H2020 FETHPC-1-2014

Enabling Exascale Fluid Dynamics Simulations
Project Number 671571

D1.2 - Initial proof-of-concept formulation of ExaFLOW algorithms

WP1: Algorithmic improvements towards exascale

DOCUMENT INFORMATION

Deliverable Number	D1.2
Deliverable Name	Initial proof-of-concept formulation of ExaFLOW algorithms
Due Date	31/03/2017 (PM 18)
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) 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	31/03/2017
Reviewed by	Nick Johnson (UEDIN) Niclas Jansson (KTH) Erwin Laure (KTH)
MGT Board Approval	31/03/2017

DOCUMENT HISTORY

Partner	Date	Comment	Version
Imperial	25/01/2017	Initial version with task 1.3	0.1
SOTON	03/02/2017	Add contribution to task 1.2	0.2
KTH	23/02/2017	Add contribution to task 1.1	0.3
USTUTT	24/02/2017	Add contribution to task 1.4	0.4
EPFL	26/02/2017	Add contribution to task 1.5	0.5
Imperial	17/03/2017	Corrections from KTH and UEDIN	0.6

Executive Summary

In this deliverable, we will outline the current proof-of-concept algorithms that are being developed by the work package 1 (WP1) ExaFLOW partners, and highlight progress made during the first half of the project. The goals of the report are to:

- describe the efforts of WP1 partners in extending the current state-of-the-art algorithms that have already been outlined in deliverable 1.1 (D1.1);
- summarise progress made in achieving the goals of each of the WP1 tasks, and how they apply to current and expected developments in WP2 and WP3;
- look ahead to further progress in each of the WP1 tasks, highlighting a work plan for each partner towards the project conclusion.

The introduction summarises the aims and contributions of WP1 within ExaFLOW and gives a brief overview of the tasks being undertaken in this work package. Progress on each of the tasks is then outlined in detail in the rest of the document. We conclude this summary with an overview of progress on each of the tasks:

- **Mesh quality and grid adaptivity:** The aim of this task is to develop novel adaptive mesh refinement techniques for exascale flow simulations. This takes the form of refinement in terms of element size (h), element locations (r) and polynomial order (p). Across the first half of the project, substantial progress has been made in developing both the mechanisms for achieving highly-parallel h-adaptive simulations, and in examining a range of error estimators in order to drive the adaptive process, refining or coarsening the underlying mesh in order to better capture flow structures.

  The development of these h-adaptive processes has focused around algorithms based on a nonconformal mesh. This has advantages both in terms of potential for parallel scalability (thus making it suited to exascale simulations), geometric flexibility and numerical conditioning of the underlying system, as nonconformal meshes will reduce the aspect ratio of elements. However, in order to develop these methods, careful consideration must be given to both the elemental connectivity, as well as the coarse space and overlapping Schwarz preconditioners that are used to address the challenges of efficiently solving the pressure Poisson equation. These issues have been the focus of much of the effort in WP1, and are outlined in detail in this deliverable.

  With a strategy identified to achieve a scalable and parallel mesh refinement, our other efforts have focused around developing methods to drive this adaptive process, so that adaption can be targeted in areas of the simulation where underresolution occurs, in order to more accurately model the flow dynamics. Two types of error indicators and estimators are being developed, and are outlined in this document. The first is a local (elemental) *a posteriori* error indicator, which is based on the expected decay rates of an expansion of spectral coefficients. The efficacy of this indictor is studied in the context of a number of examples, and its local nature makes it inherently scalable at

large processor counts. The second error estimator is currently under development, and is based around a goal-based adjoint procedure, which yields a more accurate estimation of the error throughout the domain, albeit at a higher computational cost.

In summary, this task is proceeding well. Future efforts in the short term will aim to further develop algorithms for error detection based on the adjoint solver, before applying these to the incompressible NACA4412 airfoil in WP3.

- **Error control for heterogeneous modelling:** The aim of this task is to leverage a heterogeneous approach to fluid simulations, whereby different models are used in different regions of the domain depending on the level of detail required, which will reduce the computational effort required and increase scalability. A key component of this algorithm is developing techniques for error indication, to highlight where the grid or model can be coarsened or refined appropriately within the domain to adapt to the flow dynamics. In this deliverable we examine two error indicators that have been developed as part of this task, based on performing small-scale Fourier transformations and examining the resultant energy spectrum. These are then applied to a canonical Taylor-Green vortex breakdown at various levels of resolution in order to validate the proposed error indicators.

  Progress on this task has been good in the period since deliverable 1.1, however was somewhat limited in the initial phases of the project due to late staff appointments. More effort has also necessarily been exerted in the initial phases of the project as part of WP2 to develop parts of the OpenSBLI code, which will provide a better platform in which to further develop the heterogeneous modelling of WP1. Future plans in this task are, in the short term, to further develop and validate the error indicators, by applying them to the compressible NACA4412 aerofoil case in WP3.

