H2020 FETHPC-1-2014

Enabling Exascale Fluid Dynamics Simulations
Project Number 671571

D3.2 - Use Case Perspective of Algorithmic Developments

WP3: Validation & Case Studies

DOCUMENT INFORMATION

Deliverable Number	D3.2
Deliverable Name	Use Case Perspective of Algorithmic Developments
Due Date	31/03/2017 (PM 18)
Deliverable lead	University of Southampton
Authors	Christian T. Jacobs (University of Southampton) Neil D. Sandham (University of Southampton) Nico De Tullio (University of Southampton) Adam Peplinski (KTH) Ricardo Vinuesa (KTH) Nicolas Offermans (KTH) Philipp Schlatter (KTH) Martin Vymazal (Imperial College London) David Moxey (Imperial College London) Spencer J. Sherwin (Imperial College London) C. J. Falconi D. (ASCS) M. Woo (ASCS) D. Lautenschlager (Adam Opel AG) J. F. A. Hoessler (McLaren) J.-E. W. Lombard (McLaren/Imperial)
Responsible Author	Christian T. Jacobs (University of Southampton) e-mail: `C.T.Jacobs@soton.ac.uk`
Keywords	simulations, use cases, requirements, performance, evaluation
WP	WP3
Nature	R
Dissemination Level	PU
Final Version Date	31/03/2017
Reviewed by	Niclas Jansson (KTH), Erwin Laure (KTH)
MGT Board Approval	31/03/2017

DOCUMENT HISTORY

Partner	**Date**	**Comment**	**Version**
SOTON	18/01/2017	Initial skeleton document started	0.1.0
SOTON	18/01/2017	Contributions from SOTON added	0.1.1
KTH	23/02/2017	Contributions from KTH added	0.1.2
Imperial	23/02/2017	Contributions from Imperial added	0.1.3
ASCS	24/02/2017	Contributions from ASCS added	0.1.4
McLaren	02/03/2017	Contributions from McLaren added	0.1.5
SOTON	03/03/2017	Initial draft ready for internal review	0.2
SOTON	23/03/2017	Revisions based on internal review submitted	0.3

Executive Summary

This document contains an initial assessment of the ExaFLOW algorithms formulated in WP1 and implemented in WP2, in the context of the five use cases that were defined as part of Deliverable 3.1. The use cases feature simulations of a jet in crossflow, both incompressible and compressible flow past a NACA4412 airfoil, an automotive simulation, and flow past various aspects of a race car such as the Imperial Front Wing design. The specific algorithmic developments from WP1/WP2 that have been used in each use case are given, along with a discussion on the effects that these algorithmic improvements have had on the use case with respect to the initial results presented in Deliverable 3.1. The report also provides feedback to WP1 and WP2 in light of these effects, and a plan for the next 6–12 months is formulated.

The key findings from the use cases are as follows. For the NACA-4412 compressible use case, preliminary studies have shown that the application of the error indicators developed in WP1/WP2 has helped to uncover where more grid resolution is required in the flow simulation, namely around the wake and developing boundary layers. For the jet in crossflow and incompressible NACA-4412 use cases, it has been shown that the AMG solver for preconditioning the pressure equation is significantly faster than XXT for large cases; the AMG setup using the Hypre library is more than one order of magnitude faster than the Matlab version; and the setup should be parallelized and incorporated directly into Nek5000. For the automotive use case, there is a need to increase accuracy and be able to select a bigger domain size. Finally, for the Imperial Front Wing use case, initial scaling results with the new algorithmic developers show good speed-up for small-scale runs.

Contents

1 Introduction

Work package (WP) 3 concentrates on analysing the efficiency of the methods, formulated in WP1 and implemented in WP,2 through a set of synthetic and real-world benchmarks/use cases. The use cases have been chosen according to the requirements from our industrial partners to ensure that the project will have impact both in academia and industry. This document comprises an initial evaluation of the following five such use cases:

- Jet in crossflow (KTH Royal Institute of Technology)
- NACA4412 airfoil in incompressible flow (KTH Royal Institute of Technology and University of Stuttgart)
- NACA4412 airfoil in compressible flow (University of Southampton)
- Automotive use case (Automotive Simulation Center Stuttgart and Adam Opel AG)
- Imperial Front Wing and related cases (Imperial College London and McLaren Racing Limited)

Preliminary results are presented which help to highlight the benefits of the new algorithms developed. Each section also states the feedback that will be given to WP1 and WP2 in light of the initial evaluation, and an outlook for each use case and the applied algorithms for the next 6–12 months is provided. It is worth noting that, while the use cases are intended to fully exercise the various codes and expose known issues relating to exascale computing, the results presented here are only expected to provide a preliminary assessment of the new algorithms and are therefore not large-scale 'flagship' calculations which push the limits of the algorithms and available computing resources; this will come later at PM36 as part of Deliverable 3.3.

2 NACA-4412 airfoil in compressible flow

The error indicators formulated as part of Task 1.2 of WP1 (see Deliverable 1.2 for details) have been successfully applied to the simulation of compressible flow past an airfoil. Note that for the initial assessment of the error indicator algorithms, results from a NACA-0012 wing profile were considered instead of the NACA-4412 profile originally mentioned in the Use Case document.

The airfoil incidence α was set to 5°, the Reynolds number based on the aerofoil chord Re_c was 5×10^4, and Mach number M was 0.4. The computational domain is composed of three blocks, as can be seen in Figure 1(a). Block 2 is a C-type structured grid fitted around the airfoil surface; it interfaces with the structured blocks 1 and 3, which resolve the wake of the airfoil. Since block 1 and block 3 both contain the wake line, the wake line solution at each time step is obtained by averaging between the solutions obtained in the two blocks. This is necessary because flow asymmetries near the airfoil trailing edge and/or

small differences in initial conditions will cause the wake line solutions in the two blocks to diverge. Table 1 gives the dimensions and number of grid points used.

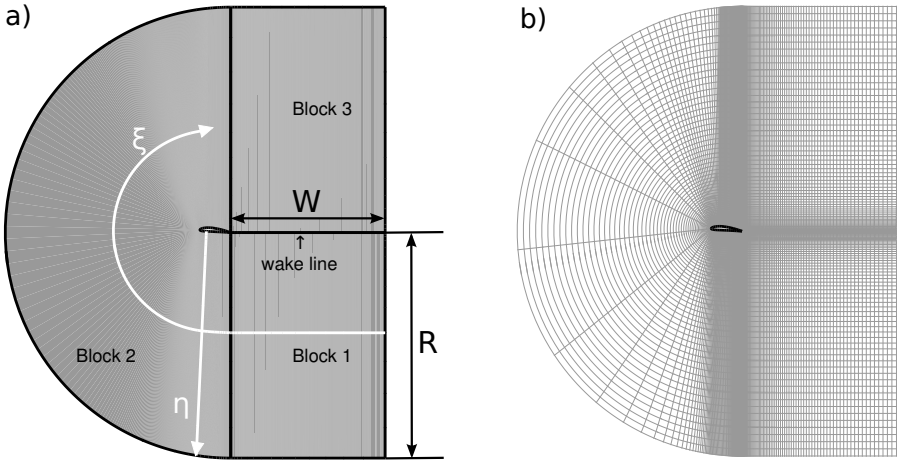

Figure 1: Computational domain arrangement. (a) multi-block domain set up, (b) computational grid, showing only one in every 10 grid points.

R/c	W/c	S/c	N_{foil}	N_{wake}	N_{ξ}	N_{η}	N_z
7.3	5.0	0.4	1799	1602	3401	692	240

Table 1: Numerical grid details. Note that N_{foil} and N_{wake} are the number of grid points around the aerofoil and along the wake line in the ξ direction.

The simulation was performed using the legacy Fortran-based SBLI code. The error indicators were implemented in a Python program, which read in the solution fields stored as a binary file and computed the error severity values at each error block, with each block comprising 16^3 grid points. The z-component of vorticity was used to compute the error severity.

2.1 Initial evaluation

The error severity values for the integer-based measure I_i are shown in Figure 2, and were computed at time-step 500,000 once the flow became fully developed. The uniform flow away from the aerofoil is well-resolved as suggested by I_i values of 0 or 1. However, a significant number of $I_i = 2$ values can be seen along the boundary layers either side of the aerofoil. Furthermore, a cluster of high values are present near the trailing edge where eddy shedding occurs. This suggests that more resolution would be required in these areas to adequately resolve the wake.

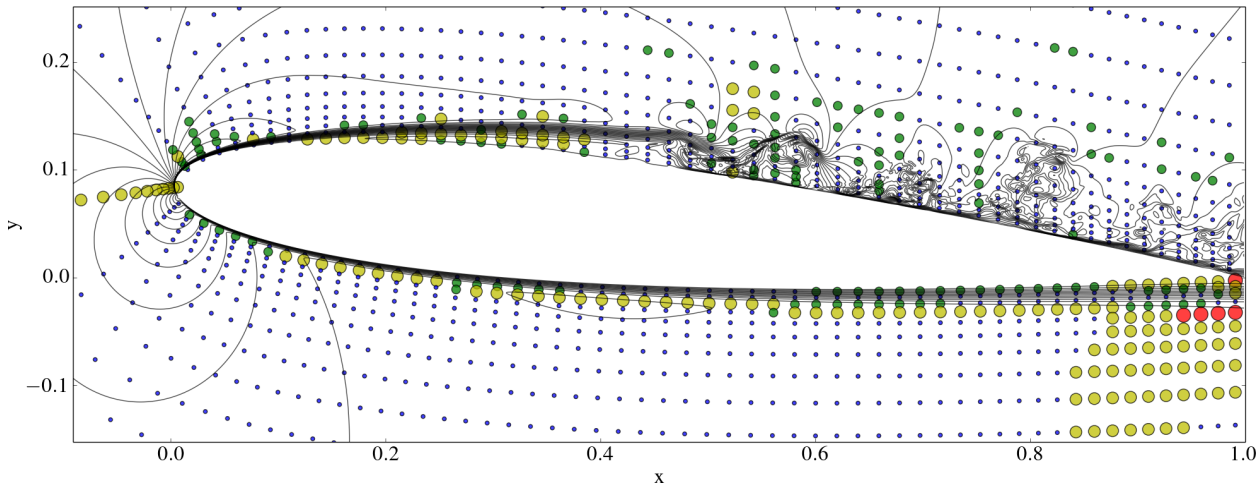

Figure 2: Contours of velocity magnitude in a two-dimensional slice (in the x–y plane at $z = 0$) from a NACA-0012 aerofoil simulation. Circles filled with blue, green, yellow and red indicate I_i error severity values of 0, 1, 2 and 3, respectively.

With respect to the goals of ExaFLOW, the error indicators have suggested where more resolution needs to be placed and also where the grid may be coarsened, thereby facilitating the highly efficient simulation of large-scale industry-relevant problems.

2.2 Feedback provided to WP1/WP2

One major limitation is currently the run-time of the error indicator algorithm. For the Taylor-Green vortex test case discussed in Deliverable 1.2 the run-time more than doubled for the 64^3 grid case. Further work is therefore required in WP1/WP2 to improve the efficiency of the implementation.

The simulation presented here was performed with the legacy Fortran-based SBLI code, since several features needed to perform such a simulation in the new OpenSBLI code [7] are not yet available, such as generalised coordinates and characteristic boundary conditions. Work to implement these features in OpenSBLI as part of WP2 is on-going.

Finally, while the error indicators have been implemented within the OpenSBLI automated model development framework, this implementation has so-far been done 'by hand' (i.e. writing the C code that computes the error indicators without any help from OpenSBLI's code generator and manually calling it from the model's automatically-generated C kernels). Further work in WP2 is necessary to *automatically* produce the C code that computes these indicators.

2.3 Outlook and future work

The application of the error indicators has helped to uncover where more grid resolution is required in the flow simulation. Therefore, over the next 6 months, we will produce a refined

grid, re-run the simulation and re-apply our error indicators to investigate to what extent the error severity has decreased. We also aim to present this work at the ParCFD 2017 conference; an abstract has been submitted and accepted for presentation [8].

3 Jet in cross-flow and incompressible NACA-4412 airfoil

This section describes the evaluation of the AMG setup for Nek5000 using Hypre for the jet in cross-flow and incompressible NACA-4412 airfoil use cases.

