H2020 FETHPC-1-2014

Enabling Exascale Fluid Dynamics Simulations
Project Number 671571

D1.3 - Final report on ExaFLOW algorithmic developments

WP1: Algorithmic improvements towards exascale

DOCUMENT INFORMATION

Deliverable Number	D1.3
Deliverable Name	Final report on ExaFLOW algorithmic developments
Due Date	TBC (PM 36)
Deliverable lead	KTH
Authors	David Moxey (Imperial) Chris Cantwell (Imperial) Martin Vymazal (Imperial) Nicolas Offermans (KTH) Adam Peplinski (KTH) Niclas Jansson (KTH) Philipp Schlatter (KTH) Satya Jammy (SOTON) Christian T. Jacobs (SOTON) Neil Sandham (SOTON) Björn Dick (USTUTT) Jing Zhang (USTUTT) Uwe Küster (USTUTT) Patrick Vogler (USTUTT) Ulrich Rist (USTUTT) Nielsen Allan Svejstrup (EPFL) Jan Hesthaven (EPFL)
Responsible Author	David Moxey (Imperial) e-mail: `d.moxey@imperial.ac.uk`
Keywords	exascale algorithms, scalability, modelling, input/output
WP	WP1
Nature	R
Dissemination Level	PU
Final Version Date	TBC
Reviewed by	Niclas Jansson (KTH), Rahul Bale (RIKEN)
MGT Board Approval	30/09/2018

DOCUMENT HISTORY

Partner	Date	Comment	Version
Imperial	30/07/2018	Initial skeleton document	0.1
Imperial	28/08/2018	Incorporate changes from KTH	0.2
Imperial	30/08/2018	Incorporate changes from SOTON	0.3
Imperial	31/08/2018	Incorporate changes from USTUTT	0.4
Imperial	03/09/2018	Incorporate fault tolerance work from IC and EPFL	0.5
Imperial	18/09/2018	Incorporate changes from IC and final draft	0.6
Imperial	23/09/2018	Address reviewer comments	0.7
KTH	28/09/2018	Address reviewer comments	0.8
KTH	30/09/2018	Final Version	1.0

Executive Summary

In this deliverable, we will outline the final algorthmic developments made by the work package 1 (WP1) ExaFLOW partners, summarising the work performed across the 36 months of ExaFLOW and focusing on the final 18 months in particular, since the previous deliverable (D1.2). Many of the developments here have led to journal or conference publications, and these are cited where appropriate. Finally, we conclude with overall progress and a brief discussion of future directions of research.

- **Error control and mesh refinement:** A central theme of WP1 activity has been the development of new algorithmic techniques for adaptive simulations at extreme scales, which we discuss in section 2. Adaptive simulations have the advantage that *a priori* knowledge of the flow physics is not required and resolution can be automatically tuned depending on local flow features, which is important in the context of extremely large simulations that will occur at exascale. However, this requires the development of robust, reliable error indicators and estimators to determine the regions of the domain that require additional resolution, as well as efficient techniques to perform adaption at extreme scales.

  In terms of error estimation, we have followed two approaches within the project. The first, which we discuss in section 2.1, attempts to detect regions of error by observing the decay of a spectrum coefficients, comparing this against the expected decay rates that correspond to the underlying numerical method. This spectral error indicator has been examined for both finite difference methods, within the *OpenSBLI* code, and additionally for the spectral element method within *Nek5000*. The main advantage of this approach is that it is local in nature and so negligible communication is required, making it both highly scalable and cheap to compute. The efficacy of this indicator has been examined in various examples which we document here for *OpenSBLI* and previously in D1.2 for *Nek5000*. However, one drawback of the spectral error indicator is that it has no particular knowledge of a fluid dynamics setting, and therefore may lead to increases in resolution that are not necessarily required in the context of improving, for example, the calculation of a drag coefficient. The second error estimator, outlined in section 2.2, aims to address this issue through the use of a goal-based adjoint error estimator. This yields a more accurate estimation of the error using a global approach throughout the domain, but does come at a higher computational cost.

  For the purposes of adaption, we have developed algorithms that combine h-type adaptive mesh refinement (i.e. a reduction in element size h) with a nonconformal high-order spectral element method. This approach has been shown to be highly effective in effectively utilising computational hardware (through the use of higher-order finite elements) alongside massive scalability and the ability to adapt to more complex geometries through the nonconformal implementation. In section 2.3, we focus on the requirements for preconditioning, solver stabilisation under this approach and algorithmic developments required to overcome challenges in partitioning. Full implementation details are outlined in D2.4.

Finally, we have considered an alternative approach to error control by leverageing a heterogeneous approach to fluid simulations. In this regime, different fluid models are used in different regions of the domain depending on the level of detail required, which both reduces the computational effort required and increase scalability. Our efforts in this area are summarised in section 2.4. Here, we examine the possibility of reducing computational cost by combining a more expensive direct numerical simulation (DNS) together with a cheaper turbulence closure (RANS) for a turbulent channel flow, where reductions of 25% in terms of grid points yield accuracy to within 2% of a full DNS of the equivalent case.

- **Mixed CG-HDG discretisation:** In this task we have developed a formulation for a mixed finite element discretisation, which aims to improve strong scalability by combining two complementary methods. The first classical continuous Galerkin (CG) method, is locally compact and exhibits excellent computational performance on modern hardware but at the expense of potentially complex communication patterns. The second, hybridizable discontinous Galerkin (HDG) method, relaxes the communication requirements by removing continuity constraints between elements, leading to more efficient pairwise communication patterns. However this comes at a higher computational cost as HDG utilises a larger number of degrees of freedom compared to CG. In section 3, we outline a formulation that uses 'macro-elements' that are composed of CG elements and connected using the HDG method, with the hope of arriving at a method that is both computationally efficient and strong-scalable. When doing the cost analysis of the CG-DG algorithm, however, the assembly and solution of the global system for the inter-node variable incurs significantly larger asymptotic cost. This means that, although the strong scaling of this method would indeed be greatly improved, it would come at the cost of significant additional runtime when compared to the CG or HDG methods. We discuss this aspect further in deliverable D2.4. However, this method has led to the development of a novel weakly-imposed boundary condition for the incompressible fluid dynamics simulations in a parameter-free manner, unlike classical formulations such as the Nitsche method, which we discuss in this section and demonstrate through a number of numerical examples.

- **Data compression:** Another key theme throughout the ExaFLOW project has been to investigate I/O and techniques that can be used to deal with the extremely large datasets that will be produced at exascale. In WP1, we have developed a number of techniques for compression of datasets. In this approach, we aim to retain the key features of the flow whilst significantly reducing the volume of data that is required for storage, thereby alleiviating pressure on parallel filesystems and longer-term storage requirements. In section 4, we outline two of the approaches that have been considered and tested during the project. The first, discussed in section 4.1, uses the JPEG-2000 format with a wavelet-based compression algorithm to develop an algorithm that is able to conserve most of the internal energy of the fluid domain and minimize the introduction of compression artifacts to support both visual and statistical evaluation.

The second approach investigates a reduced-order modelling technique based around the singular value decomposition (SVD) of the underlying flow data. Here, the aim is to mathematically extract key features of the flow by determining a small set of so-called singular values for the flow data, therefore obtaining substantial data reduction savings. In this deliverable, we summarise the theory behind these two approaches, and consider the implementation and a number of test cases within deliverable D2.4 as part of WP2.

- **Fault tolerance:** With the current mean time to failure of current computing hardware, it is expected that errors that are either hard (e.g. hardware failure) or soft (e.g. memory corrpution) will occur within a matter of only a few minutes at exascale. In ExaFLOW, we have been developing algorithms to ensure resilience at exascale for fluid dynamics simulations.

  These developments have broadly followed two lines of enquiry. In the first approach, knowing that fluid dynamics solvers are typically complex codes, we have developed a minimially-intrusive approach to dealing with hard failures specifically tailored to applications that deal with transient simulations. This approach leverages user-level failure mitigation (ULFM) extensions to the MPI communiation framework, combined with spare nodes, to handle node failure during a simulation. In particular, the simulation is broken up into an initial short 'static' phase that handles e.g. initial matrix setup, and a 'dynamic' phase that represents timestepping of the equations of state. By combining fast, in-memory backups of these states, which enable frequent and low-overhead checkpointing during the dynamic phase, we are able to construct an efficient and highly resilient method for transient flow simulations. To demonstrate the efficacy of this scheme, the incompressible Navier-Stokes solver of Nektar++ has been extended to incorporporate this algorithm. A full description of this implementation, alongside performance analysis at larger scales, is described further in D2.4.

  In our second approach, we have considered more advanced techniques that are more intrusive within fluid dynamics codes, but offer the ability to increase resilience in a highly memory efficient manner. Our approach is based on a Reed-Solomon erasure code, which is used for error correction in, for example, storage services. This approach allows for recovery of data whilst balancing the amount of memory storage required using a parity technique. A new library, named 'Llama' has been implemented around ths for this method has been implemented as part of WP2 and is described in further detail in D2.4. In this deliverable, section 5.2 describes how this technique can be modified from an algorithmic perspective to allow for partial data recovery, even when the parity information is only partially available, providing the ability to significantly increase robustness of fluid dynamics codes.

Contents

List of Figures **8**

Abbreviations **11**

1 Introduction **12**

2 Error control and mesh refinement **13**
2.1 Spectral error indicators . . . 13
2.1.1 Spectral element discretisations . . . 13
2.1.2 Finite difference discretisations . . . 13
2.2 Adjoint error estimators . . . 17
2.2.1 Weak form of the steady Navier–Stokes equations . . . 17
2.2.2 Expression of the functional . . . 17
2.2.3 Lagrange optimisation . . . 18
2.2.4 Adjoint equations . . . 18
2.2.5 Error on the functional . . . 18
2.2.6 Contributions to the error . . . 19
2.3 Adaptive mesh refinement . . . 21
2.3.1 Adaptation of the hybrid Schwarz-multigrid preconditioner to nonconforming meshes . . . 22
2.3.2 Solver stabilisation based on high-pass filter . . . 23
2.3.3 Two-level partitioning . . . 25
2.4 Error control for heterogeneous modelling . . . 27
2.4.1 Methodology . . . 27
2.4.2 Results . . . 28

3 CG-HDG formulation **31**
3.1 Motivation . . . 31
3.1.1 Weak boundary condititions as part of CG-HDG formulation . . . 31
3.1.2 Weak boundary conditions in underresolved flows . . . 32
3.1.3 Review of existing algorithms for boundary discretization . . . 33
3.2 Overview of the formulation of HDG method . . . 34
3.3 Continuous problem . . . 34
3.4 HDG interpolation spaces and discretization . . . 34
3.5 Approximation spaces . . . 36
3.6 Global formulation for HDG problem . . . 36
3.7 Local solvers in the HDG method . . . 37
3.8 Global problem for trace variable . . . 37
3.9 Discrete form of HDG local solver . . . 38
3.10 Continuous finite elements with weak Dirichlet boundary conditions . . . 39
3.11 Results . . . 41

3.12 Convergence of continuous Galerkin solver with weak boundary conditions . 41
3.13 Comparison with classical penalty techniques 43
3.14 Navier-Stokes results 45
3.14.1 NACA6412 45
3.14.2 Unsteady flow past a turbine blade 49
3.15 Discretization of global transmission condition in the CG-HDG algorithm . . 52

4 Compression algorithms **52**
4.1 JPEG 2000 54
4.1.1 Theory 54
4.2 Singular Value Decomposition 56

5 Fault tolerance **57**
5.1 Minimally intrusive resiliance for transient solvers 58
5.1.1 State Protection 59
5.1.2 State Recovery 61
5.2 Partial Information Recovery with Incomplete Checksums 61
5.2.1 The Weighted Checksum Scheme 64
5.2.2 Partial information recovery for incomplete checksums 66
5.2.3 Uniqueness by Minimizing Distance to Inexact Data 67
5.2.4 An Iterative, Self-Feeding, Recovery Scheme 68
5.2.5 Summary 71

6 Summary and outlook **75**

List of Figures

1 Computational domain arrangement. Image by [39]. 15
2 Contours of velocity magnitude in a two-dimensional slice (in the x–y plane at $z = 0$) from a V2C aerofoil simulation. Circles filled with blue, green, yellow and red indicate I_i error severity values of 0, 1, 2 and 3, respectively. The top and bottom images are for the coarse and refined grids, respectively. Images by [39]. 16
3 Vortical structures (λ_2 criterion) of the velocity field around the wing (left panel) and the part of the domain covered by refinement levels higher and equal 2 (right panel) at simulation time 7.2 for $Re_c = 2 * 10^5$ case. 21
4 1 Linear masking function restricting action of the high-pass filter to the nonconforming interfaces for 2D wing simulation. 24
5 Domain decomposition for one-level (left) and two-level (right) partitioning in 3D wing simulation at $Re = 2 * 10^5$. In red is visualised a part of the domain residing on the second node (cores 32-64). The wing position is marked in blue. 25

6 Time of pressure preconditioner execution (left) and time per time step (right) for one- and two-level partitioning. Results of a 3D wing simulation at $Re = 2 * 10^5$. 26
7 Mean velocity profiles using a cutoff β. The dashed lines indicate β_1 which is where β switches from 0 to 1. 29
8 Mean velocity profiles using β following a sinusoid of half-period $20\delta_\nu$. The dashed lines indicate the bounds where β varies from 0 to 1. The location at which the sinusoid starts is identified by β_1. 29
9 Mean velocity profiles using β following a sinusoid of half-period $40\delta_\nu$. The dashed lines indicate the bounds where β varies from 0 to 1. The location at which the sinusoid starts is identified by β_1. 30
10 Mean velocity profiles obtained with DNS and blended DNS/RANS on a fine (identical to the DNS) and coarse (4 times fewer points in the wall normal direction) grids. 30
11 Distribution of unknowns for continuous and discontinuous Galerkin methods. 31
12 Mesh decomposition in CG-HDG setting. Each partition is discretized by a continuous Galerkin method. Coupling between partitions is weak and information is transmitted through a hybrid variable (red degrees of freedom). 32
13 Implicit LES of a wingtip vortex using a high-order hp/spectral element solver. Reproduced from [52]. 33
14 Computational domain and its tesselation demonstrating notation used in the text. 35
15 Index mappings relating edge and element ids. 35
16 Meshes for Helmholtz convergence test. 42
17 Convergence to the exact solution in L_2 norm. 43
18 Anisotropic mesh of a unit square. 45
19 Viscous incompressible flow past NACA airfoil: velocity magnitude obtained with strong Dirichlet boundary conditions imposed on the airfoil surface (top) and with weak boundary conditions (bottom). 47
20 Viscous incompressible flow past NACA airfoil: drag and lift on P_4 elements (top) and P_8 elements (bottom). 48
21 Turbine blade geometry: domain and leading edge detail 49
22 Averaged velocity field obtained with strong boundary conditions on blade wall (left column) and weak boundary conditions (right column). 50
23 Instantaneous velocity field obtained with strong boundary conditions on blade wall (left) and weak boundary conditions (right). 51
24 Kinetic energy vs. time. 51
25 Structure of wavelet-based compression algorithm for volumetric floating-point arrays. Encircled letters indicate the floating-point to fixed-point transform (Q), discrete wavelet transform (W) and entropy encoding stage (E). . 55

26 2-dimensional representation of geometric operations performed during Embedded Block Coding with Optimized Truncation stage. Red squares signal a precinct, blue squares a code-block. 56
27 The form of Singular Value Decomposition 57
28 Data reduction with Singular Value Decomposition 58
29 Diagrammatic representation of the protection algorithm. Initialisation of solver, showing three processes – two active and one spare – with ranks i, j, and N, respectively. Rank i communicates static recovery data to rank j. After a number of steps, at time t_{C_0}, a remote in-memory checkpoint occurs. The spare rank, N remains idle throughout. Red MPI regions denote collective communication, while green regions denote pairwise communication. 60
30 Diagrammatic representation of the recovery algorithm. Process A with rank i fails and, after enrolment and rank translation during recovery, process S is assigned rank i and receives recovery data from rank j. Static and dynamic data is subsequently recovered to the last checkpoint at time t_{C_k} without requiring any further communication with surviving ranks. The simulation then continues. Red MPI regions denote collective communication, while green regions denote pairwise communication. 62
31 Numerical experiment testing the method (69) for partial information recovery in incomplete checksums. The data indicated by the black line was stored in $k = 100$ separate containers. $m = 15$ checksum vectors were created to protect the content of the containers. The $k_{lost} = 20$ first containers are removed, i.e. 33% more data vectors are lost than checksum code vectors created. 69
32 Numerical experiment testing the method (69) for partial information recovery in incomplete checksums. The data indicated by the black line was stored in $k = 100$ separate containers. $m = 18$ checksum vectors were created to protect the content of the containers. The $k_{lost} = 20$ first containers are removed, i.e. 11% more data vectors are lost than checksum code vectors created. 70
33 The original version of the 16 images used to demonstrate the method outlined. Each image is 512x512 pixels, and stored in separate data containers that are assumed to be failure prone. The images are taken from the The USC-SIPI Image Database[73]. In figures 34 and 35, the partial information recovery procedure is demonstrated when removing 4 and 6 images respectively. 72
34 Three checksums were computed to protect the content of the 16 containers. (a) Shows four images removed. (b) Depicts the recovered images. 73
35 Three checksums were computed to protect the content of the 16 containers. (a) Shows six images removed. (b) Depicts the recovered images. 74

Abbreviations

AMR	Adaptive Mesh Refinement
CFD	Computational Fluid Dynamics
CFL	Courant-Friedrichs-Lewy (condition)
CG	Continuous Galerkin
DCT	Discrete Cosine Transform
DMD	Dynamic Mode Decomposition
DWT	Discrete Wavelet Transform
DNS	Direct Numerical Simulation
FLOPS	Floating-Point Operations
GL	Gauss-Legendre (quadrature rules)
GLL	Gauss-Lobatto-Legendre (quadrature rules)
GMRES	Generalised Minimal Residual Method
HDG	Hybridisable Discontinuous Galerkin
HEVC	High Efficiency Video Coding (Standard)
HPC	High Performance Computing
I/O	Input/Output
JPEG	Joint Photographic Experts Group
Ma	Mach number
PCG	Preconditioned Conjugate Gradient
PDE	Partial Differential Equation
POD	Proper Orthogonal Decomposition
PSNR	Peak Signal-to-Noise Ratio
Re	Reynolds number
ROI	Regions Of Interest
SA-DWT	Shape-Adaptive Discrete Wavelet Transform
SEM	Spectral Element Method
SPD	Symmetric Positive Definite
WP	Work Package

1 Introduction

Computational fluid dynamics (CFD) is an application area that can greatly benefit from the use of exascale computing. In theory, we can use the massive computational power than exascale platforms will offer to observe the many scales of flow physics that govern important problems, such as the transition to turbulence across the wing of a commercial airliner. In practice, however, making efficient fluid dynamics codes that can effectively, efficiently and reliably utilise exascale resources is a significant challenge. A key issue in this is the lack of massively parallel algorithms for fluid dynamics code in a wide range of areas including adaptivity and error control, scalability, the amount of data generated at exascale and consideration of fault tolerance.

Across the last 36 months, ExaFLOW partners in work package 1 (WP1) have been working to develop new algorithms to tackle this challenge and enable fluid dynamics solvers to make use of exascale systems. We have been developing algorithms in several of these areas to address these key challenges:

- *Mesh quality and grid adaptivity*
  Here, we focus on the challenge of developing scalable adaptive methods, where error estimators drive an adaption process in order to make highly efficient use of large-scale computational resources without *a priori* knowledge of the flow solution.

- *Error control for heterogeneous modelling*
  By considering that different regions of the flow may be modelled with different approaches, we can reduce computational cost and increase scalability. In particular, we have been investigating the interfaces between two modelling zones, as well as ensuring that the distribution of work across nodes is regulated according to the variation of flow scales.

- *Mixed CG-HDG formulation*
  This work has investigated how to improve the scalability of state-of-the-art spectral element methods and make them suitable for exascale computations by developing a new mixed formulation based on continuous Galerkin (CG) and hybridisable discontinuous Galerkin (HDG) discretisations, where each node performs a computationally efficient CG solve, and combines this with a HDG system between nodes to minimize communication costs.

- *Data reduction*
  Simulations at exascale will produce huge amounts of raw data, which poses a significant problem for post-processing and data analysis. We have been investigating methods to reduce the amount of data that must be transferred from memory to disks by using filters for structure extraction and data reduction, i.e. transforming the large "raw" data to feature- or structure-based data which are orders of magnitude more compact.

- *Fault tolerance and resilience*
  This work focuses on the development of fault tolerant algorithms to ensure resilience to hardware faults. Activities will address the development of suitable in-situ models and strategies for detection and mitigation of hardware faults.

This deliverable, in particular, summarises the work achieved in these areas across the last 18 months since the previous deliverable D1.2.

2 Error control and mesh refinement

A central theme of work package 1 has been the development of adaptive methods that can automatically refine or coarsen a domain dependent upon the flow physics. In order to drive this adaptive process, we require two processes: techniques for automatically determining where the domain requires additional resolution or errors in the solution field, and the adaptive process that is used to apply this resolution in a large-scale simulation. In this section, we consider error indicators and estimators for both the spectral element method (particularly that used by Nek5000) and finite difference methods (for use in SBLI/OpenSBLI). We then discuss how these are applied in the context of adaptive mesh refinement for Nek5000.

2.1 Spectral error indicators

The first type of error detection we consider is the *spectral error indicator*, which uses the decay of an energy spectrum in order to determine whether the solution is underresolved in a given area. The following sections outline this in the context of spectral elements and finite difference methods.

2.1.1 Spectral element discretisations

The spectral error indicators for the spectral element method are based on a method developed by C. Mavriplis [55] and have been presented in detail in the deliverable D1.2. As a reminder, these indicators are based on the knowledge of the spectral expansion of the solution and give an indication of the L^2-norm of the error on the solution. They are easy to compute and induce a low overhead but have a tendency to refine regions of the mesh that might be irrelevant for computations engineering interest. These indicators have been chosen as a reference against which to compare the adjoint error estimators.

2.1.2 Finite difference discretisations

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. For the initial assessment of the error indicator algorithms, results from a compressible NACA-0012 wing profile were considered. This was presented at the ParCFD 2017 conference (see [38] for more details). For the flagship calculation presented here, we are using a low-drag

R/c	W/c	L_z/c	$N_{\xi,2}$	$N_{\eta,2}$	$N_{\xi,1/3}$	$N_{\eta,1/3}$	N_z	N_{total}
7.5	6.0	0.05	2095	999	999	1023	50	2.07×10^8
7.5	6.0	0.05	3045	999	1999	1023	150	1.07×10^9

Table 1: Numerical grid details for the grid before (first row) and after refinement (second row), detailed by [39].

compressible flow V2C profile for which the data was readily available and PRACE resources were awarded for a project aimed at finding the limits of buffetting for this airfoil. The V2C results described here have been published in Computers & Fluids (see [39]).

The metric upon which we will evaluate improvements is the reduction in the number of superfluous grid points through the focussing of resolution with the aid of the error indicators. In order to show such a reduction, two grids were considered: a relatively coarse grid and a refined version which focussed resolution in areas of high error severity determined by the indicators. Table 1 gives the dimensions and number of grid points used.

The airfoil incidence α was set to 4°, the Reynolds number based on the aerofoil chord Re_c was 5×10^5 (compared to 5×10^4 for the NACA-0012 case), and Mach number M was 0.7 (compared to 0.4 for the NACA-0012 case). The computational domain is composed of three blocks, as can be seen in Figure 1. Block 2 is a C-type structured grid fitted around the airfoil surface; it interfaces with the structured Cartesian blocks 1 and 3, which resolve the wake of the airfoil.

The simulation was performed using the legacy Fortran-based SBLI code. The error indicators were implemented in a program whose code was mostly generated using the OpenSBLI model development framework [37]. The solution fields stored in an HDF5 file were read in by the program, and the error severity values at each error block were computed. Each error block comprised 16^3 grid points. The z-component of vorticity was used to compute the error severity.

The error severity values for the integer-based measure I_i (see D1.2 for details) are shown in Figure 2 for both grids, and were computed once the flow became fully developed. In both cases the uniform flow away from the aerofoil is mostly well-resolved as suggested by I_i values of 0 or 1. However, in the coarse grid case high error severity can be found near the trailing edge where eddy shedding occurs and where pressure waves are present. This suggests more resolution would be required in these areas to adequately resolve the wake and potentially less elsewhere, which facilitates the highly efficient simulation of large-scale industry-relevant problems. Indeed, refining the grid clearly reduced the error in these regions. The persisting small cluster of high error severity at the leading edge in both grids is due to a very thin boundary layer, which was later treated by further grid refinement and localised filtering.

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. The error indicators code runs on various architectures (CPUs and GPUs) and is inherently scalable to large

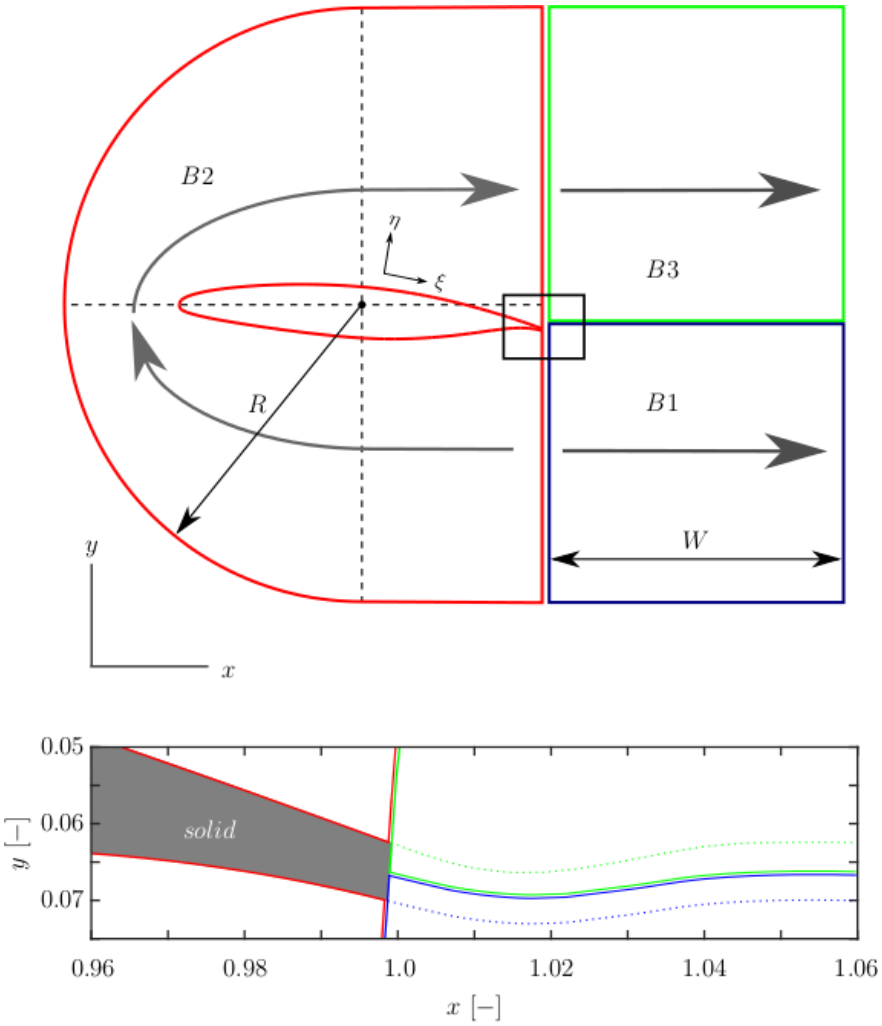


Figure 1: Computational domain arrangement. Image by [39].

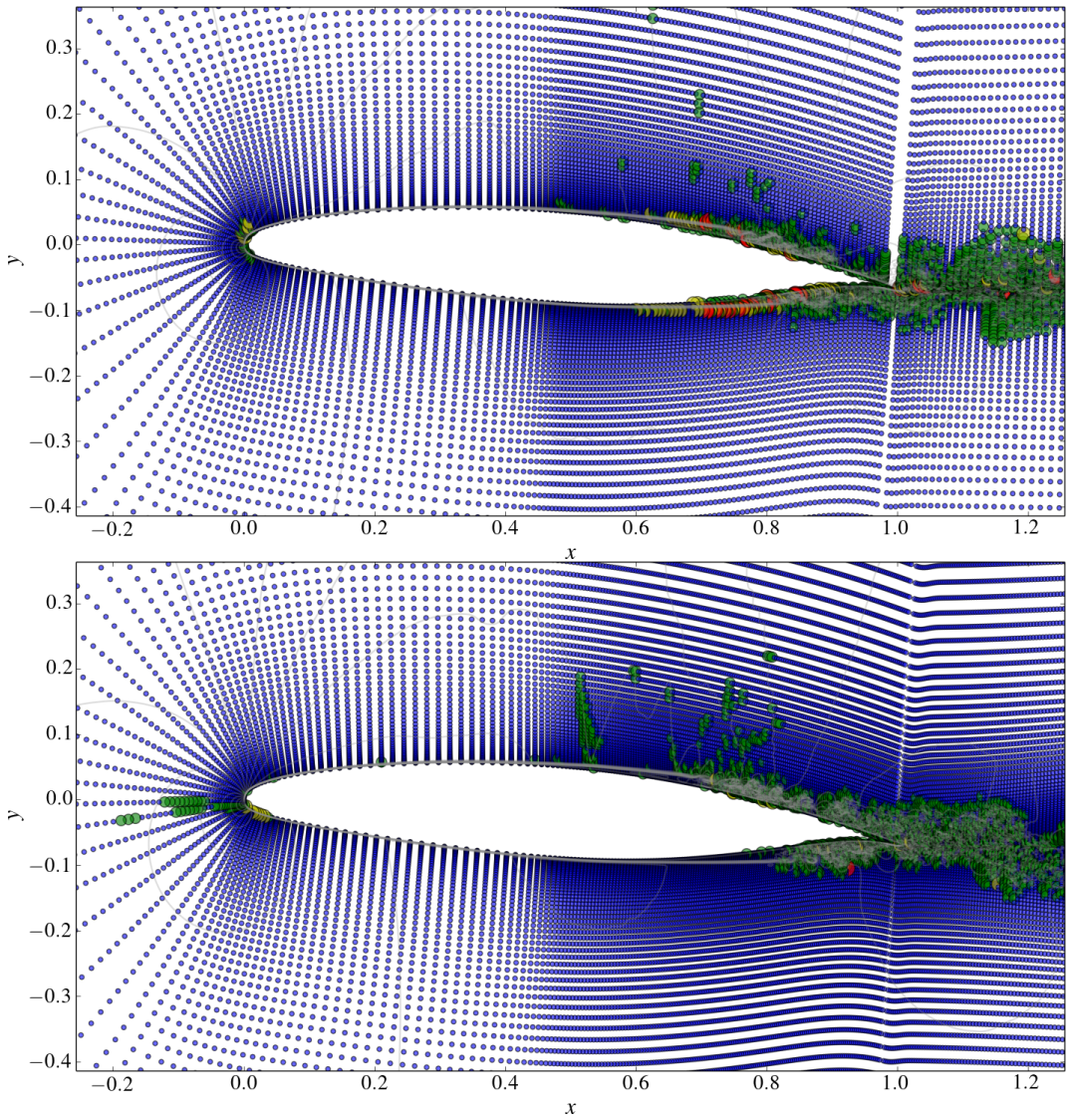


Figure 2: Contours of velocity magnitude in a two-dimensional slice (in the x–y plane at $z = 0$) from a V2C aerofoil simulation. Circles filled with blue, green, yellow and red indicate I_i error severity values of 0, 1, 2 and 3, respectively. The top and bottom images are for the coarse and refined grids, respectively. Images by [39].

cases. The parallel efficiency of the OpenSBLI code on the UK supercomputing facility at EPCC (ARCHER CPUs), the US supercomputing facility TITAN (CPUs and GPUs), and the Cambridge supercomputing facility (latest GPUs) is documented in the WP2 deliverable 2.4. The grid generation approach is documneted in the WP3 deliverable 3.3.

