AIAA Journal*, vol. 38, no. 1, pp. 22–29, Jan. 2000. DOI: 10.2514/2.949

[12] R. Ewert and W. Schröder, "On the simulation of trailing edge noise with a hybrid LES/APE method," *J. Sound Vibration*, vol. 270, no. 3, pp. 509–524, 2004. DOI: 10.1016/j.jsv.2003.09.047

[13] L. Schneiders, C. Günther, M. Meinke, and W. Schröder, "An efficient conservative cut-cell method for

rigid bodies interacting with viscous compressible flows," *J. Comput. Phys.*, vol. 311, pp. 62–86, 2016. DOI: 10.1016/j.jcp.2016.01.026

[14] E. Gröschel, W. Schröder, P. Renze, M. Meinke, and P. Comte, "Noise prediction for a turbulent jet using different hybrid methods," *Comput. Fluids*, vol. 37, no. 4, pp. 414–426, 2008. DOI: 10.1016/j.compfluid.2007.02.010

[15] D. Hartmann, M. Meinke, and W. Schröder, "A strictly conservative Cartesian cut-cell method for compressible viscous flows on adaptive grids," *Comp. Meth. Appl. Mech. Eng.*, vol. 200, no. 9-12, pp. 1038–1052, 2011. DOI: 10.1016/j.cma.2010.05.015

[16] B. van Leer, "Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method," *J. Comput. Phys.*, vol. 32, no. 1, pp. 101–136, 1979. DOI: 10.1016/0021-9991(79)90145-1

[17] M.-S. Liou and C. J. Steffen, "A New Flux Splitting Scheme," *J. Comput. Phys.*, vol. 107, no. 1, pp. 23–39, 1993. DOI: 10.1006/jcph.1993.1122

[18] M. Meinke, W. Schröder, E. Krause, and T. Rister, "A comparison of second and sixth-order methods for large-eddy simulations," *Comput. Fluids*, vol. 31, pp. 695–718, 2002. DOI: 10.1016/S0045-7930(01)00073-1

[19] L. Schneiders, D. Hartmann, M. Meinke, and W. Schröder, "An accurate moving boundary formulation in cut-cell methods," *J. Comput. Phys.*, vol. 235, pp. 786–809, 2013. DOI: 10.1016/j.jcp.2012.09.038

[20] S. Schlimpert, S. R. Koh, K. Pausch, M. Meinke, and W. Schröder, "Analysis of combustion noise of a turbulent premixed slot jet flame," *Combust. Flame*, vol. 175, pp. 292–306, 2017. DOI: 10.1016/j.combustflame.2016.08.001

[21] D. Hartmann, M. Meinke, and W. Schröder, "An adaptive multilevel multigrid formulation for Cartesian hierarchical grid methods," *Comput. Fluids*, vol. 37, no. 9, pp. 1103–1125, 2008. DOI: 10.1016/j.compfluid.2007.06.007

[22] C. Günther, M. Meinke, and W. Schröder, "A flexible level-set approach for tracking multiple interacting interfaces in embedded boundary methods," *Comput. Fluids*, vol. 102, pp. 182–202, 2014. DOI: 10.1016/j.compfluid.2014.06.023

[23] D. Kopriva, S. Woodruff, and M. Hussaini, "Discontinuous spectral element approximation of Maxwell's Equations," in *Int. Symp. Discontinuous Galerkin Methods*, 2000. DOI: 10.1007/978-3-642-59721-3_33

[24] J. S. Hesthaven and T. Warburton, *Nodal Discontinuous Galerkin Methods.* Springer, 2008. DOI: 10.1007/978-0-387-72067-8

[25] D. A. Kopriva, S. L. Woodruff, and M. Hussaini, "Computation of electromagnetic scattering with a non-conforming discontinuous spectral element method," *Int. J. Numer. Meth. Eng.*, vol. 53, pp. 105–222, 2002. DOI: 10.1002/nme.394

[26] M. H. Carpenter and C. Kennedy, "Fourth-order 2N-storage Runge-Kutta schemes," NASA Langley Research Center, NASA Report TM 109112, 1994.

[27] A. Lintermann, S. Schlimpert, J. H. Grimmen, C. Günther, M. Meinke, and W. Schröder, "Massively parallel grid generation on HPC systems," *Comput. Methods in Appl. Mech. Eng.*, vol. 277, pp. 131–153, 2014. DOI: 10.1016/j.cma.2014.04.009

[28] S. Miguet and J.-M. Pierson, "Heuristics for 1D Rectilinear Partitioning as a Low Cost and High Quality Answer to Dynamic Load Balancing," in *High-Performance Computing and Networking*, ser. LNCS, vol. 1225. Springer, 1997, pp. 550–564. DOI: 10.1007/BFb0031628

[29] M. Schlottke-Lakemper, A. Niemöller, M. Meinke, and W. Schröder, "Efficient parallelization for volume-coupled multiphysics simulations," *Comput. Methods. Appl. Mech. Eng.*, submitted 2018.