3.1 Initial evaluation: Timing results when using the Hypre setup for the AMG solver

As already described in WP2, the preconditioner for the pressure equation includes the resolution of a coarse grid problem on the elements' vertices, which can be solved using an algebraic multigrid (AMG) method or using a direct solver called XXT. The use of AMG first requires a setup step performed by an external Matlab code. This code is now replaced by a faster one, written in C and using the Hypre library for linear algebra. In this section, we compare the timing results for the three possible ways to solve the coarse grid solver: AMG with Matlab setup, AMG with Hypre setup and XXT. The computational times for both AMG setups and the total time per timestep during a simulation are presented for several test cases: the turbulent flow in a pipe at $Re_\tau = 550$, the jet in crossflow and the NACA4412 airfoil.

3.1.1 Test cases

As a brief reminder, the name, number of elements and polynomial order for each case is summarized in table 2. The turbulent flow in a straight pipe has been used previously for scaling tests and is part of the jet in crossflow. The two other cases are part of the reference test cases within ExaFLOW.

Case name	Number of el.	Pol. order
Pipe $Re_\tau = 550$	853632	7
Jet in crossflow	47960	7
Wing DNS	1847664	11

Table 2: Summary of the test cases

3.1.2 AMG setup

The total time to perform the setup when using Hypre and Matlab is compared for all test cases. The algorithms for the Matlab setup have been presented as part of WP2. The coarsening method for the Hypre setup is the Falgout-CLJP one. Therefore, both setups produce a different coarsening and different interpolation operators. However, the algorithm and the tolerance for computing the smoother are the same for both setups. The total setup time t_{tot} is split in 5 parts:

- Time to read the binary files written out by *Nek5000* and setup the matrix A_0: t_{ini},
- Time to perform coarsening: t_{crs},
- Time to perform interpolation: t_{int},
- Time to compute the smoother: t_{smo},
- Time to export the data: t_{exp},
- Rest: t_{res}.

When using the Hypre setup, we do not have access to separate timings for coarsening and interpolation and the combined time for both parts is therefore presented. All setups are performed on the same local machine (CPU: Intel Core I7 990 Extreme, RAM: 24gb) in serial and timings are shown in table 3.

Matlab	Pipe 550	Jet	Wing DNS
t_{tot}	2231.04	98.64	3755.92
t_{ini}	6.03	0.34	14.65
t_{crs}	123.11	6.68	657.17
t_{int}	2044.2	89.22	2938.28
t_{smo}	37.2	1.48	65.21
t_{exp}	20.46	0.83	79.33
t_{res}	0.02	0.09	1.28

Hypre	Pipe 550	Jet	Wing DNS
t_{tot}	77.15	2.42	182.1
t_{ini}	2.38	0.14	5
t_{crs}+ t_{int}	24.54	1.3	56.4
t_{smo}	19.85	0.72	34.9
t_{exp}	30.37	0.26	85.4
t_{res}	0.01	0	0.4

Table 3: Setup times (in seconds)

For all three cases, the total setup time has been reduced by more than one order of magnitude. The decrease is most significant for the coarsening and interpolation steps, which are almost two orders of magnitude faster. Timing for the smoother is also reduced by half, which is a priori unexpected, given the fact that they are the same. The reason behind this is still unknown but might be due to a better implementation in C compared to Matlab. Reading the matrix data is also faster with the C code than with Matlab. Exporting the data, on the other hand, is performed in the same way for both setups and leads therefore to roughly similar timings.

The significant gain in setup time will hopefully foster users to use AMG for large cases instead of XXT, which is sometimes the first choice because of the bottleneck induced by the setup despite an expected higher total computational time.

3.1.3 Execution time

Despite the reduction in setup time, the use of Hypre for the AMG setup can be validated only if it yields to similar results in terms of total computational time in the case of real, massively parallel simulations. As the AMG solver used for the coarse grid problem is the same irrespectively of the setup method, it is expected that differences are small. We verify this by running some simulations of the aforementioned test cases for both setups. We also compare the results with XXT. The simulations have been performed on Beskow, the Cray XC40 supercomputer from the PDC centre for high performance computing at KTH, Stockholm. For each case, the timings reported are the average wall-clock times per timestep. Input and output are ignored. The average is obtained over 20 timesteps and each simulation is performed once.

Pipe $Re_\tau = 550$ The computation, communication and total times for simulations of the pipe $Re_\tau = 550$ as a function of the number of cores are plotted in figure 3. This case is large enough for AMG to take advantage over XXT ($> 100,000$ elements and $> 10,000$ cores). Indeed, it is observed that all AMG setups lead to roughly the same computation time, while XXT is systematically slightly higher. On the other hand, communication time using XXT is globaly lower than for the AMG setups for large number of cores. The total time for XXT is always greater than for the AMG, by a few percents.

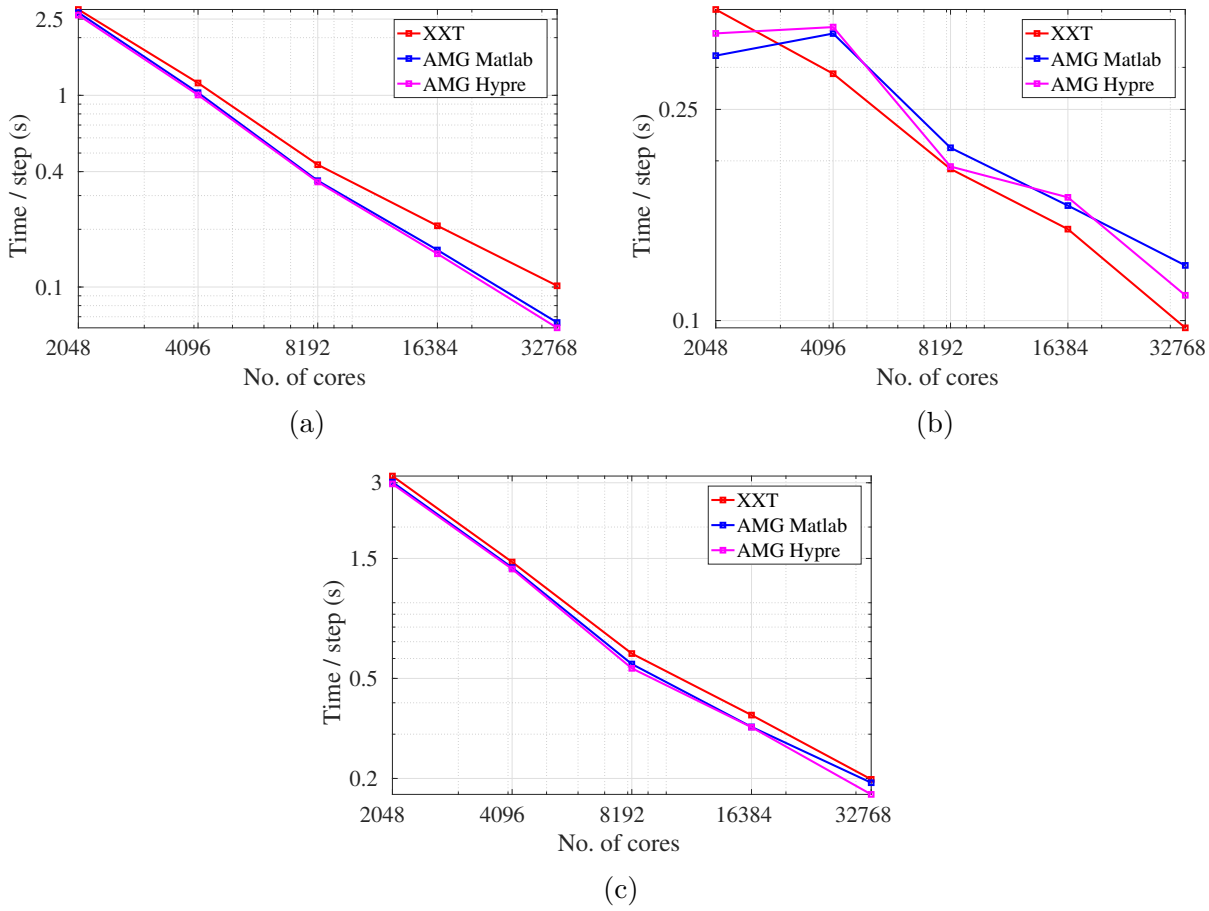

Figure 3: Timings for the pipe $Re_\tau = 550$. (a) Computation; (b) Communication; (c) Total.

Jet in Crossflow The computation, communication and total times for simulations of the jet in a crossflow for several numbers of cores are presented in figure 4. Both AMG setups lead to very similar results. XXT is slower on 256 and 512 cores but performs better on 1024 cores. Given the relatively low numbers of elements ($< 100,000$) and cores ($< 10,000$), we do not observe a clear advantage of AMG over XXT. Furthermore, each case is run using a different node allocation on the computer, which makes it difficult to draw clear conclusions that are independent on the computational environment. The main conclusion is that the AMG setup using Hypre performs as well as the Matlab version.

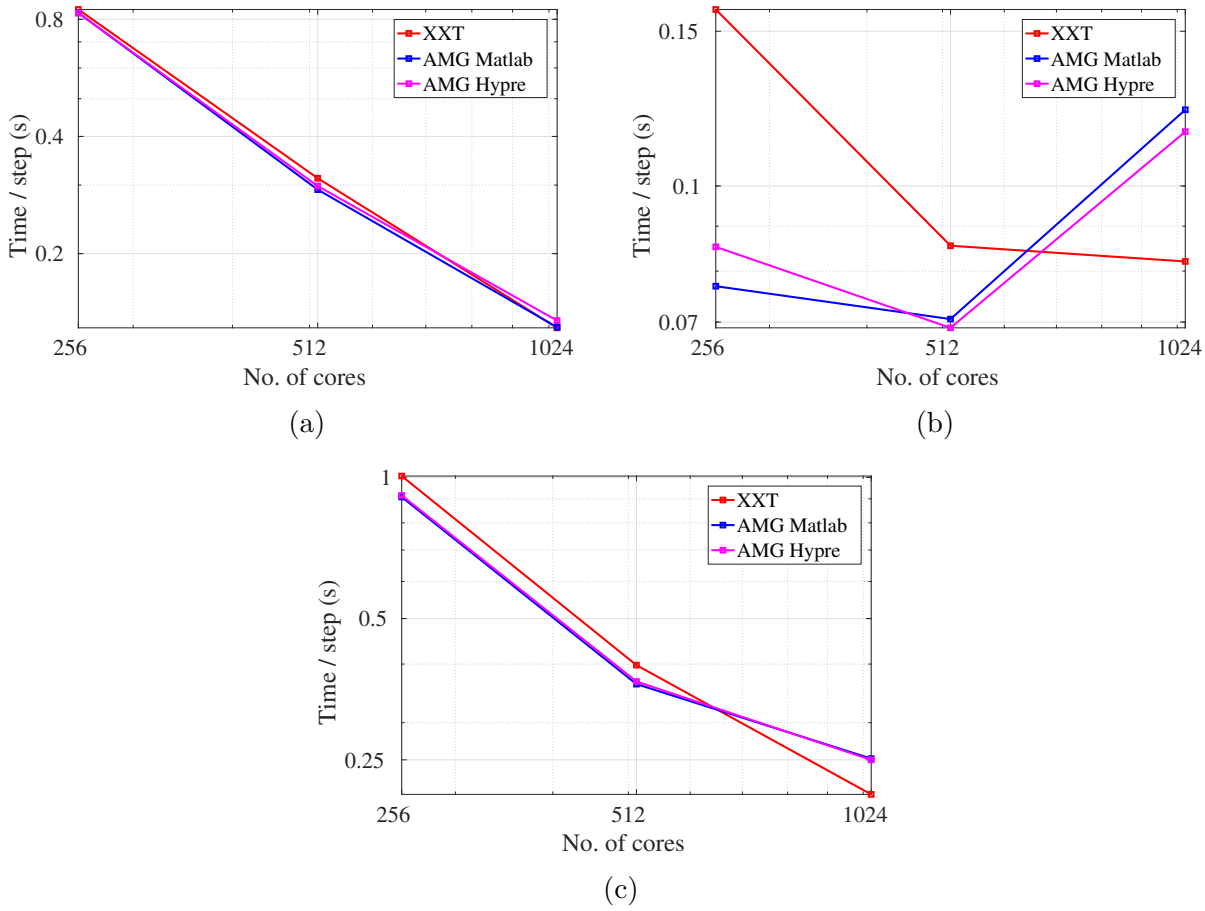

Figure 4: Timings for the jet in crossflow. (a) Computation; (b) Communication; (c) Total.

Incompressible NACA4412 airfoil The computation, communication and total times as well as the time for solving the coarse grid solver in the case of the NACA4412 airfoil are presented in figure 5. The computations have been performed on 16,384 cores only and the same node allocation has been used, making the comparison less dependent on the environment. As expected for a case that large, AMG clearly outperforms XXT. The time for solving the coarse grid solver is reduced by a factor 3, while the total computational time is reduced by about 10%.

