HORIZON-EUROHPC-JU-2021-COE-01

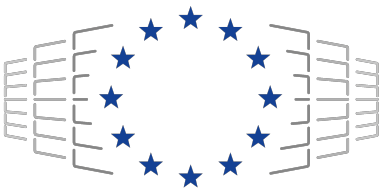
Centre of Excellence in Exascale CFD

CEEC – Centre of Excellence in Exascale CFD

Grant Agreement Number: 101093393

D1.1 – CEEC exascale lighthouse cases and their needs

WP1: Exascale light-house cases



EuroHPC
Joint Undertaking

Document Information

Deliverable Number	D1.1
Deliverable Name	CEEC exascale lighthouse cases and their needs
Due Date	31/08/2023 (PM 08)
Deliverable lead	USTUTT
Authors	Daniel Kempf (USTUTT), Anna Schwarz (USTUTT), Gerard Guillamet (BSC), Guillaume Houzeaux (BSC), Adria Quintanas (BSC), Martin Berggren (UmU), Samuel Kemmler (FAU), Ananias Tomboulides (AUTH), Timofey Mukha (KTH)
Responsible Author	Daniel Kempf (USTUTT) daniel.kempf@iag.uni-stuttgart.de
Keywords	CFD lighthouse cases, resolution/resource requirements, HPC
WP	WP1
Nature	R
Dissemination Level	PU
Final Version Date	31/08/2023
Reviewed by	Niclas Jansson (KTH), Manuel Münsch (FAU)
MGT Board Approval	31/08/2023

Acknowledgment:

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Sweden, Germany, Spain, Greece, and Denmark under grant agreement No 101093393.

Document History

Partner	Date	Comment	Version
USTUTT, FAU BSC	09/04/2023	Define questionnaire for the consortium codes	0.1
USTUTT	16/05/2023	Initial analysis of the questionnaire outputs	0.2
USTUTT	10/07/2023	Preliminary version with initial input for each LHC	0.3
USTUTT, KTH, BSC, FAU, UmU, DTU, BAM	21/07/2023	Draft of the deliverable for proofreading by each LHC representative	0.4
USTUTT	25/08/2023	Incorporation of the remarks from the first reviewer	0.5
USTUTT	30/08/2023	Incorporation of the remarks from the second reviewer	0.6
USTUTT	31/08/2023	Final version	1.0

Executive Summary

This document represents deliverable 1 of work package 1 — 'Exascale light-house cases' of the Centre of Excellence in Exascale CFD (CEEC). The CEEC project aims to develop CFD frameworks that efficiently exploit future exascale systems. For this purpose, the project is organized around six light-house cases (LHCs), representing challenging problems in the field of CFD that require the capabilities of these future HPC systems.

Deliverable 1.1 — 'CEEC exascale lighthouse cases and their needs' marks the completion of the 'requirements' phase of this project and represents the second milestone of the project. After the introduction to the LHCs and the associated codes, an estimation of the requirements for each LHC will be presented. Here, the targeted problem size in terms of degrees of freedom, the required memory consumption, target HPC architecture including the required numbers of CPUs and/or GPUs, I/O needs, and the status of exascale readiness is presented for each LHC. Synergies that arise between the codes involved and the developments in the technical work packages are also discussed.

The requirements stated in this deliverable represent a preliminary status. These cannot yet be finalized because they depend strongly on the further developments within the project as well as on the further development of the HPC resources and timely availability. These will be updated internally on a regular basis.

Contents

1 Introduction

The Center of Excellence in Exascale CFD (CEEC) aims to advance state-of-the-art CFD algorithms and models with the clear goal to enable exascale performance. This will be demonstrated by the example of six lighthouse cases (LHCs) which have high relevance for scientific and industrial applications.

To maximize the impact of the CEEC and to ensure that the developments help to meet the requirements of specified real-world CFD applications for future exascale systems, the CEEC will employ a co-design process. For this purpose, the CEEC project is organized around six LHCs, representing challenging problems in the field of CFD that require the capabilities of these future HPC systems. The LHCs will drive the development of the codes and workflows such that they will be ready to be used on exascale systems.

The overall goal of the work within this work package (WP1 — 'exascale light-house cases'), is to deliver exascale-ready implementations of the codes and workflows which are developed within the technical work packages WP 2 to WP 5. These developments are designed to meet the requirements of the six CEEC LHCs. The six LHCs considered cover a broad range of CFD applications:

- Shock-Boundary layer interaction and buffet on wings at the edge of the flight envelope
- High fidelity aeroelastic simulation of the SFB 401 wing in flight conditions
- Topology optimization of static mixers
- Localized erosion of an offshore wind-turbine foundation
- Simulation of Atmospheric Boundary Layer flows
- Merchant ship hull

Following this introduction, an individual section is devoted to each of the six LHCs. Each of the sections covers an introduction to the LHC, a description of the linked code and the associated status of exascale readiness, synergies that will be deployed within the CEEC, and finally the computational requirements for the LHC. For the estimation of the required computational requirements an estimation of the targeted problem size in terms of degrees of freedom, the required memory consumption, and target HPC architecture including the required numbers of CPUs and GPUs, I/O needs, and the status of exascale readiness is presented. However, due to the still early project phase, the figures given are initial estimations to the best of our knowledge, but all figures are subject to change during the project.

2 LHC1: Shock-Boundary layer interaction and buffet on wings at the edge of the flight envelope

2.1 Introduction to the lighthouse case

Modern transport aircraft cruise at transonic flight conditions, where the local acceleration over the suction side of the wing leads to localized supersonic flow. These supersonic pockets are terminated by normal shock waves, i.e., confined and highly localized regions

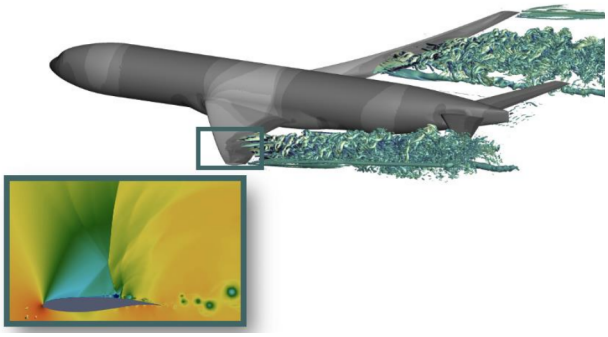


Figure 1: Main wing wake structures at high-speed conditions on the main wing of the Common Research Model (CRM) aircraft. The inset shows a detailed shock-boundary layer interaction on a wing cross-section. CRM simulation used with friendly permission from Th. Lutz, Institute for Aerodynamics and Gas Dynamics, University of Stuttgart.

in the flow where all flow quantities undergo drastic changes — the flow becomes 'shocked'. This is shown in Fig. 1. The strength of these shocks increases with flight Mach number and Angle of Attack (AoA) and the associated pressure gradients can trigger flow separation and thus loss of lift. Under certain conditions near the edge of the flight envelope, a complex interaction between the separated flow field passing over the trailing edge of the wing and the shock itself can occur, which leads to a low frequency, high amplitude periodic oscillation of the shock position. This self-sustaining feedback mechanism is called a shock buffet. It not only leads to high dynamic loads on the wings, a highly unsteady wing wake, and a rough flight but increased structural fatigue of the wing in the long run. Thus, understanding and reliably predicting such phenomena is of utmost importance for flight safety, efficiency, and radical new aircraft design concepts.