2.2 Adjoint error estimators

Contrary to spectral error indicators, adjoint error estimators are goal–oriented and aim at better computing a specific target value. We present the method for computing the adjoint error estimators for the spectral element method, following the work in [4]. So far, only the steady formulation has been considered. The method combines local information on the solution with a global adjoint solution to the problem and gives an indication on where to optimally refine the mesh to reduce the error on the functional.

2.2.1 Weak form of the steady Navier–Stokes equations

We start from the expression of the steady Navier–Stokes equations in the strong form

$$
\begin{aligned}
(\boldsymbol{u}\cdot\nabla)\boldsymbol{u} + \nabla p - \frac{1}{Re}\Delta\boldsymbol{u} &= \boldsymbol{f}, && (1)\\
\nabla\cdot\boldsymbol{u} &= 0, && (2)\\
\boldsymbol{u}|_{\Gamma_D} &= \boldsymbol{g}_D, && (3)\\
\boldsymbol{n}\cdot(\nabla\boldsymbol{u} - p\boldsymbol{I})\,|_{\Gamma_o} &= \boldsymbol{g}_o, && (4)
\end{aligned}
$$

where the velocity field $\boldsymbol{u}(\boldsymbol{x}) = (u, v, w)$ and the pressure $p(\boldsymbol{x})$ are defined on some domain $\boldsymbol{x} \in \Omega$. The boundary of the domain $\partial\Omega$ is split into Γ_D and Γ_o corresponding to Dirichlet and outflow boundary conditions. Then, we express the corresponding weak form of the set of equations as

Find $\hat{\boldsymbol{u}} = (\boldsymbol{u}, p), \boldsymbol{u} \in X, p \in Z$ *s.t.*

$$
A(\hat{\boldsymbol{u}})(\hat{\boldsymbol{\psi}}) = 0 \qquad \forall\hat{\boldsymbol{\psi}} = (\boldsymbol{\psi}, \chi),\ \boldsymbol{\psi} \in X_0, \chi \in Z \tag{5}
$$

where

$$
A(\hat{\boldsymbol{u}})(\hat{\boldsymbol{\psi}}) = \left\{ \begin{array}{l} ((\boldsymbol{u}\cdot\nabla)\boldsymbol{u}, \boldsymbol{\psi}) - (\nabla\cdot\boldsymbol{\psi}, p) + \frac{1}{Re}(\nabla\boldsymbol{u}, \nabla\boldsymbol{\psi}) - (\boldsymbol{f}, \boldsymbol{\psi}) \\ (\nabla\cdot\boldsymbol{u}, \chi) \end{array} \right\} = 0. \tag{6}
$$

The spaces X and Z are appropriate spaces satisfying the boundary conditions [20]. The notation $(\boldsymbol{a}, \boldsymbol{b})$ denotes the inner product $\int_\Omega \boldsymbol{a}\cdot\boldsymbol{b}\,\mathrm{d}\boldsymbol{x}$ between two vector fields $\boldsymbol{a}$ and $\boldsymbol{b}$.

2.2.2 Expression of the functional

As mentioned before, adjoint error estimators are based on the computation of a functional of interest. We consider the following general form for the functional:

$$
J(\hat{\boldsymbol{u}}) = \int_\Omega \hat{\boldsymbol{u}}\cdot\boldsymbol{j}_\Omega\,\mathrm{d}V + \int_{\Gamma_D}\left(\frac{1}{Re}\nabla_{\boldsymbol{n}}\boldsymbol{u} - p\boldsymbol{n}\right)\cdot\boldsymbol{j}_{\Gamma_D}\,\mathrm{d}s + \int_{\Gamma_o}\boldsymbol{u}\cdot\boldsymbol{j}_{\Gamma_o}\,\mathrm{d}s, \tag{7}
$$

where $\boldsymbol{j_\Omega} = (\boldsymbol{j}_{\Omega,\boldsymbol{u}}, \boldsymbol{j}_{\Omega,p})$, $\boldsymbol{j}_{\Gamma_D}$ and $\boldsymbol{j}_{\Gamma_o}$ are some coefficients. In the case of the drag on a body in a flow for example, $\boldsymbol{j}_\Omega = \boldsymbol{0}$, $\boldsymbol{j}_{\Gamma_o} = \boldsymbol{0}$ and $\boldsymbol{j}_{\Gamma_D}$ is a unit vector oriented parallel to the bulk flow, defined on the body of interest, zero otherwise.

2.2.3 Lagrange optimisation

To estimate the error, the following Lagrangian operator is defined as a function of the adjoint solution $\hat{\boldsymbol{u}}^\dagger = (\boldsymbol{u}^\dagger, p^\dagger)$

$$\mathcal{L}(\hat{\boldsymbol{u}}, \hat{\boldsymbol{u}}^\dagger) = J(\hat{\boldsymbol{u}}) - A(\hat{\boldsymbol{u}})(\hat{\boldsymbol{u}}^\dagger) \tag{8}$$

and its stationary points are found by solving the problem [4]

$$\mathcal{L}'(\hat{\boldsymbol{u}}, \hat{\boldsymbol{u}}^\dagger)(\hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\psi}}) = \left\{ \begin{array}{l} J'(\hat{\boldsymbol{u}})(\hat{\boldsymbol{\phi}}) - A'(\hat{\boldsymbol{u}})(\hat{\boldsymbol{\phi}}, \hat{\boldsymbol{u}}^\dagger) \\ -A(\hat{\boldsymbol{u}})(\hat{\boldsymbol{\psi}}) \end{array} \right\} = 0 \qquad \forall \{\hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\psi}}\}. \tag{9}$$

The various terms appearing are

- $J'(\hat{\boldsymbol{u}})(\hat{\boldsymbol{\phi}})$, the derivative of $J(\hat{\boldsymbol{u}})$ with respect to $\hat{\boldsymbol{u}}$ (*i.e.* the linearisation of the functional around a solution $\hat{\boldsymbol{u}}$),
- $A'(\hat{\boldsymbol{u}})(\hat{\boldsymbol{\phi}}, \hat{\boldsymbol{u}}^\dagger)$, the derivative of $A(\hat{\boldsymbol{u}})(\hat{\boldsymbol{u}}^\dagger)$ with respect to $\hat{\boldsymbol{u}}$ (*i.e.* the linearised adjoint Navier–Stokes equations in weak form around a solution $\hat{\boldsymbol{u}}$),
- $A(\hat{\boldsymbol{u}})(\hat{\boldsymbol{\psi}})$, the original problem in weak form.

2.2.4 Adjoint equations

Using integration by parts, it is possible to derive the following adjoint problem around a solution $\hat{\boldsymbol{u}}$ associated to the functional $J(\hat{\boldsymbol{u}})$ as

$$-(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}^\dagger + \boldsymbol{u}^T\boldsymbol{u}^\dagger - \nabla p^\dagger - \frac{1}{Re}\Delta\boldsymbol{u}^\dagger = \boldsymbol{j}_{\Omega,\boldsymbol{u}}, \tag{10}$$

$$-\nabla\cdot\boldsymbol{u}^\dagger = \boldsymbol{j}_{\Omega,p}, \tag{11}$$

$$\boldsymbol{u}^\dagger|_{\Gamma_D} = \boldsymbol{j}_{\Gamma_D}, \tag{12}$$

$$\left(\frac{1}{Re}\nabla_{\boldsymbol{n}}\boldsymbol{u}^\dagger + p^\dagger\boldsymbol{n} + (\boldsymbol{u}\cdot\boldsymbol{n})\boldsymbol{u}^\dagger\right)|_{\Gamma_o} = \boldsymbol{j}_{\Gamma_o}. \tag{13}$$

An analysis of the consistency of the adjoint solution and the extension to non-linear functionals are discussed in [33].

2.2.5 Error on the functional

If the Galerkin approximations $\hat{\boldsymbol{u}}_h$ and $\hat{\boldsymbol{u}}_h^\dagger$ are used instead of the exact solutions, Rannacher and Bangerth [4] prove that the error on the functional can be approximated by

$$J(\hat{\boldsymbol{u}}) - J(\hat{\boldsymbol{u}}_h) = \delta J \approx \delta J_{\text{est}} = A(\hat{\boldsymbol{u}}_h)(\hat{\boldsymbol{u}}^\dagger - \hat{\boldsymbol{u}}_h^\dagger), \tag{14}$$

where δJ_{est} denotes the estimated error on the functional of interest. This expression is valid in the general case of the Galerkin approximation and should be specified in more details for the case of a SEM solution.

2.2.6 Contributions to the error

The approximate solutions of polynomial order N, denoted $\hat{\boldsymbol{u}}_N = (\boldsymbol{u}_N, p_N)$ and $\hat{\boldsymbol{u}}_N^\dagger = \left(\boldsymbol{u}_N^\dagger, p_N^\dagger\right)$, arising from the SEM discretisation are now introduced into the expression of the error. Assuming that the domain Ω is split in E subdomains Ω_e with boundaries Γ_e, the error δJ_{est} can be computed as a sum over all the elements as

$$\begin{aligned}
\delta J_{\text{est}} &= A(\hat{\boldsymbol{u}}_N)(\hat{\boldsymbol{u}}^\dagger - \hat{\boldsymbol{u}}_N^\dagger) && (15)\\
&= \sum_{e=1}^{E} \left((\boldsymbol{u}_N \cdot \nabla)\boldsymbol{u}_N, \boldsymbol{u}^\dagger - \boldsymbol{u}_N^\dagger\right)_{\Omega_e} - \left(\nabla \cdot (\boldsymbol{u}^\dagger - \boldsymbol{u}_N^\dagger), p_N\right)_{\Omega_e} \\
&\quad + \frac{1}{Re}\left(\nabla \boldsymbol{u}_N, \nabla(\boldsymbol{u}^\dagger - \boldsymbol{u}_N^\dagger)\right)_{\Omega_e} \\
&\quad - \left(\boldsymbol{f}, \boldsymbol{u}^\dagger - \boldsymbol{u}_N^\dagger\right)_{\Omega_e} + \left(\nabla \cdot \boldsymbol{u}_N, p^\dagger - p_N^\dagger\right)_{\Omega_e}. && (16)
\end{aligned}$$

This is then integrated by parts

$$\begin{aligned}
\delta J_{\text{est}} = \sum_{e=1}^{E} & \left(\underbrace{(\boldsymbol{u}_N \cdot \nabla)\,\boldsymbol{u}_N - \frac{1}{Re}\Delta \boldsymbol{u}_N + \nabla p_N - \boldsymbol{f}_N}_{R_1(\hat{\boldsymbol{u}})}, \boldsymbol{u}^\dagger - \boldsymbol{u}_N^\dagger \right)_{\Omega_e} \\
+ & \left(\underbrace{\frac{1}{Re}\nabla_{\boldsymbol{n}} \boldsymbol{u}_N - p_N \boldsymbol{n}}_{R_2(\hat{\boldsymbol{u}})}, \boldsymbol{u}^\dagger - \boldsymbol{u}_N^\dagger \right)_{\Gamma_e} \\
+ & \left(\underbrace{\nabla \cdot \boldsymbol{u}_N}_{R_3(\hat{\boldsymbol{u}})}, p^\dagger - p_N^\dagger \right)_{\Omega_e}. \qquad (17)
\end{aligned}$$

The quantities $R_1(\hat{\boldsymbol{u}})$ and $R_3(\hat{\boldsymbol{u}})$ are the strong cell residuals of the momentum and continuity equations, respectively, while $R_2(\hat{\boldsymbol{u}})$ represents the normal stresses along the boundary of each element. The second term can be rewritten by noting that the normal vectors $\boldsymbol{n}$ along the shared edge between two adjacent cells have opposite signs and by attributing half of the jump to each element. Therefore, the face residuals $r(\hat{\boldsymbol{u}})|_\Gamma$ along the edge Γ can be seen as $\frac{1}{2}[R_2]$, half the jump of R_2 across the edge, such that the residuals are rewritten as

$$r(\hat{\boldsymbol{u}})|_\Gamma = \begin{cases} \frac{1}{2}[-\frac{1}{Re}\nabla_{\boldsymbol{n}} \boldsymbol{u}_N + p_N \boldsymbol{n}] & \text{if } \Gamma \subset \Gamma_e \backslash \partial\Omega, \\ 0 & \text{if } \Gamma \subset \partial\Gamma_D, \\ \frac{1}{Re}\nabla_{\boldsymbol{n}} \boldsymbol{u}_N - p_N \boldsymbol{n} & \text{if } \Gamma \subset \partial\Gamma_o. \end{cases} \qquad (18)$$

Consequently, the error on the functional is bounded by

$$\begin{aligned}
\delta J_{\text{est}} \leq & \sum_{e=1}^{E} \underbrace{\left\|R_1(\hat{\boldsymbol{u}})\right\|_{L^2(\Omega_e)}}_{\rho_{1,e}} \underbrace{\left\|\boldsymbol{u}^\dagger - \boldsymbol{u}_N^\dagger\right\|_{L^2(\Omega_e)}}_{\omega_{1,e}} \\
& + \underbrace{\left\|r(\hat{\boldsymbol{u}})\right\|_{L^2(\Gamma_e)}}_{\rho_{2,e}} \underbrace{\left\|\boldsymbol{u}^\dagger - \boldsymbol{u}_N^\dagger\right\|_{L^2(\Gamma_e)}}_{\omega_{2,e}} \\
& + \underbrace{\left\|R_3(\hat{\boldsymbol{u}})\right\|_{L^2(\Omega_e)}}_{\rho_{3,e}} \underbrace{\left\|p^\dagger - p_N^\dagger\right\|_{L^2(\Omega_e)}}_{\omega_{3,e}} \qquad (19) \\
\leq & \sum_{e=1}^{E} \rho_{1,e}\omega_{1,e} + \rho_{2,e}\omega_{2,e} + \rho_{3,e}\omega_{3,e}. \qquad (20)
\end{aligned}$$

Two categories of terms appear in Eq. (20). The residuals $\rho_{i,e}$, which are a measure of the strong residuals of the solution and $\omega_{i,e}$, or adjoint weights, a measure of the interpolation error on the adjoint solution in the L^2-norm. To start with the computation of the $\rho_{i,e}$ terms, we note that they are already available from the solver. Presuming continuous and smooth data and also that the solution is spectrally accurate, the boundary terms and associated jumps are ignored (*i.e.* the terms $\rho_{2,e}$). The remainders $R_1(\hat{\boldsymbol{u}})$ and $R_3(\hat{\boldsymbol{u}})$ can be computed in a straightforward fashion by reusing the operators available in the solver.

For high Reynolds numbers, it is worth noting that the jump in the viscous stresses becomes negligible. However, since the $\mathbb{P}_N\mathbb{P}_{N-2}$ version of Nek5000 is used for the simulations, the pressure is not continuous at the interface and the term cannot be completely dropped out a priori. However, the pressure discontinuity is also comparably small and therefore it is expected that this term does not constitute the main source of error, as will be shown by the a posteriori analysis further down.

The interpolation errors $\omega_{1,e}$ and $\omega_{3,e}$ can be estimated by the a priori estimate of the interpolation error [20]

$$\left\|\boldsymbol{u} - \boldsymbol{u}_N\right\|_{L^2(\Omega)} \leq C N^{1/2} N^{-m} \left\|\boldsymbol{u}\right\|_{H^m(\Omega)}, \qquad (21)$$

where C is a constant independent of N and m is the order of the H-norm. In practice, what is computed instead is

$$\left\|\boldsymbol{u}^\dagger - \boldsymbol{u}_N^\dagger\right\|_{L^2(\Omega_e)} \lesssim C^e N^{1/2} N^{-m} \left\|\boldsymbol{u}_N^\dagger\right\|_{H^m(\Omega_e)}, \qquad (22)$$

where it is assumed that the interpolation constant C^e is different for each element. Both the H-norm for orders $m = 0$ and $m = 1$ have been implemented but only the case $m = 1$ has been used for the results that follow.

The choice of C^e, the interpolation constant on element e, is capital to obtain a good estimate of the interpolation error and a tight bound on the adjoint error estimator. The actual value of the constant is difficult to determine, but it should be dependent on properties of the element it is evaluated on. Therefore, it is assumed that the constant is

$$C^e = (vol_e)^{\frac{1}{d}}, \qquad (23)$$

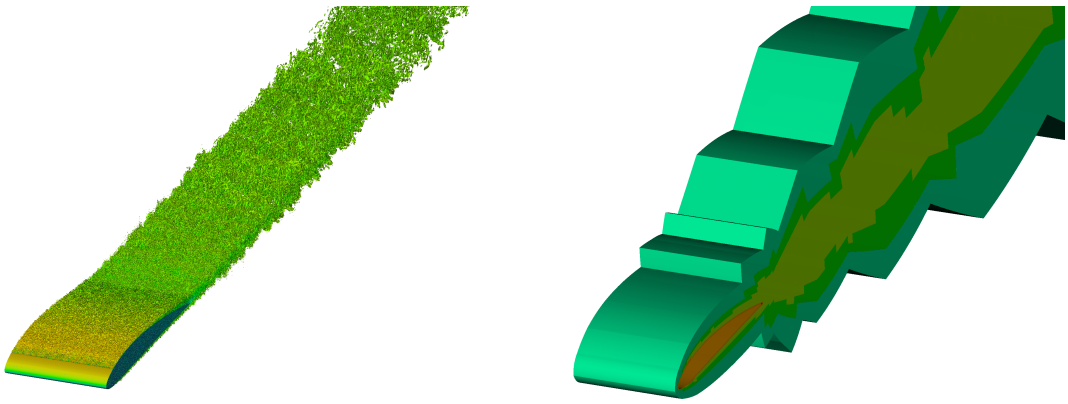


Figure 3: Vortical structures (λ_2 criterion) of the velocity field around the wing (left panel) and the part of the domain covered by refinement levels higher and equal 2 (right panel) at simulation time 7.2 for $Re_c = 2 * 10^5$ case.

where vol_e is the volume of element e (area in 2D) and d is the dimension of the problem. One reference [4] mentions that the interpolation constant is typically of size 0.2 in the case of the FEM. A more detailed discussion on the choice of the interpolation constant is presented by [40], where some indications on how to estimate the constant are given depending on the norm considered and the type of element in the case of the finite element method.

2.3 Adaptive mesh refinement

In this section we describe the progress we have made in adapting Spectral Element Method (SEM) to h-type Adaptive Mesh Refinement (AMR) framework, which we consider to be an important component of the future exascale CFD solvers. The algorithms developed in this work package were implemented in SEM solver *Nek5000* and tested within work packages 2 and 3, respectively. The result of our work is fully functional nonconforming incompressible Navier-Stokes equations solver that estimates computational error and accordingly modifies the computational mesh during the simulation. This method was used to perform one of the ExaFLOW flagship simulations: the fully adaptive 3D simulation of the turbulent flow around NACA 4412 wing profile at Reynolds number Re_c equal $2 * 10^5$ and $1 * 10^6$. Fig. 3 presents initial results of $Re_c = 2 * 10^5$ case. Left and right panels show vortical structures of the velocity field around the wing and the part of the domain covered by refinement levels higher and equal 2 at simulation time 7.2, respectively. These simulations are discussed in detail in WP3 deliverable D3.3.

Our main goal is to develop efficient tools for dynamic mesh modification and to adapt algorithms implemented in *Nek5000* to nonconforming meshes. This work was started with development of nonconforming advection-diffusion equation solver (EU project CRESTA) which was later extended to the incompressible Navier-Stokes problems within ExaFLOW. In the previous WP1 and WP2 deliverables D1.2 and D2.2, we described number of components of AMR framework, one of the components was the efficient pressure preconditioner. There

are three major modifications required to adapt *Nek5000* for nonconforming meshes:

- diagonalisation of the global mass matrix $Q^T B_L Q$,
- redefinition of the base functions for the assembly of the coarse-grid operator $b_i^T A^k b_j$ to neglect hanging nodes,
- redefinition of the direct stiffness summation operator QQ^T to include spectral interpolation J at the nonconforming faces and edges.

In following paragraphs we describe current development concerning pressure preconditioners, solver stabilisation and mesh partitioning.

2.3.1 Adaptation of the hybrid Schwarz-multigrid preconditioner to nonconforming meshes

Nonconforming pressure preconditioners are important because pressure operator is a main source of stiffness for incompressible flow simulations. There are two different pressure preconditioners implemented in *Nek5000* based on the additive overlapping Schwarz method [24, 26] and on the hybrid Schwarz-multigrid method [29, 53], respectively. The first one was described in detail in WP1 deliverable D1.2 and is based on combining solutions of the local Poisson problems in overlapping subdomains $R_k^T \hat{A}_k^{-1} R_k$ with the coarse grid problem $R_0^T \hat{A}_0^{-1} R_0$, which is solved on few degrees of freedom, but covers the entire domain

$$M^{-1} = R_0^T \hat{A}_0^{-1} R_0 + \sum_k R_k^T \hat{A}_k^{-1} R_k,$$

with R and R^T being the restriction and prolongation operators respectively. In the local problem, R_k and R_k^T transfer data to and from the subdomain, $\hat{A}_k$ is just a local stiffness matrix. On the other hand, the coarse grid problem is solved on the element vertices only with R_0^T being the operator interpolating the coarse grid solution onto the tensor product array of GL points in the reference element. We have to stress here the difference between local and coarse restriction/prolongation operators, which became important for nonconforming solver. The first one connects overlapping subdomains at the same resolution level, and the second one acts between different levels. To retain performance of the conforming solver, the nonconforming versions of the local and coarse operators have to execute different direct stiffness operators, i.e. $J_L Q Q^T J_L^{-1}$ and $J_L Q Q^T J_L^T$, respectively.

On the other hand, the hybrid Schwarz-multigrid prconditioner is based on the multiplicative Schwarz method, which for two levels scheme takes the form

$$M^{-1} = R_0^T \hat{A}_0^{-1} R_0 \left[\sum_k R_k^T \hat{A}_k^{-1} R_k \right],$$

and leads to the following two-level multigrid scheme:

$$\begin{aligned} i) \quad u^1 &= \sum_k R_k^T \hat{A}_k^{-1} R_k g, \\ ii) \quad r &= g - Au^1, \\ iii) \quad e &= R_0^T \hat{A}_0^{-1} R_0 r, \\ iv) \quad u &= u^1 + e, \end{aligned}$$

where g, r, e and u are right hand side, residual, coarse-grid error and solution of equation $Au = g$, respectively. This method can be extended to general multilevel solver performing full V cycle (see e.g. [26]). Notice that by replacing $ii)$ with $r = g$ we obtain the additive Schwarz preconditioner.

As both additive and multiplicative methods share number of features (e.g. coarse grid solver), their adaptation to nonconforming meshes is similar and consists of proper redefinition of operators at each multigrid level. Analogous to the additive Schwarz method we distinguish between Schwarz (acting at single level) and restriction (connecting different levels) operators, and apply different direct stiffness summation for each of them.

Unlike the additive preconditioner the hybrid one requires redefinition of the diagonal weight matrix as well, that indicates the number of sub-domains sharing a given node, and is used to accommodate overlapping regions. Its value is important as it reduces maximum eigenvalue of MA operator and defines smoothing properties of the additive Schwarz step (see [26] and references therein). In conforming case its definition is straightforward, however the nonconforming case is more problematic as hanging nodes are not real degrees of freedom. In the current implementation the information about node multiplicity on the nonconforming faces is hidden to the parent element, so the parent element sees only one neighbour instead of two (four in 3D). Although this choice gives preconditioner, that significantly reduces number of pressure iterations, its performance for the studied cases is slightly worse than performance of the additive Schwarz preconditioner. This can be caused by an nonoptimal value of the weight matrix, or by the fact that the hybrid preconditioner is superior over the additive one for high-aspect ratio elements, that are not present in our adaptive simulations. In the future we are going to investigate other definitions of the weight matrix. In addition we are going to investigate and adapt for AMR framework pressure preconditioners based on the restricted additive Schwarz method [12, 22].

2.3.2 Solver stabilisation based on high-pass filter

Stability of numerical methods is a challenging problem, especially for solvers with little numerical viscosity like *Nek5000*. That is why number of different stabilisation methods were developed and two of them are currently implemented in *Nek5000*. The original one is the filter-based approach [25], which is an explicit filter that suppresses short waves in the spectra of each element by directly decreasing their amplitude. Unfortunately, this violates divergence-free condition, so a new relaxation-term-based approach was developed [57]. In this case the damping term is added to the forcing that remedies some of the explicit filter

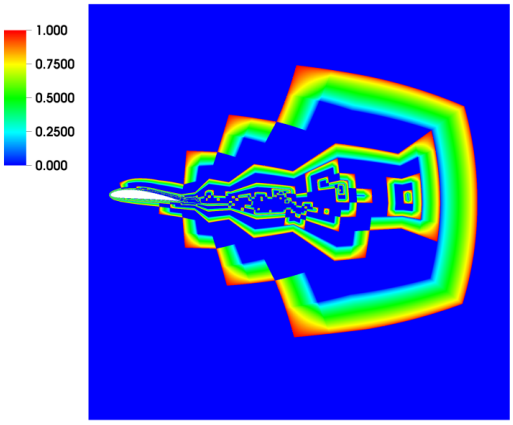


Figure 4: 1 Linear masking function restricting action of the high-pass filter to the nonconforming interfaces for 2D wing simulation.

drawbacks. We have to mention here that relaxation technique is a general method used for different purposes e.g. LES simulations [63].

Unfortunately, the stability of nonconforming solver is even more challenging than the conforming one. It is related to conforming-space/nonconforming-mesh approach [45, 28] we have adapted in our implementation. In this case functional space is continuous across nonconforming faces, that simplifies implementation as mortar elements can be replaced by simple interpolation operator, but in some specific situations it leads to instability. This instability can be observed if the lower resolution element (parent) is located downstream with respect to the high resolution region (children), and the advected field contains more information (smaller scale structures) than can be represented by the parent element. As the field values at the interface are set upwind by the low resolution parent, the interface acts as partially reflective boundary and leads to the oscillations. In most cases these oscillations remain small and are suppressed by AMR itself, as it would take care of the unresolved regions. However in some situations (e.g. user overwriting error indicator marks to additionally reduce resolution in not interesting regions) they can grow and even cause the simulation to crash. To avoid such a situation we added stabilisation mechanism that is supposed to partially diffuse small scale structures in the vicinity of the nonconforming interfaces. We achieve this using relaxation-term-based approach by adding high-pass filter to the forcing term:

$$\frac{du}{dt} = L(u) - \chi * m(r) * H(u),$$

where $L(u)$, χ, $m(r)$ and $H(u)$ are evolution operator, relaxation coefficient, masking function and transfer function removing half of high frequency modes, respectively.

The masking function is used to restrict forcing to the interior of the children elements touching nonconforming faces only. This function is equal to 1 at nonconforming interface,

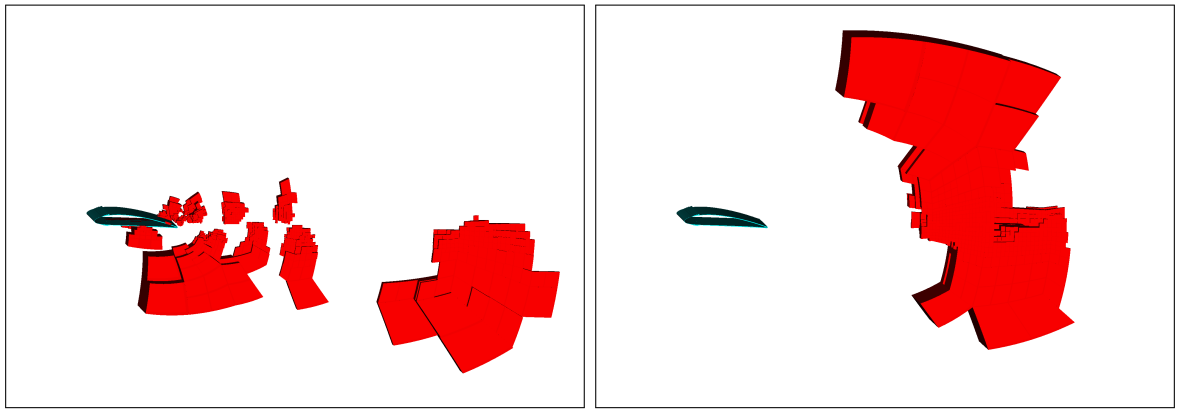


Figure 5: Domain decomposition for one-level (left) and two-level (right) partitioning in 3D wing simulation at $Re = 2 * 10^5$. In red is visualised a part of the domain residing on the second node (cores 32-64). The wing position is marked in blue.

it linearly decreases inside child element, and is 0 everywhere else as is presented in Fig. 4. Notice that $m(r)$ forms a set of closed loops at every interface between different refinement levels, as we do not distinguish between local inflow and outflow conditions. This however does not influence significantly the solution, as the flow entering high resolution region does not contain small scale features that would be damped. Moreover, the forcing is discontinuous at the nonconforming interface, as it is set to 0 inside all the parents, but this does not cause a problem, as forcing is not required to be continuous.