[30] S. R. Koh, W. Schröder, and M. Meinke, "Turbulence and heat excited noise sources in single and coaxial jets," *J. Sound Vibration*, vol. 329, pp. 786–803, 2010. DOI: 10.1016/j.jsv.2009.10.012

[31] S. Ashby *et al.*, "The Opportunities and Challenges of Exascale Computing." *ASCAC subcommittee, US - DOE Report*, 2010.

[32] D. Böhme, *Characterizing Load and Communication Imbalance in Parallel Applications*, ser. IAS. Forschungszentrum Jülich, 2014, vol. 23. DOI: 10.1109/IPDPSW.2012.321

[33] L. DeRose, B. Homer, and D. Johnson, "Detecting Application Load Imbalance on High End Massively Parallel Systems," *Parallel Processing*, pp. 150–159, Aug. 2007. DOI: 10.1007/978-3-540-74466-5_17

[34] K. D. Devine, E. G. Boman, and G. Karypis, "Partitioning and load balancing for emerging parallel applications and architectures," in *Parallel Processing for Scientific Computing.* SIAM, 2006, pp.

99–126. DOI: 10.1137/1.9780898718133.ch6

[35] G. Cybenko, “Dynamic load balancing for distributed memory multiprocessors,” *J. Parallel Distrib. Comput.*, vol. 7, no. 2, pp. 279–301, Oct. 1989. DOI: 10.1016/0743-7315(89)90021-X

[36] A. Arulananthan *et al.*, “A generic strategy for dynamic load balancing of distributed memory parallel computational mechanics using unstructured meshes,” in *Parallel CFD*, 1998, pp. 43–50.

[37] B. Hendrickson, “Load balancing fictions, falsehoods and fallacies,” *Appl. Math. Model.*, vol. 25, no. 2, pp. 99–108, 2000. DOI: 10.1016/S0307-904X(00)00042-1

[38] J. Watts and S. Taylor, “A practical approach to dynamic load balancing,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 9, no. 3, pp. 235–248, 1998. DOI: 10.1109/71.674316

[39] P. Jetley, F. Gioachin, C. Mendes, L. Kalé, and T. Quinn, “Massively parallel cosmological simulations with ChaNGa,” in *IEEE IPDPS*, Apr. 2008, pp. 1–12. DOI: 10.1109/IPDPS.2008.4536319

[40] J. Dongarra *et al.*, “The international exascale software project roadmap,” *Int. J. High Perform. Comput. Appl.*, vol. 25, no. 1, pp. 3–60, Feb. 2011. DOI: 10.1177/1094342010391989

[41] J. C. Phillips, G. Zheng, S. Kumar, and L. V. Kalé, “NAMD: Biomolecular simulation on thousands of processors,” in *ACM/IEEE Supercomputing*, 2002, pp. 1–18. DOI: 10.1109/SC.2002.10019

[42] F. Petrini, D. J. Kerbyson, and S. Pakin, “The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q,” *ACM/IEEE Supercomputing*, Nov. 2003. DOI: 10.1145/1048935.1050204

[43] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, 3rd ed. SIAM, 1999. ISBN 0-89871-447-8

[44] M. Lieber, K. Gößner, and W. E. Nagel, “The potential of diffusive load balancing at large scale,” in *EuroMPI*, 2016, pp. 154–157. DOI: 10.1145/2966884.2966887

[45] X. Zhang and Y. Yan, “Modeling and characterizing parallel computing performance on heterogeneous networks of workstations,” *Parallel Distrib. Comput.*, pp. 25–34, Oct. 1995. DOI: 10.1109/SPDP.1995.530661

[46] G. Zheng, A. Bhatelé, E. Meneses, and L. V. Kalé, “Periodic hierarchical load balancing for large supercomputers,” *Int. J. High Perform. Comput. Appl.*, vol. 25, no. 4, pp. 371–385, Nov. 2011. DOI: 10.1177/1094342010394383

[47] C. Xu *et al.*, “Collaborating CPU and GPU for large-scale high-order CFD simulations with complex grids on the TianHe-1A supercomputer,” *J. Comput. Phys.*, vol. 278, no. C, pp. 275–297, Dec. 2014. DOI: 10.1016/j.jcp.2014.08.024

[48] C. Bogey and C. Bailly, “Computation of a high Reynolds number jet and its radiated noise using large eddy simulation based on explicit filtering,” *Comput. Fluids*, vol. 35, no. 10, pp. 1344–1358, 2006. DOI: 10.1016/j.compfluid.2005.04.008

[49] H. K. Tanna, “An experimental study of jet noise Part I: Turbulent mixing noise,” *J. Sound Vibration*, vol. 50, no. 3, pp. 405–428, Feb. 1977. DOI: 10.1016/0022-460X(77)90493-X

[50] V. Pauz, A. Niemöller, M. Meinke, and W. Schröder, “Numerical analysis of chevron nozzle noise,” *AIAA Paper 2017–3853*, 2017. DOI: 10.2514/6.2017-3853

[51] Jülich Supercomputing Centre, “JUQUEEN: IBM Blue Gene/Q Supercomputer System at the Jülich Supercomputing Centre,” *Journal of large-scale research facilities*, vol. 1, no. A1, 2015. DOI: 10.17815/jlsrf-1-18