2.2 Linked code and exascale computing

2.2.1 Introduction of the code

FLEXI[1] is a high-order accurate, open-source solver for general partial differential equations of hyperbolic/parabolic-type based on the discontinuous variant of the spectral element method (SEM), the discontinuous Galerkin (DG) spectral element method (DGSEM) [4]. This enables an element-local and efficient scheme, even on highly parallel systems. Its primary area of application are direct numerical and large eddy simulations (LES) of multiscale- and multi-physics problems, where the fluid phase is governed by the compressible Navier–Stokes–Fourier equations (NSE), which includes turbulent flows and shock-turbulence interaction. Since high-order schemes are subject to oscillations at discontinuities, additional stabilization techniques are necessary to accurately predict strong compression shock waves across which the flow properties change drastically. In FLEXI, discontinuities in the solution such as shock waves are handled by shock-capturing strategies through an elementwise h-refinement strategy in a localized and stable manner. This approach is based on a finite volume (FV) sub-cell approach, where troubled cells are switched from a DG to a FV discretization. Another more stable approach relies on a convex blending of the low-order FV sub-cell with a high-order DG scheme. To ensure nonlinear stability in underresolved flow regions as occurring in a LES, FLEXI uses a

[1] https://github.com/flexi-framework/flexi

kinetic energy or entropy stable discretization based on the split form of the advective terms. Moreover, various closure strategies for the NSE in a LES formulation based on implicit or explicit modeling are available. High temporal accuracy is ensured through high-order, low-storage explicit Runge-Kutta (RK) schemes. FLEXI supports unstructured grids of tensor-product based elements due to the DGSEM and high-order accurate geometry representation including curved grid cells and hanging nodes. Grid adaptation is handled through a conservative mortar interface definition.

FLEXI is equipped with high-order capable pre- and post-processing tools. Grid preparation for high-order schemes, in particular the element surface and volume curving as well as the mortar connectivity, is handled by the HOPR framework[2].

2.2.2 Current code status and steps towards exascale readiness

FLEXI has demonstrated excellent scaling on several supercomputer architectures, including Jülich JUGENE, HLRS Hazel Hen/Hawk, and LRZ SuperMUC-NG. Since FLEXI is designed as a massively parallel code with the MPI paradigm and written in modern FORTRAN, it consistently uses data structures optimized for continuous, non-overlapping access. Mesh and solution files are ordered along a space-filling curve, fully utilizing the multi-threaded capabilities of the underlying file system. During the initialization phase, the initially unstructured data is reordered into continuous segments optimized for the selected thread distribution, thereby facilitating single-instruction multiple-data (SIMD) vector operations and cache utilization. Combined with the small memory footprint of the DGSEM scheme, FLEXI can achieve optimal performance with as long as 3000 degrees of freedom (DoF)/core. To further adapt to the individual performance characteristics of the target machine, the framework utilizes compiler and linker-level techniques such as profile-guided optimization. To transfer the serial performance to the parallel context, FLEXI leverages extensively the hardware offloading capabilities provided by the interconnect. The highly local DGSEM operator requires only the exchange of surface flux information which is achieved through non-blocking communication of linear 1D buffers. Local volume work is utilized for latency hiding with ongoing research to achieve zero-copy remote memory access communication which eliminates CPU operations and conserves memory bandwidth. Global communication during runtime is reduced to the minimum required amount of one data exchange that is necessary to calculate the time increment of the explicit time-stepping scheme.

In its current state, FLEXI has demonstrated its suitability for both large setups ($> 10^{10}$ DoFs) and high parallel efficiency [1]. It offers excellent scaling behavior regarding strong and weak scaling. While the unstructured code sections are well suited for general-purpose CPUs, the next step in its development is to offload the entire core DGSEM solver to GPUs to minimize restrictive slow data transfers. Here, we expect that the code has a much broader optimum and a larger computational load per GPU can be utilized. For this, the routines are adapted to benefit from the massive parallelism of GPUs. Due to the high locality and computational intensity, the existing strategies for latency minimization of the communication will be implemented. When combined with zero-copy communication, the resulting framework will be able to fully utilize modern hardware capabilities while remaining portable over a wide range of architectures.

[2]https://github.com/hopr-framework/hopr

2.2.3 Synergies within CEEC

The following synergies will be leveraged to achieve the common goals of CEEC with greater efficiency and from which FLEXI and LHC1 can benefit: The performance engineering, continuous integration, testing, and performance tracking strategies and methods developed within WP2 — 'software and performance engineering' will be applied. This includes, for example, a high-performance implementation of relevant compute kernels for different architectures. Here, an intensive exchange and transfer of knowledge between the developers of FLEXI and waLBerla will be considered. The joint efforts in WP3 — 'exascale algorithms' consider possibilities from which FLEXI can benefit, e.g., mixed-precision algorithms apart from the explicit kernel routine. Moreover, codes that utilize similar numerical schemes, such as FLEXI and Alya (explicit, compressible CFD solvers which are SEM-based) can potentially exploit synergies in algorithmic design. In WP4 — 'exascale techniques', we will share our experience and existing methods in the area of ML and HPC computing with other project partners, namely the codes Alya and Neko. This is among others the integration of machine learning models in FLEXI utilizing SmartRedis to enable the coupling with TensorFlow.

2.3 Computational requirements for the LHC

Several intermediate stages are planned on the way to the exascale LHC. The final lighthouse case will represent the simulation of a three-dimensional airfoil in a buffet condition, where a shock oscillates cyclically on the suction side of the airfoil in a transonic flight condition. For LHC1, a minimum computation time of three buffet cycles is planned. An increase to at least six cycles for greater statistical confidence in the evaluation is envisioned for the LHC. Further, based on our experience, about three buffet cycles are necessary for the start-up process until the cyclic buffet sets in. This value may differ for the three-dimensional wing configuration. Thereby, one buffet cycle corresponds to approximately 15 convective time units $T^* = c/u_\infty$. The exact period length for the three-dimensional airfoil is currently unknown and a longer period length would cause a linear increase in computation time.

The targeted lighthouse case requires the simulation of a three-dimensional airfoil with a spanwise extent of $3.8c$, with c being the chord length, including the consideration of the wing tip. Before that, intermediate steps are planned where a wing segment with a $0.5c$ span as a development model and one with a $2c$ span are investigated. These numbers serve only as a first estimate since the intermediate steps related to the spanwise expansion will be appropriately adjusted as the project progresses.

Currently, the following airfoil mesh is assumed for the simulation: The boundary layer is resolved using a boundary layer grid with a resolution in the range of $\Delta x/c = 5.1 \cdot 10^{-3}$, $\Delta y/c = 8.7 \cdot 10^{-4}$ and $\Delta z/c = 2.3 \cdot 10^{-3}$ per element using a polynomial degree of $N = 7$. This accounts for the use of a wall model in the LES (WMLES). The use of a wall model further leads to no relevant Reynolds number dependence in the considered region and the boundary layer mesh is not adapted to variations in the Reynolds number. Moreover, the grid is coarsened towards the free-stream region. To significantly reduce the DoFs, a stepwise coarsening of the mesh resolution in the spanwise direction towards the far field is pursued using non-conforming mesh transitions.

Tab. 1 provides estimates for the LHC requirements at a polynomial degree of $N = 7$.

This includes the number of elements and DoFs, the number of CPUs and GPUs required for the simulation, the total memory requirements, the total simulation costs without post-processing, and the estimated file sizes. Data is given for the case normalized to unit span, the intermediate step 1 with a $0.5c$ span and $90T^*$ simulation duration, the intermediate step 2 with a $2c$ span and $90T^*$ simulation duration, and the final LHC1 case with $3.8c$ span and $180T^*$ simulation duration.

For the estimation of the required hardware resources, a load of 12 000 DoFs/core was assumed in the CPU case. In the GPU case, the load per GPU is assumed higher compared to a CPU core. It is estimated by the performance equivalent of one A100 to two nodes with 128 CPU cores, e.g., AMD EPYC 7742 CPUs. This results in an estimated load of 3.1M DoFs/device. Concerning memory bandwidth, the DGSEM scheme is computationally intensive and is generally limited by memory bandwidth such that the program benefits from an increase in memory bandwidth. Currently, we are not aiming at a specific HPC system, but are working on having the code run efficiently on as many European systems as possible.

Since the physical problem in LHC1 is a dynamic process, the output of time-resolved data is required. A completely resolved output of the volume solution is not possible, therefore a restartable state as well as a time average with associated turbulence statistics is stored at each T^*. Point probes are used for the output of the temporal solution. Approximately $500\,000/c$ samples are utilized for this purpose. The output of the solution takes place after every 10^{th} time step, resulting in an output of 3500 time steps per convective time unit T^*.