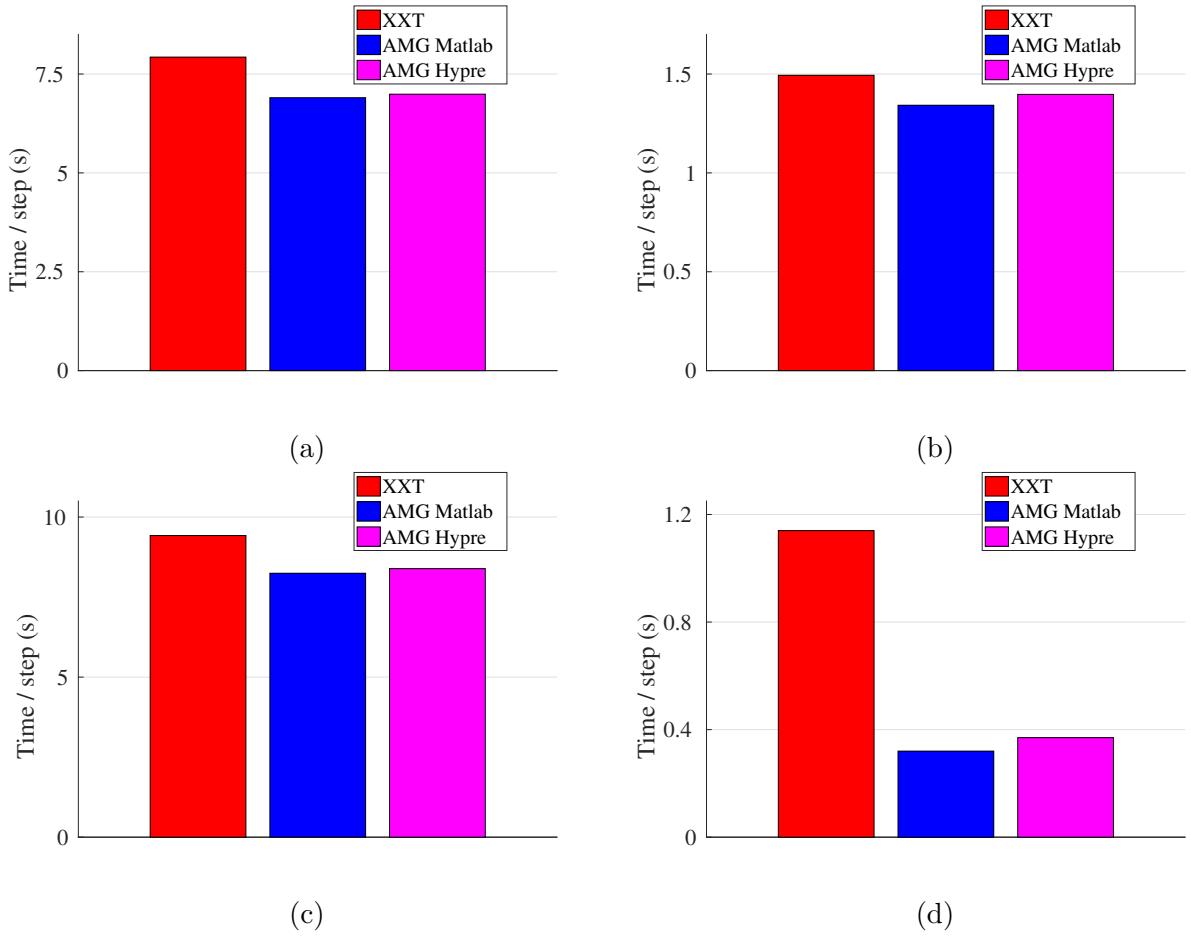

Figure 5: Timings for the NACA4412 airfoil on 16,384 cores. (a) Computation; (b) Communication; (c) Total; (d) Coarse grid solver (AMG Matlab: 26 Chebyshev iterations. AMG Hypre: 33 Chebyshev iterations).

The difference between both AMG setups may be explained by the difference in the number of Chebyshev iterations for the AMG solver. Indeed, both setups have different coarsening and interpolation algorithms, leading to different numbers of levels and different operators. With Matlab, the total number of Chebyshev iterations is 26 and it is 33 with Hypre. As a global communication operation is performed per iteration, this has a direct impact on the computational time for the coarse grid solver. Let us mention that the number of levels and the number of Chebyshev iterations could be indirectly changed by tuning the tolerances for the setup.

3.2 Feedback provided to WP1/WP2

The simulations of large test cases using Nek5000 show that the use of the algebraic multigrid (AMG) solver as a preconditioner should be favored over the direct solver XXT. In the particular case of the NACA4412 wing, run on 16384 cores, the gain in total computational

time reached approximately 10%. In order to facilitate the use of AMG, the setup phase, currently performed by an external serial code using the Hypre library, should be parallelized and incorporated directly in Nek5000.

3.3 Outlook and future work

The desciption of the algorithm for the AMG setup and its implementation using Hypre haved been presented as part of WP2. In this section, we have shown that the setup time is reduced by at least one order of magnitude, while the computational times are similar. This new setup facilitates the use of AMG for large cases, which allows a significant reduction in computational resources. Indeed, based on simulations of the NACA4412 airfoil, it has been shown that the use of AMG is preferred over XXT for large cases. The decrease in total computational time reaches approximately 10% for this case. For small cases however, no clear advantage of AMG over XXT has been observed.

In the future, large simulations ($N > 1 \times 10^5$) on massively parallel computers ($P > 1 \times 10^4$) will be preferably performed using AMG.

4 Automotive use case

The simulation of external aerodynamics surrounding a vehicle is essential for the development of an efficient car body design. The Reynold Averaged Navier-Stokes (RANS) approach is a widely used method for this purpose. However, the RANS predicts only the mean flow field which gives rise to difficulties to analyze the time-dependent phenomena such as vortex shedding and flow separation. A promising solution to overcome this limitation is the Detached Eddy Simulation (DES) approach which utilizes both the RANS turbulent model for the boundary layer and the Large Eddy Simulation (LES) in separated region to capture the fully three-dimensional turbulent movement [4]. Direct Numerical Simulation (DNS) provides even more accurate solutions with resolving all scales with a high number of mesh cells and requires therefore much higher computational efforts as compared to the DES calculation. The computation efficiency is a key parameter for effective use of such massive computation methods.

This section describes the state-of the art modelling of a commercial software (Fluent) and the first development implemented in the ExaFLOW open source code (nektar++ [1]) with the automotive use case defined in [4]. The computations have been performed on the supercomputer Hazel Hen (position 8 of TOP500, 11/2015), a Cray XC40-system in HLRS [5].

The simulation focuses only on the rear wake of a sporty vehicle so that the half-left part of rear vehicle region namely submodel is only taken into account. By means of the submodel, the computational performances of Fluent (DES) and nektar++ (Continuous Galerkin method) have been compared with different number of processors. Based on the results, the relationship between the computational time and costs is presented, which comes up with the optimal range of the number of cores for each computation method.

4.1 Problem definition

A submodel is defined to explore the turbulent characteristic at the rear upper region of the vehicle. Due to the lateral symmetric of the car body, only the left-half part is chosen for computation. Figure 6 illustrates the computational domain and boundary conditions for the submodel. The size of the domain is 7m, 1.3m, and 0.15m in the x-, y- and z-direction, respectively. The RANS simulation of the complete vehicle [4] was performed with ANSYS.

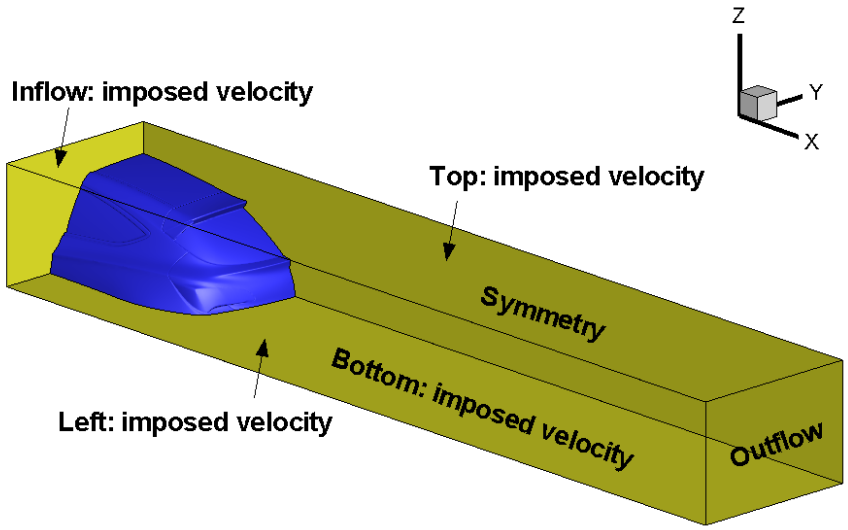

Figure 6: Computational domain of submodel and its boundary conditions.

Fluent version 14.0 with a Reynolds number of 6.3×10^6 based on the vehicle speed of 140 km/h and 2.695 m wheel base. Since the submodel is generated from the computational domain for the RANS simulation, the velocity fields outside from submodel already exist as a result of RANS calculation. Therefore, the velocity at the boundary is imposed from the RANS results except for symmetric plane and outflow region. Vehicle surface is set as a no-slip boundary condition.

4.1.1 Computational meshes

For generation of the computational meshes, the 3D computational domain is discretized by volumetric cells projected from a 2D surface mesh. For DES simulation, the identical mesh used for the RANS simulation (ANSYS Fluent) has been employed. The typical length of the elements of the two-dimensional surface mesh is between 1-6 mm. 7 prism layers with a growing cell size are defined on the exterior surface of the vehicle to take into account the sharp velocity gradient in the boundary layer. A non-equilibrium wall function is utilized for the first cell near the wall. The 3D volumetric mesh is composed of a non-conformal

Hexcore mesh (Hexaeder and Tetraeder) [4]. The total number of cells is 20,424,186 for DES calculation.

Since the mesh used for DES simulation (finite-volume method) is not compatible with nektar++ (finite-element method), a new mesh is required for the simulation of the submodel wih nektar++.

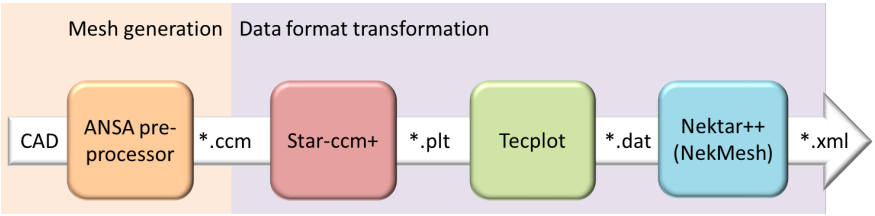

Figure 7: Pre-processing procedure of nektar++ for submodel computation.