Efficiency of this method depends on the value of relaxation coefficient and setting its value is not straightforward. Too low value of χ would make filter inefficient, but on the other hand too strong forcing could remove important flow features. Based on our current experience we set $\chi = 0.5$, but more investigation of this problem is necessary. In the future we are going to drop assumption of conforming functional space and implement mortar elements in the solver that should increase stability.

2.3.3 Two-level partitioning

Our previous investigation have shown the grid partitioning to be an important constraint for AMR simulations. For advection-diffusion problems (results of CRESTA EU project) the main limitation came from the strong scaling limit of the library performing bisection itself (in this case ParMetis). Although *Nek5000* can efficiently scale up to a million MPI ranks with only a few elements (approximately 2000 grid points for BG Mira at Argonne National Laboratory) per single MPI rank, the calculations performed on Cray machine Lindgren at PDC (Sweden) showed that AMR simulation is dominated by graph bisection (spending more than 50% of computing time) if there was less than 50 elements per core with total 32000 MPI ranks.

In the case of incompressible Navier-Stokes equations the problem gets more complicated, as the pressure calculations require an expensive coarse grid solver, that sets the new strong

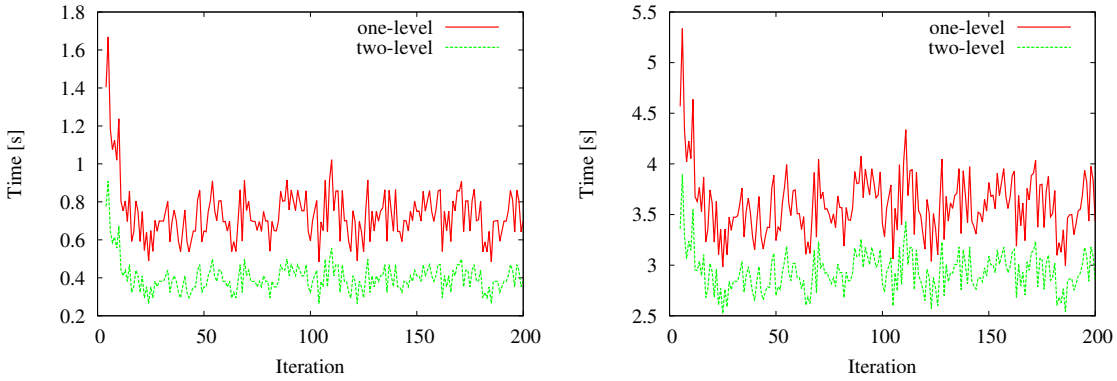


Figure 6: Time of pressure preconditioner execution (left) and time per time step (right) for one- and two-level partitioning. Results of a 3D wing simulation at $Re = 2 * 10^5$.

scaling limit [27]. There are two coarse grid solvers implemented in *Nek5000*: AMG and XXT. The ExaFLOW development related to the first one is described in the previous section, and in this section we focus on the second one. XXT [69] is based upon a (quasi-) sparse factorisation of the inverse of operator matrix and its performance is strongly dependent on domain decomposition. In addition, in the case of AMR we have to take into account the solver setup phase as well, as it is executed after each refinement phase. This setup phase can be very expensive and depending on the grid topology can take several minutes. That is why is it crucial to optimise mesh decomposition for XXT and to use grid partitioners based on graph bisection. Our tests revealed that a naive use of parallel graph partitioners such as the ParMetis library, where all MPI ranks take part in single step (one-level) partitioning, can lead to a suboptimal mesh distribution. We identified a possible cause of this being that most graph partitioners often do not take into account the distribution of the cores between the nodes, and the resulting partitioning is not continuous in space. The outcome is a mesh decomposition in which cores residing on single node contain grid parts corresponding to different graph branches, so the domain on single node is not continuous, as is visible on the left plot in Fig. 5. This has a strong impact on XXT setup and solver performance, as at each computational level it has to use slow inter-node communication. The optimal approach would be placing the whole graph branch on a single node and execute fast intra-node communication for some set of levels. We have to mention here that ParMetis gives such suboptimal partitions, even though we provide physical coordinates of each graph node.

The possible remedy to this problem could be proper renumbering of the graph nodes or keeping additional information about neighbouring domain subsets. However, in our case the node numbering is provided by grid manager (p4est library) and there is no simple way to alter it. That is why we have tested simple two-level partitioning, in which we split inter- and intra-node graph bisection. The first step is executed on all the MPI ranks at the same time with a single communicator, this provides a partitioning in which a whole graph branch is located on a single node (see left plot in Fig. 5). In the next step we generate in parallel on all the nodes independently, an intra-node graph and perform local graph bisection using only

fast intra-node communication. According to our observation, the intra-node partitioning is usually two orders of magnitude faster than the inter-node one. This method allows to optimise mesh decomposition and speed up XXT setup and solver phases, but is strongly dependent on the quality of the first partitioning. If the first step would place disjoint graph on a single node, the second step could fail.

We investigated the performance of one- and two-level partitioning using 3D turbulent wing case at $Re = 2 * 10^2$ with 224328 element on 512 cores distributed among 16 nodes. The partitioning time and element imbalance was similar for both cases and was equal to 1.2 sec and 34 elements versus 1.2 sec and 33 element for one- and two-level partitioning respectively. On the other hand there was a strong impact on XXT setup time, where the initial setup time of 765.2 sec. (one-level) was reduced to 76.9 sec. (two-level). We have to mention here that the gain on the 2048 was much higher, and the setup time was reduced by more than two orders of magnitude showing strong dependence of XXT scaling on mesh partitioning. The benefit for XXT solver phase is is less prominent, but still significant, as the pressure preconditioner time was reduced by almost 50% (left plot in Fig. 6), giving time per time-step shorter by 15% (right plot in Fig. 6).

2.4 Error control for heterogeneous modelling

In many applications, such as strongly separated flows or rough boundary layers, it may be desireable to solve the Navier-Stokes equations directly on a fine grid and model the turbulence in regions where one is confident with the accuracy of the chosen turbulent model.

The present work focuses on incompressible wall-bounded flows and combines direct numerical simulations (DNS) with one-point turbulence closures (RANS), to investigate the grid point requirement to perform simulations for industrial surfaces at higher Reynolds number. These methods are incorporated into a second-order accurate finite-differences on a staggered grid for the spatial discretisation and a second-order accurate Adams-Bashforth mothod for the time integration [10].

2.4.1 Methodology

Mean quantities are defined based on averaging out the dimensions x_i over which the solution is expected to be statistically homogeneous. In the case of the channel flow (studied here) these are x_1 and x_2, the stream- and span-wise directions, respectively.

Indicating averages by $\langle\rangle$ and mean quantities by the capitals, the equation to be solved is

$$\frac{\partial u_1}{\partial t} + \frac{\partial u_i u_j}{\partial x_j} - \beta\langle\frac{\partial u_i u_j}{\partial x_j}\rangle = \frac{\partial p}{\partial x_i} + \beta\langle\frac{\partial p}{\partial x_i}\rangle + \nu\frac{\partial^2 u_i}{\partial x_j \partial x_j} + G_i + \beta\frac{\partial}{\partial x_3}\left(\nu_t\frac{\partial U_i}{\partial x_3}\right), \qquad (24)$$

where β acts as a switch between the DNS and RANS regions, G_i is the force driving the flow and ν_t is the eddy viscosity.

Notice that when $\beta = 0$, equation 24 reduces to the classical incompressible Navier-Stokes equations while for $\beta = 1$ we obtain the RANS equation (simplified for the turbulent channel flow). Thus, the mean velocity U_1 ($U_2 = 0 = U_3$) satisfies

$$\frac{\partial U_1}{\partial t} + \overline{\beta}\langle\frac{\partial u_1' u_3'}{\partial x_3}\rangle + \beta\frac{\partial}{\partial x_3}\left(\nu_t\frac{\partial U_1}{\partial x_3}\right) = \nu\frac{\partial^2 U_1}{\partial x_3^2} + G_1 \tag{25}$$

with $\overline{\beta} = 1 - \beta$ and $()'$ represents the turbulent quantities.

Equations 24 and 25 ensure that in the region where the DNS is active the mean flow is driven by the correct Reynolds stresses while in regions where the DNS may be under-resolved the RANS solution drives the under-resolved DNS. Thus reducing the grid point requirement compared to full scale DNS.

2.4.2 Results

The hybrid DNS/RANS methodology is incorporated and tested for a turbulent channel flow and the mixing length hypothesis was used to determine ν_t used in the RANS equations. A domain of $2\pi\delta \times \pi\delta \times 2\delta$ is used in the stream-, span-wise and wall-normal directions, respectively, where δ is the channel's half-height. Periodic boundary conditions are applied in the streamwise and spanwise directions (x and z).

A fully resolved DNS of a turbulent channel flow is performed using $128 \times 128 \times 128$ grid points at a Reynolds number of 180 based on the friction velocity. The grid points are clustered near the walls using a hyperbolic tan function. Statistics are recorded for 100 flow through times after the transients. These are used to compare the results for the hybrid DNS/RANS approach.

Different approaches can be used to switch the solution from DNS to RANS and in this work three different forms are used to determine their impact on the solution. The first is a cutoff function approach in which a prescribed set value of β is used to switch between DNS and RANS, in the second and third approaches we follow a sinusoidal profile with half-periods $20, 40$ times the viscous length scale (δ_ν) respectively to switch.

Figure 7 shows the mean velocity profile in the wall-normal direciton using a cutoff function which varies from $0 - 180$ with 0 representing a full RANS simulation and 180 represents a full DNS. It should be noted that the same grid used in the DNS is used for these simulations. The cutoff function approach doesn't follow a simple trend, this might be due to sudden change from DNS to RANS and not all information on the small scale fluctuations are transferred to RANS through the eddy viscosity.

Similarly, Figures 8 and 9 shows the behaviour of mean velocity profiles in the wall-normal direction for the sinusoidal blending of the solution. The hybrid solutions become slightly distorted when β varies slowly and approach the DNS solution when the solution starts blending just above the buffer region.

In the final analysis, the effect of grid points on the mean velocity profile is studied for a hybrid simulation with β following a sinusoid of half-period $40\delta_\nu$. Figure 10, the mean velocity obtained with this methodology is rather close to the DNS solution, even when the grid is coarsened by a factor of 4 in the wall normal direction.

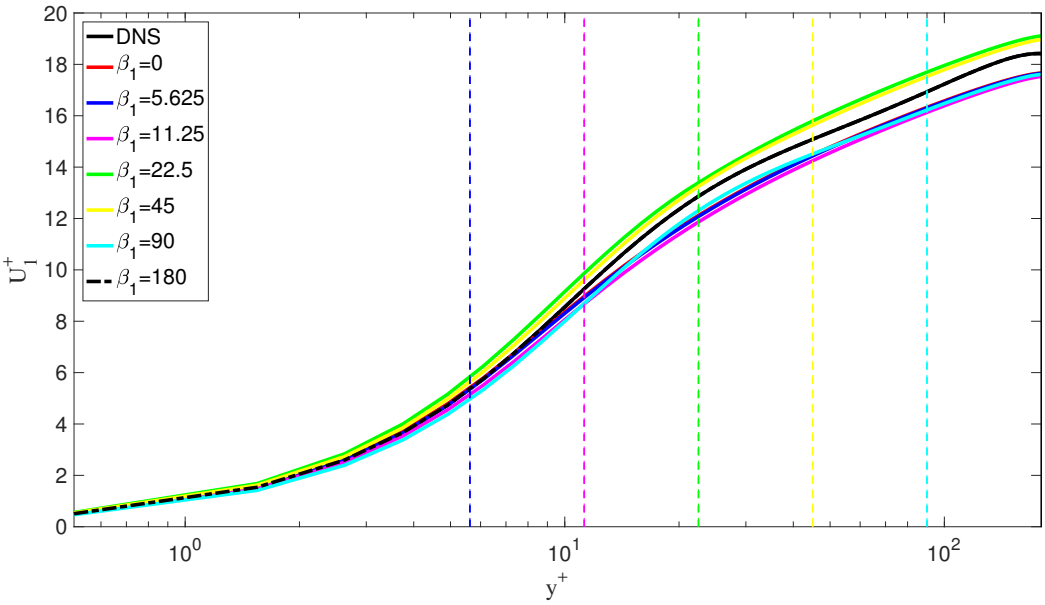


Figure 7: Mean velocity profiles using a cutoff β. The dashed lines indicate β_1 which is where β switches from 0 to 1.

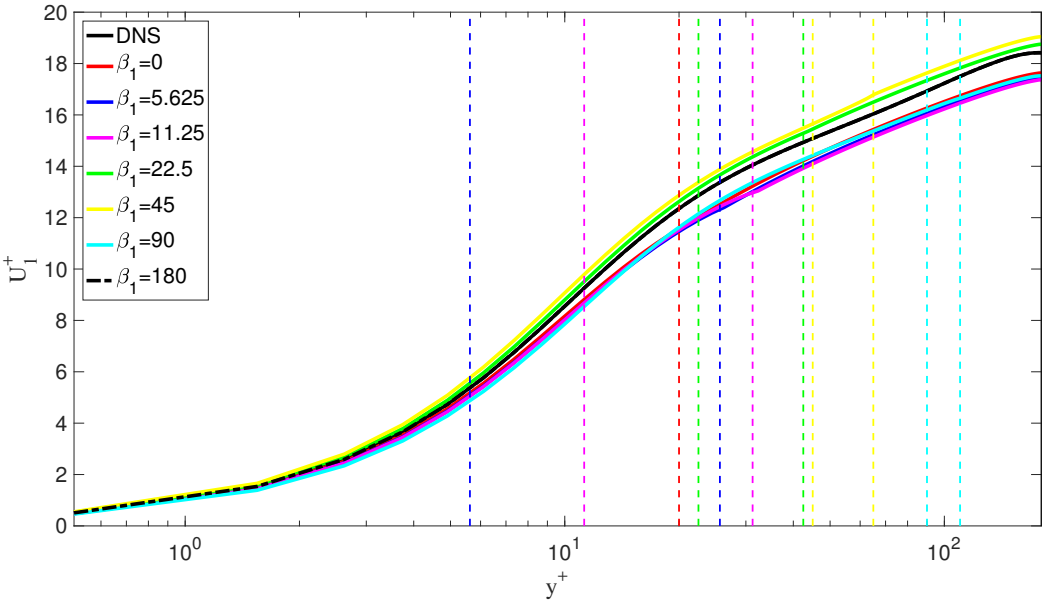


Figure 8: Mean velocity profiles using β following a sinusoid of half-period $20\delta_\nu$. The dashed lines indicate the bounds where β varies from 0 to 1. The location at which the sinusoid starts is identified by β_1.

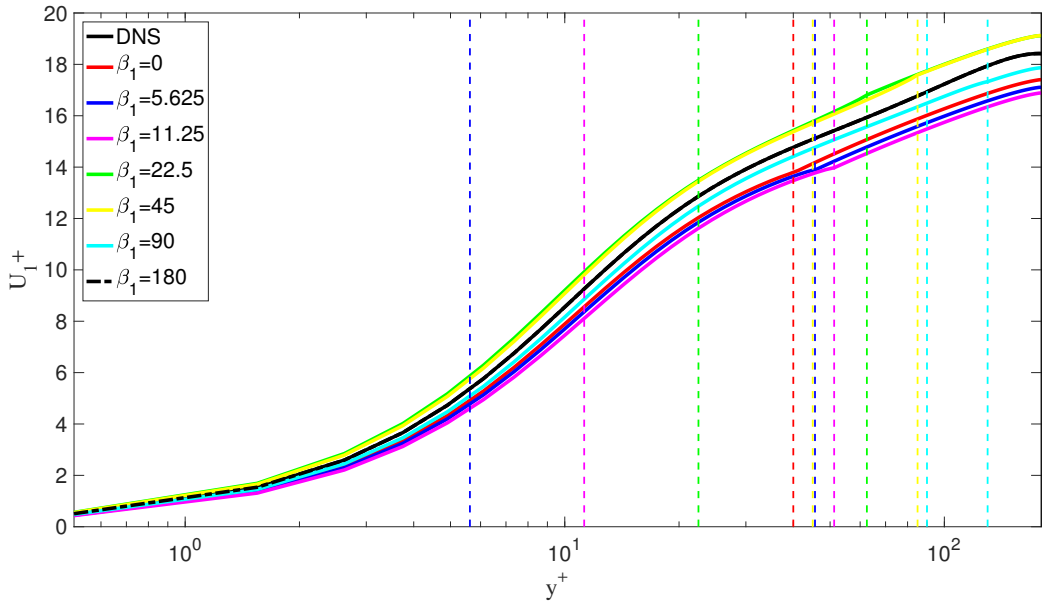


Figure 9: Mean velocity profiles using β following a sinusoid of half-period $40\delta_\nu$. The dashed lines indicate the bounds where β varies from 0 to 1. The location at which the sinusoid starts is identified by β_1.

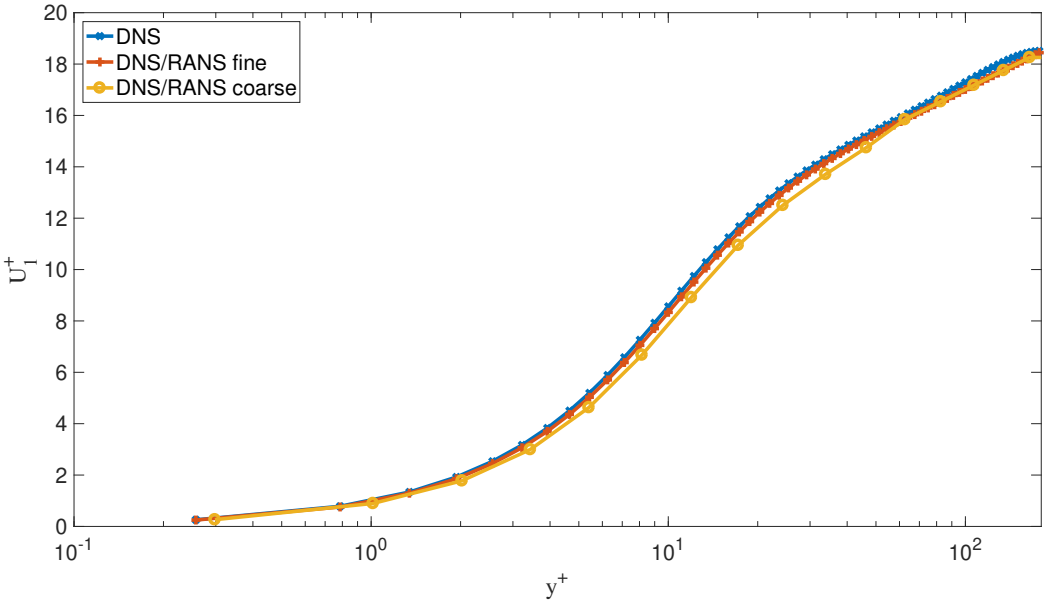


Figure 10: Mean velocity profiles obtained with DNS and blended DNS/RANS on a fine (identical to the DNS) and coarse (4 times fewer points in the wall normal direction) grids.

In the current simulations, the hybrid approach has used 25% of grid points to produce a result within 2% of the DNS results. Hybrid simulations can drastically reduce the grid point requirement for practical flows and replicating the flow physics at a reasonable cost.

3 CG-HDG formulation

3.1 Motivation

3.1.1 Weak boundary condititions as part of CG-HDG formulation

The combined CG-HDG formulation was motivated by our search for efficient algorithms for incompressible Navier-Stokes equations that employ high-order space discretization and a time splitting scheme. The cost of one step in time is largely determined by the amount of work needed to obtain the pressure field, which is defined as a solution to a scalar elliptic problem. Several Galerkin-type methods are available for this task, each of them have specific advantages and drawbacks.

High-order continuous Galerkin (CG) method is the oldest. Compared to its discontinuous counterparts, it involves a smaller number of unknowns (figure 11), especially in a low-order setting. The CG solution can be accelerated by means of static condensation, which produces a globally coupled system involving only those degrees of freedom on the mesh skeleton. The element interior unknowns are subsequently obtained from the mesh skeleton data by solving independent local problems that do not require any parallel communication. The amount of information interchanged while constructing and solving the

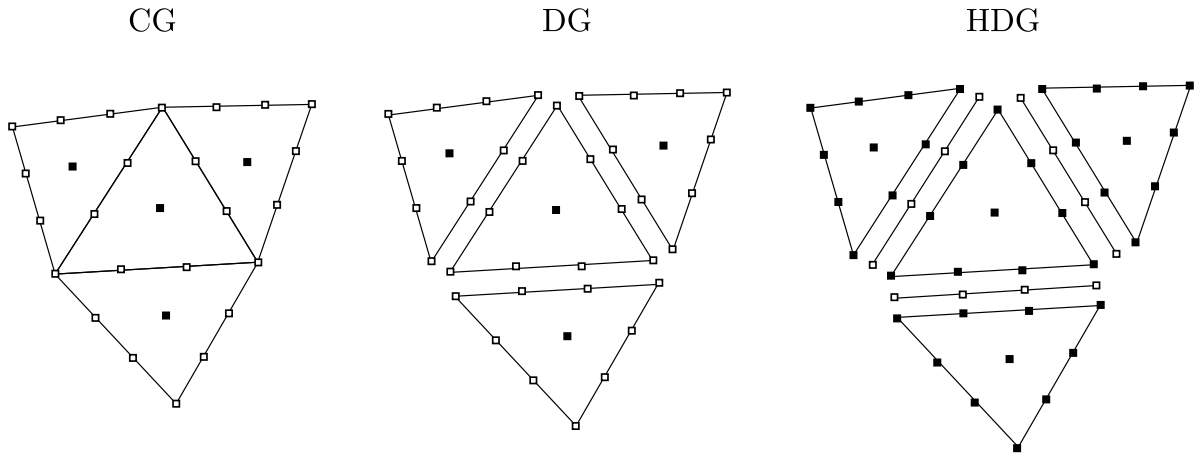


Figure 11: Distribution of unknowns for continuous and discontinuous Galerkin methods.

statically condensed system, however, is determined by the topology of the underlying grid. Unstructured mesh generators often produce meshes with high vertex valency (number of elements incident to given vertex) and CG therefore has rather complex communication patterns in parallel runs, which has a negative impact on scaling [75].

Discontinuous Galerkin (DG) methods [3], on the other hand, duplicate discrete variables on element boundaries, thus decoupling mesh elements and requiring at most pairwise communication between them. This is at the expense of larger linear system and more time spent in the linear solver. Discontinuous discretization is therefore expected to scale better on parallel computers, but the improved scaling is not necessarily reflected in significantly smaller CPU times when compared to a CG solver.

Hybrid discontinuous Galerkin (HDG) methods [17] address this problem by introducing an additional (hybrid) variable on the mesh skeleton. The hybrid degrees of freedom determine the rank of the global system matrix and HDG therefore produces a statically condensed system that is similar in size to the CG case. In contrast with CG, the static condensation in HDG takes place by construction rather than being an optional iterative technique. Similarly to the classical DG method, HDG scales favourably in comparison with CG, but the work-to-communication ratio is once again improved due to increased amount of intra-node work rather than due to better overall efficiency.

To take advantage of the efficiency and lower memory requirements of continuous Galerkin method together with the flexibility and more favorable communication patterns of discontinuous Galerkin methods in domain-decomposition setting, we combine both as follows. Each mesh partition is seen as a 'macro-element', where the governing equation is discretized by continuous Galerkin solver, while the patches are coupled together weakly as in HDG (Figure 12). This setting leads naturally to a formulation of weak Dirichlet boundary conditions for the CG method, which is the main focus of discussion for this section. We note that the following text is adapted from a publication submitted to *J. Comp. Phys.*

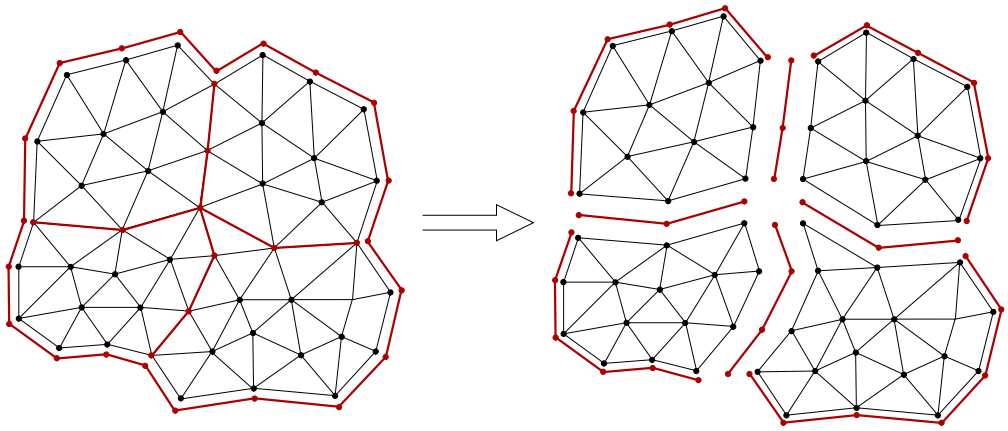


Figure 12: Mesh decomposition in CG-HDG setting. Each partition is discretized by a continuous Galerkin method. Coupling between partitions is weak and information is transmitted through a hybrid variable (red degrees of freedom).

3.1.2 Weak boundary conditions in underresolved flows

Weak boundary conditions have the potential to provide additional stability to high-order methods in high-fidelity simulations. The geometric flexibility and ability to efficiently uti-

lize modern computing hardware [72] make high-order methods particularly attractive in various application areas such as Large-Eddy Simulations (LES) over complex industrial geometries [52], which can be used to gain detailed insight into flow physics. In our case, these are based on the incompressible Navier-Stokes equations and can be efficiently tackled using discretizations which employ high-order elements in space and a time-splitting scheme [42] that involves the solution of four scalar elliptic equations for pressure and velocity components respectively. The cost of one time step in this scheme is then largely determined by the amount of work needed to obtain the pressure field, which is defined as a solution to a (frequently ill-conditioned) scalar elliptic Poisson equation.

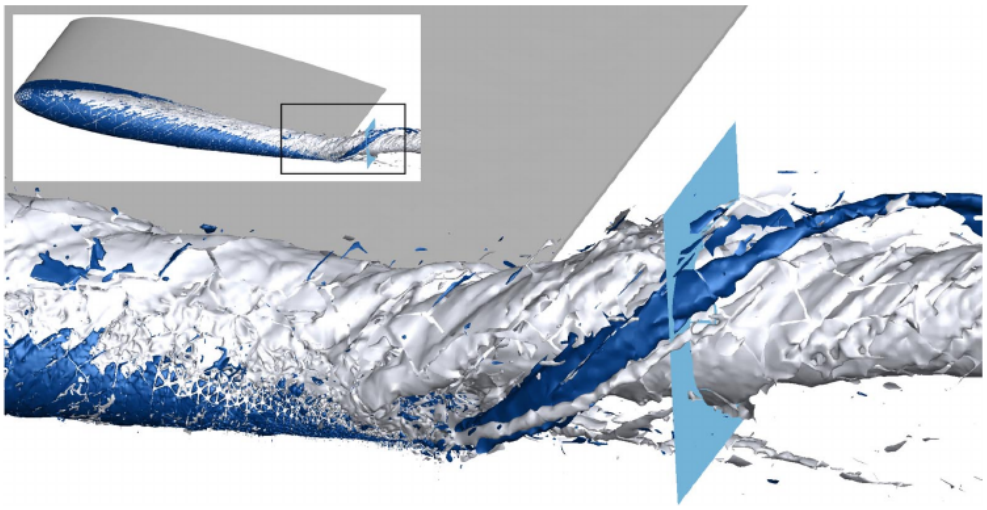


Figure 13: Implicit LES of a wingtip vortex using a high-order hp/spectral element solver. Reproduced from [52].

The difficulty in solving the governing equation for pressure is further increased when one considers complex flow features, for example the formation and evolution of a wingtip vortex simulated by a high-order method (Figure 13). Under-integration of nonlinear terms in Navier-Stokes equations introduces an aliasing error which may compromise the stability of the simulation [43]. This is usually not problematic when the flow features are adequately resolved. Once the buildup of aliasing errors becomes an issue, however, the stability of the solver is often compromised in vicinity of (wall) boundaries. This problem further motivated our search for a weak formulation of Dirichlet boundary terms, which are needed for the CG-HDG formulation.

3.1.3 Review of existing algorithms for boundary discretization

Boundary conditions in the form of penalty terms can be incorporated into the variational form using the approach first described by Nitsche [58] and later on developed in a number of other papers, for example in [30] or [41]. We compare its accuracy and performance with our method in Section 3.13.

Other work on weak imposition of Dirichlet boundary conditions includes the contribution of Bazilevs et al. [6]. The authors used their formulation to solve an advection-diffusion problem and incompressible Navier-Stokes equations with low-order stabilized finite element methods. The Dirichlet constraints were incorporated directly into the variational form as Euler-Lagrange equations. In their paper [7], the authors note that the conditions render the simulation more robust on coarse meshes where near-wall resolution is low.

Conceptually similar mechanisms can also be found in the works of Liakos [49], Liakos and Caglar [11], Layton [47] and Urquiza et al. [70], where the discrete form of the governing law together with weak boundary conditions is again enforced either by Lagrange multipliers or by penalizing certain components of the velocity field on wall boundaries. The existing body of work differs from our contribution in terms of scope, however. We are interested in elliptic problems and high-order Galerkin finite element methods, while the references cited above consider conservation laws of advection-diffusion type and use low-order stabilized methods.

3.2 Overview of the formulation of HDG method

We begin with a brief recap of the standard HDG formulation for a finite element mesh, following a similar approach to that taken in [44] and [75].