	Normalized by c and T^*	Interm. 1 ($0.5c$, $90T^*$)	Interm. 2 ($2.0c$, $90T^*$)	LHC1 ($3.8c$, $180T^*$)
Elements	2M	1M	3.9M	8.3M
DoFs	1B	0.5B	2B	4.5B
#Cores	84,000	42,000	170,000	375,000
#GPU	330	165	650	1470
Total memory	3 TB	1.5 TB	6 TB	14 TB
Core-h / GPU-h	440K / 2K	10.8M/39K	40M/160K	176M/0.7M
Size states	45 GB	2 TB	8 TB	36 TB
Size average	120 GB	5 TB	21TB	96 TB
Size probes	30 GB	14 TB	6 TB	21 TB

Table 1: Estimation of the required DoFs, hardware resources, and file sizes for LHC1 using FLEXI.

3 LHC2: High fidelity aeroelastic simulation of the SFB 401 wing in flight conditions

3.1 Introduction to the lighthouse case

Wind tunnel simulations are usually stiff and therefore avoid elastic deformations of the structure and thus aeroelastic effects. However, aeroelastic effects on the aircraft wing may appear for specific flight conditions, e.g., cruise at transonic speed, and tend to be

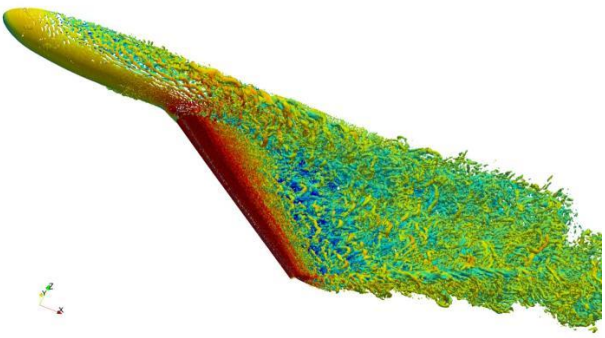


Figure 2: Application of a WMLES for the aerodynamic prediction of the flow over a full aircraft [3] (AoA = 21°, Reynolds number: Re = $1.7M$, Mach number: Ma = 0.17 with a mesh of 2B of cells and solved with 100K CPUs).

a more complex phenomenon to be solved numerically. Thus, multi-physics simulations of the wing structure and the flow field are necessary to generate realistic and accurate solutions for the certification of the structure. Typical flight conditions under subsonic and transonic regimes of civil aircraft ($Ma < 1.0$) are characterized by strong nonlinearities. This is depicted in Fig. 2, which illustrates the aerodynamic prediction of the turbulent flow over a full aircraft in stall conditions. The occurring nonlinearities strongly depend on high Reynolds numbers and the influence of the transient and steady flow and may induce shock flow separation or even shock-buffets with unsteady fluid-structure interaction. Due to the lack of computational methods to solve this kind of problem together with the elevated computational cost of the coupled problem, this type of simulation may be impractical without the use of High-Performance Computing (HPC).

In this LHC, an aeroelastic simulation of the SFB 401 wing, also known as the HIRENASD wing model, in a transonic regime ($Ma = 0.8$) will be conducted using existing LES models for compressible flows and the multi-physics code Alya. This simulation reproduces typical flight conditions under transonic conditions with high Reynolds numbers and elastic deformations for the structural part. This LHC allows to address existing methodology gaps in the coupling of advanced LES models with structural dynamics. Examples include the understanding of existing nonlinearities that occur in transonic regimes in real wing reference configurations, the accurate determination of aeroelastic deformations in flight conditions, and finally, computational advances for coupled problems towards exascale computing. Most of these gaps are not covered due to the high cost of coupled simulations which tends to be not feasible in a production setting and also the lack of computational methods to solve these problems. With the existing computational cost of aeroelastic simulations in mind, the LHC2 is attractive to be solved by means of exascale computing. Thus, advances in efficient parallel CFD (flow part) and CSD (structure part) solvers for heterogeneous architectures will be strong drivers for aeroelastic simulations.

3.2 Linked code and exascale computing

3.2.1 Introduction of the code

Alya[3] is a multi-physics simulation code developed at Barcelona Supercomputing Center (BSC). From its inception, the Alya code is designed using advanced HPC programming techniques to solve coupled problems on supercomputers efficiently, especially taking heterogeneous architectures into account. The target domain are engineering applications, with all their particular features. Examples include complex geometries and unstructured meshes, coupled multi-physics with exotic coupling schemes and physical models, ill-posed problems or flexibility needs for rapidly including new models. Since its beginnings in 2004, Alya has scaled well on an increasing number of processors when solving single-physics problems such as fluid mechanics, solid mechanics, heat transfer or combustion. Over time, we have strived a concerted effort to maintain and even improve scalability for multi-physics problems. This poses challenges on multiple fronts, including numerical models, parallel implementation, physical coupling models, algorithms and solution schemes, and meshing processes. Alya is present in the PRACE benchmark suite (UEABS) together with Code_Saturne in the field of CFD and thus, is considered as the reference in the EU framework for supercomputing. The core Alya Dev Team has today around 50 members, distributed among the BSC and its spinoff, ELEM Biotech.

The existing capabilities of the Alya code in terms of LES modeling and multi-physics couplings render this code suitable for the proposed lighthouse case. The fluid-structure interaction simulation concerning this LHC will be performed by coupling Alya and SOD2D (Spectral high-Order coDe 2 solve partial Differential equations), which is also developed and maintained at BSC. Alya will control the workflow, perform the projections between both domains, and solve the structural part. In turn, SOD2D will solve the turbulent fluid domain employing an LES model and spectral high-order elements. Regarding the LHC, the flow (CFD part) will be solved using WMLES for compressible flows, while the structure (CSD part) will be solved by a solid mechanics framework specifically devoted to transient non-linear problems with large deformations. The implicit Newmark time-integration scheme will be utilized together with an iterative solver for the resolution of the algebraic system for the dynamic analysis of the structure, while the CFD part will be treated explicitly in time. With regards to the coupling algorithm and to preserve the advantages of the highly adapted CFD and CSM solvers, a partitioned approach with weak coupling will be conducted. To account for the elastic deformation of the structure, the Alya mesh deformation tool will be employed which is based on an arbitrary Lagrangian-Eulerian (ALE) formulation.

3.2.2 Current code status and steps towards exascale readiness

The current implementation of the software incorporates the following features with respect to parallelization. Alya's kernel and structural mechanics solver employs a master-worker strategy through MPI to communicate between nodes (coarse level). In turn, it employs MPI within the node (mid-level) but is also ready to use OpenMP. Alya can also be used with a runtime dynamic load balance (DLB) strategy to leverage load imbalance issues. Finally, it exploits the SIMD instructions through the compiler to execute concurrent operations in the CPU. The solid domain will be solved using an implicit algorithm

[3]https://gitlab.com/bsc-alya/alya

which can be a bottleneck of the simulation for two reasons: the stiffness of the matrix, which can be large, requiring complex preconditioners and the communications of the solver.

SOD2D uses a fully distributed approach by means of OpenACC and MPI or NCCL. Depending on the considered hardware, it is possible to utilize CUDA aware MPI communications, therefore exploiting peer to peer communications. On GPUs, the performance is greatly enhanced by combining a good fine-grained parallelism strategy with re-usage of data structures already present on the GPU and single precision arithmetic. Indeed, all the kernel execution is performed on the GPU, which almost diminishes the issue of host/device data transfers. Parallelism at the GPU level is achieved by distributing each element on a mesh to a thread block on the GPU and allowing each nodal operation to be computed by a single thread of this block. Since each thread has only a few operations to compute, the GPU warps are able to be activated rather quickly, resulting in excellent performance over the most taxing kernels. The code has also a small memory footprint, which would allow for far greater loads per GPU, although at the detriment of scalability. The bottlenecks of SOD2D are expected to be related with load balancing, efficient use of shared memory of the GPUs, and the data movement between host and device.

For both Alya and SOD2D codes, the mesh is partitioned using the GEMPA[4] library, included as a submodule in these codes. Regarding the parallel I/O operations, Alya uses MPIO-tools, a built-in tool to manage binary files, while SOD2D uses HDF5 library.