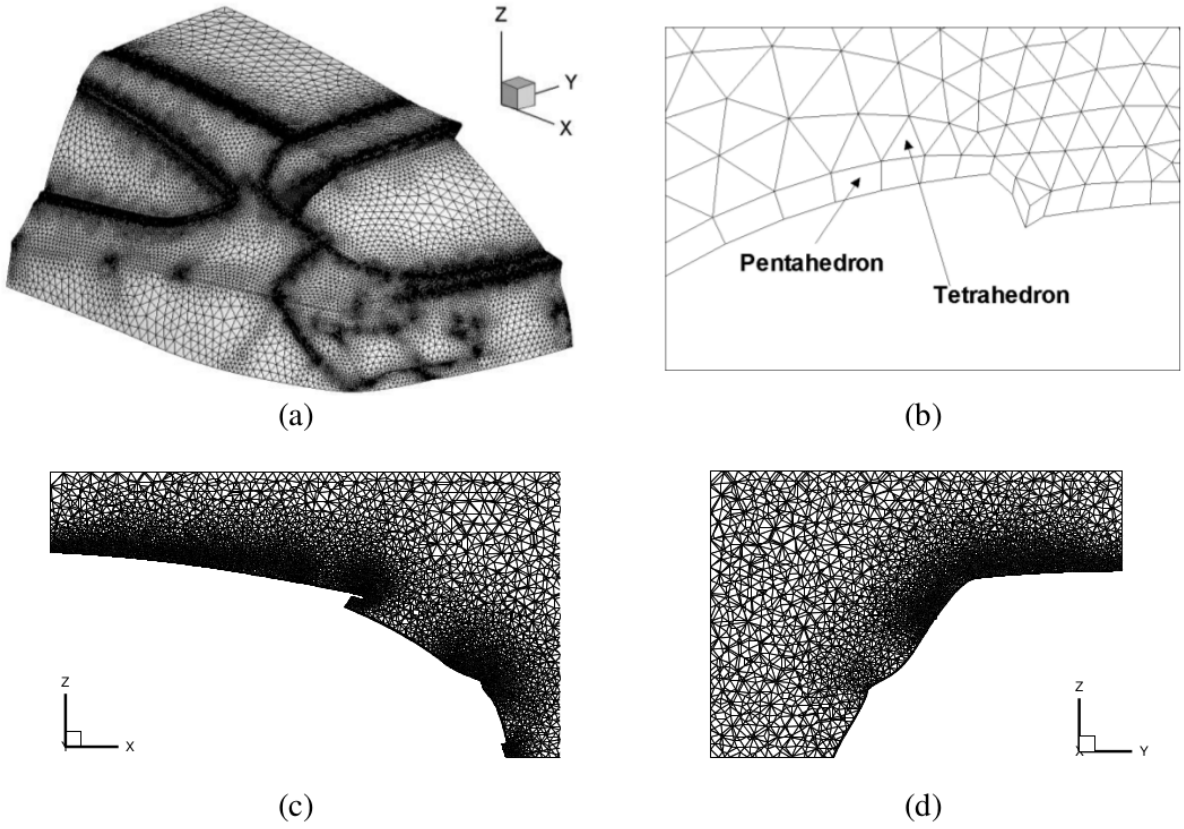

Figure 8: Computational mesh cells for submodel. Surface mesh on the vehicle (a), zoom-in near the vehicle surface (b), volume mesh on xz plane (c) and yz plane (d).

Figure 7 explains the pre-processing procedure for nektar++. ANSA pre-processor is used for the mesh generation. Once the mesh is created by ANSA pre-processor, the mesh is converted to xml format for nektar++. In the data converting procedure, additional software such as Star-ccm+, Tecplot and the nektar++ tool 'NekMesh' is utilized.

According to the different computational method, the mesh for nektar++ (finite-element method) is different from the mesh for DES (finite-volume method). In nektar++, within each cell a polynomial basis function is defined, which is a feature of spectral/hp method implemented in the code [1]. Increasing the expansion factor of the polynomial the mesh resolution can be refined flexibly without regeneration of a new mesh. Figure 8 shows the computational mesh of nektar++ for the submodel domain.

As for the DES mesh, the 2D surface mesh (Figure 8 (a)) is firstly generated. The size of two-dimensional mesh cells is in the range between 3mm and 40mm. Pentahedrons are generated in boundary layer region by an extrusion of the surface mesh Figure 8 (b). The biggest surface mesh size is bigger than that of DES mesh using the polynomial expansion factor of 3 for velocity. Therefore, only one boundary layer cell is applied to the mesh in the boundary layer region and it as shown in Figure 8 (b). The height for boundary layer mesh is set to 0.6 mm. No wall function is required.

The rest of the 3D computational domain is discretized by a tetrahedral mesh (Figure 8 (b)). In ANSA pre-processor, the growth rate, Max shell size and FLUENT_skewness are set to 1.1, 40 mm and 0.6 respectively, resulting in total number of elements of 2,266,492. Figure 8 (c) and (d) show the mesh on a certain xy and yz plane inside the subdomain.

4.1.2 Imposed boundary condition

The boundary conditions for the submodel simulation with nektar++ are extracted from the RANS results obtained with ANSYS Fluent. Figure 9 shows the data transformation process of velocity boundary condition from Fluent to nektar++. Due to the mismatch of the mesh utilized for Fluent and nektar++, the Fluent velocity fields at the boundaries are interpolated to the boundary mesh of nektar++. The Kriging method is employed for the interpolation without drift option. In the boundary condition, Tecplot plays also a crucial role for data exchange between Fluent and nektar++ (see Figure 9). After the interpolation, the nektar++ tool 'FieldConvert' transforms the boundary conditions into the nektar++ format. Figure 10, Figure 11, Figure 12 and Figure 13 display the velocity components and pressure at each boundary and compares the original field from Fluent and the interpolated field for nektar++. This comparison figures out that there is no significant difference between two velocity fields.

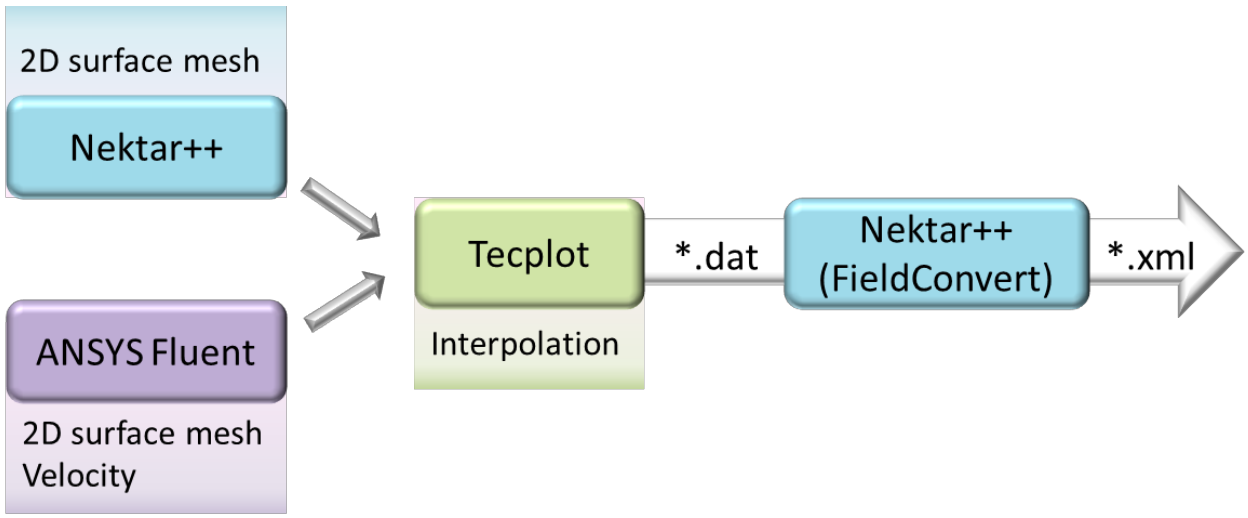

Figure 9: Data exchange process for imposing velocity boundary conditions.

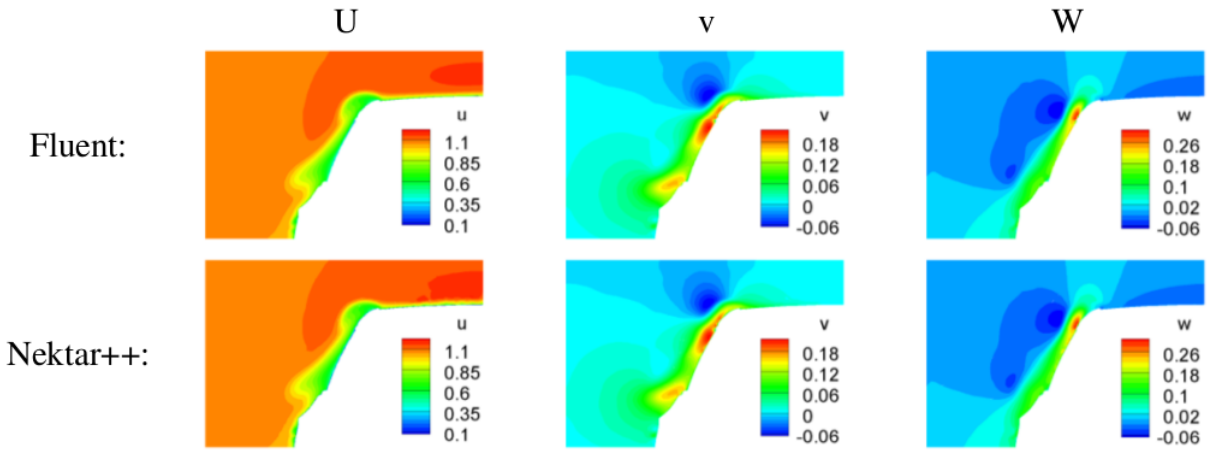

Figure 10: Imposed velocity from Fluent to nektar++ for inflow boundary condition.

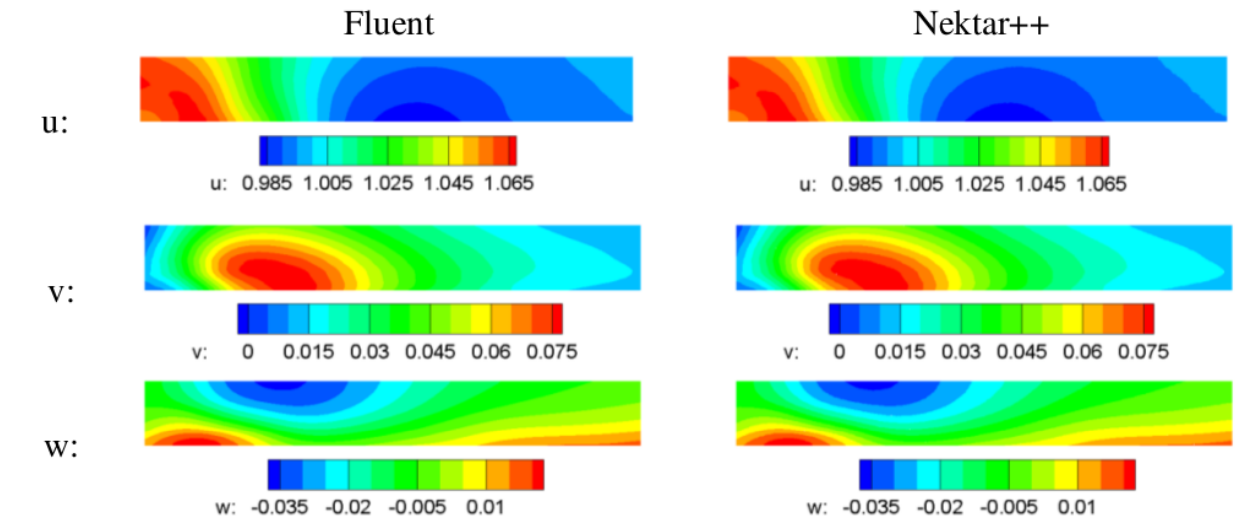

Figure 11: Imposed velocity from Fluent to nektar++ for left boundary condition.

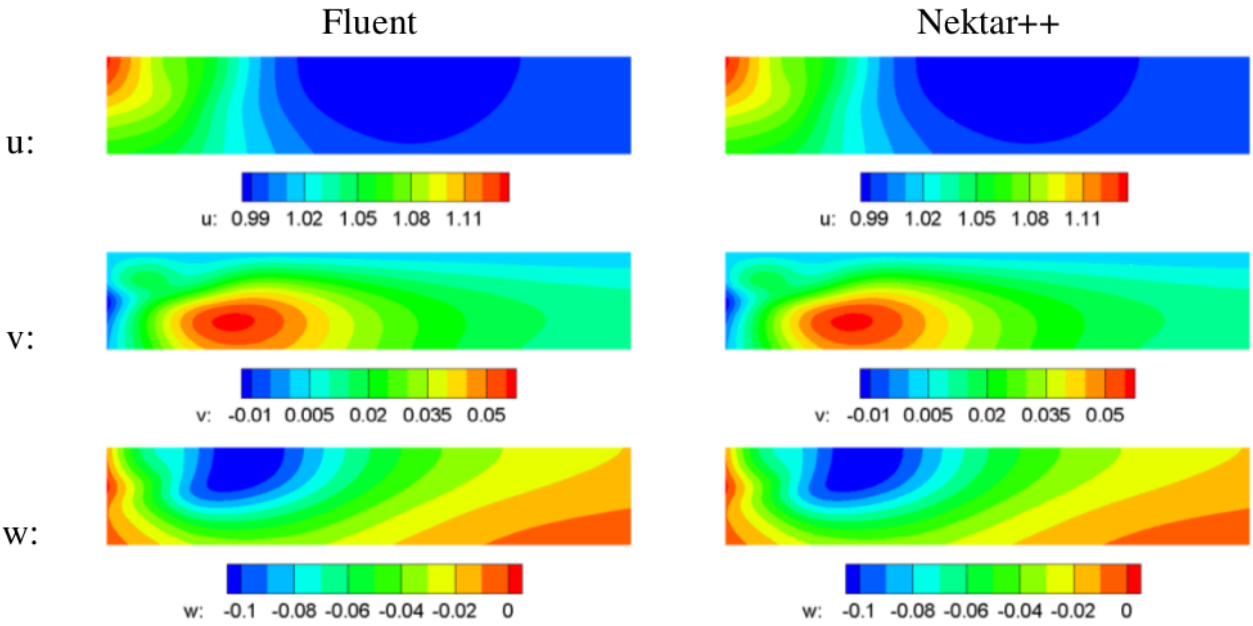

Figure 12: Imposed velocity from Fluent to nektar++ for top boundary condition.