3.3 Continuous problem

We seek the solution of a Poisson equation as a representative elliptic problem

$$\begin{aligned} -\nabla^2 u(\boldsymbol{x}) &= f(\boldsymbol{x}) && \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) &= g_D(\boldsymbol{x}) && \boldsymbol{x} \in \partial\Omega_D, \\ \boldsymbol{n} \cdot \nabla u(\boldsymbol{x}) &= g_N(\boldsymbol{x}) && \boldsymbol{x} \in \partial\Omega_N, \end{aligned} \tag{26}$$

on a domain Ω with Dirichlet ($\partial\Omega_D$) and Neumann ($\partial\Omega_N$) boundary conditions, where $\partial\Omega_D \bigcup \partial\Omega_N = \partial\Omega$ and $\partial\Omega_D \bigcap \partial\Omega_N = \emptyset$. To formulate the HDG method, we consider a mixed form of (26) by introducing an auxiliary variable $\boldsymbol{q} = \nabla u$:

$$-\nabla \cdot \boldsymbol{q} = f(\boldsymbol{x}) \qquad \boldsymbol{x} \in \Omega, \tag{27}$$

$$\boldsymbol{q} = \nabla u(\boldsymbol{x}) \qquad \boldsymbol{x} \in \Omega, \tag{28}$$

$$u(\boldsymbol{x}) = g_D(\boldsymbol{x}) \qquad \boldsymbol{x} \in \partial\Omega_D, \tag{29}$$

$$\boldsymbol{q} \cdot \boldsymbol{n} = g_N(\boldsymbol{x}) \qquad \boldsymbol{x} \in \partial\Omega_N. \tag{30}$$

The gradient variable $\boldsymbol{q}$ is approximated together with the primal variable u, which contrasts with the CG method and other discontinuous methods for (26).

3.4 HDG interpolation spaces and discretization

We limit ourselves to two-dimensional problems for sake of simplicity, but the formal description remains unchanged in three dimensions. We assume that in the discrete setting, the

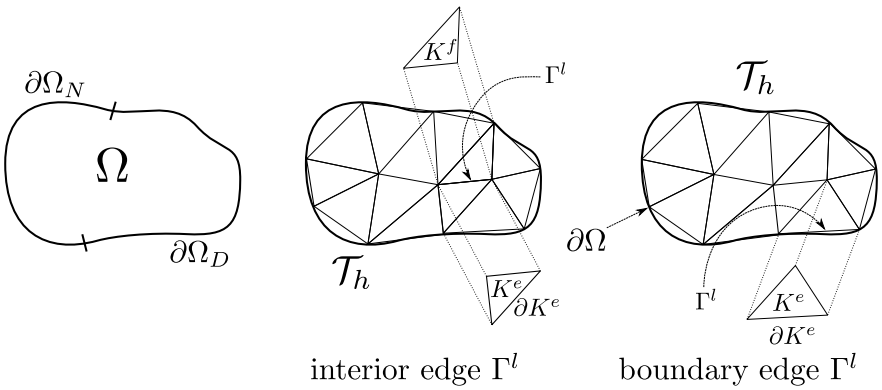


Figure 14: Computational domain and its tesselation demonstrating notation used in the text.

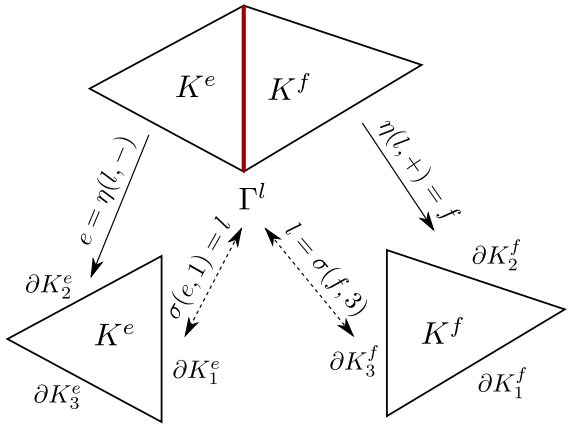


Figure 15: Index mappings relating edge and element ids.

computational domain Ω is approximated by its tesselation $\mathcal{T}_h$ consisting of non-overlapping and conformal elements K^e such that for each pair of distinct indices $e_i \neq e_j$, $K^{e_i} \cap K^{e_j} = \emptyset$. The symbol Γ^l denotes an interior edge of the tesselation $\mathcal{T}_h$, i.e. an edge $\Gamma^l = \bar{K}^i \cap \bar{K}^j$ where K^i and K^j are two distinct elements of the tesselation. We say that Γ^l is a boundary edge of the tesselation $\mathcal{T}_h$ if there exists an element K^e such that $\Gamma^l = K^e \cap \partial\Omega$ and the length of Γ^l is not zero, as shown in figure 14. The set of all internal edges is denoted by $\mathcal{E}_h^0$, while $\mathcal{E}_h^\partial$ is a set of all boundary edges. Their union $\mathcal{E}_h$ comprises of all mesh edges, $\mathcal{E}_h = \mathcal{E}_h^0 \cup \mathcal{E}_h^\partial$.

In order to describe some terms in the HDG formulation, it is also useful to introduce mappings that relate elements to their local edges, as shown in figure 15. Let ∂K_j^e be the j-th edge of element K^e, and suppose that this is also the l-th edge Γ^l in the global edge numbering. Then we define the local-to-global edge mapping σ by setting $\sigma(e,j) = l$ so that we can write $\partial K_j^e = \Gamma^{\sigma(e,j)}$. An interior edge Γ^l is the intersection of the boundaries of two elements K^e and K^f, hence we set $\eta(l,+) = e$ and $\eta(l,-) = f$ in order to be able to write $\Gamma^l = \partial K^{\eta(l,+)} \cap \partial K^{\eta(l,-)}$.

3.5 Approximation spaces

The finite element spaces supported by the (two-dimensional) tesselation $\mathcal{T}_h$ are defined as follows:

$$\begin{aligned} V_h &\coloneqq \{v \in L^2(\Omega) \quad : \quad v|_{K^e} \in \mathcal{P}(K^e) \quad \forall K^e \in \mathcal{T}_h\}, \\ \boldsymbol{\Sigma}_h &\coloneqq \{\boldsymbol{w} \in [L^2(\Omega)]^2 : \quad \boldsymbol{w}|_{K^e} \in \boldsymbol{\Sigma}(K^e) \quad \forall K^e \in \mathcal{T}_h\}, \\ \mathcal{M}_h &\coloneqq \{\mu \in L^2(\Omega) \quad : \quad \mu|_{\Gamma^l} \in \mathcal{P}(\Gamma^l) \quad \forall \Gamma^l \in \Gamma\}, \end{aligned}$$

where $\mathcal{P}(\Gamma^l) = \mathcal{S}_P(\Gamma^l)$ is the polynomial space over the standard segment

$$\mathcal{S}_P(\Gamma^l) = \{s^p \ : \ 0 \le p \le P, \ [x_1(s), x_2(s)] \in \Gamma^l, \ -1 \le s \le 1\},$$

and $\mathcal{P}(K^e)$ is the space of polynomials of degree P defined on a standard region, which can either be the standard triangle

$$\mathcal{P}(K^e) = \mathcal{T}_P(K^e) = \{\xi_1^p \xi_w^q \ : \ 0 \le p+q \le P, \ [x_1(\xi_1,\xi_2), x_2(\xi_1,\xi_2)] \in K^e, \ -1 \le \xi_1 + \xi_2 \le 0\},$$

or standard quadrilateral

$$\mathcal{P}(K^e) = \mathcal{Q}_P(K^e) = \{\xi_1^p \xi_2^q \ : \ 0 \le p,q \le P, \ [x_1(\xi_1,\xi_2), (x_2(\xi_1,\xi_2)] \in K^e, \ -1 \le \xi_1, \xi_2 \le 1\}.$$

Similarly $\boldsymbol{\Sigma}(K^e) = [\mathcal{T}_P(K^e)]^2$ or $\boldsymbol{\Sigma}(K^e) = [\mathcal{Q}_P(K^e)]^2$. There is no requirement on global continuity of the expansion. This is also true for the trace space $\mathcal{M}_h$: a discrete variable $\lambda \in \mathcal{M}_h$ is multi-valued at every mesh vertex shared by multiple interior edges.

3.6 Global formulation for HDG problem

Given an element $K \in \mathcal{T}_h$ and two functions $u, v \in L^2(\mathcal{T}_h)$, we define their L^2 scalar product by

$$(u,v)_{\mathcal{T}_h} = \sum_{K \in \mathcal{T}_h} (u,v)_K, \qquad \text{where} \qquad (u,v)_K = \int_K uv \, d\boldsymbol{x}.$$

Similarly, the L^2 product of functions u and v that are square-integrable on element traces are defined by:

$$\langle u,v \rangle_{\partial \mathcal{T}_h} = \sum_{K \in \mathcal{T}_h} \langle u,v \rangle_{\partial K} \qquad \text{where} \qquad \langle u,v \rangle_{\partial K} = \int_{\partial K} uv \, d\boldsymbol{s}$$

The DG method seeks an approximation pair $(u^{DG}, \boldsymbol{q}^{DG})$ to u and $\boldsymbol{q}$, respectively, in the space $V_h \times \boldsymbol{\Sigma}_h$. The solution is required to satisfy the weak form of (27) and (28)

$$\left(\boldsymbol{q}^{DG}, \nabla v\right)_{\mathcal{T}_h} = (f,v)_{\mathcal{T}_h} + \left\langle \boldsymbol{n}^e \cdot \tilde{\boldsymbol{q}}^{DG}, v \right\rangle_{\partial \mathcal{T}_h} \tag{31}$$

$$\left(\boldsymbol{q}^{DG}, \boldsymbol{w}\right)_{\mathcal{T}_h} = -\left(u^{DG}, \nabla \cdot \boldsymbol{w}\right)_{\mathcal{T}_h} + \left\langle \tilde{u}^{DG}, \boldsymbol{w} \cdot \boldsymbol{n}^e \right\rangle_{\partial \mathcal{T}_h} \tag{32}$$

for all $(v, \boldsymbol{w}) \in V_h(\Omega) \times \boldsymbol{\Sigma}_h(\Omega)$, where the numerical traces $\tilde{u}^{DG}$ and $\tilde{\boldsymbol{q}}^{DG}$ have to be suitably defined in terms of the approximate solution $(u^{DG}, \boldsymbol{q}^{DG})$. For details, we refer the reader to [17]. This choice of trace variables allows us to construct the discrete HDG system involving only trace degrees of freedom $\tilde{u}^{DG}$. Once $\tilde{u}$ is known, the element-interior degrees of freedom represented by both the primal variable u and gradient $\boldsymbol{q}$ can be reconstructed from element-boundary values.

We note that the element-interior variable u restricted to element traces is not equal to the hybrid variable $\tilde{u}$, but only approximates it: due to the definition of approximation spaces V_h and $\mathcal{M}_h$, u must be continuous along element boundaries, while $\tilde{u}^{DG}$ is allowed to have jumps in element vertices.

3.7 Local solvers in the HDG method

Assume that the function

$$\lambda \coloneqq \tilde{u}^{DG} \in \mathcal{M}_h, \tag{33}$$

is given. Then the solution restricted to element K^e is a function $u^e, \boldsymbol{q}^e$ in $P(K^e) \times \boldsymbol{\Sigma}(K^e)$ that satisfies the following equations:

$$(\boldsymbol{q}^e, \nabla v)_{K^e} = (f, v)_{K^e} + \langle \boldsymbol{n}^e \cdot \tilde{\boldsymbol{q}}^e, v \rangle_{\partial K^e} \tag{34}$$

$$(\boldsymbol{q}^e, \boldsymbol{w})_{K^e} = -(u^e, \nabla \cdot \boldsymbol{w})_{K^e} + \langle \lambda, \boldsymbol{w} \cdot \boldsymbol{n}^e \rangle_{\partial K^e}, \tag{35}$$

for all $(v, \boldsymbol{w}) \in P(K^e) \times \boldsymbol{\Sigma}(K^e)$. For a unique solution of the above equations to exist, the numerical trace of the flux must depend only on λ and on $(u^e, \boldsymbol{q}^e)$:

$$\tilde{\boldsymbol{q}}^e(\boldsymbol{x}) = \boldsymbol{q}^e(\boldsymbol{x}) - \tau\big(u^e(\boldsymbol{x}) - \lambda(\boldsymbol{x})\big)\boldsymbol{n}^e \quad \text{on } \partial K^e \tag{36}$$

for some positive function τ. The analysis presented in [17] reveals that as long as $\tau > 0$, its value can be arbitrary without degrading the robustness of the solver. For the limiting value of $\tau \to \infty$, one obtains a statically condensed continuous Galerkin formulation. In this sense, τ plays the role of a method selector as opposed to traditional penalty parameter used in Nitsche's method, for example.

3.8 Global problem for trace variable

We denote by $(U_\lambda, \boldsymbol{Q}_\lambda)$ and by $(U_f, \boldsymbol{Q}_f)$ the solution to the local problem (34), (35) when $\lambda = 0$ and $f = 0$, respectively. Due to the linearity of the original problem (26) and its mixed form, the solution satisfies

$$(u^{HDG}, \boldsymbol{q}^{HDG}) = (U_\lambda, \boldsymbol{Q}_\lambda) + (U_f, \boldsymbol{Q}_f). \tag{37}$$

In order to uniquely determine λ, we require that the boundary conditions be weakly satisfied and the normal component of the numerical trace of the flux $\tilde{\boldsymbol{q}}$ given by (36) is single valued, rendering the numerical trace conservative.

We say that λ is the element of $\mathcal{M}_h$ such that

$$\lambda = \mathbb{P}_h(g_D) \quad \text{on } \partial\Omega_D \tag{38}$$

$$\langle \mu, \tilde{\boldsymbol{q}} \cdot \boldsymbol{n} \rangle_{\partial\mathcal{T}} = \langle \mu, g_N \rangle_{\partial\Omega_N}, \tag{39}$$

for all $\mu \in \mathcal{M}_h^0$ such that $\mu = 0$ on $\partial\Omega_D$. Here $\mathbb{P}_h$ denotes the L^2-projection into the space of restrictions to $\partial\Omega_D$ of functions of $\mathcal{M}_h$.

In the following, we consider $u^e(\boldsymbol{x}), \boldsymbol{q}^e(\boldsymbol{x}) = [q_1, q_2]^T$ and $\lambda^l(\boldsymbol{x})$ to be finite expansions in terms of basis functions $\phi_j^e(\boldsymbol{x})$ for the expansions over elements and the basis $\psi_j^l(\boldsymbol{x})$ over the traces of the form:

$$u^e(\boldsymbol{x}) = \sum_{j=1}^{N_u^e} \phi_j^e(\boldsymbol{x}) \hat{\underline{u}}^e[j] \qquad \boldsymbol{q}_k^e(\boldsymbol{x}) = \sum_{j=1}^{N_q^e} \phi_j^e(\boldsymbol{x}) \hat{\underline{q}}_k^e[j] \qquad \lambda^l(\boldsymbol{x}) = \sum_{j=1}^{N_\lambda^l} \psi_j^l(\boldsymbol{x}) \hat{\underline{\lambda}}^l[j]$$

3.9 Discrete form of HDG local solver

We now define several local matrices stemming from standard Galerkin formulation, where scalar test functions v^e are represented by $\phi_i^e(\boldsymbol{x})$, with $i = 1, \ldots, N_u^e$.

$$\mathbf{D}_k^e[i,j] = \left(\phi_i^e, \frac{\partial \phi_j^e}{\partial x_k}\right)_{K_e} \qquad \mathbf{M}^e[i,j] = \left(\phi_i^e, \phi_j^e\right)_{K_e}$$

$$\mathbf{E}_l^e[i,j] = \left\langle \phi_i^e, \phi_j^e \right\rangle_{\partial K_l^e} \qquad \widetilde{\mathbf{E}}_{kl}^e[i,j] = \left\langle \phi_i^e, \phi_j^e n_k^e \right\rangle_{\partial K_l^e}$$

$$\mathbf{F}_l^e[i,j] = \left\langle \phi_i^e, \psi_j^{\sigma(e,l)} \right\rangle_{\partial K_l^e} \qquad \widetilde{\mathbf{F}}_{kl}^e[i,j] = \left\langle \phi_i^e, \psi_j^{\sigma(e,l)} n_k^e \right\rangle_{\partial K_l^e}$$

If the trace expansion matches the expansions used along the edge of the elemental expansion and the local coordinates are aligned, that is $\psi_i^{\sigma(e,l)}(s) = \phi_{k(i)}(s)$ then $\mathbf{E}_l^e$ contains the same entries as $\mathbf{F}_l^e$ and similarly $\widetilde{\mathbf{E}}_{kl}^e$ contains the same entries as $\widetilde{\mathbf{F}}_{kl}^e$.

Inserting the finite expansions of the trial functions into equations (34) and (35), and using the definition of the flux (36) yields the matrix form of *local solvers*

$$\sum_{k=1,2} \left\{ (\mathbf{D}_k^e)^T - \sum_{l=1}^{N_b^e} \left[\widetilde{\mathbf{E}}_{kl}^e \right] \right\} \hat{\underline{q}}_k^e + \sum_{l=1}^{N_b^e} \tau^{e,l} \left[\mathbf{E}_l^e \hat{\underline{u}}^e - \mathbf{F}_l^e \hat{\underline{\lambda}}^{\sigma(e,l)} \right] = \underline{f}^e \tag{40}$$

$$\mathbf{M}^e \hat{\underline{q}}_k^e = -(\mathbf{D}_k^e)^T \hat{\underline{u}}^e + \sum_{l=1}^{N_b^e} \widetilde{\mathbf{F}}_{kl}^e \hat{\underline{\lambda}}^{\sigma(e,l)} \qquad k = 1, 2 \tag{41}$$

The *global equation for* λ can be obtained by discretizing the transmission condition (39). We introduce local element-based and edge-based matrices

$$\overline{\mathbf{F}}^{l,e}[i,j] = \left\langle \psi_i^l, \phi_j^e \right\rangle_{\Gamma^l} \qquad \widetilde{\overline{\mathbf{F}}}_k^{l,e}[i,j] = \left\langle \psi_i^l, \phi_j^e n_k^e \right\rangle_{\Gamma^l} \qquad \bar{\mathbf{G}}^l[i,j] = \left\langle \psi_i^l, \psi_j^l \right\rangle_{\Gamma^l}$$

and define

$$\underline{g}_N^l[i] = \left\langle g_n, \psi_i^l \right\rangle_{\Gamma^l \bigcap \partial\Omega_N}.$$

The transmission condition in matrix form is then

$$\begin{bmatrix} \tilde{\bar{\mathbf{F}}}_1^{l,e} & \tilde{\bar{\mathbf{F}}}_2^{l,e} \end{bmatrix} \begin{bmatrix} \hat{\underline{q}}_1^e \\ \hat{\underline{q}}_2^e \end{bmatrix} + \begin{bmatrix} \tilde{\bar{\mathbf{F}}}_1^{l,f} & \tilde{\bar{\mathbf{F}}}_2^{l,f} \end{bmatrix} \begin{bmatrix} \hat{\underline{q}}_1^f \\ \hat{\underline{q}}_2^f \end{bmatrix} + (\tau^{e,i} + \tau^{f,j}) \bar{\mathbf{G}}^l \hat{\underline{\lambda}}^l - \tau^{e,i} \bar{\mathbf{F}}^{l,e} \underline{u}^e - \tau^{f,j} \bar{\mathbf{F}}^{l,f} \underline{u}^f = \underline{g}_N^l,$$

where we are assuming that $l = \sigma(e,i) = \sigma(f,j)$.

3.10 Continuous finite elements with weak Dirichlet boundary conditions

With the standard HDG formulation now outlined, we investigate how this approach can be applied to derive a weak Dirichlet boundary condition implementation for a continuous Galerkin problem. Since the HDG local solver naturally imposes a weak boundary condition on a single element, we choose to apply the HDG local solver to a single 'macro element' that covers the whole domain tesselation $\mathcal{T}_h$. The term 'macro element' in this setting denotes a conformal triangulation (as described in section 3.2) which supports a piecewise polynomial expansion, as is generally common in Galerkin methods.

We start again from the weak mixed problem (34), (35), but integrate the second term in the flux equation (35) by parts once again. This modified flux form allows for a symmetric boundary contribution to the linear system as will be explained shortly. In order to distinguish between the standard HDG local solver within a single element and HDG applied to the whole domain tesselation $\mathcal{T}_h$, the superscript 'e' has been replaced by $\mathcal{T}_h$ where appropriate. The 'macro element' form yields a system

$$\left(\boldsymbol{q}^{\mathcal{T}_h}, \nabla v\right)_{\mathcal{T}_h} = (f, v)_{\mathcal{T}_h} + \left\langle \boldsymbol{n}^{\mathcal{T}_h} \cdot \tilde{\boldsymbol{q}}^{\mathcal{T}_h}, v \right\rangle_{\partial\mathcal{T}_n} \tag{42}$$

$$\left(\boldsymbol{q}^{\mathcal{T}_h}, \boldsymbol{w}\right)_{\mathcal{T}_h} = \left(\nabla u^{\mathcal{T}_h}, \boldsymbol{w}\right)_{\mathcal{T}_h} - \left\langle u^{\mathcal{T}_h}, \boldsymbol{w} \cdot \boldsymbol{n}^{\mathcal{T}_h} \right\rangle_{\partial\mathcal{T}_h} + \left\langle \lambda, \boldsymbol{w} \cdot \boldsymbol{n}^{\mathcal{T}_h} \right\rangle_{\partial\mathcal{T}_h} \tag{43}$$

The numerical approximation $u^{\mathcal{T}_h}$ belongs to the space $V_h^{\mathcal{T}_h}$ and $\boldsymbol{q}^{\mathcal{T}_h}$ lies in $\boldsymbol{\Sigma}_h^{\mathcal{T}_h}$, which are defined as

$$\begin{aligned} V_h^{\mathcal{T}} &:= \{v \in \mathcal{C}^0(\Omega) \quad : \quad v|_{K^e} \in P(K^e) \quad \forall K^e \in \mathcal{T}_h\}, \\ \boldsymbol{\Sigma}_h^{\mathcal{T}} &:= \{\boldsymbol{w} \in [L^2(\Omega)]^2 : \quad \boldsymbol{w}|_{K^e} \in \boldsymbol{\Sigma}(K^e) \quad \forall K^e \in \mathcal{T}_h\}. \end{aligned}$$

Using the definition of the trace flux

$$\tilde{\boldsymbol{q}}^{\mathcal{T}_h}(\boldsymbol{x}) = \boldsymbol{q}^{\mathcal{T}_h}(\boldsymbol{x}) - \tau(u^{\mathcal{T}_h}(\boldsymbol{x}) - \lambda(\boldsymbol{x}))\boldsymbol{n}^{\mathcal{T}_h} \quad \text{on } \partial\mathcal{T}_h,$$

and the fact that the integral over $\mathcal{T}_h$ can be written as a sum of integrals over all $K^e \in \mathcal{T}_h$,

equations (42) and (43) become

$$\sum_{K^e\in\mathcal{T}_h}(\nabla v,\boldsymbol{q}^e)_{K^e}-\sum_{\substack{K^e\\ \partial K^e\cap\partial\mathcal{T}_{h,D}\neq\emptyset}}\langle v,\boldsymbol{n}^e\cdot\boldsymbol{q}^e\rangle_{\partial K^e}+\tau\sum_{\substack{K^e\\ \partial K^e\cap\partial\mathcal{T}_{h,D}\neq\emptyset}}\langle v,u^e\rangle_{\partial K^e}$$
$$-\tau\sum_{\substack{K^e\\ \partial K^e\cap\partial\mathcal{T}_{h,D}\neq\emptyset}}\langle v,\lambda\rangle_{\partial K^e}=\sum_{K^e\in\mathcal{T}_h}(v,f)_{K^e} \tag{44}$$

$$\sum_{K^e\in\mathcal{T}_h}(\boldsymbol{w},\boldsymbol{q}^e)_{K^e}+\sum_{\substack{K^e\\ \partial K^e\cap\partial\mathcal{T}_{h,D}\neq\emptyset}}\langle u^e,\boldsymbol{w}\cdot\boldsymbol{n}^e\rangle_{\partial K^e}$$
$$-\sum_{K^e\in\mathcal{T}_h}(\boldsymbol{w},\nabla u^e)_{K^e}-\sum_{\substack{K^e\\ \partial K^e\cap\partial\mathcal{T}_{h,D}\neq\emptyset}}\langle\boldsymbol{w}\cdot\boldsymbol{n}^e,\lambda\rangle_{\partial K^e}=0 \tag{45}$$

A continuous Galerkin solver with Dirichlet data prescribed by the variable λ can be obtained by eliminating the flux variable from the system and reverting back to primal form for the unknown u. The mass matrix which appears in the second equation of the local solver after evaluating the dot product $\left(\boldsymbol{q}^{\mathcal{T}_h},\boldsymbol{w}\right)_{\mathcal{T}_h}$ is now block-diagonal as a consequence of the discontinuous nature of the discrete flux $\boldsymbol{q}^{\mathcal{T}_h}$, hence the elimination of $\boldsymbol{q}^{\mathcal{T}_h}$ from the system can be performed element-wise. The matrix equivalent of (44), (45) written for a single element $K^e\in\mathcal{T}_h$ adjacent to Dirichlet boundary reads

$$\sum_{k=1,2}\left\{(\mathbf{D}_k^e)^T-\sum_{l=1}^{N_b^e}\widetilde{\mathbf{E}}_{kl}^e\right\}\underline{\hat{q}}_k^e+\sum_{l=1}^{N_b^e}\tau^{e,l}\left[\mathbf{E}_l^e\underline{\hat{u}}^e-\mathbf{F}_l^e\hat{\underline{\lambda}}^{\sigma(e,l)}\right]=\underline{f}^e \tag{46}$$

$$\mathbf{M}^e\underline{\hat{q}}_k^e=\left\{(\mathbf{D}_k^e)-\sum_{l=1}^{N_b^e}\widetilde{\mathbf{E}}_{kl}^e\right\}\underline{\hat{u}}^e+\sum_{l=1}^{N_b^e}\widetilde{\mathbf{F}}_{kl}^e\hat{\underline{\lambda}}^{\sigma(e,l)}\qquad k=1,2 \tag{47}$$

The discrete flux $\underline{\hat{q}}_k^e$ expressed from (47) and substituted in equation (46) yields element-wise contribution to the left- and right-hand side of the linear system which can be expressed as

$$\sum_{k=1,2}\Bigg\{\underbrace{(\mathbf{D}_k^e)^T\left(\mathbf{M}^e\right)^{-1}\mathbf{D}_k^e}_{\boxed{1}}-\underbrace{\left(\sum_{l=1}^{N_b^e}\tilde{\mathbf{E}}_{kl}^e\right)\left(\mathbf{M}^e\right)^{-1}\mathbf{D}_k^e}_{\boxed{2a}}\Bigg\}\underline{\hat{u}}^e$$
$$-\sum_{k=1,2}\Bigg\{\underbrace{(\mathbf{D}_k^e)^T\left(\mathbf{M}^e\right)^{-1}\left(\sum_{l=1}^{N_b^e}\tilde{\mathbf{E}}_{kl}^e\right)}_{\boxed{2b}}+\underbrace{\left(\sum_{l=1}^{N_b^e}\tilde{\mathbf{E}}_{kl}^e\right)\left(\mathbf{M}^e\right)^{-1}\left(\sum_{l=1}^{N_b^e}\tilde{\mathbf{E}}_{kl}^e\right)}_{\boxed{3}}\Bigg\}\underline{\hat{u}}^e+\underbrace{\sum_{l=1}^{N_b^e}\tau^{(e,l)}\mathbf{E}_l^e\underline{\hat{u}}^e}_{\boxed{4}}$$
$$=\underline{\boldsymbol{f}}^e+\sum_{k=1,2}\left\{\left(\sum_{l=1}^{N_b^e}\tilde{\mathbf{E}}_{kl}^e-(\mathbf{D}_k^e)^T\right)\left(\mathbf{M}^e\right)^{-1}\left(\sum_{l=1}^{N_b^e}\tilde{\mathbf{F}}_{kl}^e\hat{\underline{\lambda}}^{\sigma(e,l)}\right)\right\}+\sum_{l=1}^{N_b^e}\tau^{(e,l)}\mathbf{F}_l^e\hat{\underline{\lambda}}^{\sigma(e,l)}$$

Term $\boxed{1}$ on the left-hand side is a discrete Laplacian that arises from the standard continuous Galerkin discretization, which would typically be accompanied by the forcing term $\underline{\boldsymbol{f}}^e$ on the right hand side. This new expression therefore denotes a modification of the existing matrix system and right hand side, which makes implementation relatively straightforward. The matrix expressions $\boxed{2a}$, $\boxed{2b}$, $\boxed{3}$ and $\boxed{4}$ appear in the formulation only for elements K^e containing at least one edge on Dirichlet boundary of Ω. In addition, expressions $\boxed{3}$ and $\boxed{4}$ are symmetric as a consequence of symmetry of $\tilde{\mathbf{E}}^e_{kl}, \mathbf{E}^e_l$ and $\left(\mathbf{M}^e\right)^{-1}$. The products $\boxed{2a}$ and $\boxed{2b}$ are transposes of each other, hence their sum is again symmetric. The modifications to the symmetric discrete Laplacian therefore preserve symmetry of the discrete weak form, meaning that efficient iterative solvers such as the conjugate gradient method can be used to obtain solutions.

When the domain trace λ and forcing term $\boldsymbol{f}$ are both zero, the bilinear forms (44), (45) yield a homogeneous linear system with a regular matrix. This can be shown by testing the two forms with $v = u$ and $\boldsymbol{w} = \boldsymbol{q}$:

$$\sum_{K^e \in \mathcal{T}_h} (\nabla u^e, \boldsymbol{q}^e)_{K^e} - \sum_{\substack{K^e \\ \partial K^e \cap \partial \mathcal{T}_{h,D} \neq \emptyset}} \langle u^e, \boldsymbol{n}^e \cdot \boldsymbol{q}^e \rangle_{\partial K^e} + \tau \sum_{\substack{K^e \\ \partial K^e \cap \partial \mathcal{T}_{h,D} \neq \emptyset}} \langle u^e, u^e \rangle_{\partial K^e} = 0$$

$$\sum_{K^e \in \mathcal{T}_h} (\boldsymbol{q}^e, \boldsymbol{q}^e)_{K^e} + \sum_{\substack{K^e \\ \partial K^e \cap \partial \mathcal{T}_{h,D} \neq \emptyset}} \langle u^e, \boldsymbol{q}^e \cdot \boldsymbol{n}^e \rangle_{\partial K^e} - \sum_{K^e \in \mathcal{T}_h} (\boldsymbol{q}^e, \nabla u^e)_{K^e} = 0$$

Their sum

$$\sum_{K^e \in \mathcal{T}_h} (\boldsymbol{q}^e, \boldsymbol{q}^e)_{K^e} + \tau \sum_{\substack{K^e \\ \partial K^e \cap \partial \mathcal{T}_{h,D} \neq \emptyset}} \langle u^e, u^e \rangle_{\partial K^e} = 0$$

has a unique solution $\boldsymbol{q} = \mathbf{0}$ and $u = 0$ provided $\tau > 0$. This means that the CG system with weakly imposed Dirichlet boundary conditions is uniquely solvable.