The next steps towards exascale readiness can be divided into three blocks. First, the coupling schemes available in Alya's kernel will be recycled and extended for projecting solutions between non-matching meshes. Specifically, the current algorithm will be extended to ensure conservative projections between spectral meshes using third order polynomials involved in the fluid domain and Lagrangian meshes using second order polynomials in the solid one. Second, the structural solver in the solid domain will be remanufactured for high-order elements and ported to GPU. The first milestone will be the extension of the available element library to deal with elements of order higher than two and several node distribution schemes. The second milestone will be concerned with the porting of the elemental assembly to GPU using OpenACC directives. The final milestone will address the porting of solvers to the GPU. For this, Jacobian-free methods or third-party libraries will be explored. Their use will depend on the requirements of the physical problem, which are unknown at the current stage. Third, the main work on the fluid side will be the coupling of Alya and SOD2D, harmonization of data structures, organization, and improvement of the performance at large scale GPU clusters. A standalone version of SOD2D is already able to work up to 400 GPUs and meshes up to 4B grid nodes. The code is based on hybrid OpenACC and MPI or NCCL.

3.2.3 Synergies within CEEC

The following synergies will be leveraged to achieve the common goals of CEEC with greater efficiency, which will benefit Alya and LHC2: The joint efforts in WP3 – 'exascale algorithms' consider possibilities from which Alya could benefit, e.g., mixed-precision algorithms. Moreover, codes that utilize similar numerical schemes, such as FLEXI and Alya (explicit, compressible CFD solvers which are SEM-based) could potentially exploit

[4]https://gitlab.com/rickbp/gempa

synergies. In WP4 – 'exascale techniques', the integration of the machine learning models in SOD2D will be carried out using SmartRedis to enable the coupling with TensorForce and PyTorch. This could be shared by Alya, FLEXI, and Neko.

3.3 Computational requirements for the LHC

For LHC2, the estimation of the requirements to properly resolve the fluid and solid phases are the following: Based on similar investigations, our estimation is, that 4B - 8B DoFs are required to resolve the fluid phase and about 5M - 10M for the solid phase. In Tab. 2, an estimation of the number of DoFs required to properly resolve the physical behavior of the coupled simulation is summarized in comparison to similar investigations.

Based on our current information, to compute the LHC simulation with this number of DoFs, a system with approximately 1000 GPUs (e.g. NVIDIA H100) and 8000 CPUs (e.g. Intel Sapphire) will be required.

At this point, we estimate that with those required DoFs, each simulation run will generate about 10 TB of data which is required for further analysis and needs to be stored.

	Chwalowski (2011)	Hassan (2012)	Ritter (2021)	CEEC LHC2 (2026)
Fluid	10 - 30M	3.5M	3.7M	4B - 8B
Solid	170 - 200K	200K	200K - 358K	5M - 10M

Table 2: Estimation of the required DoFs for LHC2 for the fluid and solid solver in comparison to existing simulations.

4 LHC3: Topology optimization of static mixers

4.1 Introduction to the lighthouse case

Static mixers are used in practice since the 1970s for blending fluids in a wide range of applications ranging from waste-water treatment, food processing to pharmaceutical and chemical applications. Essentially, a static mixer is a part of a pipe in which the incoming fluid streams are mixed by the deflection effect of the internal geometry, with the goal of achieving highest mixing with the least pressure drop. A static mixer is a passive device without any moving parts. Static mixers are thus very reliable with low maintenance needs and typically low production costs. To optimize static mixers, a profound understanding of the fluid mechanics is necessary: The three main mechanisms needed to achieve these objectives are i) creating and promoting disturbances by additional obstacles in the main flow, ii) modifying the flow topology by sequentially splitting and recombining the fluid streams and iii) generating chaotic advection by sequences of bends which promote so-called secondary flow. In particular, in the turbulent regime, enhanced fluctuation levels are promoted to exploit turbulent diffusion processes. Given the multiscale nature of turbulence, both macromixing of the large convective scales connected to the mean flow, and mesomixing related to the smaller inertial scales, and micromixing at the diffusive scales need to be considered. The exact working principles of static mixers are typically part of the trade secrecy, and thus only limited literature exists.

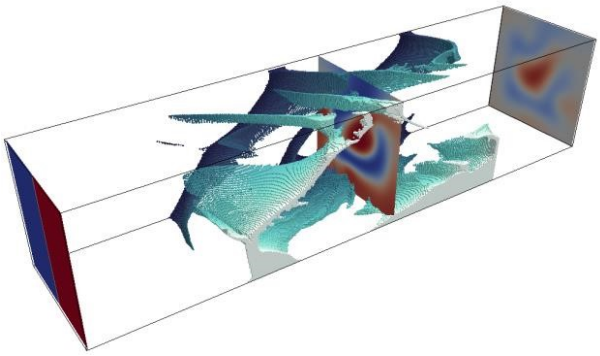


Figure 3: Topology optimized static microfluidic mixer with intricate structure and three slices illustrating stretching and folding of diffusive interface in convection dominated transport.

Preliminary research on topology optimization on static mixers in strict convection-dominated laminar flows, as illustrated in Fig. 3, shows great improvement in the ability to change the flow topology and increase the diffusion interface. Current implementations are limited to steady-state incompressible laminar single phase flow resolved by tens of millions of grid cells; however, industrial scale optimization, flow rates, and relevant flow details in also turbulent flows require a leap in algorithms and implementations.

Here, topology optimization, based on a mathematical formulation of the respective objective functions, will be implemented using the code Neko to obtain novel mixer designs, where the fluid mechanical aspects of the problem are fully resolved. A hierarchy of complexity, ranging from laminar single-phase flow to very complicated chaotic turbulence flow with multiple phases will be considered, tracking the Pareto front corresponding to the multi-objective optimization problem. Using the material distribution method for topology optimization, initial designs may be either open channels or, e.g., innovative static mixer designs by Westfall, with a specific sequence of fins oriented in vertical and horizontal planes.

Optimization of large-scale fluid systems is clearly a challenging problem, both from a purely fluid-dynamics point of view, and from the optimization. Depending on the flow speed, small scales related to turbulent chaotic motion are created, potentially further mediated by aspects such as compressibility, heat transfer, and multiple phases including bubbles and particles. As a point of reference, currently, the largest simulation of a single phase turbulent pipe flow was run by KTH on 80 billion grid points, consuming several million CPU hours; for a Reynolds number relevant for turbulent static mixers. For the optimization, there are two main challenges to be addressed: i) the optimization methods, including the sensitivity calculation, need to be able to be coupled to high-performance CFD codes, and need to scale to millions of cores. ii) as static mixers will operate in the turbulence regime, methods need to be developed to be able to compute sensitivities for a chaotic system, without leading to a blowup of the adjoint solution. For any further advancement of the field, such methods are urgently needed.

LHC3 marks the first demonstration of topology optimization of turbulent static mixers in complex flow situations. This LHC will advance the current state-of-the-art on various levels: algorithmically for large-scale optimization, mathematically for developing ways to

treat a chaotic system, and computationally to perform large-scale 3D optimization. The code to be used in the optimization is Neko, given its high accuracy and efficiency. The impact on industries is immediate, given the various industries ranging from biomedical to petro-chemical applications.

4.2 Linked code and exascale computing

4.2.1 Introduction of the code

Neko[5] is a portable framework for high-order spectral element based simulations, focusing on the incompressible regime. The framework is written in modern Fortran and adopts an object-oriented approach, allowing for multi-tier abstractions of the solver stack and facilitating various hardware backends, ranging from general-purpose processors, accelerators, and vector processors, to limited support of FPGA. Neko focuses on single core/single accelerator efficiency via fast tensor-product operator evaluations. A key to achieving this is a matrix-free formulation, where one always works with the unassembled matrix on a per-element basis. Gather–scatter operations are used to ensure continuity of functions on the element level, operating on both intra-node and inter-node element data.