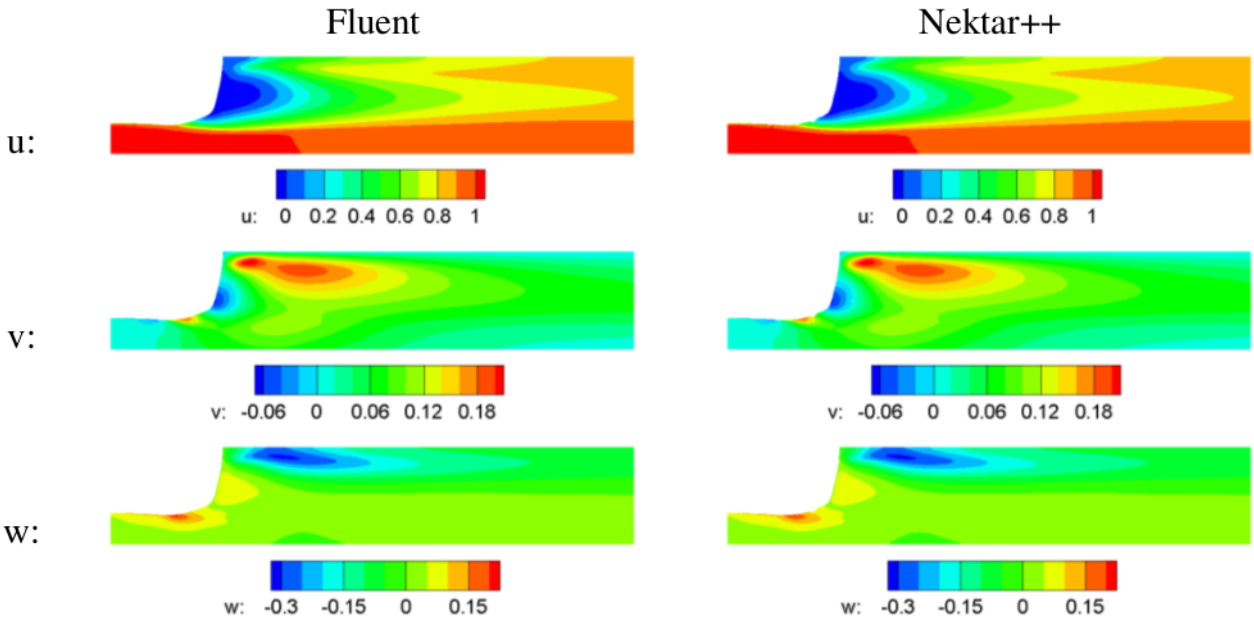

Figure 13: Imposed velocity from Fluent to nektar++ for bottom boundary condition.

4.2 Strong scalability test

As mentioned earlier, the computational methods which fully take into account the evolution of three-dimensional turbulent behaviors such as DES, LES and DNS require huge computation cost according to the temporal and spatial resolution of those approaches. Using a code with a good scalability behavior would enable more efficient computation in order to meet the fast turnaround times (<36 hours) required from the automotive industry. To find the optimal number of processors for massive computation, the computation time with respect to the number of cores namely strong scalability test has been carried out.

The strong scalability test is firstly conducted for a DES calculation of the submodel by ANSYS Fluent 17 in the HLRS facility [5]. The Reynolds number for this calculation [4] is $Re = 6.3 \times 10^6$ and the boundary conditions are shown in Figure 10–13. The time step size is set to 1×10^{-4} and it is computed till 500 time steps (0.05s) only for the scalability analysis.

The strong scalability test has been performed by nektar++ as well. For nektar++ the zero initial velocity is specified and Reynolds number is set to 1,000. The influence of Reynolds number has also been investigated with Reynolds number of 10,000. There is almost no noticeable difference between the computation times for both Reynolds numbers. In the future, the Reynolds number will be increased to reach $Re = 6.3 \times 10^6$ as used for DES simulation. The expansion factors used are 3 and 2 for velocity and pressure, respectively. The calculation runs till 5000 iteration with time step size of 1×10^{-5} so that it comes up with the same physical time 0.05s as the Fluent scalability test.

4.2.1 Real time comparison

The real time represents the waiting time for user from the job submission till the end of the job. Details about the other times are described in the next subsection, 4.2.2. Figure 14 shows the real time of both scalability tests. In Fluent calculation, the computation time shows decreasing aspect by increasing number of cores till 960 cores, but does not further decrease when more than 1000 cores are used. This represents that there is a limitation to increase the number of cores for this specific test scenario. Using more than 1000 cores is not further reducing the computation time, but the computational time stays around the constant value. The average real time for the range 960-7680 cores is 1.58h. The real time values of nektar++ are much smaller than those of fluent DES simulation. As the processor number increases, the real time is decreasing till the 3840 number of cores. With more than this number of cores, the real time is increasing again. The average time of nektar++ for the range 960-7680 cores is 0.55h and is almost three times smaller than the time required for the Fluent DES simulation.

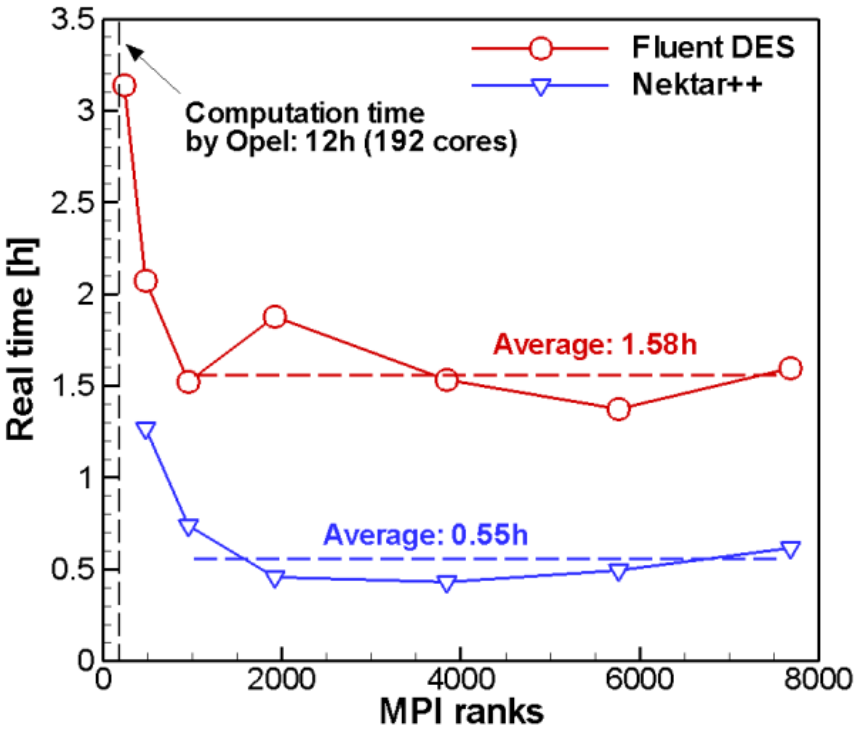

Figure 14: Comparison of real time between fluent DES and nektar++ for the physical time of 0.05s with different number of processors.

In addition, the original use case description [4] reports that the computational time for DES simulation is 8 days for 0.4s of physical time using 192 cores. The DES calculation covers whole rear part of the vehicle without symmetric condition, while the submodel used

in this report considers only the left-half of the rear vehicle region. Thus, the size of the domain in [4] is twice the domain size contemplated here. Considering that the equivalent computation time for DES (physical time= 0.05s) is approximately 12h. This industrial simulation was not performed on HLRS but on an internal cluster at GM Tech Center.

4.2.2 Solver time vs real time

As described in section 4.2.1, there is another time for the computation in nektar++ namely solver time. This solver time represents the time required for the nektar++ solver IncNavierStokesSolver to accomplish one specific number of time steps. This information is written at each output frequency specified by user and finally the sum of these times is also written as Time-integration at the end of output file. The solver time is therefore always less than the real time. The solver time for the first time step is given in Figure 15 with different number of processors. The time for 1st time step is much longer than the average time per time step. The time for first time step may contain all the time required for preparation of the calculation such as decomposition before starting the solver. Therefore, this time is excluded for the average solver time evaluation.

MPI ranks	480	960	1920	3840	5760	7680
1st time step	97s	67s	51s	42s	33s	41s
Average time per time step	0.8s	0.4s	0.21s	0.11s	0.073s	0.066s

Figure 15: The solver time required for the first time step and average time per time steps obtained from the calculation of 5000 time steps with different MPI ranks.

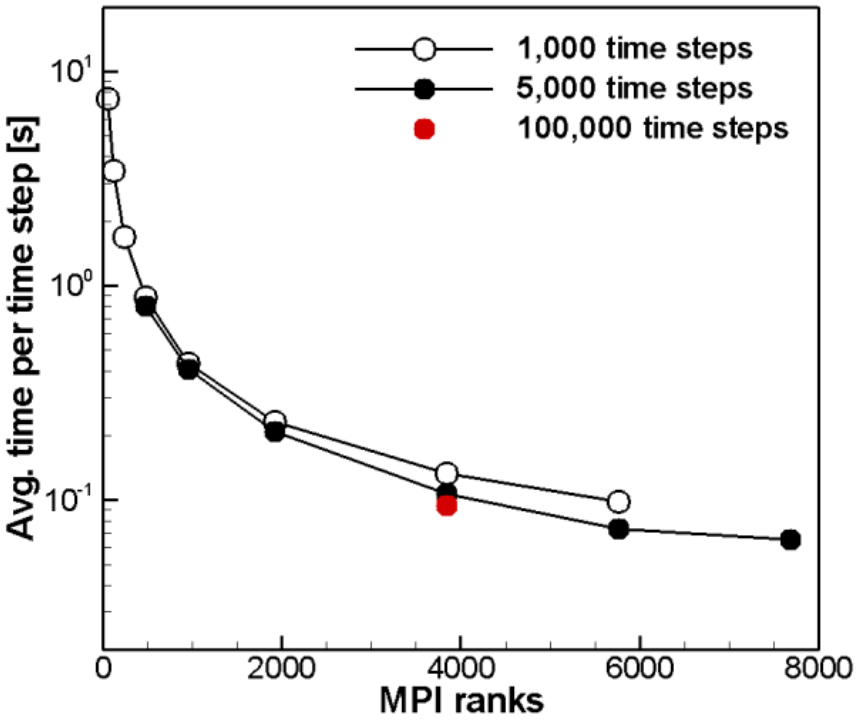

Figure 16: Average solver time per time step from three different calculations with different time steps (1,000, 5,000 and 100,000).

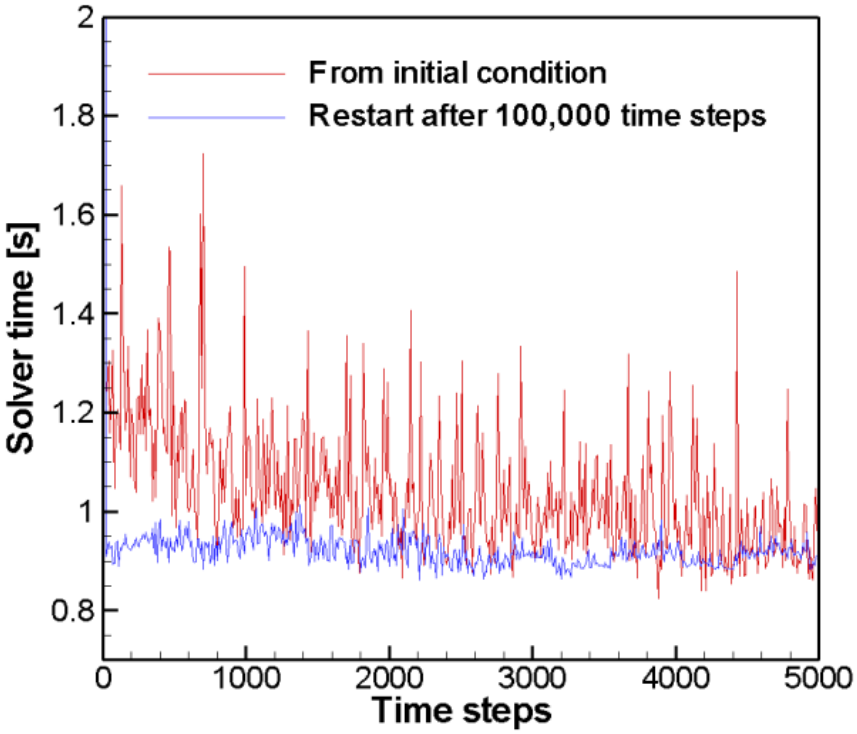

Figure 17: Variation of solver time during the calculation. Comparison between the cases with initial condition and restart after 100,000 time steps.