3.11 Results

In this section, we apply the weak Dirichlet boundary conditions to various elliptic problems. We demonstrate that this technique preserves the expected convergence properties of high-order methods, and then apply it to a standard fluid dynamics test case in order to showcase its use in a more realistic application.

3.12 Convergence of continuous Galerkin solver with weak boundary conditions

We first present a straightforward evaluation of the convergence properties of weakly imposed Dirichlet boundary conditions on a scalar Helmholtz problem

$$\nabla^2 u - \lambda u = f \tag{48}$$

in a square domain $(-1, 1)^2$ with $\lambda = 1$ and $f(x, y)$ chosen so that the exact solution is of the form

$$u(x, y) = \sin(10\pi x)\cos(10\pi y) + x + y \tag{49}$$

Two meshes were considered: a structured Cartesian grid and an unstructured mesh consisting of triangles. Figure 17 compares the L^2 error for polynomial orders varying between

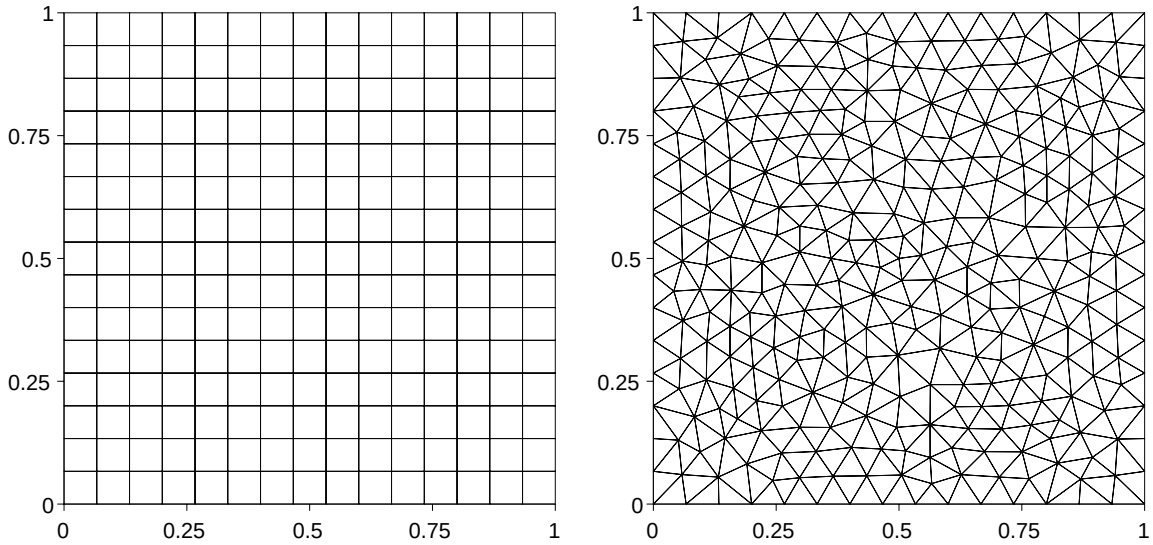
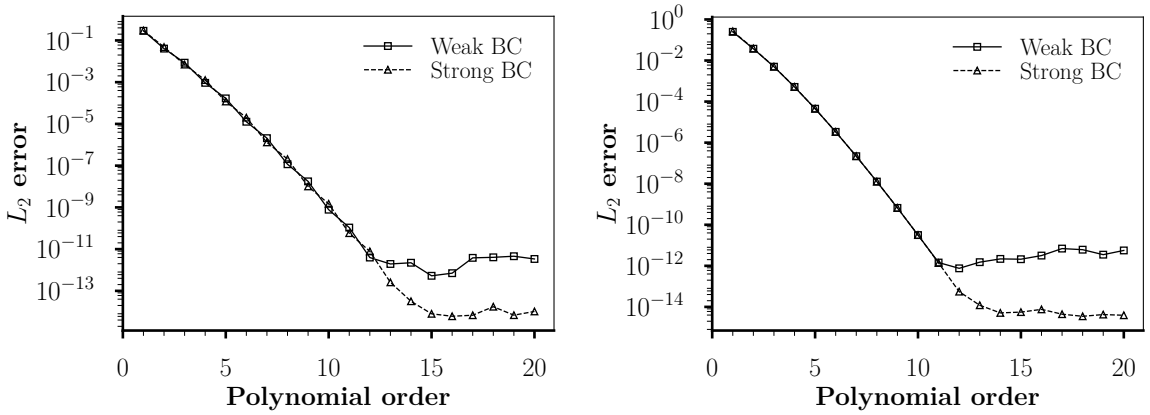


Figure 16: Meshes for Helmholtz convergence test.

1 and 20 when the Dirichlet boundary conditions are imposed strongly and weakly. The behaviour of both strong and weak methods produces nearly identical errors up to $p = 12$ on the structured grid and $p = 11$ on triangles. With further increase of polynomial degree of the basis, however, the weak errors fail to further decrease. The observed differences are not surprising, because the HDG-based algorithm only penalizes the solution in order to satisfy boundary conditions, while the strong implementation completely eliminates known degrees of freedom and moves them to the right-hand side of the linear system, thus fulfilling the boundary conditions exactly by construction. Furthermore, the stiffness matrix with weak constraints is larger, hence less favourably conditioned and round-off errors in the linear solver become important as the error values approach the limits of finite-precision arithmetic on given machine.



(a) Convergence to exact solution, strong vs. weak boundary conditions on triangular mesh.

(b) Convergence to exact solution, strong vs. weak boundary conditions on quadrilateral mesh.

Figure 17: Convergence to the exact solution in L_2 norm.

3.13 Comparison with classical penalty techniques

We conclude the paper with a brief comparison of our method against an existing technique for imposing weak boundary conditions. A classical penalty approach for boundary conditions in finite element methods is due to Nitsche [58]. Consider a Poisson equation

$$\begin{aligned} -\nabla^2 u(\boldsymbol{x}) &= f(\boldsymbol{x}) \quad \boldsymbol{x} \text{ in } \Omega \\ u(\boldsymbol{x}) &= g_D(\boldsymbol{x}) \quad \boldsymbol{x} \in \partial\Omega, \end{aligned}$$

Multiplying both sides of the equation by a test function v and adding a term $\langle u - g_D, \nabla v \cdot \boldsymbol{n} \rangle_{\partial\Omega}$ which should vanish for u satisfying the boundary condition yields

$$(\nabla u, \nabla v)_\Omega - \langle \nabla u \cdot \boldsymbol{n}, v \rangle_{\partial\Omega} - \langle u - g_D, \nabla v \cdot \boldsymbol{n} \rangle_{\partial\Omega} = (f, v)_\Omega$$

A coercive bilinear form can be obtained by adding a penalty term $\tau_N \langle u - g_D, v \rangle_{\partial\Omega}$ which should again be equal to zero for the exact solution. Nitsche's method is therefore defined as: find $u_h \in V_h$ such that

$$\mathcal{B}(u_h, v) = \mathcal{F}(v) \quad \forall v \in V_h, \tag{50}$$

where

$$\begin{aligned} \mathcal{B}(u, v) &= (\nabla u, \nabla v)_\Omega - \langle v, \nabla u \cdot \boldsymbol{n} \rangle_{\partial\Omega} - \langle u, \nabla v \cdot \boldsymbol{n} \rangle_{\partial\Omega} + \tau_N \langle u, v \rangle_{\partial\Omega}, \\ \mathcal{F}(v) &= (f, v)_\Omega - \langle g, \nabla v \cdot \boldsymbol{n} \rangle_{\partial\Omega} + \tau_N \langle g, v \rangle_{\partial\Omega} \end{aligned}$$

and

$$V_h := \{ v \in H^1(\Omega) : \, v|_{K^e} \in P(K^e) \, \forall K^e \in \mathcal{T}_h \}.$$

The main drawback of the above formulation is that the penalty parameter τ_N is problem-dependent; estimates are discussed in more detail in papers [30] or [41].

To demonstrate the differences in behaviour of (50) and our method in a concrete setting, we solve a two-dimensional Laplace problem with exact solution given by

$$u = \sin(10\pi x)\cos(10\pi y) + x + y, \tag{51}$$

i.e. the same exact solution as in Section 3.12. The numerical approximation was represented by Lagrange finite elements with polynomial degree varying between 1 and 7 and we solved the underlying linear system by a preconditioned conjugate gradient (PCG) method with algebraic multigrid as preconditioner. Each computation was required to reach a relative tolerance threshold of 10^{-9}.

The results summarized in Table 2 show that the weak boundary algorithm has little sensitivity to values of τ with respect to the obtained L_2 errors. Large values of the penalty parameter help reduce the number of PCG iterations by approximately 10%. Nitsche's method, on the other hand, yields larger variations in L_2 errors when the penalty parameter is changed, and this can be observed even for low orders. Too low values of τ_N initially lead to larger errors and with increasing p eventually prevent the method from converging.

	Value of τ in weak BCs						Value of τ_N in Nitsche's method			
	10^{-6}		1		10^{6}		10^{6}		10^{8}	
p	N_{AMG}	$\|e\|_{L_2}$	N_{AMG}	$\|e\|_{L_2}$	N_{AMG}	$\|e\|_{L_2}$	N_{AMG}	$\|e\|_{L_2}$	N_{AMG}	$\|e\|_{L_2}$
1	17	$3.15 \cdot 10^{-1}$	17	$3.15 \cdot 10^{-1}$	15	$3.29 \cdot 10^{-1}$	15	$4.18 \cdot 10^{-1}$	14	$4.15 \cdot 10^{-1}$
2	23	$4.51 \cdot 10^{-2}$	23	$4.51 \cdot 10^{-2}$	21	$4.66 \cdot 10^{-2}$	22	$5.42 \cdot 10^{-2}$	20	$5.25 \cdot 10^{-2}$
3	35	$2.90 \cdot 10^{-3}$	35	$2.89 \cdot 10^{-3}$	31	$2.47 \cdot 10^{-3}$	32	$8.68 \cdot 10^{-3}$	29	$5.06 \cdot 10^{-3}$
4	49	$6.34 \cdot 10^{-4}$	49	$6.34 \cdot 10^{-4}$	46	$6.39 \cdot 10^{-4}$	47	$1.17 \cdot 10^{-2}$	42	$6.79 \cdot 10^{-4}$
5	66	$1.45 \cdot 10^{-4}$	66	$1.45 \cdot 10^{-4}$	61	$1.54 \cdot 10^{-4}$	–	–	55	$1.64 \cdot 10^{-4}$
6	95	$1.06 \cdot 10^{-5}$	95	$1.06 \cdot 10^{-5}$	87	$1.09 \cdot 10^{-5}$	–	–	79	$5.04 \cdot 10^{-5}$
7	141	$2.08 \cdot 10^{-6}$	141	$2.08 \cdot 10^{-6}$	128	$2.19 \cdot 10^{-6}$	–	–	116	$5.17 \cdot 10^{-5}$

Table 2: Iterative convergence and L_2 errors for different values of penalty parameters in weak boundary conditions and Nitsche's method. Missing entries in the table indicate cases where Nitsche's algorithm did not converge. N_{PMG} represents the number of iterations in preconditioned conjugate gradient solver using an algebraic multigrid solver as preconditioner and $\|e\|_{L_2}$ is the obtained L_2 error norm.

The situation is different when an anisotropic mesh such as the one depicted in figure 18 is considered. Despite the fact that errors computed with weak boundary conditions are now larger, the formulation remains stable and yields consistent results across a range of values of τ.

Nitsche's method, however, lacks in robustness in this case. Table 3b contains two different values of stabilization parameter τ for each polynomial order: the first is chosen as the smallest power of 10 for which the method converged, the second is then equal to the

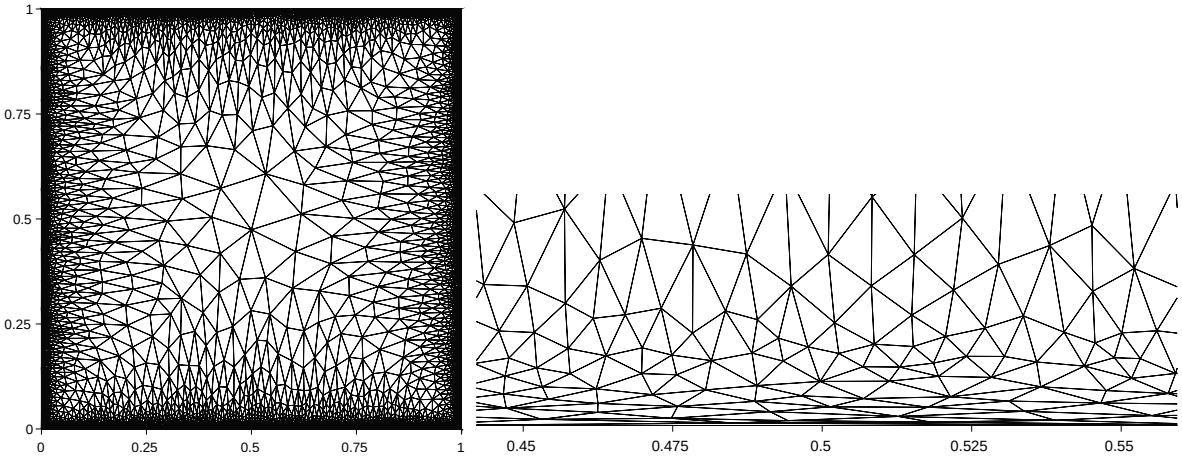


Figure 18: Anisotropic mesh of a unit square.

first multiplied by 10^5. This is to demonstrate that not even a significant increase of the stabilization parameter helps reduce the error.

	Value of τ in weak BCs					
	10^{-6}		1		10^{6}	
p	N_{AMG}	$\|e\|_{L_2}$	N_{AMG}	$\|e\|_{L_2}$	N_{AMG}	$\|e\|_{L_2}$
1	19	$3.73 \cdot 10^{-1}$	19	$3.73 \cdot 10^{-1}$	19	$3.73 \cdot 10^{-1}$
2	37	$8.44 \cdot 10^{-2}$	37	$8.44 \cdot 10^{-2}$	37	$8.44 \cdot 10^{-2}$
3	57	$1.34 \cdot 10^{-2}$	57	$1.34 \cdot 10^{-2}$	57	$1.34 \cdot 10^{-2}$
4	81	$3.44 \cdot 10^{-3}$	81	$3.44 \cdot 10^{-3}$	81	$3.44 \cdot 10^{-3}$
5	112	$6.84 \cdot 10^{-4}$	112	$6.84 \cdot 10^{-4}$	112	$6.84 \cdot 10^{-4}$
6	166	$1.24 \cdot 10^{-4}$	166	$1.24 \cdot 10^{-4}$	166	$1.24 \cdot 10^{-4}$
7	252	$2.28 \cdot 10^{-5}$	252	$2.28 \cdot 10^{-5}$	252	$2.28 \cdot 10^{-5}$

(a) Iterative convergence and L_2 errors for different values of penalty parameters in weak boundary conditions on anisotropic mesh.

p	τ_N	N_{AMG}	$\|e\|_{L_2}$
1	10^{14}	18	1102.71
1	10^{19}	14	927.93
2	10^{15}	30	22.76
2	10^{20}	22	22.53
3	10^{15}	47	55.57
3	10^{20}	32	54.17
4	10^{15}	64	3.05
4	10^{20}	46	2.89
5	10^{16}	90	1.53
5	10^{21}	58	1.52

(b) Iterative convergence and L_2 errors for Nitsche's method on anisotropic mesh.

Table 3: Performance of Nitsche's method and weak boundary conditions on anisotropic mesh.

3.14 Navier-Stokes results

3.14.1 NACA6412

The incompressible flow past a NACA6412 airfoil was used to evaluate the performance of weak boundary conditions when computing derived quantities such as aerodynamic forces.

The airfoil is placed in the flow with angle of attack $\alpha = 15^\circ$ and the Reynolds number based on chord length is $Re = 500$. No-slip condition on airfoil surface was imposed weakly, while inlet values were prescribed using the classical strong algorithm. The simulation ran for 20,000 time steps with $\Delta t = 5 \times 10^{-3}$ using a velocity correction scheme implemented in the open-source library Nektar++ [13]. The obtained flow field at $t = 10$ is plotted in Figure 19 for both strong and weak boundary conditions.

We compared lift and drag computed on the same mesh using a modal expansion with degrees 4 and 8 (Figure 20). In both cases, the forces computed with weak and strong approach are in excellent agreement.

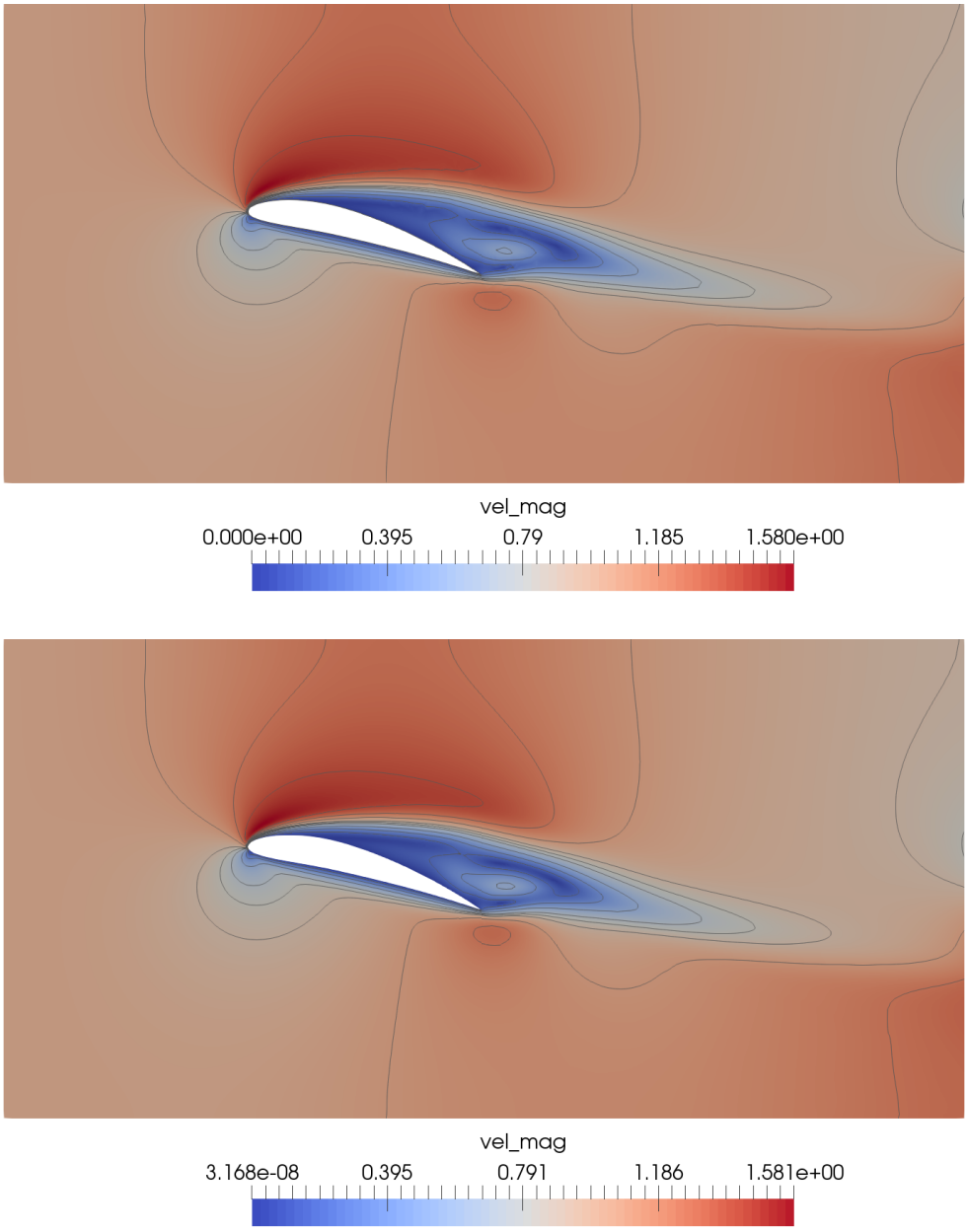


Figure 19: Viscous incompressible flow past NACA airfoil: velocity magnitude obtained with strong Dirichlet boundary conditions imposed on the airfoil surface (top) and with weak boundary conditions (bottom).

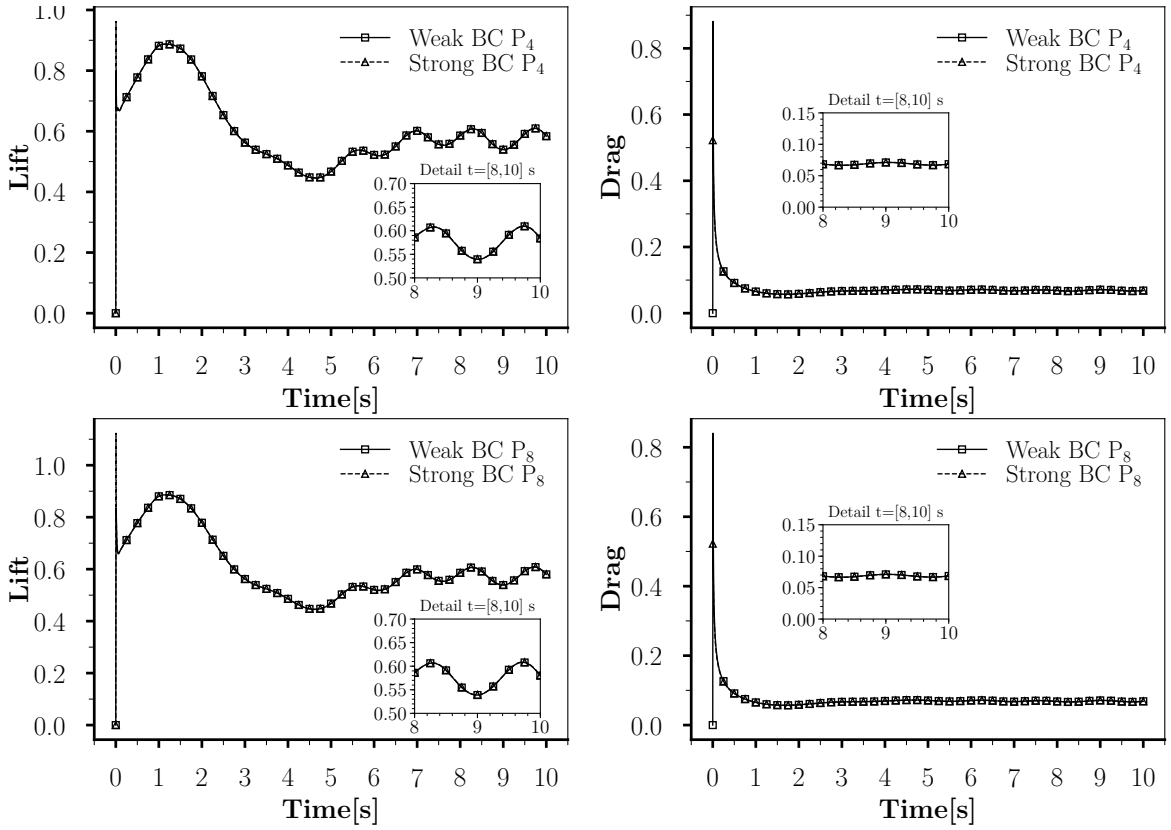


Figure 20: Viscous incompressible flow past NACA airfoil: drag and lift on P_4 elements (top) and P_8 elements (bottom).

3.14.2 Unsteady flow past a turbine blade

This test case serves as means of comparing the performance of weak and strong Dirichlet boundary conditions in turbulent flow. We considered the T106 turbine blade and a high-order hybrid mesh consisting of triangles and quadrilaterals (Figure 21). The simulation

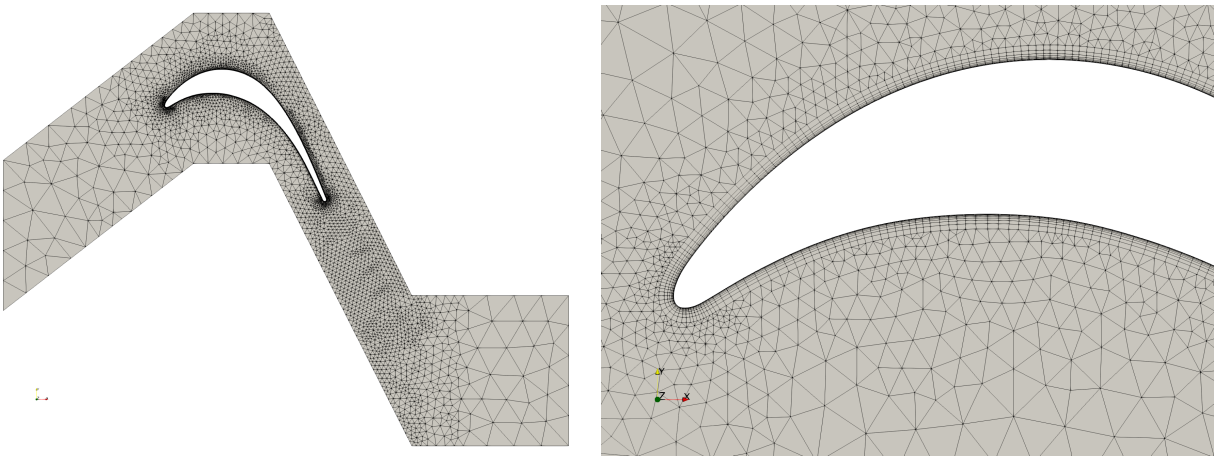


Figure 21: Turbine blade geometry: domain and leading edge detail

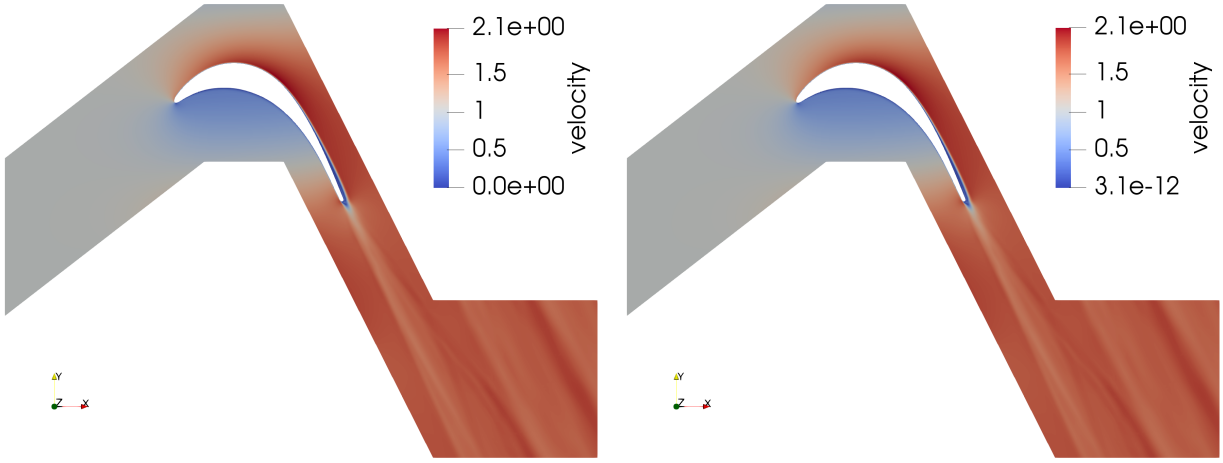
was initiated from a restart file containing flow field data that correspond to physical time $T = 24$ and ran until $T = 32s$. The time step was $\Delta t = 2.5 \times 10^{-5}$ and Reynolds number $Re = 50\,000$. The flow enters the domain with angle of incidence $\alpha = 37.7°$.

The configuration files for strong and weak boundary conditions only differed in the definition of boundary treatment on blade wall: the former used standard Dirichlet boundary condition algorithm, while the latter used weakly imposed boundary condition based on a modified HDG algorithm. Any other boundary conditions prescribed in the symmetry and inflow/outflow boundaries were identical.

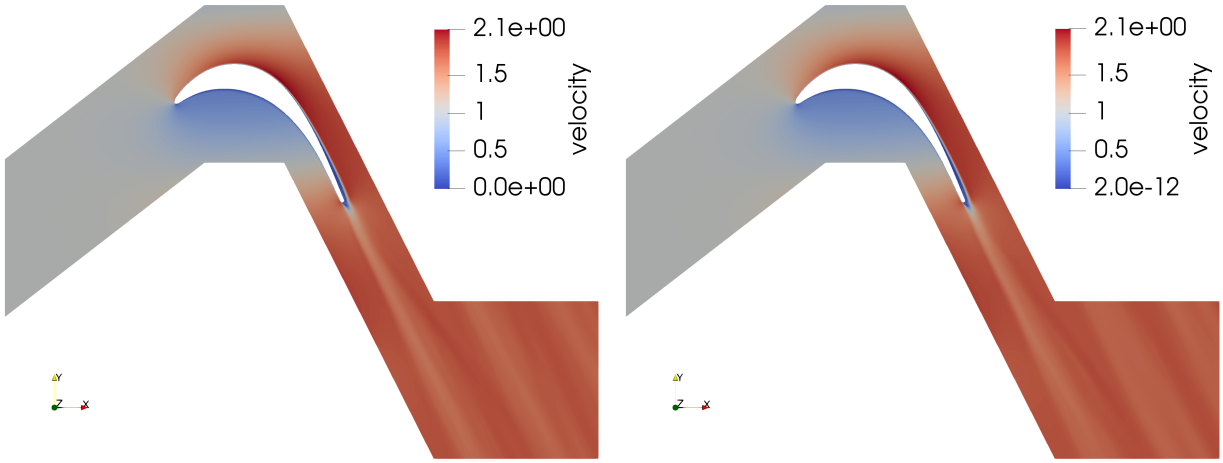
An averaged flow field was saved every 5000 iterations. Figure 22 shows two of these averaged fields. Both simulations produce nearly identical result, with weak boundary enforcement allowing for small nonzero values on blade wall.