The primary consideration in Neko is how to efficiently utilize different computer hardware without re-implementing the entire framework for each supported backend. We solve this problem by hiding the implementation of backend-dependent low-level kernels behind a common interface realized as an abstract Fortran type. This way, types implementing higher-level concepts, for example, the fluid solver, can remain completely backend-agnostic.

4.2.2 Current code status and steps towards exascale readiness

Neko has demonstrated excellent strong scaling on several general-purpose processor-based supercomputers, including Cray XC40 and EX systems (Beskow and Dardel at PDC), HPE Apollo 9000 (Hawk at HLRS) to clusters with recent vector processors, i.e., SX-Aurora (Vulcan at HLRS), achieving more than 70% parallel efficiency with as few as 4-8 elements per processing element. Each backend in Neko have been optimized to best exploit the underlying hardware and achieves 30% of the theoretical peak performance when running the CEED BK5 benchmark on AMD EPYC 7742 Rome and 7763 Milan, and 20% of theoretical peak on a NEC SX-Aurora 10B Vector Engine.

Extreme-scale strong scalability has been demonstrated on the accelerator partition of the 309 PFlop/s European pre-exascale machine LUMI at CSC. The results show that Neko achieves close to 80% parallel efficiency for a large direct numerical simulation, going from 4096 up to 16,384 AMD MI250X Graphics Compute Dies (GCDs), representing 20%, 40% and 80% of the entire LUMI supercomputer. Therefore, Neko is, in principle, ready to be deployed and tested on an exascale machine. However, advanced fault-tolerance and check-pointing mechanisms needs to be incorporated to address the light-house cases.

[5]https://github.com/ExtremeFLOW/neko

4.2.3 Synergies within CEEC

The following synergies will be leveraged to achieve the common goals of CEEC with greater efficiency, which will benefit Neko and LHC3: Through the collaborative efforts in WP3 – 'exascale algorithms', we could benefit from mixed-precision in preconditioners. This concept will also be propagated to the underlying Krylov-type solvers. Since the codes Neko, Nek5000, and NekRS have the same roots and therefore, share many similarities, the efforts in WP3 could potentially be shared by these codes. Moreover, these similarities can be leveraged for further developments, including strategies for performance measurements and efficient portability to different computing hardware. Furthermore, there are possibilities to develop common approaches for dealing with fault tolerance, and the software stack for enabling machine learning inference in parallel with the solvers' time stepping. Codes based on similar numerical schemes, such as Neko and FLEXI (both SEM-based, but different flavors) will certainly benefit from discussions of common problems that arise during the lifetime of the project.

4.3 Computational requirements for the LHC

The main code for this LHC is Neko, and as such both CPU and GPU based machines can be used. Topology optimization leads to an iterative procedure where the sensitivity of the density (solid–fluid distribution) is computed using a direct–adjoint simulation, and update of the design (using a gradient-based algorithm), a new computation of the sensitivity etc. Therefore, the final compute time is not only given by the resolution requirements of the case at hand (static mixer), but rather the complexity of the objective function, and the expected design. Note that one iteration is the combination of a forward/direct simulation using the usual Navier–Stokes equations, followed by the solution of the adjoint equations. This latter step required the forward solution to be available at each time step; this can be achieved with so-called revolve algorithms and require a re-computation of the forward solution. Thus, the cost of one iteration can be approximated conservatively by the four-fold cost of one forward solution. Assuming $\mathcal{O}(100)$ iterations for a reasonable design leads to order 400 forward evaluations for one converged design. This factor obviously limits the size of the flow case, keeping the required computer time high. Note that additional parallelization techniques can also be explored, including ensembles, and parallel computation of different designs.

The resolution requirements for a specific case are driven by the complexity and resolution of the geometry (which is imposed using an immersed boundary method - IBM), and the expected steady or unsteady flow structures. We focus here on direct numerical simulation, i.e. without the use of a turbulence model. Based on our previous experience using the spectral-element method for topology optimization, for a moderately complex case with walls, the number of DoFs is on the order of millions to perhaps 100 million points. Depending on available resources, we may want to extend this grid count progressively. To estimate the computational requirements, we consider 15 elements per CPU core as the minimum, after which parallel efficiency becomes too low (strong scaling limit). For 10^8 unknowns and polynomial order 7, the number of elements is $\approx 2 \cdot 10^5$. Dividing by 15 gives the estimate of $1.3 \cdot 10^4$ cores. To estimate the number of GPUs, we relate to existing data for Neko that shows that a single GPU roughly corresponds to a single CPU node with 128 cores (on e.g. LUMI). This gives $\approx 10^2$ GPUs. Note that Neko has been

DoFs	CPUs	GPUs	Output size
~ 100M	$\sim 10^4$	$\sim 10^2$	≤ 10TB

Table 3: Estimation of the required hardware resources for topology optimization using Neko and the disc space used for a whole iteration.

shown to scale with nearly ideal parallel efficiency on up to 16 386 GPUs on LUMI.

Memory size can be relevant for the computation of the adjoint equation, and the number of snapshots that can be kept in the memory without re-computing. However, the revolve algorithm ensures and optimal use of the various hierarchies of the memory (RAM, flash, disk). For I/O, the main issue is the disk space for the storage of the forward solution (checkpointing), which can be a few TB per simulation/iteration. Thus, tens of TB may be required for a reasonable simulation campaign. On the other hand, the number of files per simulation is expected to be unproblematic due to the consistent use of MPI-IO in a distributed environment. A summary of the required resources is provided in Table 3.

5 LHC4: Localized erosion of an offshore wind-turbine foundation

5.1 Introduction to the lighthouse case

Suction foundations for offshore wind turbines gain growing relevance as an environmentally friendly alternative to monopile foundations. However, if the operating suction exceeds a critical threshold during the suction-driven installation, the large hydraulic gradients within the flow network may cause a localized erosion of soil channels, which would prevent further installation. For complex marine soils, deriving simple analytical solutions for the critical suction threshold is often impossible. Instead, physical tests or predictive numerical simulations are used. Unfortunately, the local fluidization occurring during piping often appears at the grain-scale, which is not tractable with the conventional macromechanical simulation methods employed for engineering problems. Thus, this LHC should produce a demonstrator of 3D particle-resolved simulation approaches for a real-scale geotechnical application, as illustrated in Fig. 4. The first focus is on the model validation with available experimental results, and then on a grand application focusing on a representative cut of the full-scale foundation during the first meters of the suction-driven installation. It should elucidate the unseen microscopic physics of localized erosive failure of large offshore foundations.

5.2 Linked code and exascale computing

5.2.1 Introduction of the code

waLBerla (widely applicable Lattice Boltzmann (LB) from Erlangen) is a modern open source multi-physics simulation software framework with a focus on CFD applications. The main unique feature of waLBerla is its uncompromising focus on large-scale simulations and scalability. It supports the massive parallelism of current peta- and future exascale supercomputers with a framework of carefully designed distributed data structures. Automated testing ensures the correctness of the functionality across a wide range

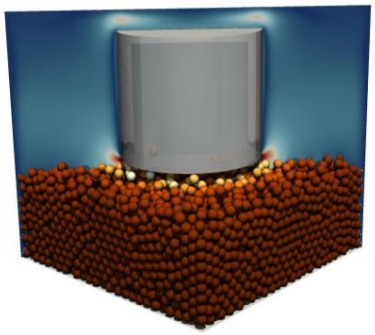


Figure 4: Prototype implementation of the fully-coupled 3D suction-bucket setup at lab-model scale with waLBerla.

of target hardware and software environments such that it is well-suited for robust further developments. Its current version also features adaptive techniques. Moreover, it includes functionalities for load balancing, for checkpoint-restart, and even automatic resilience techniques that may become essential on future extreme-scale systems. waLBerla contains efficient, hardware specific compute kernels to achieve optimal performance on most common supercomputing CPU and GPU architectures. Simulations of particulate flows have always been a special focus of the group's effort throughout the continuous development of waLBerla. The integration of the rigid body physics engines PE and MESA-PD allows us to model multi-physics scenarios with a granular phase.