Figure 16 compares the average solver time per time step with respect to the number of processors. This average solver time is evaluated by dividing the Time-integration by number of time steps. As depicted in Figure 16, the average solver time is decreasing as MPI rank increases. Even if the slope of decrease is being smaller, the solver time is gradually decreasing as the number of core increases. For 5000 time steps, using more than 5760 cores requires less than 1s per time step while the time for 5760 and 7680 cores are almost similar. In order to investigate the influence of time step, three different calculations with different numbers of time steps are repeated for the scalability plot. With more time steps the average solver time is slightly being shorter. This behavior can be explained with the solver time variation during the calculation.

Figure 17 presents the solver time obtained each output frequency of the solver. The case started from initial condition (u=v=w=0) shows longer solver times and bigger oscillation of those times than the case restarted after 100,000 time steps. In the case with initial condition (u=v=w=0) the solver time is decreasing as calculation goes on, while the solver time for the case that started with the restart-File looks almost in the uniform range. It represents that the solver requires more time when the velocity fields are far different from the physical condition and the time becomes shorter as the solution is converging by iterative procedure. This corresponds to the decrease of average solver time in Figure 16 as well. Hence, the

decrease of average solver time appears with increasing time steps, but actually the solver time is independent from the number of time steps.

4.3 Initial evaluation

The usual output method of nektar++ is the xml-based format which writes separate files per each MPI rank. Recently, hdf5 method, a hierarchical method with a combined output file, is implemented to nektar++ [11]. This implementation has been accomplished during this project period. As an evaluation of the new development the scalability test for hdf5 has been also performed to compare the computational time from the original xml-based format.

4.3.1 Improvement of real time

The computation times with two different output formats (xml, hdf5) in nektar++ are compared in this section. Figure 19 shows the results of strong scalability test for nektar++ with both output formats. This plot compares the total solver time and real time with increasing the number of processors. As already shown in Figure 16, the solver time using the xml format is decreasing as MPI rank increases. Hdf5 results show also similar trends of solver time against increasing number of cores. However, the real times for both methods deviate remarkably. The real time of xml format starts increasing with more than 3840 cores so that this number of cores is optimal to minimize computation time using this format. In contrary, the real time with hdf5 format is gradually declining by increasing number of processors. The new output format shows notable reduction of computation time for high MPI ranks. The computation times depicted in Figure 19 and the reduction rate (see Equation 1 below) with hdf5 format are given in Figure 18.

$$\text{Reduction rate} = (\text{Hdf5 real time - Xml real time})/(\text{Xml real time}) \times 100 \qquad (1)$$

The real time of hdf5 format with highest MPI ranks is 64% smaller than that of xml format. However, hdf5 requires longer time where the number of CPUs is smaller than 1920. With smallest number of cores, the real time of hdf5 format is almost four times longer than the real time of xml format. Currently, the hdf5 format is therefore suitable only for use more than 3000 processors.

MPI ranks		480	960	1920	3840	5760	7680
Xml	Total solver time, h	1.12	0.567	0.290	0.147	0.102	0.091
	Real time, h	1.26	0.736	0.460	0.433	0.494	0.616
Hdf5	Total solver time, h	0.953	0.608	0.275	0.144	0.124	0.065
	Real time, h	4.16	1.277	0.594	0.292	0.264	0.221
Reduction rate of real time, %		-229	-74	29	33	47	64

Figure 18: Total solver time and real time for strong scalability test of nektar++

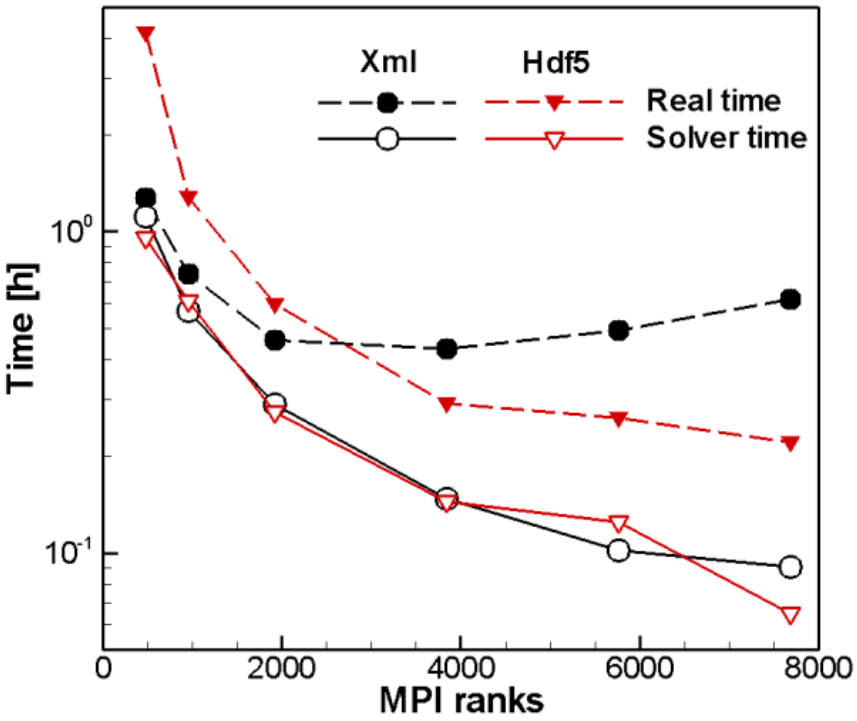

Figure 19: Total solver time and real time with respect to the number of cores. Comparison between Xml (black colored) and hdf5 (red colored) output format.

4.3.2 Energy consumption and cost analysis

Based on the computational time for different number of cores, the actual energy consumption, the required electricity cost and the cost required for the use of high performance computing (HPC) are described in this section.

Energy consumption and Electricity cost: The CrayPat performance analysis tool [3] is utilized for the energy consumption measurement. Each case of strong scalability test is repeated with this measuring tool. Hdf5 format is employed for this investigation. Figure 20 presents the energy consumption against increasing number of cores and corresponding real time and electricity cost. The energy usage is decreasing as the number of cores increases. The noticeable point is that the energy consumption is directly proportional to the real time used for the computation as shown in Figure 20 as well. Using more number of cores requires less energy computation due to the less time for computation.

Based on this result, the approximate cost for electricity is also given in Figure 20. The cost refers to the tariff information (26.3 Cent/kWh) from one of the electricity provider in Germany (Stadtwerke Karlsruhe [13]). The relation between the energy and the cost is

defined by Equation 2:

$$\text{Cost [Euro]} = 0.263\frac{1}{\text{kWh}}[\text{Euro}] = 0.073\frac{1}{\text{MJ}}[\text{Euro}] \tag{2}$$

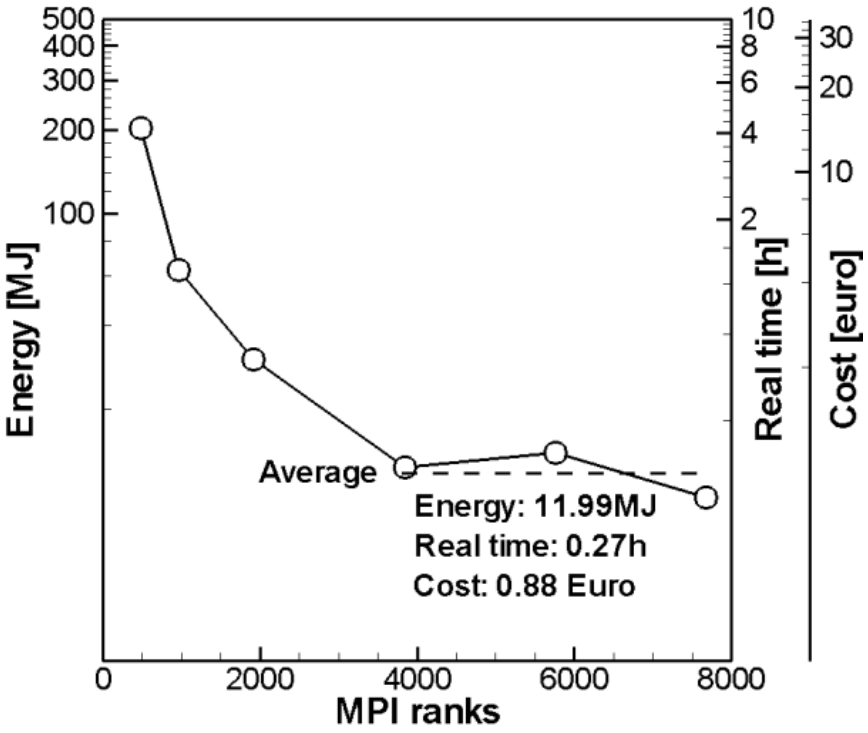

Figure 20: Energy consumptions and corresponding real times and costs for 5000 time steps of nektar++ with respect to the MPI rank.

The use of more cores offers definite benefits for the cost of energy till 4000 cores. After this number of cores, the cost remains in a certain range. The average energy consumption for 3840, 5760 and 7680 cores is 11.99MJ and the corresponding real time and cost are 0.27h and 0.88 Euros, respectively.

Cost of the HPC usage: Additionally, the cost of use of high performance computing system is also estimated from the user point of view. The formulation of user cost [6] is given by Equation 3,

$$\text{Cost} = 0.89[\text{Euro}] \times \text{Time [h]} \times \text{Number of Nodes} \tag{3}$$

where the time represents the real time of computation. This cost rate (0.89 Euros) is only valid for the users at one university and research facility in Germany [6].

Current calculation uses 20 processors per node. Based on this formula Equation 3, Figure 21 (a) compares the user cost for both output formats. The user cost with xml format is

gradually increasing as MPI rank grows, while the hdf5 format shows non-monotonic behavior against the number of processors. In Figure 19, the real time of hdf5 increases drastically with lower number of cores. It makes the concave shape of the cost distribution with hdf5 format. As shown in the scalability analysis, this user cost analysis also clearly shows that there is an intersection between two output formats. Hdf5 format is appropriate only with more than approximate 3000 number of core, while xml is economic with less number of cores. With a linear extrapolation of the number of processor per node, it is found out that using 24 processors per node would further reduce the user cost as shown in Figure 21 (a).

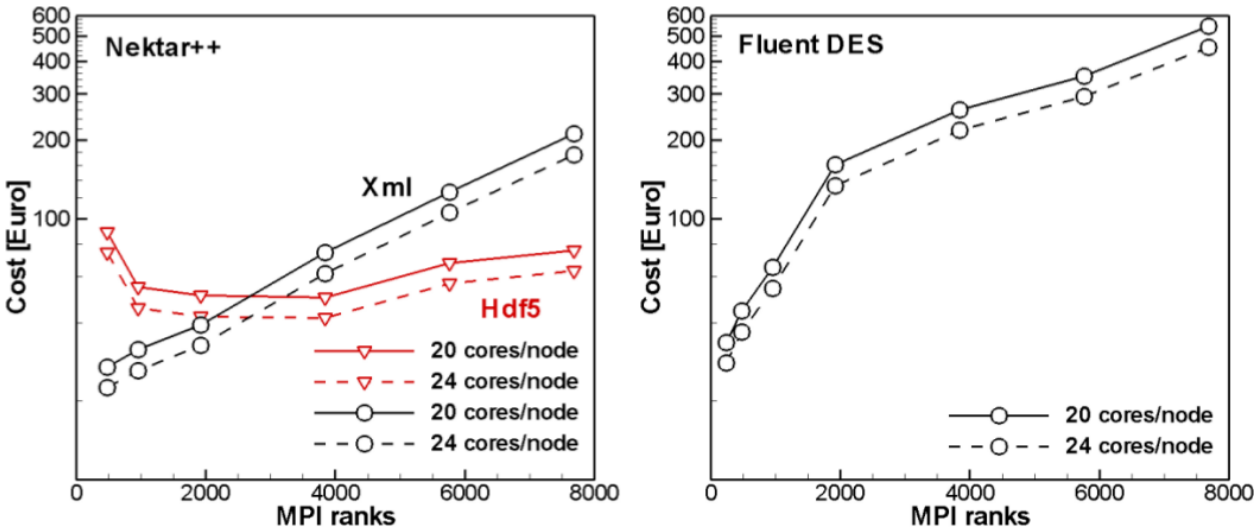

Figure 21: Cost for user of high performance computing facility. Solid line: current calculation with 20 cores per node, dashed line: imaginary line for calculation with 24 cores per node.