Instantaneous velocity fields, however, differ more significantly. Figure 23 shows instantaneous velocity fields with strong and weak boundary conditions at 192 000 iterations.

In figure 24, we present the kinetic energy as function of time.



50000 iterations.



300000 iterations.

Figure 22: Averaged velocity field obtained with strong boundary conditions on blade wall (left column) and weak boundary conditions (right column).

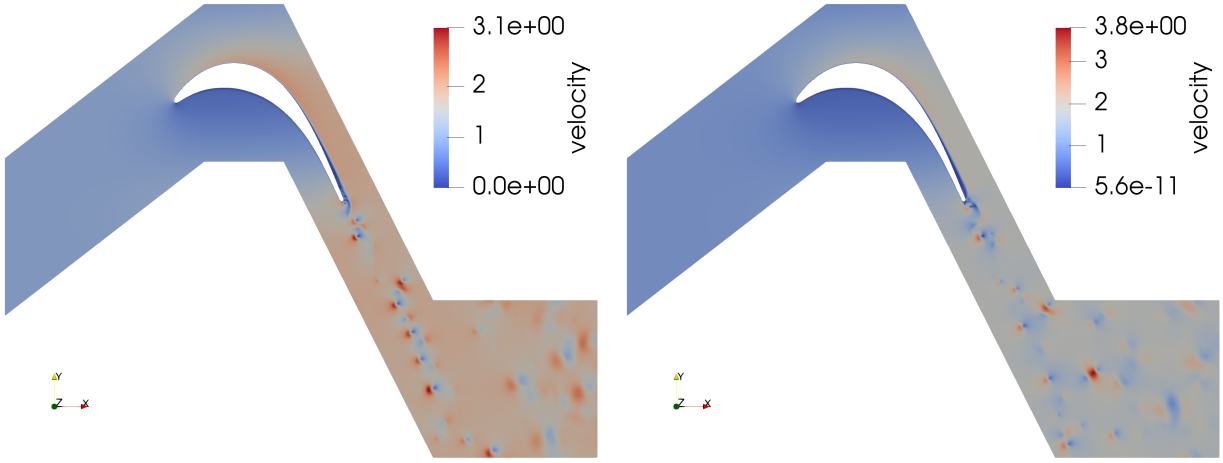


Figure 23: Instantaneous velocity field obtained with strong boundary conditions on blade wall (left) and weak boundary conditions (right).

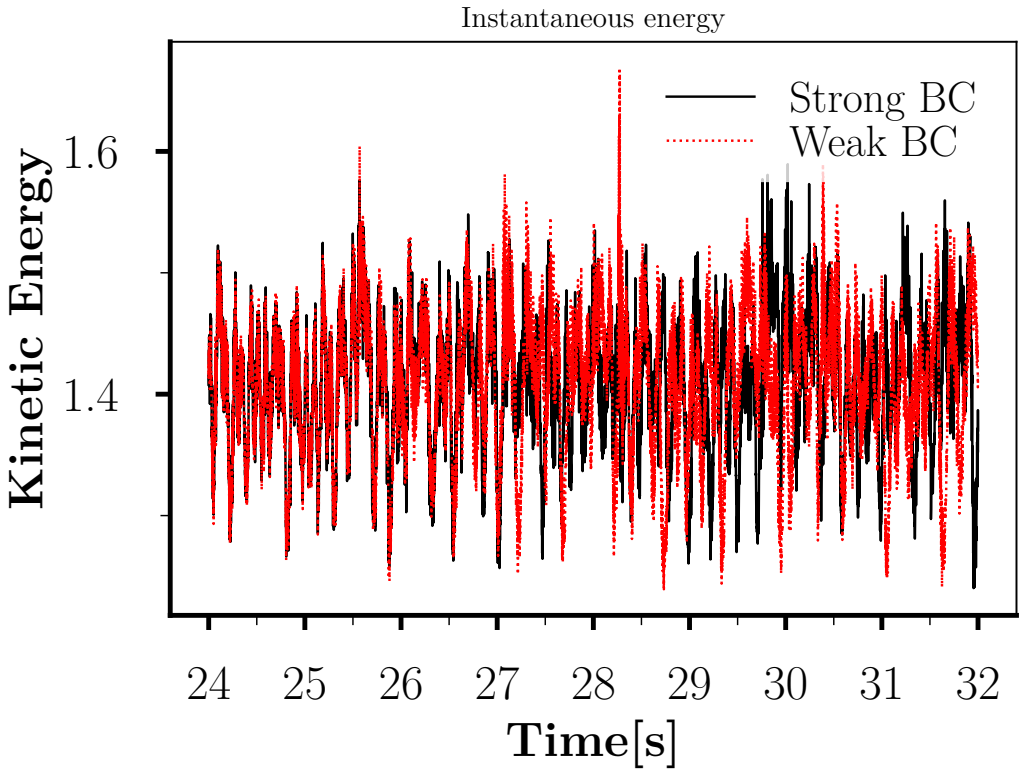


Figure 24: Kinetic energy vs. time.

3.15 Discretization of global transmission condition in the CG-HDG algorithm

The previous section detailed the formulation and implementation of weak Dirichlet boundary algorithm, which constitutes the second part of the algorithm. The first step, however, requries the assembly of the global transmission condition (i.e. the CG-HDG equivalent of equation (39)) and the solution of the corresponding linear system. Contrary to our initial expectation, however, the performance analysis of this step indicates that it would be significantly slower that the classical HDG (or statically condensed CG) solver. For this reason, we decided not to follow through with the complete implementation. More detailed analysis pertaining to the performance of the CG-HDG algorithm can be found in deliverable D2.4. We note, however, that the weak boundary condition formulation is useful as such as already mentioned in the introduction.

4 Compression algorithms

A steady increase in computing power has enabled scientists and engineers to use progressively more complex models to simulate a myriad of fluid-flow problems. As we approach the end of Moores Law, however, a significant gain in performance can only be achieved by an increase in core-count. Concurrently with higher levels of parallelism, the overall efficiency of modern high-performance systems is reduced by an ever widening I/O bottleneck. Developing I/O strategies that can decrease the data footprint of numerical simulations is therefore of paramount importance.

Since effective data storage is a pervasive IT problem, much effort has already been expended on developing and refining compression schemes for various applications. Prominent algorithms that allow for the lossy compression of structured data-sets can be found in the world of entertainment technology. In this context, Schmalzl, Loddoch[51] and Lindstrom[50] have extended the Joint Photographic Experts Group (JPEG) standard for volumetric floating-point arrays. These compression algorithms are simple and efficient in exploiting the low-frequency nature of most numerical simulations. The non-local basis functions of the discrete cosine transform, however, will result in a heavily distorted data-set [1].

In contrast to the base-line JPEG standard, JPEG 2000 (JP2) employs a lifting-based one-dimensional discrete wavelet transform (DWT) that can be performed by either the reversible LeGall-5/3-Wavelet (5/3-CDF-Wavelet) for lossless or the non-reversible Cohen-Daubechies-Feauveau-9/7-Wavelet (9/7-CDF-Wavelet) filter for lossy coding [19]. The resulting time-freqeuncy representation facilitates random access as well as region-dependent rate-distortion-control operations, reducing the introduction of large-scale artifacts during a lossy compression stage. Furthermore, the JP2 standard offers an embedded block-coding algorithm that generates a quality- and resolution-scalable bit-stream which is optimally truncated with regards to the induced distortion [1].

Efforts have already been made to adapt the full range of JP2 features to floating-point numbers. Usevitch [71] proposed an extended integer representation of single precision IEEE

754 floating-point numbers which stores the 24 consecutive mantissa bits losslessly using 278 bit locations. While this approach requires little alteration of the base code, the extended integer representation results in a significant increase in memory consumption and overall decrease in compression performance. Gamito [32], on the other hand, split the floating-point values into their sign, bit and mantissa fields. The so-called shape-adaptive discrete wavelet transform is then applied inside the smooth regions of the exponent and mantissa fields to avoid high-frequency information that is introduced during the splitting operation. The increase in computational complexity the shape-adaptive wavelet transform incurs, however, limits the usefulness of this approach for large-scale CFD simulations.

Since Lindstrom [50] demonstrated that lossy compression of numerical data-sets is not only possible but also necessary to achieve high compression ratios, we chose to develop a lossy encoder based on the JPEG 2000 standard. Our goal was to develop an algorithm that is able to conserve most of the internal energy of the fluid domain and minimize the introduction of compression artifacts to support both visual and statistical evaluation. To allow for easier storage, the compressed bit-stream was required to embed all the information needed to decompress the simulation file. The resulting wavelet based compression algorithm (WBC), as described in deliverable 1.2, can efficiently compress a wide variety of CFD problems without sacrificing crucial flow-field information. We here give a brief summary of the work that has been done for deliverable 1.2 and highlight the improvements to the compression code that have been implemented since then (see subsubsection 4.1.1).

In subsection 4.2, we will evaluate the existing mathematics method, called singular value decomposition (SVD), and emerging new ideas. Special attention will be paid to algorithm that identify, extract and preserve physical structures in the flow field for instance. This alone has the potential of reducing the initial "raw" data by more than an order of magnitude. SVD has already been surveyed during this project.

To allow for a quantitative and qualitative evaluation of our data-compression strategies, the following parameters were selected. The most popular metrics used to measure the performance of a data compression algorithm is the compression size. It is defined as the ratio of the number of bits used to represent the original to the number of bits used to represent the compressed bit-stream:

$$CR = \frac{Uncompressed\ Size}{Compressed\ Size}. \tag{52}$$

With regards to the SVD algorithm, the compression ratio can be rewritten as

$$CR = \frac{m \times n}{m \times r + r + r \times n} = \frac{m \times n}{r \times (m + 1 + n)}, \tag{53}$$

where m and n define the rows and columns of the original matrix and r the rank of its approximation. The goal of the singular-value decomposition is to reduce the storage required for the compressed data-stream when compared to the original data-set. Therefore, the limit

on the value of rank r is given by:

$$r < \frac{m \times n}{m + 1 + n}. \tag{54}$$

where r defines an integer value. From this follows that a reduction in rank r will result in an increase in compression ratio and a decrease in the flow field reconstruction quality. This implies smaller ranked SVD approximations are preferable as long as the compression induced distortion remains minimal.

To assess the overall quality of the decompressed files we used the mean-squared error (MSE) and peak signal-to-noise ratio (PSNR) metric. The mean squared error is defined as the average square of the difference between the original matrix I and its approximation I':

$$MSE \; = \; \frac{1}{ijk} \sum_{x=1}^{i} \sum_{y=1}^{j} \sum_{z=1}^{k} |I(x,y,z) - I'(x,y,z)|^2, \tag{55}$$

where i, j, k represent the row, column and slice of a three-dimensional matrix. The compression induced distortion between the original and reconstructed data is measured using the peak signal-to-noise ratio. It can be defined as the ratio of maximum signal to the corrupting noise power. PSNR is usually expressed in terms of the logarithmic decibel scale (dB) and is most easily defined via the MSE:

$$PSNR \; = \; 20 \log_{10} \left(\frac{max(I(x,x,z)) - min(I(x,x,z))}{\sqrt{MSE}} \right), \tag{56}$$

Typically, a PSNR of 35 dB or higher is a good indicator for a well reconstructed data-set [34]. In our own investigation we found that a PSNR of 50 dB or higher is needed for the evaluation of turbulence statistics.

We note that the sections below serve to summarise some of the main development efforts made within the ExaFLOW project. A full description of the mathematical formulation can be seen in the previous deliverable D1.2, and a more thorough investigation of the initial implementation of these methods can be seen in D2.4.

4.1 JPEG 2000

4.1.1 Theory

The fundamental structure of our wavelet-based compressor is shown in Figure 25. Since it has been adapted from the integer based JP2 standard, our first course of action was to transform the volumetric floating-point into an integer array. This was accomplished by using the fixed-point number format Q. To this end, the dynamic range of each flow-field variable is first centered around zero. This is done to reduce the number of significant bits needed to represent the data samples and exploit the full range of the number format Q.

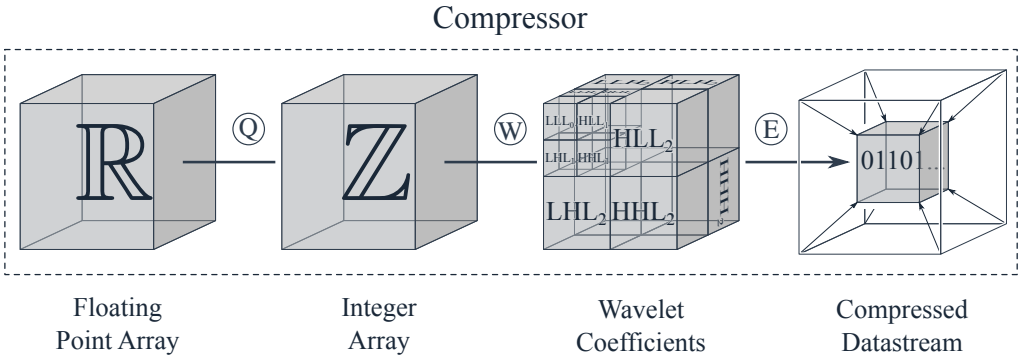


Figure 25: Structure of wavelet-based compression algorithm for volumetric floating-point arrays. Encircled letters indicate the floating-point to fixed-point transform (Q), discrete wavelet transform (W) and entropy encoding stage (E).

Next, all values are normalized to the range (-1, +1) and then multiplied by the number of fractional bits Q_m. In this context, Q_m can be used as a primary rate control parameter by decreasing the information entropy of the volumetric floating point array. On the other hand, the number of fractional bits should not exceed the width of the integer type used to store the fixed-point values. For our purpose, we use $Q_m = 28$ to prevent an integer overflow during the subsequent DWT stage and to allow for a fast compression scheme.

The next step was to generate a time-frequency representation of the transformed data samples. Here, the discrete wavelet transform is applied to the volumetric integer field to decorrelate its inherent spatial frequency information. The discrete wavelet transform implemented in our compression algorithm is the 9-tab/7-tab real-to-real filter bank, commonly known as the 9/7-CDF-Wavelet. The 9/7-filter bank is split into two predictor (high-band) and two update (low-band) operations, followed by a dual normalization step:

$$\begin{aligned}
y(2n+1) &\leftarrow x(2n+1) + (\alpha \times |x(2n) + x(2n+2)|),\\
y(2n) &\leftarrow x(2n) + (\beta \times |y(2n-1) + y(2n+1)|),\\
y(2n+1) &\leftarrow y(2n+1) + (\gamma \times |y(2n) + y(2n+2)|),\\
y(2n) &\leftarrow y(2n) + (\delta \times |y(2n-1) + y(2n+1)|),\\
y(2n+1) &\leftarrow -K \times y(2n+1),\\
y(2n) &\leftarrow (1/K) \times y(2n).
\end{aligned} \tag{57}$$

Here, $\alpha = -1.59$ and $\gamma = 0.88$ are the coefficients for the first and second predictor stage. The coefficients $\beta = -0.053$ and $\delta = 0.44$ define the first and second update stage. The dual normalization step is described by the coefficient K = 1.23 [60]. Since the discrete-wavelet-transform is a one-dimensional transformation defined for unbounded signals, we can extend the transformation stage to volumetric data-sets by applying the analysis filter-bank along each spatial dimension successively. For bounded data-sets, the undefined samples outside of the finite-length signal need to be related to values inside the signal segment. For odd-

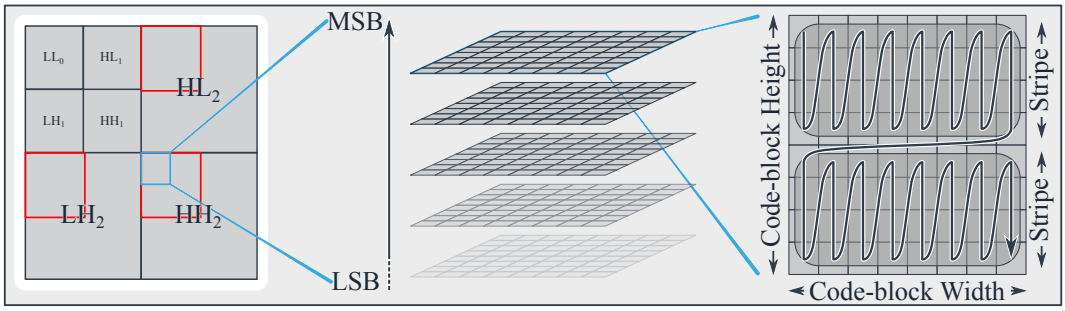


Figure 26: 2-dimensional representation of geometric operations performed during Embedded Block Coding with Optimized Truncation stage. Red squares signal a precinct, blue squares a code-block.

length filter taps, this is achieved by applying a whole-point symmetric extend on the signal boundaries [19].

Finally, each wavelet sub-band is independently encoded using the Embedded Block Coding with Optimized Truncation (EBCOT) algorithm described in the JPEG 2000 standard. First, the wavelet coefficients are rounded down to the nearest integer. Next, each sub-band is subdivided into non-overlapping cubes (see Figure 26). For every wavelet level, spatially related rectangles from the 7 high-bands form a precinct. Each precinct is further divided into $32 \times 32 \times 32$ sized code-blocks, which form the input signal of the entropy encoding stage. These code-blocks are then split into their bit-plane fields. The bit-planes are scanned in a zigzag pattern, starting from the most significant bit-plane (MSB) to the least significant bit-plane (LSB). Each bit-plane is encoded using three distinct coding passes: The significant propagation, refinement and cleanup pass. During these coding operations, every bit of a bit-plane is only encoded once in one of the three coding passes. The sign bit is encoding once a coefficient becomes significant for the first time. In a post-processing operation, the encoded bit-stream is subdivided into so-called quality layers. These quality layers represent optimized bit-stream truncation points that minimize the distortion for a specific bit-rate. Each successive quality layer will monotonically improve the quality of the data-set reconstruction. The encoded information for each code-block is distributed across all layers. Rate control is handled by defining quality layer 0 to be rate-distortion optimized for the specified compression ratio.

4.2 Singular Value Decomposition

SVD is one of the most useful tools in matrix algebra and includes the concept of the eigenvalue/eigenvector decomposition and is as a technique offering adequate approximations to represent fluid flows with reduced data/dimension.

We start with the definition of SVD. The SVD allows an exact representation of any $m \times n$ data matrix A with rank r $(r = min(m, n))$ in the following form [23]

$$A = U\Sigma V^T, \Sigma = diag(\sigma_1, \sigma_2, \ldots, \sigma_r), \tag{58}$$

where the columns of U and V are orthonormal with $U^TU = I = V^TV$. Σ is a diagonal matrix of non-negative numbers $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0$, arranged in descending order. An equivalent way of writing is:

$$A = \sum_{r=1}^{min(m,n)} u_r \sigma_r v_r^T. \tag{59}$$

We can also find the fundamental theorem of SVD (with $m > n$) as shown in Figure 27 . The vectors u_r of the orthonormal U, called the left singular vectors, are the eigenvectors of AA^T with the associated eigenvalues σ_r. The vectors v_r of the orthonormal V, called the right singular vectors, are the eigenvectors of A^TA with the same associated eigenvalues σ_r.

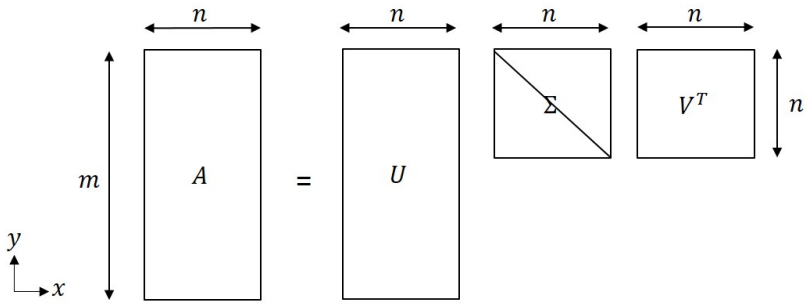


Figure 27: The form of Singular Value Decomposition

Using SVD, low-dimensional matrices may be used to represent a high-dimensional matrix. It makes easy to eliminate the less important parts of the representation to produce an approximation with any desired number of dimension, i.e. rank r. Obviously the fewer the dimensions are chosen, the less accurate will be the approximation[48]. The approximation is given by:

$$A = \sum_{r=1}^{min(m,n)} u_r \sigma_r v_r^T \approx \sum_{r=1}^{r} u_r \sigma_r v_r^T = A_r, \tag{60}$$

where A_r is the approximated matrix by using SVD. The low-dimension approximation of matrix A is illustrated in Figure 28. The product of low-dimension representations U_r, Σ_r and V_r equals the approximated matrix A_r.

The purpose of applying SVD is to replace a larger original matrix by three other matrices whose sizes are much smaller than the original, but from which the original can be approximately reconstructed, usually by taking their product.

5 Fault tolerance

Algorithm and software resilience is now one of the greatest concerns in striving towards exascale and interruption, due to component failure, is now considered a major barrier to

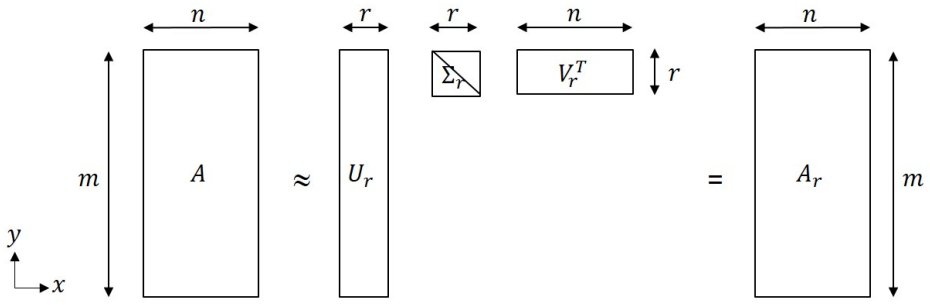


Figure 28: Data reduction with Singular Value Decomposition

effectively using an exascale system with current numerical codes [15, 68]. Both hardware and software errors, such as component failures or operating system crashes, may interrupt simulations or lead to non-deterministic results [65]. This is further exacerbated by the trend towards heterogeneous computing where nodes are composed of multiple processing components and additional system-level software layers. Failures typically necessitate a restart of the computation and results in wasted time, energy and resources. Even with the use of high-quality hardware, the number of components necessary to reach this level of throughput leads to an overall system failure rate of once every few hours.

In this section, we consider two approaches to implementing fault tolerant algorithms for fluid dynamics applications. The first, in section 5.1, considers a minimally intrusive approach to existing fluid dynamics solvers, building upon user-level failure mitigation extensions to the MPI library. In deliverable D2.4, a new C++/MPI library for fault-tolerance using checkpoints and automatic rollback is presented. The library, called Llama, uses multi-level layered group-local in-memory checksums to protect distributed arrays. Checksum encoding schemes are widely used to protect data from hardware failures. Data protected through the creation of checksum parity code is, however, only recoverable if the number of lost data and/or code blocks is smaller than the number of checksums parity codes created. In section 5.2, a newly developed method for partial information recovery in incomplete checksums is presented.

5.1 Minimally intrusive resiliance for transient solvers

A number of techniques already exist to improve the resilience of application codes in the event of system degradation or failure, including checkpoint/restart, redundant computing and application-based resilience. These methods can be classified as either forward recovery or backward recovery. In the former, the algorithm continues and corrects errors introduced by failures. Examples of this include redundant computing or some algorithm-specific approaches. Forward-correcting algorithms exist for some sub-components of the transient solvers considered, such as conjugate gradient solvers [2], but these approaches do not typically provide a comprehensive solution in the case of an error occurring outside of these components. Furthermore, they require a significant intrusion into the application code. In

contrast, backward-recovery rolls back to the last previously recorded globally consistent state and repeats calculations. Without the addition of resilience, the failure of a single MPI process would typically lead to the termination of the entire simulation, requiring a complete restart and backward-recovery from the last checkpoint.

Existing approaches to incorporating resilience in scientific codes involve substantial modifications to the application code in order to add protection mechanisms to all the necessary data structures and, in some cases, may require a complete redesign. Many of the demonstrators of these approaches are stencil applications, written specifically for illustrating the resilience algorithm and are not necessarily representative of production codes.

In the event of a process failure (e.g. due to a hardware fault) we would like to avoid the complete restart of the simulation on all processes, avoid checkpointing to disk and instead substitute the failed process with a *spare* process which recovers from data provided by a surviving process in order to continue the calculation. User-Level Failure Mitigation (ULFM) [8, 9] is a proposed extension to the MPI 4.0 standard which adds fault tolerance semantics for *application-based* recovery. ULFM provides three key additions to the MPI API. The `MPIX_Comm_revoke` method invalidates a communicator and allows a process to notify other processes that a failure has occurred, for example to initiate recovery. The `MPIX_Comm_shrink` method then reconstructs a revoked communicator containing failed processes into a working communicator with those failed processes omitted. Finally, `MPIX_Comm_agree` implements an agreement algorithm, performing a logical AND operation on the boolean parameter across processes; this succeeds even if there are failed processes.

Here, we instead outline our low-intrusion application-based resilience approach, building on ULFM, specifically for tightly-coupled transient solvers. We illustrate our strategy through a prototype implementation in Nektar++, a production-ready high-order spectral/hp element framework for the solution of a wide range of partial differential equations [13], which is described further in deliverable D2.4.

Full details of the algorithm and performance analysis have been published [14].

5.1.1 State Protection

The resilience approach we describe here is independent of the particular numerical discretisation used (finite difference, element, volume, etc), time-integration scheme and the specific time-dependent PDE to be solved. In general, the implementation fo solvers for PDEs can be broken down into two distinct phases. The first is a set-up phase in which the necessary discrete operators or stencils are constructed. These are typically in the form of matrices which remain fixed throughout the simulation. The cumulative storage of such operators is often large, compared to solution vectors, and their construction requires global communication in order to associate neighbouring degrees of freedom across partitions. The second phase is the time-advancement of the initial conditions to the final state. The value of the solution variables change during time integration but the discrete operators do not.

To leverage the capabilities of ULFM, we provide a custom error handler on all communicators, rather than allowing MPI to automatically defer to calling `MPI_Abort` when a failure occurs. Execution of the application proceeds as normal but with a number of additional

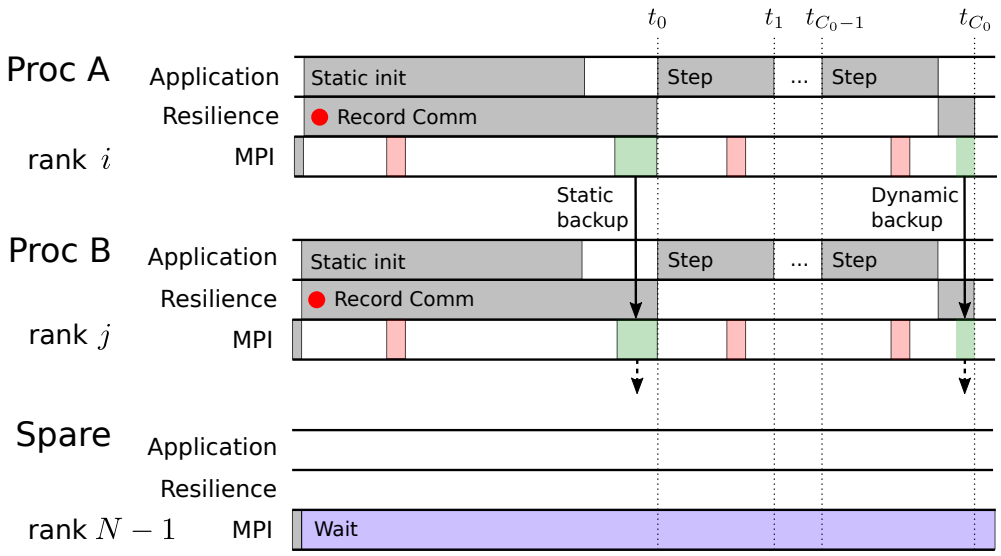


Figure 29: Diagrammatic representation of the protection algorithm. Initialisation of solver, showing three processes – two active and one spare – with ranks i, j, and N, respectively. Rank i communicates static recovery data to rank j. After a number of steps, at time t_{C_0}, a remote in-memory checkpoint occurs. The spare rank, N remains idle throughout. Red MPI regions denote collective communication, while green regions denote pairwise communication.

ranks allocated. The set of all processes are partitioned into *worker* and *spare* ranks immediately after MPI is initialised using `MPI_Comm_split`. This creates a sub-communicator in which the worker processes participate. Spare processes are assigned to a null communicator and wait until a failure occurs. They should proceed only on two events: a failure occurring, or; the application terminating normally in which case `MPI_Finalise` must be called. To achieve this on spare processes we call `MPI_Comm_agree` with a value of *true* immediately after `MPI_Init`. On worker threads, the same call is made immediately prior to `MPI_Finalise` with a value of *true* or, in the event of an error being detected, from the error handler with a value of *false*. Therefore, if the resulting conjugation evaluates to true, the application code must have completed successfully. If it evaluates to false, this implies a process has failed, identifying that recovery is required.

In order to enable recovery on a replacement MPI process, the data structures within the code must be protected in a way which allows their reconstruction on a spare process. State protection is depicted diagrammatically in Fig. 29. In our approach, rather than writing to disk, we opt for remote in-memory check-pointing. Data on each process is backed up to a partner process which requires only pairwise communication and is denoted by green blocks in Fig. 29. This 'buddy' process is chosen to balance resilience and performance and is typically on an adjacent node.