5.2.2 Current code status and steps towards exascale readiness

waLBerla has demonstrated excellent strong scaling on a wide variety of supercomputers representing different architectures. The automatic code generation is a key component to ensure portability across most existing architectures, including GPU based systems. For example, scalable simulations with adaptive mesh refinement and load balancing have been demonstrated already on up to 2M CPU threads (on Juqueen) when processing meshes with a trillion unknowns. This represents some of the largest CFD computations that have been feasible on this class of pre-exascale machines. On the large GPU cluster PizDaint, waLBerla has exhibited almost perfect scalability for a two-phase benchmark case with up to 2048 GPUs. A careful roofline analysis for this case has shown that waLBerla could sustain a high fraction of aggregate peak memory bandwidth. This study demonstrates that waLBerla operates probably close to the architecture-specific optimum. As many top EU HPC machines are heterogeneous CPU-GPU clusters, the steps towards exascale readiness will mainly involve porting all simulation parts to GPU and supporting GPUs from different vendors, namely AMD, as they are used in LUMI. Furthermore, more advanced fault-tolerance and checkpointing mechanisms will be incorporated.

5.2.3 Synergies within CEEC

The following synergies will be leveraged to achieve the common goals of CEEC with greater efficiency, which will benefit waLBerla and LHC4: Through the collaborative efforts in WP3 — 'exascale algorithms', the performance of waLBerla could be improved using a mixed-precision Lattice Boltzmann approach. This reduces the required memory

traffic which is the bottleneck of the fluid simulation using the Lattice Boltzmann method. The performance engineering, continuous integration, testing, and performance tracking strategies and methods developed within WP2 – 'software and performance engineering' will be applied. This includes, for example, a high-performance implementation of relevant compute kernels for different architectures. Here, intensive exchange and transfer of knowledge between the developers of waLBerla and FLEXI will be conceivable.

5.3 Computational requirements for the LHC

The domain of our LHC in its final form will be a cuboid of size $3\,\mathrm{m} \times 0.5\,\mathrm{m} \times 1\,\mathrm{m}$. This domain will be discretized using cubic fluid cells of size $5 \times 10^{-4}\,\mathrm{m}$. This results in $6000 \times 1000 \times 2000 = 1.2 \times 10^{10}$ cells. This fluid domain will be filled with 1×10^{7} spherical particles for the particle-resolved simulation. Since there are three orders of magnitude between the number of particles and the number of fluid cells, the computational requirements are dominated by the fluid cells.

The computational requirements can be estimated as follows. We store in the order of 40-50 double-precision values per fluid cell and we typically have two grids. Therefore, we arrive at a main memory requirement of 5-10 TB for the whole simulation. The known bottleneck of this simulation will be the memory bandwidth. Here, we aim for hardware with high bandwidth, e.g., A100 GPUs. The final number of the required GPUs is still to be evaluated. Our target HPC architectures are clusters with NVIDIA and AMD GPUs, namely Leonardo and LUMI.

Regarding the required storage for the simulation, the I/O needs are highly dependent on the VTK output frequency and resolution. This will be adjusted depending on the resources available. All the estimated numbers are summarized in Table 4.

Lattice cells	Particles	Main memory
12B	10M	5-10 TB

Table 4: Estimation of the requirements for LHC4.

6 LHC5: Simulation of Atmospheric Boundary Layer flows

6.1 Introduction to the lighthouse case

Atmospheric boundary layer (ABL) flows, in addition to their role in vertical exchanges of moisture and aerosols in the atmosphere, also affect transportation, power generation by renewable sources (wind and solar), pollutant dispersion, and others. Efficient simulation of ABL flows is important for the study of wind farms, urban canyons, and basic weather modeling. Density stratification from surface heating and cooling directly affects these flows which are highly turbulent, with Coriolis effects caused by the earth's rotation complicating them even further. Regional weather patterns and terrain morphology add additional complexity to ABL flows.

In LHC5, the state-of-the-art LES of the stable and convective ABL is extended to examine the quality of LES solutions, and in particular their dependence on the mesh,

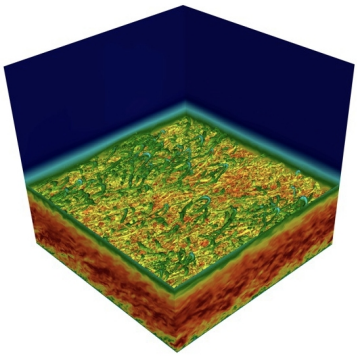


Figure 5: Isocontours of streamwise velocity in the GABLS benchmark from LES with Nek5000.

subgrid-scale (SGS) parameters, numerical discretization, and surface boundary conditions. For this, the high-order Nek5000 and NekRS codes will be used, including wall models based on the Monin-Obukhov [7] similarity theory for rough walls which are appropriate for variational formulation approaches such as the high-order SEM. We continue our collaboration with ANL and NREL scientists toward the cross-verification and validation of the LES results and corresponding wall models by performing a number of scaling studies to compare the performance of several ABL codes on CPU and GPU platforms. In the stably stratified ABL, e.g., the nocturnal ABL over land, the largest turbulent scales are often much smaller than those seen during neutral or unstable stratification such that the sensitivity to the SGS model in the LES is increased. Here, we follow the work of [9], where the SGS stress tensors are expressed in terms of a non-isotropic, an isotropic and a fluctuating part. Although continued improvement of SGS turbulence models is necessary, increasing simulation resolution is one route to reduce the dependence on SGS turbulence models. A well-documented stably stratified ABL benchmark problem that can be used for these purposes, namely, for model and code inter-comparison, is the Global Energy and Water Cycle Experiment (GEWEX) ABL Study (GABLS) benchmark, illustrated in Fig. 5, which is of direct relevance to wind-farm modeling and weather forecasting. For the convective ABL, numerical convergence is examined by means of a sheared daytime convective ABL reported in Sullivan and Patton [10] by tracking low- and high-order statistics and bulk entrainment.

Specialized kernels based on OCCA[6] will be developed for the CPU/GPU code NekRS, taking the memory hierarchy into account and minimizing memory transfer between host and device. Preliminary simulations [6] demonstrated very good agreement with results by other established ABL codes, but more importantly, excellent scalability on 4800 GPUs on Summit.

6.2 Linked code and exascale computing

6.2.1 Introduction of the code

Nek5000 and NekRS are highly-efficient and scalable open source incompressible and low Mach flow solvers[7] employing the SEM, a high-order weighted residual technique for spa-

[6]https://libocca.org
[7]https://nek5000.mcs.anl.gov

tial discretization that can accurately represent complex geometries. Globally, the SEM is based on a decomposition of the domain into E smaller subdomains (elements), which are assumed to be curvilinear hexahedra (bricks) that conform to the domain boundaries. Locally, functions within each element are expanded as Nth order polynomials cast in tensor-product fashion, which allow differential operators on N^3 grid points per element to be evaluated with only $O(N^4)$ work and $O(N^3)$ storage. The principal advantage of the SEM is that convergence is exponential in N, yielding minimal numerical dispersion and dissipation. Significantly fewer grid points per wavelength are required in order to accurately propagate a signal (or turbulent structure) over extended times in high Reynolds number simulations.

The solution procedure for solving the governing equations is based on a high-order splitting scheme, where the hydrodynamic equations are advanced with a backward difference/ characteristic-based (BDF/CHAR), time-stepping algorithm developed for the ALE method [8]. The BDF/CHAR scheme allows the simulation to overcome CFL restrictions imposed by standard schemes such as backward difference/extrapolation (BDF/EXT). Nek5000 is equipped with multilevel solvers that scale to millions of cores. The multilevel solvers require global coarse-grid solves that are based on fast direct solvers developed in Tufo and Fischer [11]. The pressure substep requires a Poisson solve at each step, which is affected through a multigrid-preconditioned GMRES iteration coupled with temporal projection to find an optimal initial guess. Particularly important components of Nek5000 are its scalable coarse-grid solvers that are central to parallel multigrid. Counts of 15 GMRES iterations per timestep for billion-gridpoint problems are typical with the current pressure solver.