According to the real time of Fluent scalability test, Figure 21 (b) shows the user cost for the DES calculation with respect to the MPI rank. This shows monotonic increasing of the cost against the number of cores. This is explained by the real time behavior shown in Figure 14. With more than 1000 cores, the real time does not change much with increasing number of cores. Therefore, the user cost is directly related to the increasing number of cores. This cost is even higher than that for the xml format of nektar++.

4.4 Feedback provided to WP1/WP2

Based on the real time and energy cost described in the previous subsection, using more processors is better to reduce the time and corresponding electricity usage. The required times and energies for 3840 cores and 7680 cores are almost similar. However, with consideration of HPC user cost, approximate 4000 number of cores is an optimal number for the current version of the solver using the Hdf5 Format. In order to be able to increase the mesh resolution in the automotive use case a better solver performance using more than 7680

cores is required. The completely successfully integration of the hybrid CG-HDG approach could help to solve this requirement. Even more increasing the performance of the solver significantly will enable a flexible choice of the computational domain (bigger domain size).

5 Imperial Front Wing

One of our goals here in McLaren as industrial partners is to demonstrate that the algorithms developed within the ExaFLOW consortium will potentially help improve the accuracy and/or throughput of our production CFD simulations on complex geometries. To that end, we have proposed and are actively setting up large demonstration cases with high geometrical complexity as part of WP3, and we are in the process of running additional wind tunnel experiments to provide a transient experimental dataset to compare against. Those demonstration cases will however require a significant amount of HPC resources to be run and obtain accurate statistics, and it is important for shorter term development/testing to provide a subset of test cases which could be run more often and on more limited resources.

5.1 Submodels

5.1.1 Wing tip vortex

The wing tip vortex was studied experimentally by [2], and later numerically, amongst other sources, by [9, 10], albeit at reduced Reynolds numbers. [10] kindly provided a mesh and a solution checkpoint.

5.1.2 McLaren Front Wing without the wheel

The second test case under consideration is based on the McLaren 17D race car, and was studied experimentally by [12], hence providing validation data for the simulations. In order to provide a step further in complexity when compared to the naca wing tip cases whilst maintaining a reasonnable turnaround time, we elected to run the front wing in isolation, as [12] provides experimental data for this configuration. This alleviates most of the CFL restrictions related to the tyre contact patch handling.
This geometry is also being used as a test case for Imperial College's high order mesher NekMesh ([14]), and we will now be assessing the effect of near wall resolution on the solution.

5.2 Numerical setup

The exact numerical setup used for both those test cases is now provided, and both setup files and checkpoints are available on demand. Both cases were run using the Incompressible Navier Stokes solver of Nektar++, and the setup parameters used are reproduced in listing 2 and 3 for the wing tip and front wing case respectively. To ensure the numerical experiments are repeatable, the git hash of the version of the code used is reproduced in listing 1.

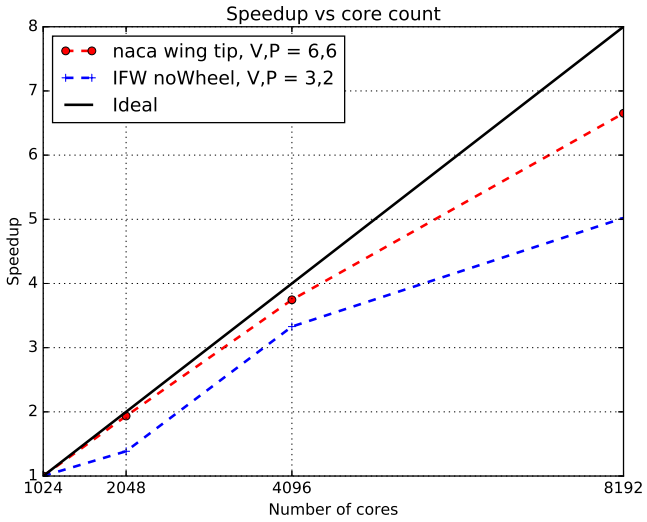

Figure 22: Speedup against core count for the naca wing tip and McLaren front wing without wheels from 1024 to 8192 cores

Listing 1: Git hash of the version of Nek++ used

```
commit 61ce4a4bfcbc3b57d5be509352f5a2660a65b945
Author: ssherw <s.sherwin@imperial.ac.uk>
Date:   Sat Oct 22 17:18:27 2016 +0100

    Updates to make merge into current master work. All but one regression test working for IncNSSolver
```

The next step is to quantify the runtime of the code on those two cases on a range of core counts.

5.3 Scaling test

5.3.1 Initial result

An initial scaling test was performed on both submodels on our local cluster, reported on Fig. 22. The wing tip case was run using sixth order accuracy for both velocity and pressure (see listing 2), while the front wing case was run at third order for velocity and second order for pressure. Both cases are using the SpectralVanishingViscosity flag in Nektar++ to improve stability, and the number of global degrees of freedom for both cases is respectively 27.5×10^6 and 18.2×10^6.

Fig. 22 shows that the low order case drops earlier than the wing tip case when increasing the core count. In order to assess whether this is solely due to the lower number of degrees of freedom or to the polynomial order chosen, a few more options will now be tested. It is also worth mentioning that this initial scaling test was run only once at each core count. Finally, we used a deprecated option for the partitionning weightings "uniform", and were advised to use the "dof" option instead.

Config name	Geometry	NumModes	NumPoints	Global DOFs	PartitionWeights
ifw32	Front wing	$3, 2$	5	18.2×10^6	Uniform
ifw43	Front wing	$4, 3$	6	18.2×10^6	Uniform
ifw32qp	Front wing	$3, 2$	6	18.2×10^6	Uniform
ifw43qp	Front wing	$4, 3$	7	64.5×10^6	Uniform
ifw43dof	Front wing	$4, 3$	6	64.5×10^6	Dof
naca66	Naca	$6, 6$	10	27.5×10^6	Uniform
naca66qp	Naca	$6, 6$	11	27.5×10^6	Uniform
naca66dof	Naca	$6, 6$	11	27.5×10^6	Dof

Table 4: Configurations considered for the scaling test

5.3.2 Second battery of tests

The protocol for the second scaling test is as follows. On both models, and for each configuration considered, a simulation of a hundred time steps was run and the cpu time per time step was computed after discarding the first 10 iterations. Each of those runs is then repeated three times and the mean is computed. To assess the effect of polynomial order, quadrature points and partition weightings on the runtime, we devised the list of configurations presented in table 4

Fig. 23 shows the results of those test. The front wing case was only ran up to 4096 cores due to limited resources, and some points are removed from the curves because of insufficient number of repeats.

The first point worth mentioning is the rather low scatter on the repeats, as shown by the tight clustering on Fig. 23 top two graphs. Secondly, the alternative option "dof" for partition weightings was proven to improve the performance on the wing tip case, but there is a crossover on the front wing case where it only seems beneficial at the higher end of the core count range.

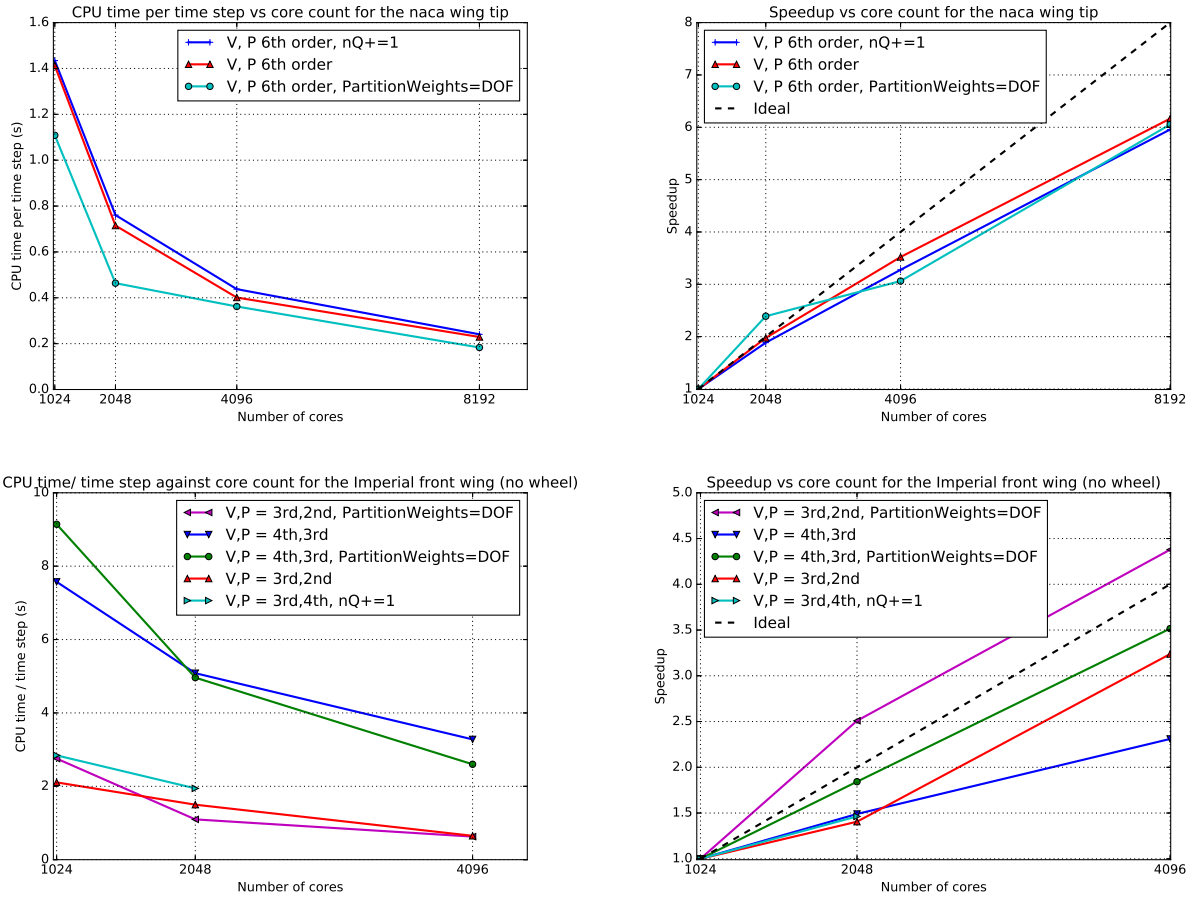

Figure 23: Scaling test: CPU time per time step (left) and speedup (right) against core count for the naca wing tip case (top) and the McLaren front wing without wheel (bottom)

Listing 2: naca wing tip setup

```
<SOLVERINFO>
    <I PROPERTY="SolverType" VALUE="VelocityCorrectionScheme" />
    <I PROPERTY="EQTYPE" VALUE="UnsteadyNavierStokes" />
    <I PROPERTY="AdvectionForm" VALUE="Convective" />
    <I PROPERTY="Projection" VALUE="Galerkin" />
    <I PROPERTY="TimeIntegrationMethod" VALUE="IMEXOrder2" />
    <I PROPERTY="GLOBALSYSSOLN" VALUE="IterativeStaticCond" />
    <I PROPERTY="SuccessiveRHS" VALUE="30" />
    <I PROPERTY="SpectralhpDealiasing" VALUE="True" />
    <I PROPERTY="PRECONDITIONER" VALUE="FullLinearSpaceWithDiagonal" />
    <I PROPERTY="WeightPartitions" VALUE="NonUniform" />
    <I PROPERTY="SpectralVanishingViscosity" VALUE="TRUE" />
</SOLVERINFO>
<PARAMETERS>
    <P> TimeStep       = 1e-5 </P>
    <P> NumSteps       = 100 </P>
    <P> Kinvis         = 1.2/1200000 </P>
    <P> IterativeSolverTolerance = 1e-8 </P>
    <P> SVVDiffCoeff   = 0.1 </P>
    <P> SVVCutoffRatio= 0.5 </P>
</PARAMETERS>
<EXPANSIONS>
<E COMPOSITE="C[0]"
   BASISTYPE="Modified_A,Modified_A,Modified_B"
   NUMMODES="6,6,6"
   POINTSTYPE="GaussLobattoLegendre,GaussLobattoLegendre,GaussRadauMAlpha1Beta0"
   NUMPOINTS="10,9,9"
   FIELDS="u,v,w,p" />
<E COMPOSITE="C[1]"
   BASISTYPE="Modified_A,Modified_B,Modified_C"
   NUMMODES="6,6,6"
   POINTSTYPE="GaussLobattoLegendre,GaussRadauMAlpha1Beta0,GaussRadauMAlpha2Beta0"
   NUMPOINTS="10,9,9"
   FIELDS="u,v,w,p" />
</EXPANSIONS>
```