We do not store the initialised static data structures themselves, but rather the outcome of any MPI communications performed by the application during the static phase, indicated by *Record Comm* in Fig. 29. This can be achieved by intercepting calls to the MPI API and

logging the result, if applicable. This provides two key advantages: the volume of data is anticipated to be smaller than the fully initialised data structures whose generation invoked the communication, since exchanges occur along partition boundaries rather than within the partition volume, and; very little modification is required to the existing application code. To complete this aspect, we must annotate the code (through function calls) to mark the beginning and end of the initialisation phase.

Dynamic data checkpointing is performed at regular intervals after the end of the initialisation phase. These data typically consist of the solution vectors at the time of checkpointing only and are relatively small in size compared to static data. These are protected through duplication to the memory of a partner node.

5.1.2 State Recovery

The first task is to modify the behaviour of the MPI routines in the event of a failure. We specify our own error handler function to be called by MPI in the event that any process detects another process has failed through any communicator. The primary role of this handler is: to revoke the communicator, thereby ensuring all other processes become aware of the failure, and; to throw an exception which propagates up the call tree to a suitable point in the application code in which backward-recovery can be managed. For transient simulation codes, this is typically the outer time-integration loop.

The second task is to enrol spare processes to replace those which have failed and rebuilding all communicators used in the application code. A call to `MPI_Comm_agree` unlocks the spare processes (see above) so they can participate in the enrolling process. The set of all processes is then shrunk to omit those which have failed, MPI group operations are used to determine the failed ranks, and spare processes are reassigned the ranks of the failed processes using `MPI_Group_translate_ranks`. A complete set of worker processes is then selected using `MPI_Comm_split` and any other sub-communicators are similarly reconstructed.

The application must now achieve a globally consistent state on all processes. Surviving processes simply rollback their dynamic data to the last in-memory checkpoint. Spare processes must recover both the static and dynamic data. This aspect of the algorithm is shown in Fig. 30.

A prototype implementation of the algorithm has been developed in Nektar++. We describe the implementation of this algorithm and analyse its performance in the work package 2 deliverable 2.4.

5.2 Partial Information Recovery with Incomplete Checksums

Hardware is inherently failure prone. For data centers and cloud services, protecting user data has been essential for as long as the services have existed. In early days, the standard approach was protection through triple redundancy. At any given moment, three separate hard drives on separate systems would store any given data[54, 36]. If at some point, a drive would fail, the two remaining hard-drives are to make a new copy by sending their data over the network to a new failure-free machine. The process of copying the full content of a

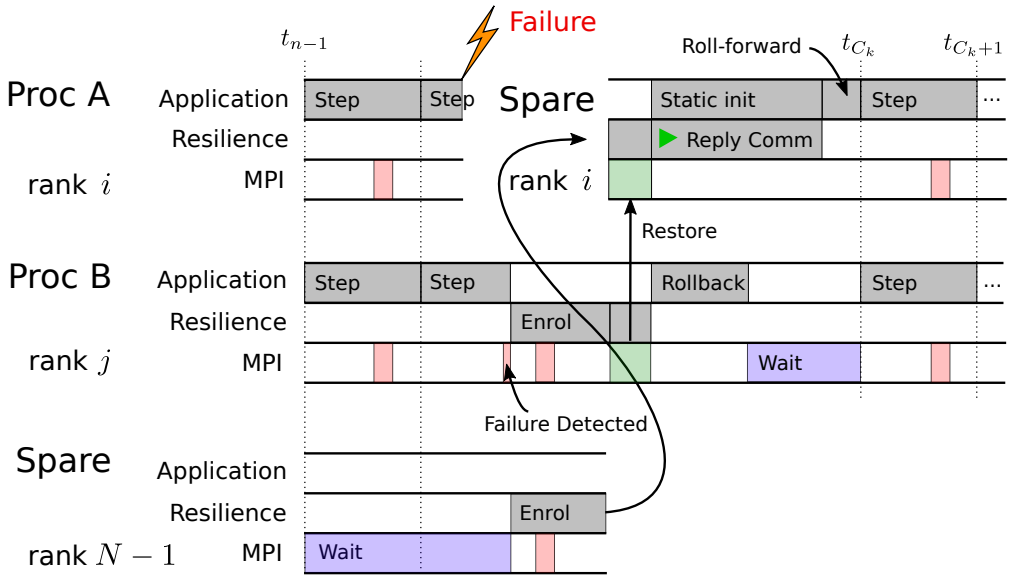


Figure 30: Diagrammatic representation of the recovery algorithm. Process A with rank i fails and, after enrolment and rank translation during recovery, process S is assigned rank i and receives recovery data from rank j. Static and dynamic data is subsequently recovered to the last checkpoint at time t_{C_k} without requiring any further communication with surviving ranks. The simulation then continues. Red MPI regions denote collective communication, while green regions denote pairwise communication.

hard-drive over network fabric might take an hour or so, depending on the state of network congestion.

With triple redundancy, three hard-drives on separate systems would have to all fail within the span of roughly an hour for the data to be lost. Consumer grade hard disk drives (HDD) have an annualized failure rate of 2 to 5 percent when running 24 hours a day, 365 days a year[64]. With these ballpark numbers, the probability of loosing one hard-drive worth of data within a year is in the order of $\sim 10^{-15}$. Thus, even with a very large number of hard-drives, the probably of loosing data becomes small. Newer Solid State Drives (SSD) are unlikely to lessen the need for failure protection techniques meaningfully. Studies have shown that they have lower annual replacement rate of 1-3 percent, but in turn they have a higher rate of uncorrectable data errors [66]. The problem with the approach of redundancy for protecting data is that it is very expensive. The main cost of running a data-center is due to energy consumption and hardware purchase. Triple redundancy essentially triples the cost of running the service.

For said reasons, the industry no longer rely solely on redundancy as a mean of fault-tolerance. Instead, erasure codes have found use. Erasure codes are a form of forward error correction that take data consisting of k symbols and turn it into a larger data set with n symbols such that the original data may be recovered from a subset of the n symbols. Erasure codes for which any k symbols are sufficient to recover the original data are called optimal, these codes are as resilient as possible, but typically scale quadratic in terms of coding and decoding complexity with respect to n. The $n - k$ extra symbols created are typically referred to as parity codes or checksums. The most commonly used erasure code is ReedSolomon error correction[61], used in RAID6 and in various storage services.

The advantage of using Reed-Solomon erasure code instead of redundancy is that it is relatively memory efficient. Imagine three hard-drives full of data that one would like to protect, i.e. $k = 3$, say another three hard-drives are used to store parity code, i.e. $n = 6$. We'd have to lose four drives among the six, before the data on the lost hard-drives become truly lost. This comes at the cost of a 100 percent overheard in hard-drive usage. Compare this to the triple-redundancy approach, here nine hard-drives would be required corresponding to a 200 percent overheard, this despite the two approaches having essentially the same probability of data-loss. The Reed-Solomon approach is thus very efficient in terms of disk-usage. The trade-off is that Reed-Solmon requires encoding and decoding of data, which increase both CPU usage and network congestion. In practice, major data-centers are typically using various forms of hierarchies of parity codes, in combination with redundancy techniques, to mitigate issues of network congestion in particular[35, 62, 59].

The use of erasure code is beginning to find its way into HPC as well. In the context of fault-tolerance, the interest is to protect data in-memory to avoid the comparatively slow parallel-file-system; this makes the memory conserving property very attractive. In addition, most clusters used for large-scale applications are equipped with high-bandwidth low-latency networks, which serves to somewhat alleviate the cost of encoding data. In the Llama library (see deliverable D2.4), erasure codes were used for this exact purpose to enable a library for fault-tolerance to create memory-conserving in-memory checkpoints.

Unfortunately Reed-Solomon too has limits, even optimal erasure codes can not recreate lost data if the number of lost hard-drives, or nodes, k_l is greater than $n - k$. Up until this point, all data lost can be recreated with bit-wise accuracy, but after, there's no known way of recreating the lost data. This is unfortunate. In the context of HPC fault-tolerance through light-weight checkpointing, as demonstrated in D2.4, this means one must revert to a more resilient checkpoint such as one written to the parallel-file-system. In the context of data-centers and cloud services protecting costumer data, it is unfortunate because they are forced to use sufficiently many resources to ensure that the probability of this situation happening is infinitely small.

The decoding procedure can be thought of, in very simple terms, as the procedure of solving a linear system. In order to recover lost data, one must solve a system for which the number of columns is the number of lost symbols k_l, while the number of rows is the number of parity code symbols $n - k$. The reason that it becomes impossible to recover the data when $n - k < k_l$ is that the system to solve become under-determined. If real numbers were used in the encoding scheme, infinitely many solutions to the decoding procedure would exist. If, as in the case of Reed-Solomon, the encoding matrix and symbols to encode str in a Galois field, the number of possible solutions would still be finite, but there would be no unique solution.

Despite the fact that no unique solution exists to the decoding problem when $n - k < k_l$, one could argue that the remaining equations still hold information about the structure of the lost data. Even though the number of possible solutions is infinite, the number of solutions not admissible is likewise infinite. One might speculate that given some other knowledge about the data that was decoded, say knowledge of how one element in a vector tends to relate to another, could this information be used to impose other conditions in such a way that the system to solve yet again become well posed? In this section we present a study demonstrating a proof-of-concept. For reasons of simplicity, we consider an erasure code in the style of Reed-Solomon, but on real numbers.

5.2.1 The Weighted Checksum Scheme

Since real numbers can only be represented with finite precision on computers, most erasure codes such as Reed-Solomon, are designed to be applied to data represented as elements in a Galois field so that bitwise exact recovery is possible, thereby allowing the encoding/decoding mechanism to be agnostic with respect to what the bits represent.

In HPC fault-tolerance applications, some form of numerical data is often what needs to be protected. A vector, or matrix, filled with real numbers. This data could be treated as bit-streams using Reed-Solomon encoding, but parity codes could also be generated directly from the floating point numbers. In [46] the authors list a number of advantages in doing so. A main argument is that one avoids the trouble of introducing Galois Field arithmetic in the encoding and decoding procedure, instead being able to rely on standard matrix operations for floating point numbers. The disadvantage is the introduction of round-off errors during the recovery procedure due to the limited precision with which floating point numbers may be represented. The latter may however be of limited concern since it has been shown that

the loss of accuracy can be limited with a clever choice of checksum encoding matrix [16, 46].

In the introduction below, we use k to indicate the number of separately stored data, m to indicate the number of parity codes to compute, i.e. checksums, and $n = m + k$ the total number of data and code blocks. In the context of the example given in the previous section, this corresponds to a total of n hard-drives, among which k separate hard-drives are used to store the data that must be protected, and m hard-drives are used to store checksum parity code.

Consider a vector $\bar{x}$ of length d times k. Let's assume that this vector is partially stored in k equal parts on k independent hard-drives. Then the "sub-vector" stored on each hard-drive contain d elements. For ease of notation, let $\bar{x}^i$ denote the i'th sub-vector. A simple way of protecting the content of the vector against hardware failures is to compute an element-wise sum, i.e. a new vector $\bar{c}$, also containing d elements,

$$\bar{c} = \sum_{i=1}^{k} \bar{x}^i \tag{61}$$

and then storing $\bar{c}$ on a separate hard-drive, $M = 1$. We will refer to the vector $\bar{c}$ as the checksum hence forward. If at some point we were to lose a hard-drive $k_l \in \{1, k\}$. No matter which one it is, the vector $\bar{c}$ can be used to recover the data through a summation on the form

$$\bar{x}^{k_l} = \bar{c} - \sum_{i \neq k_l}^{k} \bar{x}^i \tag{62}$$

The above approach is extendable to create a simple scheme for protection against multiple failures, called a weighted checksum scheme. Suppose that we can afford to store m checksum vectors on m separate hard-drives, we would then compute each checksum vector $\bar{c}^i$ as such

$$\begin{cases} a_{1,1}\bar{x}^1 + \ldots + a_{1,k}\bar{x}^k & = \bar{c}^1 \\ & \vdots \\ a_{m,1}\bar{x}^1 + \ldots + a_{m,k}\bar{x}^k & = \bar{c}^m \end{cases} \tag{63}$$

where $a_{m,k}$ are some weights to be chosen. The matrix $A = (a_{i,j})_{m,k}$ is called the checksum matrix. If multiple hard-drives fail, how could the data be recovered? Assume, without loss of generality, that all of the first k_{lost} hard-drives have failed, and the all subsequent hard-drives have survived, we may then derive an equation to recover the sub-vectors $\bar{x}^1, \bar{x}^2 \ldots, \bar{x}^{k_{lost}}$ lost by restructuring (63) to arrive at

$$\begin{cases} a_{1,1}\bar{x}^1 + \ldots + a_{1,k_{lost}}\bar{x}^{k_{lost}} & = \bar{c}^1 - \sum_{t=k_l+1}^{k} a_{1,t}\bar{x}^t \\ & \vdots \\ a_{m,1}\bar{x}^1 + \ldots + a_{m,k_{lost}}\bar{x}^{k_{lost}} & = \bar{c}^m - \sum_{t=k_l+1}^{k} a_{m,t}\bar{x}^t \end{cases} \tag{64}$$

We refer to the coefficient matrix of the left hand side matrix in the above linear system, consisting of k_{lost} columns of A, as A_r. For recovery to always be possible, the coefficients

of the weighted checksum matrix A must be chosen in such a way that for any possible A_r, a unique solution is guaranteed to exist. I.e., the elements in the checkpoint matrix A must be chosen so that any sub-matrix of A is non-singular as this guarantees that A_r will always have full rank. Many structured matrices such as Vandermonde matrix, Cauchy matrix, and Gaussian Random matrix satisfy this condition. Not all such matrices would necessarily be suited though, it is also important that any sub-matrix A_r is well conditioned. If A_r has a high condition number, round off errors will accumulate and reduce the accuracy of the recovered data[5]. Gaussian Random matrices are both well conditioned and satisfy the condition that any submatrix is non-singular[21]. They are therefore a natural choice as noted in [46], and will be used as the checksum matrix for all numerical experiments presented in this section.

5.2.2 Partial information recovery for incomplete checksums

The unfortunate limitation of the approach, as with all erasure codes, is that if the number of hard-drives lost k_{lost} is larger than the number of checksums m, there is no unique solution to the problem of recovering the lost sub-vectors $\bar{x}^1, \bar{x}^2 \ldots, \bar{x}^{k_{lost}}$ since the system (64) becomes under-determined. Of course, one could simply let m be very large to avoid that situation, but increasing m means increasing the overhead in terms of hardware and energy. Ideally, we'd like to keep the overhead due to data protection as low as possible. Therefore, it would be immensely practical if we could somehow magically find the right $\bar{x}^1, \bar{x}^2 \ldots, \bar{x}^{k_{lost}}$ among the infinite number of solutions to the under-determined system (64) when $m < k_{lost}$.

To appreciate how that might be possible, let's take a step back and consider again the case of having only a single checksum vector. If we are so unfortunate to lose $k_{lost} > 1$ number of hard-drives, the solution space of every element d in each lost sub-vector $\bar{x}^1, \bar{x}^2 \ldots, \bar{x}^{k_{lost}}$ is spanned by an $k_{lost} - 1$ dimensional affine hyperplane as can be deduced from (63). The solution space is infinitely large, so we can not naively recover the lost sub-vectors by direct computations. Here's an idea though, let's assume that the data vector $\bar{x}$ stored in a distributed manor on many hard-drives has some structure to it, and that we have some knowledge of what that structure is. Say, we might be informed that the content of the vector $\bar{x}$ represents points sampled from a $\mathcal{C}^\infty$ functional. Now, given this knowledge, what if, among the infinitely many solutions to (63), for each element dd in each of the lost data vectors $\bar{x}^1, \bar{x}^2 \ldots, \bar{x}^{k_{lost}}$, we choose the solution which makes, in some yet to be defined sense, the function that $\bar{x}$ represents as smooth as possible?

The essence of the approach is simple. Let $\dot{\bar{x}}$ represents the first order derivative of $\bar{x}$; now, find the sub-vectors $\bar{x}^1, \bar{x}^2 \ldots, \bar{x}^{kk_{lost}}$ that minimize $\|\dot{\bar{x}}\|_2$ under the constraint that all checksum equations must be satisfied. By writing $\dot{\bar{x}}$ as a function of $\bar{x}$ with the application of a finite difference stencil, and then rewriting the checksum equations (63) to depend on $\dot{\bar{x}}$ instead of $\bar{x}$, we are left with a well-posed convex optimization problem for $\dot{\bar{x}}$ that may be solved using standard methods. The approach worked fairly well, though the derivation is somewhat long, and becomes especially involved if to be extended to surfaces or volumes. In addition, due to the global nature of the optimization, it is computationally expensive. Upon further experiments, we found that though the underlying idea was right, a slightly

different path proved better.

Instead of formulating a problem that finds the smoothest possible solution that satisfy all checksum equations, we assume that some compressed data or reduced model of $\bar{x}$ is available, denoted $\tilde{x}$. We then formulate a different optimization problem, i.e., find the sub-vectors $\bar{x}^1, \bar{x}^2 \ldots, \bar{x}^{kk_{lost}}$ that minimize $\|\bar{x} - \tilde{x}\|_2$ under the constraint that all checksum equations must be satisfied. This approach turns out to both work well and, unlike the other approach, be particularly simple to formulate and solve. The method is introduced in section 5.2.3, and extended into an iterative scheme in section 5.2.4 to demonstrate partial data recovery for incomplete checksums of image data.

5.2.3 Uniqueness by Minimizing Distance to Inexact Data

Let's define $\hat{c}$ as the right hand side of (64), so that the equation for recovery may be written as

$$A_r \bar{x}^{1:k_{lost}} = \hat{c} \tag{65}$$

When $m < k_{lost}$, all possible admissible solutions z to the under-determined system above may be written as

$$\bar{z} = A_r^+ \hat{c} + \left[I - A_r^+ A_r\right] \bar{w} \tag{66}$$

where A_r^+ is the Moore-Penrose pseudo inverse of A_r. $\bar{w}$ is an arbitrary vector with k_l elements. The solution corresponding to $w = \bar{0}$, is the minimum norm solution, i.e., the solution for which $\|z\|_2$ is the smallest amongst all the admissible solutions. If $\bar{x}$ approximates a function that is never too far from zero, one might speculate that computing

$$\bar{x}^{1:k_{lost}} = A_r^+ \hat{c} \tag{67}$$

could potentially recover data that is close to what was lost. Let's take that approach a bit further, imagine that we have access to some reduced model or compressed data $\tilde{x}$ that approximates the original vector $\bar{x}$. Upon failure, we'd like to use this data in conjunction with the checksum equations to find an even better approximation to the lost data. Subtracting $A_r \tilde{x}^{1:k_{lost}}$ from (65) we recover

$$A_r \left(\bar{x}^{1:k_{lost}} - \tilde{x}^{1:k_{lost}}\right) = \hat{c} - A_r \tilde{x}^{1:k_{lost}} \tag{68}$$

Applying the M–P pseudo inverse A_r^+ on each side and isolating $\bar{x}^{1:k_{lost}}$ we arrive at

$$\bar{x}^{1:k_{lost}} = A_r^+ \hat{c} + \left[I - A_r^+ A_r\right] \tilde{x}^{1:k_{lost}} \tag{69}$$

where $\bar{x}^{1:k_{lost}}$ as computed in (69) is the solution that minimize $\left\|\bar{x}^{1:k_{lost}} - \tilde{x}^{1:k_{lost}}\right\|_2$ whilst satisfying the constraint $\bar{x}^{1:k_{lost}} = A_r^+ \hat{c}$.

Two experiment with this approach at partial information recovery for incomplete checksum equations are presented in Figures 31 and 32. The vector $\bar{x}$ contains data created from a smoothed random walk, it contains a total of 10.000 data points. The data vector $\bar{x}$ was split into $k = 100$ separate sub-vectors, each containing 100 data points, and protected through

the creation checksum vectors as in (63), encoding vectors element-wise. $\tilde{x}$ is created by taking the first 20 modes of a Fourier transform of the entire dataset $\bar{x}$. After encoding the m checksum vectors, the first $k_{lost} = 20$ sub-vectors were removed. The figure contains both the original data vector $\bar{x}$, the compressed data $\tilde{x}$, and the recovered data $\bar{x}^{1:k_{lost}}$ computed by (69). In Figure 31, the data was protected using $m = 15$ checksums, hence in the incomplete data recovery, 33.3% more sub-vectors were lost than checksum vectors created. In Figure 32, the data was protected using $m = 18$ checksums, hence in the incomplete data recovery, 10% more sub-vectors were lost than checksum vectors created. For all experiments, the checksum matrix A was a Gaussian Random matrix.

This preliminary result indicates that the method works well. An interesting aspect of the approach of ensuring uniqueness by minimizing the distance between the data to be recovered, and some inexact data, is that the algorithm may be used to feed itself in an iterative manor. An iteration could consists of first solving the constrained minimization problem, followed by the application of some regularization function that modifies the data towards some property, smoothness for example. In the next section, the method just derived is used to create an improved, iterative, self-feeding algorithm for incomplete data recovery.

5.2.4 An Iterative, Self-Feeding, Recovery Scheme

In the previous section we demonstrated that partial information recovery for incomplete checksums is possible when some compressed version, or reduced model, of the lost data was available to be used in conjunction with the under-determined checksum equations.

Here we present a further improved method in the form of an iterative algorithm. Computing (69) reconstructs the lost data by finding the solution, nearest to some guess, that satisfy the checksum equations. After performing this operation, one could continue the recovery procedure by applying some function, to the approximation found, that filters or modifies the solution in accordance with what meta knowledge one might have on the structure of the lost data. If, for example, it is known that the data to be recovered represents a continuous surface, one could apply a function that smoothe the solution a bit.

This new, filtered solution, will however in all likelihood no longer satisfy the checksum equations. So it could be fed back into (69) as a new, improved guess. In this way, one could alternate between the two, to potentially arrive at a better approximation to the data lost. In the enumerated list below, the iterative scheme is outlined step-by-step.

1. Choose an initial guess $x^* = \tilde{x}^{1:k_{lost}}$. This could be zero if need be.

2. Form the error equation

$$A_r \left(\bar{x}^{1:k_{lost}} - x^* \right) = \hat{c} - A_r x^* \quad \Rightarrow \quad A_r e = r^* \tag{70}$$

   Find the e with minimum euclidean norm $\|e\|_2$ among all admissible solutions.

3. Update $\bar{x}^{1:k_{lost}} = x^* + e$.

4. If $\|e\| \,/\, \|x^*\| < \varepsilon$, algorithm converged.

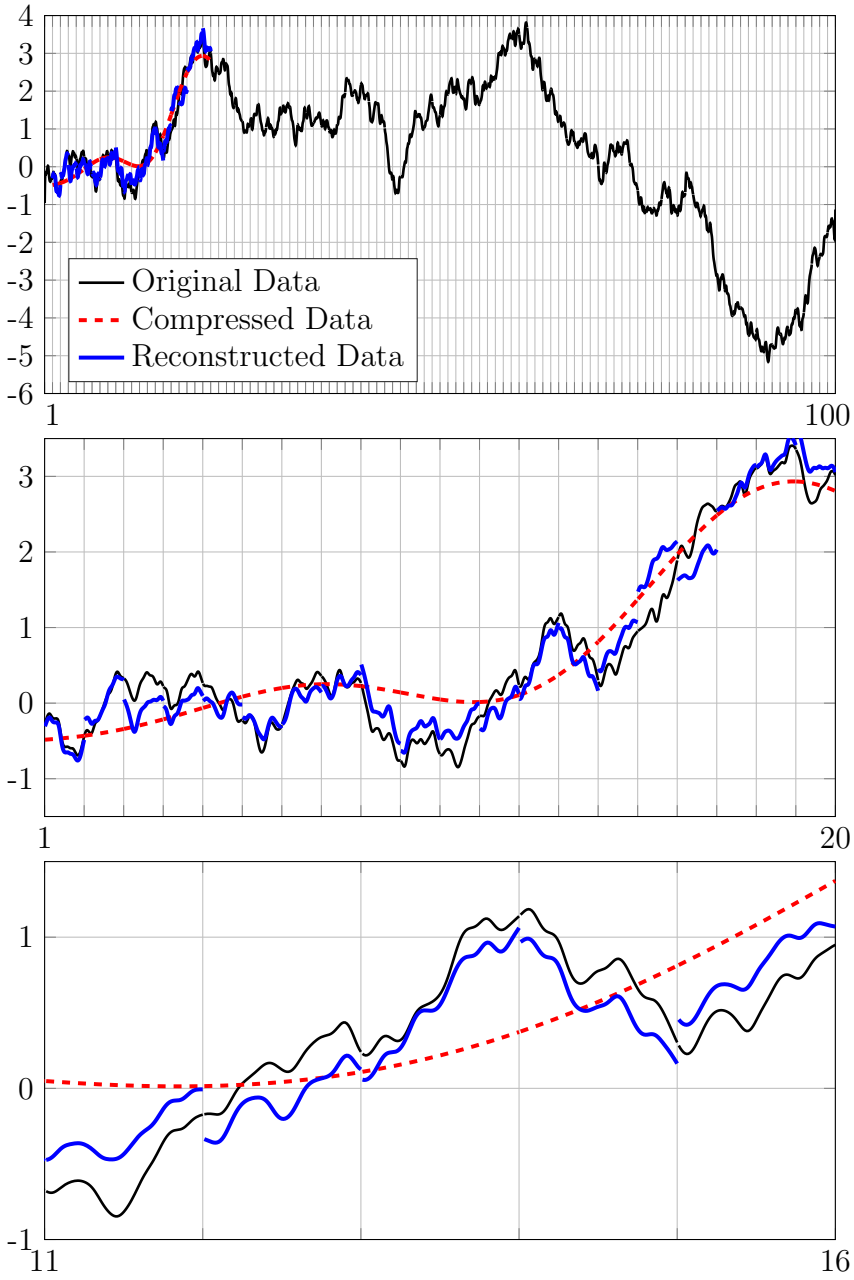


Figure 31: Numerical experiment testing the method (69) for partial information recovery in incomplete checksums. The data indicated by the black line was stored in $k = 100$ separate containers. $m = 15$ checksum vectors were created to protect the content of the containers. The $k_{lost} = 20$ first containers are removed, i.e. 33% more data vectors are lost than checksum code vectors created.

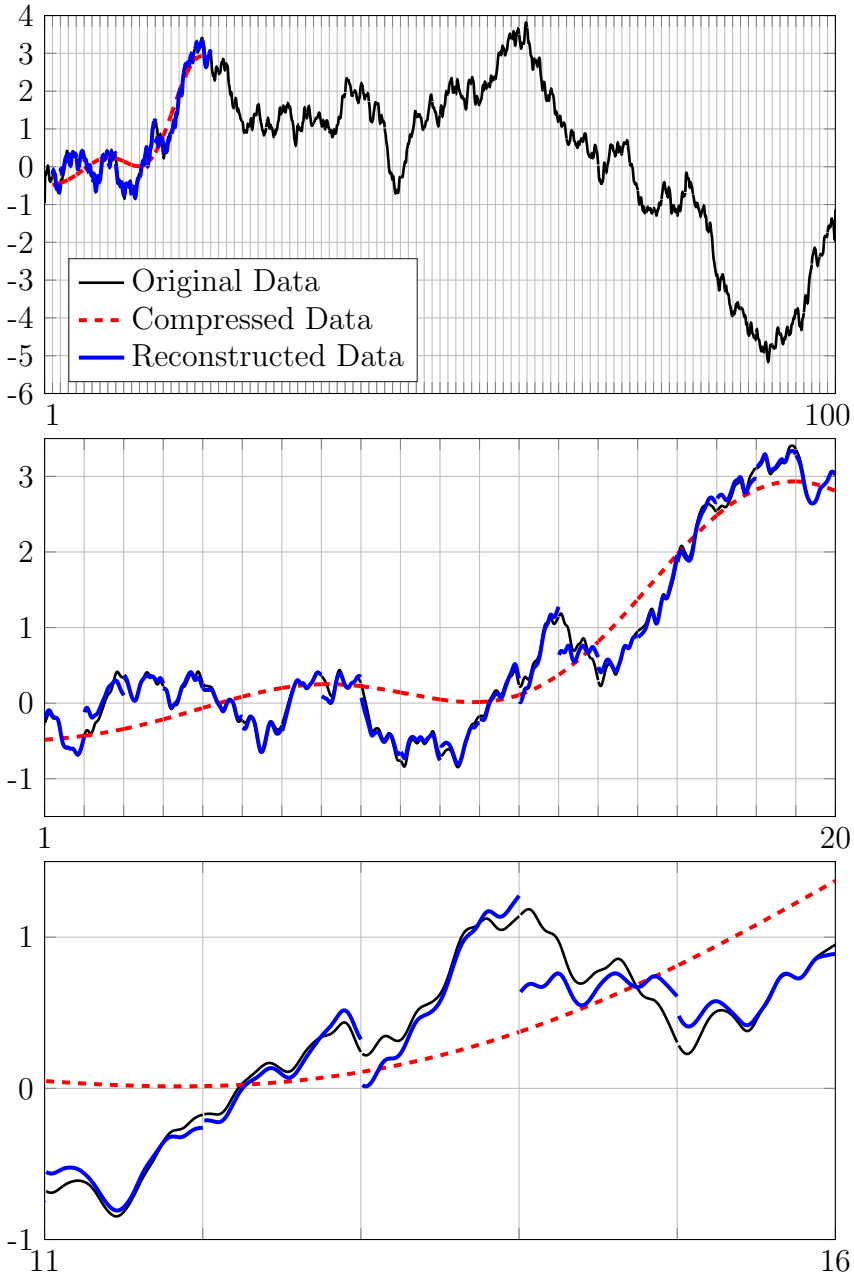


Figure 32: Numerical experiment testing the method (69) for partial information recovery in incomplete checksums. The data indicated by the black line was stored in $k = 100$ separate containers. $m = 18$ checksum vectors were created to protect the content of the containers. The $k_{lost} = 20$ first containers are removed, i.e. 11% more data vectors are lost than checksum code vectors created.

5. Otherwise, apply filter to $\bar{x}^{1:k_{lost}}$.

6. Set $x^* = \bar{x}^{1:k_{lost}}$, go to 2.

To test the method, we use it to recover images lost. In the test-case, all 16 images are stored separately in 16 data sets, each image being 512x512 pixels. Another 3 data sets, consisting of checksums, are computed element-wise, per color, using (63), to protect the images. The images used are depicted in Figure 33 in their original form. (69) are used for step 2-3 in the method outlined above.