Recent activity involves enhancement of Nek5000's support for combustion and reactive flows, multiphase (liquid/gas and fluid-particle-particle interactions), multimodel physics (e.g., drift-diffusion, combustion or RANS), and moving domains (e.g., rotating machinery, internal combustion engine or fluid-structure interaction in reactors). In the past several years, Nek5000 has been developed further to include nonconforming overlapping Schwarz discretizations, including multirate time-stepping in overlapping grids, advanced meshing and mesh optimization, and a high-order characteristic-based ALE approach for moving-domains.

6.2.2 Current code status and steps towards exascale readiness

Nek5000 has a history of scaling extremely well on a variety of architectures, including the IBM BG/P and BG/Q at ALCF and the Cray systems at the Swiss National Supercomputing Center. It also scales well on ALCF's Theta machine. Scaling studies on Theta during the ESP Workshop in 2016 showed 7:2x per node speedup in comparison with BG/Q. The low-Mach solver in Nek5000 demonstrated very good strong scaling across Theta's nodes, which exhibited 75% parallel efficiency for as few as 9000 points per MPI rank.

The GPU-oriented NekRS, developed with support from DOE's Exascale Computing Project as part of the Center for Efficient Exascale Discretizations, is built on the OCCA-based libParanumal library developed by Tim Warburton's group at Virginia Tech. The highly tuned kernels in this library run at the memory-bandwidth limit of the roofline model (meaning that they are running at the peak theoretical speed) and sustain in excess

of 2 TFLOPS (FP64) on the Nvidia V100 for polynomial degrees of $N = 14 - 16$. As described in Fischer et al. [2], the leading indicator of parallel scalability for a given algorithm-architecture coupling is n_{DoF}/P, rather than the number of processing units, P. Based on this, scalability studies for Nek5000, NekRS (CPU), and NekRS (GPU) on ORNL's Summit demonstrated excellent strong scaling for NekRS on all of Summit, pointing to an $n_{\mathrm{DoF}}/P \sim 2.5\mathrm{M}$ as the 80% efficiency level for the V100s. Currently the bottlenecks are strong scaling of gather/scatter type kernels, strong scaling of coarse polynomials and AMG multi-grid levels, data layout for low polynomial degrees, the data layout on the CPU backend and the performance imbalance between communication performance and compute power. These bottlenecks will be addressed within the CEEC project to further increase the performance of the codes.

6.2.3 Synergies within CEEC

The following synergies will be leveraged to achieve the common goals of the CEEC with greater efficiency, which will benefit the considered codes NekRS/Nek5000 and LHC5: The joint efforts in WP3 — 'exascale algorithms' consider possibilities in which NekRS/Nek5000 can benefit from, e.g., mixed-precision algorithms combined with the underlying Krylov-type solvers and in the preconditioners. Moreover, the codes Neko, Nek5000, and NekRS have the same roots and therefore, share many similarities that can be leveraged for further developments, including the efforts in WP3. The GPU-oriented NekRS, which is written in C++/OCCA, is the refactored version of Nek5000 which is written in F77/C. NekRS provides access to the standard Nek5000 interface and features (e.g., deformed geometry through an ALE formulation as well as overlapping domains), which allows users to leverage existing application-specific source code and data files on GPU-based platforms. In addition, the two codes share the same output file format and the results can be used interchangeably by both codes.

6.3 Computational requirements for the LHC

The two simulation cases within LHC5 are first the base case (GABLS) and second, the canonical daytime convective ABL. At this stage within the project, we expect the domain size of the GABLS case to be $(L_x, L_y, L_z) = (400\,\mathrm{m}, 400\,\mathrm{m}, 400\,\mathrm{m})$ and for the daytime convective ABL to be $(L_x, L_y, L_z) = (5120\,\mathrm{m}, 5120\,\mathrm{m}, 2048\,\mathrm{m})$.

For each case, four different mesh resolutions are investigated, ranging from 32^3 to 256^3 high-order spectral elements. Considering six variables per grid point (three velocity components, pressure, temperature, and turbulent kinetic energy (TKE)) results in 0.1B up to 52B DoFs. The associated resolutions and mesh data are summarized in Tab. 5.

These simulation cases will be investigated both with the CPU-based Nek5000 and the GPU-based NekRS. For the investigations with Nek5000, only the first three mesh resolutions are considered. In the case of the canonical daytime convective ABL, similar resolutions will be investigated. For those simulations, the required resources using Nek5000 and NekRS are summarized in Tab. 5. Here, an estimation of the required CPUs/GPUs and an estimated memory consumption based on the requirements of Nek5000 are provided.

Our strategy regarding I/O is the following: First, all data resides on the device, with

Elements	Grid points	DoFs	CPUs	GPUs	Required memory
32^3	256^3	0.1B	~ 1000	8	40 GB
64^3	512^3	0.8B	~ 8000	64	320 GB
128^3	1024^3	6.4B	~ 32000	512	3 TB
256^3	2048^3	51.5B		4096	20 TB

Table 5: Estimation of the required number of elements, grid points and DoFs for LHC5 as well as an estimation of the required hardware resources for Ne5000 (CPUs) and NekRS (GPUs).

Grid points	Output size (single field)	Output size (whole simulation)
256^3	0.57 GB	15 GB
512^3	4.6 GB	120 GB
1024^3	36.7 GB	954 GB
2048^3	293.5 GB	7630 GB

Table 6: File size of the solution for a single field and the whole simulation, depending on the mesh size.

a copy back to the host only when needed. We want to perform check-pointing every few thousand time steps, e.g., approximately every hour, which contains only restart information, i.e., the whole system state is not stored. For post-processing, the system will be stored every 20 minutes of physical time. At a simulation duration of the GABLS case of 9 hours of physical time, this results in about 26 field outputs. Each output file contains the field of the variables of interest (three velocity components, pressure, temperature, and TKE). The approximated file sizes depend on the resolution and are given in Tab. 6 for a single field and the expected 26 field outputs. Supplementary data such as mean and fluctuation profiles, surface data, point probes and integral values in time will be calculated during the simulation, but their size is negligible compared to the size of the field files.

For the actual output, Nek5000 and NekRS share the same I/O parallelization strategy. In the case of NekRS, where all data resides on the device, the data is copied back to the host only when memory-intensive data transfer has to be avoided. Application-level checkpointing is available in Nek5000/NekRS based on tuned MPI-I/O collective, reduced-blocking, and thread-based approaches, achieving a write performance of 70 GB/s at best. Nek5000/NekRS provide balanced I/O latency among all processors and reduce the overhead or even completely hides the I/O latency by using dedicated I/O communicators in the optimal case.

In terms of required memory bandwidth, LHC5 will gain significant performance as memory bandwidth increases as the OCCA-based kernels operate close to the bandwidth-limited roofline in the case of NekRS.

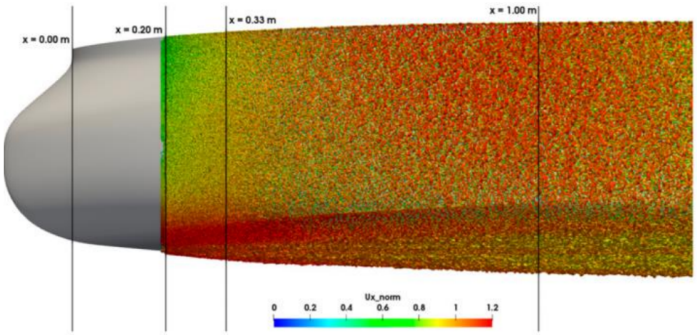


Figure 6: Resolved turbulent structures on the hull of the Japan Bulk Carrier. Reproduced from [5].

7 LHC6: Merchant ship hull

7.1 Introduction to the lighthouse case

A rapid shift towards more environment-friendly propulsion systems is a high priority for all branches of the transport sector. This includes the marine shipping industry, which currently accounts for approximately 90% of all freight transport. A critical component of evaluating the performance of a ship hull design is the accurate prediction of the associated flow structures forming near the hull's surface and in its wake. This concerns both the prediction of the friction drag, but also the performance of the propulsion system, particularly the propeller and potential Energy Saving Devices (ESDs). Both are located in the turbulent wake formed behind the hull of the vessel and are subject to associated unsteady loads.