Listing 3: McLaren front wing (no wheel)

```
<SOLVERINFO>
    <I PROPERTY="SolverType" VALUE="VelocityCorrectionScheme" />
    <I PROPERTY="EQTYPE" VALUE="UnsteadyNavierStokes" />
    <I PROPERTY="AdvectionForm" VALUE="Convective" />
    <I PROPERTY="Projection" VALUE="Continuous" />
    <I PROPERTY="TimeIntegrationMethod" VALUE="IMEXOrder2" />
    <I PROPERTY="GLOBALSYSSOLN" VALUE="IterativeStaticCond" />
    <I PROPERTY="SuccessiveRHS" VALUE="30" />
    <I PROPERTY="SpectralhpDealiasing" VALUE="True" />
    <I PROPERTY="PRECONDITIONER" VALUE="FullLinearSpaceWithDiagonal" />
    <I PROPERTY="WeightPartitions" VALUE="NonUniform" />
    <I PROPERTY="SpectralVanishingViscosity" VALUE="TRUE" />
</SOLVERINFO>
<PARAMETERS>
    <P> TimeStep = 1.00e-5 </P>
    <P> NumSteps = 100 </P>
    <P> Linf = 0.24 </P>
    <P> Re = 5.0e4 </P>
    <P> Kinvis = Linf/Re </P>
    <P> IterativeSolverTolerance = 1e-8 </P>
    <P> SVVDiffCoeff = 2.5 </P>
    <P> SVVCutoffRatio = 0.4 </P>
</PARAMETERS>
<EXPANSIONS>
    <E COMPOSITE="C[0]" BASISTYPE="Modified_A,Modified_A,Modified_B" NUMMODES="3,3,3" POINTSTYPE="
        GaussLobattoLegendre,GaussLobattoLegendre,GaussRadauMAlpha1Beta0" NUMPOINTS="5,5,4" FIELDS="u,v,w"
        />
    <E COMPOSITE="C[0]" BASISTYPE="Modified_A,Modified_A,Modified_B" NUMMODES="2,2,2" POINTSTYPE="
        GaussLobattoLegendre,GaussLobattoLegendre,GaussRadauMAlpha1Beta0" NUMPOINTS="5,5,4" FIELDS="p" />
    <E COMPOSITE="C[1]" BASISTYPE="Modified_A,Modified_B,Modified_C" NUMMODES="3,3,3" POINTSTYPE="
        GaussLobattoLegendre,GaussRadauMAlpha1Beta0,GaussRadauMAlpha2Beta0" NUMPOINTS="5,4,4" FIELDS="u,v,
        w" />
    <E COMPOSITE="C[1]" BASISTYPE="Modified_A,Modified_B,Modified_C" NUMMODES="2,2,2" POINTSTYPE="
        GaussLobattoLegendre,GaussRadauMAlpha1Beta0,GaussRadauMAlpha2Beta0" NUMPOINTS="5,4,4" FIELDS="p" /
        >
</EXPANSIONS>
```

5.4 Infinite pipe flow case

Most of Imperial's effort has concentrated on the development of a combined CG-DG method. The idea is to apply a discontinuous Galerkin discretization for elliptic problems to entities which are not mesh elements, but groups of elements (patches). Each patch is spanned by a C0-continuous basis and discretized by a standard Galerkin approach. The solution within each patch is reconstructed from values on its boundary in manner known from hybrid discontinuous Galerkin method (HDG). When considering a mesh with a single patch, this step can be interpreted as a weak enforcement of Dirichlet boundary conditions which is different from existing penalty-based approaches such as Nitsches method.

Imperial/McLaren plan to investigate the effect of weak boundary conditions on solver stability in turbulent flows, especially when the flow features are not fully resolved. Given the fact that the weak enforcement of no-slip condition on wall boundaries allows for the computed velocity to differ from imposed values to larger extent than in strong case, we would like to understand whether this relaxation of constraints would render the solver more stable in applications where stability has been previously an issue. From the perspective of WP3, this is particularly important in studying the McLaren F1 car simulations, where the contact patch between moving tyre and stationary road induces large pressure gradients that can affect the stability of the simulation. Over the coming period the use of these weakly-imposed boundary conditions will be evaluated with respect to this use case.

5.4.1 Initial evaluation

As an initial test in three-dimensions, Imperial/McLaren are considering incompressible flow in an infinite pipe using a mixed spectral/hp element and Fourier discretization available in Nektar++. This aligns with part of the jet in crossflow simulations, where a pipe flow is used to generate the jet. The Fourier expansion is used in the streamwise direction and the spectral/hp elements are used to capture the circular cross-section, so that the weak boundary conditions are imposed in circular cross-sections at the wall for the velocity field. An example solution of laminar flow with $Re = 250$ obtained on P_7 elements with 8 Fourier modes in streamwise direction is presented below. Note that in Figure 24 depicting the strongly imposed boundary conditions, the velocity field has a minimum of precisely zero, whereas for the weakly imposed conditions this is a small value above zero as expected.

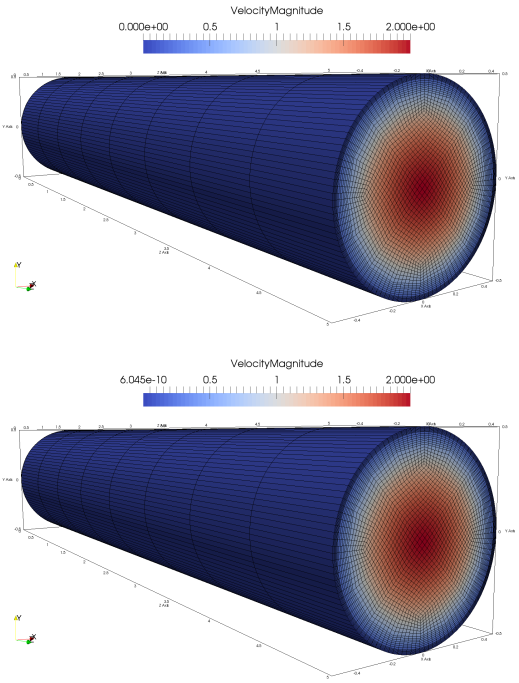

Figure 24: Laminar solution with strong (top) and weak (bottom) boundary conditions for velocity.

5.5 Feedback provided to WP1/WP2

Imperial is looking to apply the HDG weak boundary conditions to one of the industrial test cases which will most likely be the Imperial Front Wing geometry from McLaren. In order to alleviate some of the demands for resolution in these large flagship WP3 simulations, the aim will be to evaluate how well these BCs can be used in the context of regions of low interest such as the contact patch area first.

5.6 Outlook and future work

As far as the next steps are concerned, McLaren are going to speed up now to try to prepare the demonstration case:

- Jean Eloi Lombard, Imperial/McLaren's PhD candidate with Spencer Sherwin, is going to come to work for McLaren for six months from May onwards to prepare the large-scale simulation.
- Mike Turner will join McLaren from April onwards to look specifically at high order meshing on our geometry using his mesher NekMesh.

Imperial's future work (in the coming weeks) will include an update of the test case to turbulent flow and an assessment of the implementation for various polynomial orders of spatial discretization and different flow conditions. In particular Imperial/McLaren will examine the effects of under-resolution in the presence of the weakly imposed boundary conditions, in order to quantify their effect in the industrial flow regimes of interest to WP3.

6 Conclusions and Future Work

This deliverable has presented the initial assessment of the new algorithms and code developed as part of the ExaFLOW work packages 1 and 2. Many results show significant improvements in terms of run-time, and potential for running large-scale simulations in preparation for exascale computing. It is clear that there is scope for further benefits as the algorithms are improved via the feedback cycle. After an iteration with WP1 and WP2, to hone the new developments, the use cases will be run in their final large-scale configurations, for more challenging conditions (typically higher Reynolds number) at higher resolution than those simulations presented here, pushing the limits of the available hardware and confirming, or otherwise, the success of the utility of the algorithmic developments. These large-scale flagship calculations will be presented as part of Deliverable 3.3 at PM 36, along with Deliverable 3.4 which comprises a critical evaluation of the ExaFLOW developments from the perspective of the industrial partners.

References

[1] C.D. Cantwell, D. Moxey, A. Comerford, A. Bolis, G. Rocco, G. Mengaldo, D. De Grazia, S. Yakovlev, J.-E. Lombard, D. Ekelschot, B. Jordi, H. Xu, Y. Mohamied, C. Eskilsson, B. Nelson, P. Vos, C. Biotto, R.M. Kirby, and S.J. Sherwin. Nektar++: An open-source spectral/ element framework . *Computer Physics Communications*, 192:205–219, 2015. ISSN 0010-4655. doi: 10.1016/j.cpc.2015.02.008.

[2] Jim S Chow, Gregory G Zilliac, and Peter Bradshaw. Mean and turbulence measurements in the near field of a wingtip vortex. *AIAA journal*, 35(10):1561–1567, 1997.

[3] Cray Inc. Using Cray Performance Measurement and Analysis Tools. Technical Report S-2376-622, Cray Inc., 2014. URL `http://docs.cray.com/books/S-2376-622/`.

[4] C. J. Falconi D. and D. Lautenschlager. ExaFLOW use case: Numerical simulation of the rear wake of a sporty vehicle. Exaflow technical report, ASCS, 2016. URL `http://exaflow-project.eu/index.php/use-cases/automotive`.

[5] HLRS. Cray xc40 (hazel hen), 2017. URL `http://www.hlrs.de/en/systems/cray-xc40-hazel-hen/`.

[6] HLRS. Neufassung der hlrs entgeltordnung (stand 2016),, 2017. URL `https://www.hlrs.de/solutions-services/academic-users/legal-requirements/`.

[7] C. T. Jacobs, S. P. Jammy, and N. D. Sandham. OpenSBLI: A framework for the automated derivation and parallel execution of finite difference solvers on a range of computer architectures. *Journal of Computational Science*, 18:12–23, 2017. doi: 10.1016/j.jocs.2016.11.001.

[8] C. T. Jacobs, N. D. Sandham, and N. De Tullio. An error indicator for finite difference methods using spectral techniques with application to aerofoil simulation. In *Abstracts of the Parallel CFD (ParCFD) 2017 conference, Glasgow, Scotland, 15–17 May 2017*, Accepted. URL `http://www.strath.ac.uk/engineering/parcfd2017/`.

[9] Li Jiang, Jiangang Cai, and Chaoqun Liu. Large-eddy simulation of wing tip vortex in the near field. *International Journal of Computational Fluid Dynamics*, 22(5):289–330, 2008.

[10] Jean-Eloi W. Lombard, David Moxey, Spencer J. Sherwin, Julien F. A. Hoessler, Sridar Dhandapani, and Mark J. Taylor. Implicit large-eddy simulation of a wingtip vortex. *AIAA Journal*, 54(2):506–518, 2016/04/26 2015. doi: 10.2514/1.J054181. URL `http://dx.doi.org/10.2514/1.J054181`.

[11] R. Nash, S. Clifford, C. Cantwell, D. Moxey, and S. Sherwin. eCSE 02-13 technical report: Communication and I/O masking for increasing the performance of Nektar++. Technical report, 2016. URL `https://www.archer.ac.uk/community/eCSE/eCSE02-13/eCSE02-13-TechnicalReport.pdf`.

[12] J. M. Pegrum. *Experimental Study of the Vortex System Generated by a Formula 1 Front Wing*. PhD thesis, Imperial College London, 2006.

[13] Stadtwerke Karlsruhe. Preisblatt alttarife strom, 2017. URL `https://www.stadtwerke-karlsruhe.de/`.

[14] Michael Turner, David Moxey, Spencer J Sherwin, and Joaquim Peiró. Automatic generation of 3d unstructured high-order curvilinear meshes. In *Proceedings of the European Congress on Computational Methods in Applied Sciences and Engineering*, 2016.