In Figure 34, 4 images have been removed, and then recovered. The approximations recovered are almost indistinguishable to the originals in Figure 33. In figure 35, 6 images have been removed, i.e. the recovery procedure is attempting to recover having only half as many checksums parity sets as data sets lost. The results are still visually pleasing, although the recovered images have clearly been degraded in quality compared to the originals. In the test, no compressed data was used to initiate the algorithm. As a guess for the first iteration, a zero matrix was used.

An important thing to note about the results presented is that in the encoding procedure, and subsequent decoding procedure, a set of randomly generated checksum matrices A was used in a round-robin fashion, rather than just using a single A repeatedly as is normally the case with erasure codes. Doing so improved the quality of the recovered images substantially compared to using the same matrix A for all elements. In general, our observation was that the increasing the number of randomly generated checksum matrices used would also increase the quality of the images recovered. Using 4, or 16, randomly generated matrices was clearly better than using a single matrix, the improvements were found to be diminishing however, as using 64 matrices instead of 16 would yield results only marginally better.

5.2.5 Summary

In this section we set out to investigate to what extent it might be possible to find an approximate solution to the problem of recovering lost data from an under-determined checksum system, when having some knowledge of the underlying structure of the data encoded. We proposed a new method, and tested its application on a weighted checksum scheme for floating point numbers. Our preliminary finding is that the answer is yes, it is indeed possible to partially recover the data otherwise considered lost.

The data recovered when $k_{lost} > m$ is not of machine accuracy with respect to the original data, so for the approach to be of practical use in applications like the multi-level checksum checkpointing scheme presented in D2.4, one would need some way of quantifying the accuracy expected of the data recovered.

The method as presented works best when generating and using multiple checksum matrices. These checksum matrices needs to be stored as well since they are needed in the decoding scheme. The added memory overhead is however very small, in the example given the checksum matrices took up 0.8KB of space compared to 12.6MB for the image data.

Using Gaussian matrices to encode a checksum is somewhat of a niche in the context of fault-tolerance as it really only applies to floating point numbers where the exact bitwise

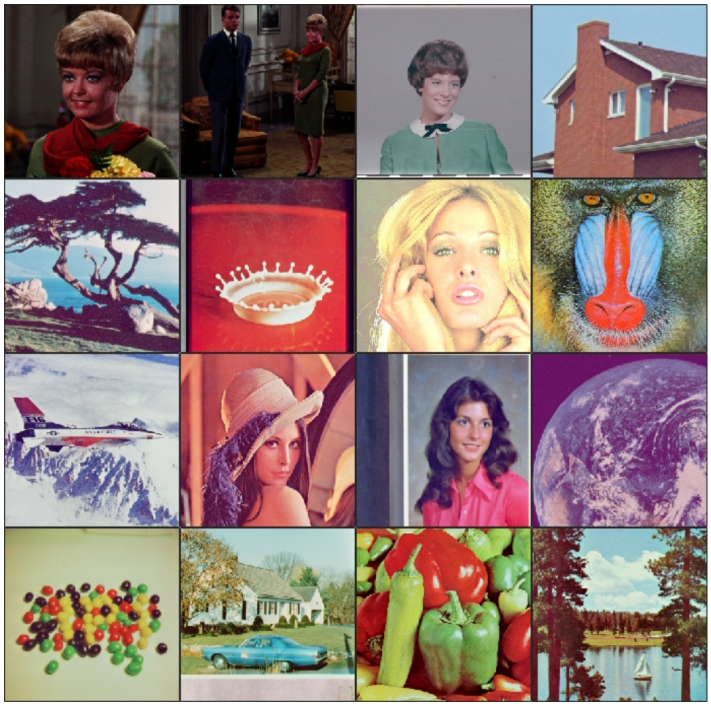
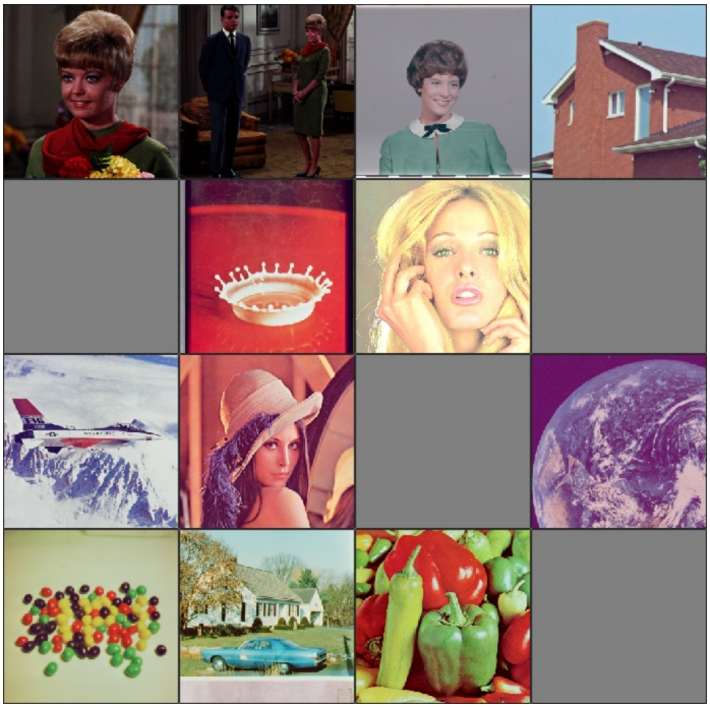


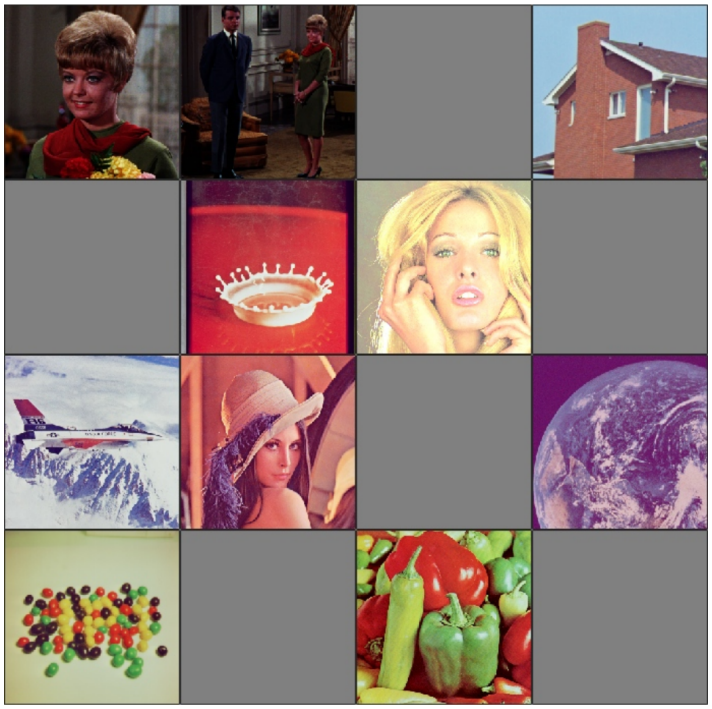
Figure 33: The original version of the 16 images used to demonstrate the method outlined. Each image is 512x512 pixels, and stored in separate data containers that are assumed to be failure prone. The images are taken from the The USC-SIPI Image Database[73]. In figures 34 and 35, the partial information recovery procedure is demonstrated when removing 4 and 6 images respectively.



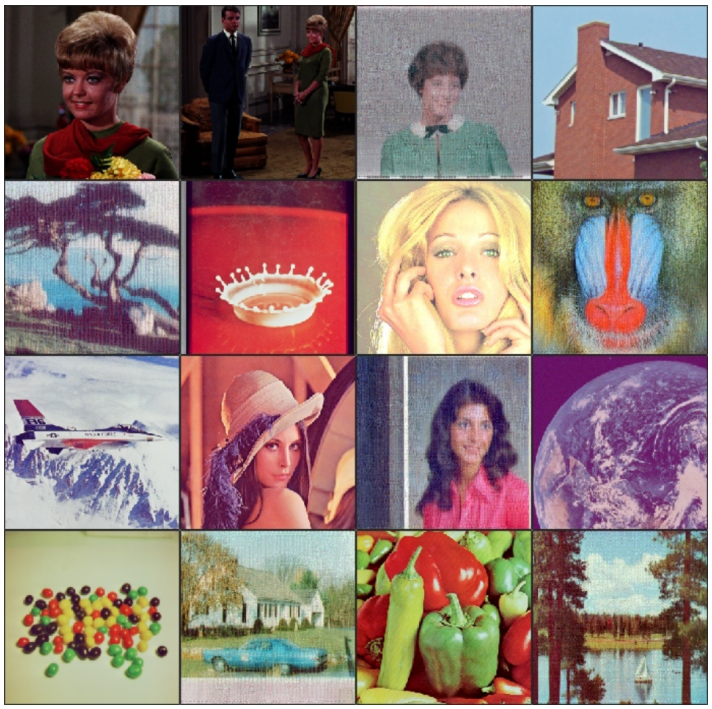
(a) Four images removed.

(b) Four images reconstructed.

Figure 34: Three checksums were computed to protect the content of the 16 containers. (a) Shows four images removed. (b) Depicts the recovered images.



(a) Six images removed.



(b) Six images reconstructed.

Figure 35: Three checksums were computed to protect the content of the 16 containers. (a) Shows six images removed. (b) Depicts the recovered images.

representation is not necessary upon recovery. For the method introduced to have practical relevance, it must be extended to the case where the checksum matrix elements are from a Galois field, i.e. Reed-Solomon codes. There are no obvious reasons to believe that the fundamental idea of alternating between enforcing the constraint of the checkpoint equations, and applying some filter, should not work. In practice though, one need a way of completing step 2-3 in the method as outlined in section 5.2.4. When $A \in \mathbb{R}^{m\times k}$, the Moore–Penrose pseudo inverse could be used as it provides the minimum euclidean norm solution to the under-determined linear system.

A solution procedure for the same problem when $A \in \mathbb{F}^{m\times k}$ is less obvious. Generalized inverses of matrices with elements from finite fields is a research topic that has received limited attention. Some first results on matrices over finite fields that satisfy the four criteria of a Moore-Penrose pseudo inverse were published in [31]. In [74], necessary and sufficient conditions on $A \in \mathbb{F}^{m\times k}$ for the existence of A^+ was given, and in [18] a method for constructing the Moore–Penrose pseudo inverse was presented. It is however not clear if/how the minimum norm property applies here. In short, further studies are needed to generalize the method.

If successfully extended, the method could potentially have a wide number of applications in fault tolerance and error correction, also outside the context of HPC. Today the average data-center consumes as much power as a small city. In the US alone, data-centers and cloud-service providers store several hundred million terabytes of data, and account for more than 2% of the nations electricty consumption[67]. Overhead related to protection of data accounts for a substantial part of the energy usage, this mainly due to extensive use of redundancy and erasure code to ensure that the probability of hard-drive data loss is extremely small. If a method existed that would allow for robust partial information recovery in the commonly used erasure codes, this would mean that failures that would otherwise result in complete data loss would instead only result in loss of data fidelity which in turn could potentially lead to relaxed requirement on data protection for certain applications.

6 Summary and outlook

In this deliverable, we have reported on the progress achieved during the last 18 months by the ExaFLOW WP1 partners, in terms of developing new algorithms and techniques that can capitalise on the potential offered for fluid dynamics simulations on future exascale platforms. Across the project, we have made great strides in improving the state-of-the-art in the objectives identified at the start of this project as current bottlenecks for exascale fluid dynamics problems. In the points below, we outline some of the key achievements and scope for future work in these areas.

- In terms of error control and mesh refinement, we have developed efficient techniques for the automatic identification of solution error based on both spectral error indicators and goal-based adjoint error estimators. Together with an efficient adaptive mesh refinement strategy that builds on the algorithmic developments in terms of coarse space

preconditioning and solver robustness, the simulations in WP2 and WP3 demonstrate significant capability improvements for realistic exascale test cases. In ExaFLOW, we have mostly focused on a complete, efficient implementation using element size h to provide the route to increasing resolution. Future work in this area can focus on the incorporation of additional aspects of refinement; in particular, the variation of polynomial order (p-refinement) and node position (r-refinement) are alternate strategies that, although briefly during this project (see e.g. [56]) could be further investigated alongside the balance of each refinement strategy. The introduction of heterogeneous modelling, in which different fluid models are introduced in different regions of the domain, also shows great potential in terms of reducing computational cost and improving time-to-solution and scalability whilst maintaining the accuracy of solutions.

- To improve strong scaling of these codes, we have investigated various algorithmic developments based around higher-order spectral element codes. In particular, as part of work on adaptive mesh refinement, significant advances have been made in terms of load balancing and the use of algebraic multigrid approaches for the improvement of coarse grid preconditioning, which is a significant obstacle for strong scaling of these methods. Future work in this area can focus on extending this work to larger-scale unstructured grids, as briefly discussed in D2.4. Finally, although the CG-HDG formulation shows good potential in terms of improving strong scaling, it was found that on current hardware, time-to-solution is unlikely to be improved. However,

- Significant effort has been invested in the development of compression techniques to alleviate the I/O burden anticipated for exascale simulations. We have adopted two approaches: first, investigating the development of a wavelet-based compression algorithm within the JPEG-2000 standard. The reported results in D2.4 for realistic flow data using the Taylor-Green vortex highlight that this approach is both viable and capable of attaining high compression ratios whilst still retaining accuracy of the underlying data sufficient for postprocessing. The second approach has focused on reduced-order modelling of the underlying datasets, by investigating decomposition of the data through mathematical approaches such as SVD. Future work in this area can focus on the producing an optimized implementation of the wavelet-based codec, alongside efficiency improvements to increase data throughput and the consideration of data attained through mesh refinement techniques.

- Finally, we have worked to make resilient exascale simulations through the development of fault tolerant algorithms. We have adopted two approaches. The first takes a minimally intrusive approach to allow for complex transient solvers, such as those included in the Nektar++ framework, to be hardened against hardware failures without excessive changes to the inner workings of the program. The efficacy of this approach is demonstrated further in D2.4, where in-memory checkpointing and MPI ULFM are combined to examine fault tolerance for a fluid dynamics solver within Nektar++. The second approach examines a method that is more intrusive, but comes with the benefit of greater flexibility in memory efficiency and lower memory overhead. The

development of a separate library, described in D2.4, provides an ideal platform for projects to benefit from this algorithmic development. Future work in this area can focus on both the incorporation of this library within fluid dynamics solvers, as well as to investigate further the use of partial data recovery. This approach should allow simulations to recover even when the parity information is only partially available, providing the ability to significantly increase robustness for fluid dynamics codes.

References

[1] Tinku Acharya and Ping-Sing Tsai. *JPEG2000 Standard for Image Compression: Concepts, Algorithms and VLSI Architectures*. John Wiley & Sons, Hoboken, New Jersey, 2005.

[2] Emmanuel Agullo, Luc Giraud, Abdou Guermouche, Jean Roman, and Mawussi Zounon. Towards resilient parallel linear krylov solvers: recover-restart strategies. Technical Report RR-8324, INRIA, 2013.

[3] Douglas N Arnold, Franco Brezzi, Bernardo Cockburn, and L Donatella Marini. Unified analysis of discontinuous galerkin methods for elliptic problems. *SIAM journal on numerical analysis*, 39(5):1749–1779, 2002.

[4] W. Bangerth and R. Rannacher. *Adaptive Finite Element Methods for Differential Equations*. Birkhäuser, Basel, 2002.

[5] Richard Barrett, Michael W Berry, Tony F Chan, James Demmel, June Donato, Jack Dongarra, Victor Eijkhout, Roldan Pozo, Charles Romine, and Henk Van der Vorst. *Templates for the solution of linear systems: building blocks for iterative methods*, volume 43. Siam, 1994.

[6] Y. Bazilevs and T.J.R. Hughes. Weak imposition of Dirichlet boundary conditions in fluid mechanics. *Computers & Fluids*, 36(1):12 – 26, 2007. Challenges and Advances in Flow Simulation and Modeling.

[7] Y. Bazilevs, C. Michler, V.M. Calo, and T.J.R. Hughes. Weak dirichlet boundary conditions for wall-bounded turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, 196(49):4853 – 4862, 2007.

[8] Wesley Bland, George Bosilca, Aurelien Bouteiller, Thomas Herault, and Jack Dongarra. A proposal for user-level failure mitigation in the MPI-3 standard. Technical report, 2 2012.

[9] Wesley Bland, Aurelien Bouteiller, Thomas Herault, George Bosilca, and Jack Dongarra. Post-failure recovery of MPI communication capability. *The International Journal of High Performance Computing Applications*, 27(3):244–254, 1 2013.

[10] A. Busse and N. D. Sandham. Parametric forcing approach to rough-wall turbulent channel flow. *Journal of Fluid Mechanics*, 712:169202, 2012.

[11] Atife Caglar and Anastasios Liakos. Weak imposition of boundary conditions for the navier–stokes equations by a penalty method. *International journal for numerical methods in fluids*, 61(4):411–431, 2009.

[12] Xiao-Chuan Cai and Marcus Sarkis. A restricted additive schwarz preconditioner for general sparse linear systems. *SIAM J. Sci. Comput.*, 21(2):792–797, September 1999.

[13] Chris D. Cantwell, David Moxey, Andrew Comerford, Alessandro Bolis, Gabriele Rocco, Gianmarco Mengaldo, Daniele De Grazia, Sergey Yakovlev, Jean-Eloi. Lombard, Dirk Ekelschot, Bastien Jordi, Hui Xu, Yumnah Mohamied, Claes Eskilsson, Blake Nelson, Peter Vos, Cristian Biotto, Robert M. Kirby, and Spencer J. Sherwin. Nektar++: An open-source spectral/ hp element framework. *Computer Physics Communications*, 192:205–219, 7 2015.

[14] Chris D Cantwell and Allan S Nielsen. A minimally intrusive low-memory approach to resilience for existing transient solvers. *Journal of Scientific Computing*, pages 1–17, 2018.

[15] Franck Cappello, Al Geist, William Gropp, Sanjay Kale, Bill Kramer, and Marc Snir. Toward exascale resilience: 2014 update. *Supercomputing Frontiers and Innovations*, 1(1), 1 2014.

[16] Zizhong Chen and Jack J Dongarra. Condition numbers of gaussian random matrices. *SIAM Journal on Matrix Analysis and Applications*, 27(3):603–620, 2005.

[17] Bernardo Cockburn, Jayadeep Gopalakrishnan, and Raytcho Lazarov. Unified Hybridization of Discontinuous Galerkin, Mixed, and Continuous Galerkin Methods for Second Order Elliptic Problems. *SIAM Journal on Numerical Analysis*, 47(2):1319–1365, 2009.

[18] Zongduo Dai and Yufeng Zhang. Partition, construction, and enumeration of m–p invertible matrices over finite fields. *Finite Fields and Their Applications*, 7(3):428–440, 2001.

[19] David David Taubman and Michael Marcellin. *JPEG2000: Image Compression Fundamentals, Standards and Practice*. Springer US, New York City, 2002.

[20] M. O. Deville, P. F. Fischer, and E. H. Mund. *High-Order Methods for Incompressible Fluid Flow*. Cambridge University Press, 2002.

[21] Alan Edelman. Eigenvalues and condition numbers of random matrices. *SIAM Journal on Matrix Analysis and Applications*, 9(4):543–560, 1988.

[22] Evridiki Efstathiou and Martin J. Gander. Why restricted additive schwarz converges faster than additive schwarz. *BIT Numerical Mathematics*, 43(5):945–959, 2003.

[23] Lars Eldén. *Matrix Methods in Data Mining and Pattern Recognition (Fundamentals of Algorithms)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2007.

[24] P. Fischer, N. Miller, and H. Tufo. An overlapping schwarz method for spectral element simulation of three-dimensional incompressible flow. In P. Bjorstad and M. Luskin, editors, *Parallel Solution of Partial Differential Equations*, volume 120 of *The IMA Volumes in Mathematics and its Applications*, pages 159–180. Springer New York, 2000.

[25] P. Fischer and J. Mullen. Filter-based stabilization of spectral element methods. *Academie des Sciences Paris Comptes Rendus Serie Sciences Mathematiques*, 332:265–270, February 2001.

[26] Paul F. Fischer. An overlapping schwarz method for spectral element solution of the incompressible navier stokes equations. *Journal of Computational Physics*, 133:84–101, May 1997.

[27] Paul F Fischer. Scaling limits for pde-based simulation. In *22nd AIAA Computational Fluid Dynamics Conference*, page 3049, 2015.

[28] Paul F. Fischer, Gerald W. Kruse, and Francis Loth. Spectral element methods for transitional flows in complex geometries. *J. Sci. Comput.*, 17(1-4):81–98, December 2002.

[29] Paul F. Fischer and James W. Lottes. *Hybrid Schwarz-Multigrid Methods for the Spectral Element Method: Extensions to Navier-Stokes*, pages 35–49. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[30] Jouni Freund and Rolf Stenberg. On weakly imposed boundary conditions for second order problems. In *Proceedings of the Ninth Int. Conf. Finite Elements in Fluids*, pages 327–336. Venice, 1995.

[31] John D Fulton. Generalized inverses of matrices over a finite field. *Discrete mathematics*, 21(1):23–29, 1978.

[32] M. N. Gamito and M. Salles Dias. Lossless coding of floating point data with JPEG 2000 Part 10. In A. G. Tescher, editor, *Applications of Digital Image Processing XXVII*, volume 5558, pages 276–287, November 2004.

[33] Ralf Hartmann. Adjoint consistency analysis of discontinuous Galerkin discretizations. *SIAM Journal on Numerical Analysis*, 45(6):2671–2696, 2007.

[34] Henri Bruno Razafindradina, Paul Auguste Randriamitantsoa, and Nicolas Raft Razafindrakoto. Image compression with svd: A new quality metric based on energy ratio. *CoRR*, abs/1701.06183, 2016.

[35] Cheng Huang, Minghua Chen, and Jin Li. Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems. *ACM Transactions on Storage (TOS)*, 9(1):3, 2013.

[36] Cheng Huang, Huseyin Simitci, Yikang Xu, Aaron Ogus, Brad Calder, Parikshit Gopalan, Jin Li, Sergey Yekhanin, et al. Erasure coding in windows azure storage. 2012.

[37] 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.

[38] 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*, 2017.

[39] Christian T. Jacobs, Markus Zauner, Nicola De Tullio, Satya P. Jammy, David J. Lusher, and Neil D. Sandham. An error indicator for finite difference methods using spectral techniques with application to aerofoil simulation. *Computers and Fluids*, 168:67 – 72, 2018.

[40] Claes Johnson and Peter Hansbo. Adaptive finite element methods in computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 101(1):143 – 181, 1992.

[41] Mika Juntunen and Rolf Stenberg. Nitsche's method for general boundary conditions. *Mathematics of computation*, 78(267):1353–1374, 2009.

[42] George Em Karniadakis, Moshe Israeli, and Steven A Orszag. High-order splitting methods for the incompressible navier-stokes equations. *Journal of computational physics*, 97(2):414–443, 1991.

[43] Robert M Kirby and George Em Karniadakis. De-aliasing on non-uniform grids: algorithms and applications. *Journal of Computational Physics*, 191(1):249–264, 2003.

[44] Robert M. Kirby, Spencer J. Sherwin, and Bernardo Cockburn. To CG or to HDG: A comparative study. *Journal of Scientific Computing*, 51(1):183–212, 2011.

[45] Gerald W. Kruse. *Parallel Nonconforming Spectral Element Solution of the Incompressible Navier-Stokes Equations in Three Dimensions*. PhD thesis, Providence, RI, USA, 1997. UMI Order No. GAX97-38573.

[46] Julien Langou, Zizhong Chen, Jack J Dongarra, and George Bosilca. Disaster survival guide in petascale computing. In *Petascale Computing: Algorithms and Applications*, pages 263–288. Chapman and Hall/CRC, 2007.

[47] W Layton. Weak imposition of no-slip conditions in finite element methods. *Computers & Mathematics with Applications*, 38(5-6):129–142, 1999.

[48] Jurij Leskovec, Anand Rajaraman, and Jeffrey D. Ullman. *Mining of massive datasets*. Cambridge University Press, Cambridge, second edition edition, 2014.

[49] Anastasios Liakos. Weak imposition of boundary conditions in the stokes problem. Technical report, University of Pittsburgh, 1999.

[50] P. Lindstrom. Fixed-rate compressed floating-point arrays. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2674–2683, Dec 2014.

[51] Alexander Loddoch and Jrg Schmalzl. Variable quality compression of fluid dynamical data sets using a 3-d dct technique. *Geochemistry, Geophysics, Geosystems*, 7(1):n/a–n/a, 2006. Q01003.

[52] Jean-Eloi W Lombard, David Moxey, Spencer J Sherwin, Julien FA Hoessler, Sridar Dhandapani, and Mark J Taylor. Implicit large-eddy simulation of a wingtip vortex. *AIAA Journal*, 2015.

[53] James W. Lottes and Paul F. Fischer. Hybrid multigrid/schwarz algorithms for the spectral element method. *Journal of Scientific Computing*, 24(1):45–78, 2005.

[54] M Manasse, C Thekkath, and A Silverberg. A reed-solomon code for disk storage, and efficient recovery computations for erasure-coded disk storage.

[55] Catherine Mavriplis. A posteriori error estimators for adaptive spectral element techniques. In Peter Wesseling, editor, *Notes on Numerical Fluid Mechanics*, pages 333–342, 1990.

[56] D. Moxey, C. D. Cantwell, G. Mengaldo, D. Serson, D. Ekelschot, J. Peiró, S. J. Sherwin, and R. M. Kirby. Towards p-adaptive spectral/hp element methods for modelling industrial flows. In *Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2016*, pages 63–79, 2017.

[57] Prabal Negi, Philipp Schlatter, and Dan Henningson. A re-examination of filter-based stabilization for spectral-element methods. Technical report, KTH, Stability, Transition and Control, 2017. QC 20171121.

[58] Joachim Nitsche. Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind. In *Abhandlungen aus dem mathematischen Seminar der Universität Hamburg*, volume 36, pages 9–15. Springer, 1971.

[59] Dimitris S Papailiopoulos and Alexandros G Dimakis. Locally repairable codes. *IEEE Transactions on Information Theory*, 60(10):5843–5855, 2014.

[60] Majid Rabbani and Rajan Joshi. An overview of the {JPEG} 2000 still image compression standard. *Signal Processing: Image Communication*, 17(1):3 – 48, 2002. {JPEG} 2000.

[61] Irving S Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.

[62] Maheswaran Sathiamoorthy, Megasthenis Asteris, Dimitris Papailiopoulos, Alexandros G Dimakis, Ramkumar Vadali, Scott Chen, and Dhruba Borthakur. Xoring elephants: Novel erasure codes for big data. In *Proceedings of the VLDB Endowment*, volume 6, pages 325–336. VLDB Endowment, 2013.

[63] Philipp Schlatter, Steffen Stolz, and Leonhard Kleiser. Les of transitional flows using the approximate deconvolution model. *International Journal of Heat and Fluid Flow*, 25(3):549 – 558, 2004. Turbulence and Shear Flow Phenomena (TSFP-3).

[64] Bianca Schroeder and Garth A Gibson. Disk failures in the real world: What does an mttf of 1,000,000 hours mean to you? In *FAST*, volume 7, pages 1–16, 2007.

[65] Bianca Schroeder and Garth A. Gibson. Understanding failures in petascale computers. *Journal of Physics: Conference Series*, 78:012022, 7 2007.

[66] Bianca Schroeder, Raghav Lagisetty, and Arif Merchant. Flash reliability in production: The expected and the unexpected. In *Proceedings of the 14th USENIX Conference on File and Storage Technologies*, 2016.

[67] Arman Shehabi, Sarah Josephine Smith, Dale A. Sartor, Richard E. Brown, Magnus Herrlin, Jonathan G. Koomey, Eric R. Masanet, Nathaniel Horner, Inês Lima Azevedo, and William Lintner. United states data center energy usage report. Technical report, 06/2016 2016.

[68] Marc Snir, Robert W Wisniewski, Jacob A Abraham, Sarita V Adve, Saurabh Bagchi, Pavan Balaji, Jim Belak, Pradip Bose, Franck Cappello, Bill Carlson, Andrew A Chien, Paul Coteus, Nathan A DeBardeleben, Pedro C Diniz, Christian Engelmann, Mattan Erez, Saverio Fazzari, Al Geist, Rinku Gupta, Fred Johnson, Sriram Krishnamoorthy, Sven Leyffer, Dean Liberty, Subhasish Mitra, Todd Munson, Rob Schreiber, Jon Stearley, and Eric Van Hensbergen. Addressing failures in exascale computing. *The International Journal of High Performance Computing Applications*, 28(2):129–173, 1 2014.

[69] H.M Tufo and P.F Fischer. Fast parallel direct solvers for coarse grid problems. *J. Parallel Distrib. Comput.*, 61(2):151–177, February 2001.

[70] José M Urquiza, André Garon, and Marie-Isabelle Farinas. Weak imposition of the slip boundary condition on curved boundaries for stokes flow. *Journal of Computational Physics*, 256:748–767, 2014.

[71] Bryan E. Usevitch. Jpeg2000 compatible lossless coding of floating-point data. *Eurasip Journal on Image and Video Processing*, 2007:1–8, 3 2007.

[72] Z.J. Wang, Krzysztof Fidkowski, Rmi Abgrall, Francesco Bassi, Doru Caraeni, Andrew Cary, Herman Deconinck, Ralf Hartmann, Koen Hillewaert, H.T. Huynh, Norbert Kroll,

Georg May, Per-Olof Persson, Bram van Leer, and Miguel Visbal. High-order cfd methods: current status and perspective. *International Journal for Numerical Methods in Fluids*, 72(8):811–845, 2013.

[73] Allan Weber. The USC-SIPI Image Database, 2018. link [Acccessed: 02.08.2018].

[74] Chuan-Kun Wu and Ed Dawson. Existence of generalized inverse of linear transformations over finite fields. *Finite Fields and Their Applications*, 4(4):307–315, 1998.

[75] Sergey Yakovlev, David Moxey, Robert M. Kirby, and Spencer J. Sherwin. To CG or to HDG: A comparative study in 3D. *Journal of Scientific Computing*, 67(1):192–220, 2016.