Low-fidelity turbulence modeling approaches are not capable of properly resolving the large vortical structures forming on the hull and directly affecting the flow in the wake. Moreover, time-averaged treatment of turbulence is, by definition, not capable of providing certain key characteristics, such as the peak load on the propeller. Here, we will use scale-resolving turbulence modeling methods (LES and WMLES) to compute the flow around the Japan Bulk Carrier (JBC), see Fig. 6, which was studied extensively at the Workshop on CFD in Ship Hydrodynamics in Tokyo, 2015. As a baseline, we will consider the hull in double-body condition, i.e., without the effects of the free surface, and absence of ESDs and the propeller. For this configuration, a wall-resolved LES is performed, which will be used to analyze the flow physics and provide reference data for lower-fidelity runs using WMLES. For the latter we will use both traditional RANS-based wall models and those based on machine learning, leveraging the developments in WP4. Neko will be used to perform all the simulations, leveraging its high-order accuracy, computational efficiency, and portability to different computing hardware.

The scientific relevance of the proposed simulations can be demonstrated from several angles. From a physical standpoint, they will provide data of unprecedented accuracy for the flow around the hull. This will allow examining the flow structures in detail and provide extremely valuable reference data for driving the development of lower-fidelity models. Since the latter will remain to be the workhorse of industrial simulations in the foreseeable future, the potential impact on both the scientific and engineering community is high. The case will also provide an excellent testing ground for methodological devel-

opments. Particularly, wall model development, immersed boundary techniques, adaptive mesh refinement, and uncertainty quantification.

7.2 Linked code and exascale computing

7.2.1 Introduction of the code

A detailed introduction to Neko is given in Section 4.2.1.

7.2.2 Current code status and steps towards exascale readiness

The status of Neko is described in Section 4.2.2. In summary, Neko has been tested on a full pre-exascale GPU machine (LUMI-G) and demonstrated excellent scaling. Therefore, Neko is well prepared for the upcoming exascale systems and LHC6.

7.2.3 Synergies within CEEC

The following synergies will be leveraged to achieve the common goals of CEEC with greater efficiency, which will benefit Neko and LHC6: Through the collaborative efforts in WP3 – 'exascale algorithms', we could benefit from mixed-precision in preconditioners. This concept will also be propagated to the underlying Krylov-type solvers. Since the codes Neko, Nek5000, and NekRS have the same roots and therefore, share many similarities, the efforts in WP3 could potentially be shared by these codes. Moreover, these similarities can be leveraged for further developments, including strategies for performance measurements and efficient portability to different computing hardware. We expect the most productive collaboration among project partners to take place within WP4 — 'exascale techniques', i.e., in the development of wall models and common approaches for efficiently conducting machine-learning-enabled simulations at scale, including fully automatic uncertainty quantification simulations on HPC systems. For example, the AI-driven part could potentially be shared by Neko, Alya and FLEXI, while the latter could be shared by Neko and FLEXI.

7.3 Computational requirements for the LHC

The resolution requirements for the turbulent boundary layers formed on the hull are driven by the sizes of turbulent structures found therein. We focus here on the wall-resolved simulations since their size will be orders of magnitude larger than the wall-modelled ones. The mesh sizes are based on the structures in the inner layer, and the requirements are well understood. Based on the literature, for a moderately wall-resolved LES of the JBC, the number of DoFs is in the order of tens of billions. Depending on available resources, we may want to aim higher than that. To estimate the computational requirements, we consider 15 elements per CPU core as the minimum, after which parallel efficiency becomes too low. For 10^{10} unknowns and polynomial order 7, the number of elements is $\approx 2 \cdot 10^7$. Dividing by 15 gives the estimate of $1.3 \cdot 10^6$ cores. To estimate the number of GPUs, we relate to existing data for Neko that shows that a single GPU roughly corresponds to a single CPU node with 128 cores (on e.g. LUMI). This gives $\approx 10^4$ GPUs. Note that Neko has been shown to scale with nearly ideal parallel efficiency on up to 16 386 GPUs on LUMI.

DoFs	CPUs	GPUs	Output size
$\sim$ 10B	$\sim 10^6$	$\sim 10^4$	$\leq$ 100 TB

Table 7: Estimation of the required hardware resources for Neko and the disc space used for the whole simulation.

Memory size is not expected to be an issue, as the memory of each compute node is very high on modern machines. For I/O, the main issue is the disk space, since each 3D data dump can be in the order of TB in size. Thus, hundreds of TB may be required for a comfortable simulation environment. On the other hand, the number of files per simulation is expected to be low enough to not pose any specific requirements worth mentioning. A summary of the required resources is provided in Table 7.

8 Summary

This deliverable describes the requirements of the six lighthouse cases (LHCs) being considered by the Center of Excellence in Exascale CFD (CEEC). Using these six lighthouse cases, the CEEC project aims to develop CFD frameworks that efficiently exploit future exascale systems.

After introducing the LHCs and the CFD codes used to compute them, the computational requirements for the LHCs were given. This includes preliminary estimates for each LHC of the target problem size in degrees of freedom, the required memory, the target HPC architecture including the required number of CPUs and/or GPUs, the I/O requirements, and the exascale maturity status. Synergies between the codes involved and the developments in the technical work packages were also discussed.

The next steps within the work package one will address common benchmarking and concepts for self- and cross-validation for the LHCs.

Bibliography

[1] M Blind, P Kopper, D Kempf, M Kurz, A Schwarz, A Beck, and CD Munz. Performance improvements for large scale simulations using the discontinuous galerkin framework flexi. *High Performance Computing in Science and Engineering*, 22, 2022.

[2] Paul Fischer, Misun Min, Thilina Rathnayake, Som Dutta, Tzanio Kolev, Veselin Dobrev, Jean-Sylvain Camier, Martin Kronbichler, Tim Warburton, Kasia Świrydowicz, et al. Scalability of high-performance pde solvers. *The International Journal of High Performance Computing Applications*, 34(5):562–586, 2020.

[3] Konrad A Goc, Oriol Lehmkuhl, George Ilhwan Park, Sanjeeb T Bose, and Parviz Moin. Large eddy simulation of aircraft at affordable cost: a milestone in computational fluid dynamics. *Flow*, 1:E14, 2021.

[4] Nico Krais, Andrea Beck, Thomas Bolemann, Hannes Frank, David Flad, Gregor Gassner, Florian Hindenlang, Malte Hoffmann, Thomas Kuhn, Matthias Sonntag, and Claus Dieter Munz. FLEXI: A high order discontinuous Galerkin framework for hyperbolic–parabolic conservation laws. *Computers and Mathematics with Applications*, 81:186–219, 2021.

[5] M. Liefvendahl and M. Johansson. Wall-modeled les for ship hydrodynamics in model scale. *Journal of Ship Research*, 65:41–54, 2021.

[6] M. Min, M. Brazell, A. Tomboulides, M. Churchfield, P. Fischer, and M. Sprague. Towards exascale for wind energy simulations. *submitted The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC22)*, 2022.

[7] A. S. Monin and A.M. Obukhov. Basic laws of turbulent mixing in the surface layer of the atmosphere. *Tr. Akad. Nauk. SSSR Geophiz. Inst.*, 24(151):163–187, 1954.

[8] Saumil Patel, Paul Fischer, Misun Min, and Ananias Tomboulides. A Characteristic-Based Spectral Element Method for Moving-Domain Problems. *Journal of Scientific Computing*, 79(1):564–592, 2019.

[9] P. Sullivan, J. McWilliams, and C. Moeng. A subgrid-scale model for large-eddy simulation of planetary boundary-layer flows. *Bound.-Layer Meteor.*, 71:247–276, 1994.

[10] Peter P Sullivan and Edward G Patton. The effect of mesh resolution on convective boundary layer statistics and structures generated by large-eddy simulation. *Journal of the Atmospheric Sciences*, 68(10):2395–2415, 2011.

[11] H.M. Tufo and P.F. Fischer. Fast parallel direct solvers for coarse grid problems. *J. Parallel Distributed Comput.*, 61:151–177, 2001.