- **Mixed CG-HDG discretisation:** The aim of this task is to develop a new, mixed finite element discretisation for elliptic problems, whereby the compact properties of the continuous Galerkin method are combined with the pairwise communication property of the hybridisable discontinuous Galerkin method in order to exploit the heterogeneous nature of inter- and intra-node computing to improve strong scaling at exascale. Across the first half of the project, good progress has been made in developing a new weak Dirichlet boundary condition. This is required for the weak imposition of the solution between nodes. This deliverable presents the development and validation of this for the solution of elliptic problems, and their application to both a Poisson equation and simulations of the incompressible Navier-Stokes equations. We examine this novel development in the context of existing techniques for weak imposition of Dirichlet conditions, in which the method developed here provides a significant advantage in a nearly parameter-free manner.

  The second half of the work presented in this deliverable outlines the anticipated performance of the mixed CG-DG discretisation for both two- and three-dimensional simulations. This examines in more detail the second half of the CG-HDG algorithm:

namely, the computational cost of solving the matrix system which allows us to obtain a solution between nodes. We hoped to exploit the known block structure of this matrix in order to keep the cost in comparable to classical CG solve in one partition. The reduced communication would then still ensure favorable scaling and wall clock times. When doing the cost analysis of the CG-DG algorithm, however, we found out that 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.

Although this task has therefore lead to the development of a significant new boundary condition implementation, in light of our performance analysis, we feel that further modelling of the computational costs are required before progressing to a full implementation of the method. In particular, in the context of exascale hardware, in which FLOPS and memory bandwidth are expected to significantly increase and be readily available, we feel that a model that can incorporate both communication costs and the balance of FLOPS and memory bandwidth will be of more use in guiding further decisions on this task. However, should these studies demonstrate that the CG-HDG formulation is unlikely to yield the appropriate performance benefits, in terms of time to compute over existing formulations, we have identified a number of areas that align with both the goals of the project and the objective of increasing strong scaling in this task, which we outline in this deliverable.

- **Data reduction:** The aim of this task is to develop data-reduction techniques specific to CFD. This is key for exascale simulations, since input/output (I/O) is expected to prove a large bottleneck. Algorithmic developments are therefore required in order to reduce the amount of data needed to be stored as the simulation runs. In this deliverable we present results from the first half of the project, in which two lines of research have been undertaken.

  The first has examined the use of image compression techniques, specifically JPEG-2000, in the lossy and lossless compression of flow data. Our developments thus far show that lossy compression, where we adopt fixed-point number format to normalise the underlying floating-point data of the flow field, can provide substantial data reduction savings. In particular, we have examined a three-dimensional turbulent flat-plate simulation at two Mach numbers, wherein compression ratios of 17:1 and 15:1 have been observed. Whilst this provides a good route to reduce data overhead for visualisation purposes, the resulting data set could not be used to, for example, restart a fluid simulation from a checkpoint. We have therefore investigated lossless compression techniques, based on a shape-adaptive discrete wavelet transformation. However thus far, this method has not proven effective, due to the underlying representation of the floating-point data type. Our future work will therefore shift to investigate other motion-compression techniques, specifically the High Efficiency Video Coding Standard (HEVC), as well as higher order signal transforms.

The second class of techniques we are investigating focus on reduced-order modelling techniques, specifically singular value decomposition (SVD) and dynamic mode decomposition (DMD), which may be used to extract and store only the key flow features and may offer very substantial data reduction savings at larger scales. Presently, due to the more experimental and complex nature of these techniques (when compared to the more well-established image and motion compression techniques considered previously), this work is still in the exploratory stages, with prototype implementations written for serial codes. Our future work in the second half of the project will therefore focus initially on the assessment of these different decomposition techniques, alongside their parallel implementation as part of WP2.

- **Fault tolerance and resilience:** The final task in this WP considers developing numerical techniques that are resilient to hardware errors, both soft (e.g. memory corruption) and hard (e.g. node failures). At exascale levels of parallelism, it is expected that such errors are likely to occur on the frequency of a few minutes, based on the expected time to failure of current computing hardware.

  Significant advances have been made in the first half of the project in terms of developing prototype techniques to mitigate hard errors. These are based on a diskless checkpoint-based approach, where node recovery is accomplished by restoring data from in-memory checkpoints that are communicated at regular frequencies to neighbouring nodes during the simulation. A major accomplishment in this task is the development of a prototype solver within the Nektar++ framework, which we outline in this deliverable. This uses proposed user-level failure mitigation (ULFM) extensions to the MPI communication framework in order to notify the nodes when a rank failure occurs during a simulation. This prototype solver, based on a diffusion equation, has been tested to correctly recover the computation when a node fails. This demonstrates the efficacy of the proposed checkpointing algorithm on a problem which can be seen as the core building block of more complex equations, such as Navier-Stokes.

  The next phase of the project will involve the investigation of more sophisticated error-correction routines, which we describe further in this deliverable. We will additionally extend the existing methods from the prototype solver into the Navier-Stokes solver, allowing us to test the effectiveness of these schemes on larger-scale problems being investigated as part of WP3.

Contents

List of Figures

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

Work package 1 (WP1) focuses on algorthmic development to overcome the main challenges and objectives identified by the ExaFLOW project. Specifically, the work being undertaken in this work package is designed to overcome key algorithmic bottlenecks that need to be addressed before computational fluid dynamics (CFD) simulations can be undertaken at exascale-level computing platforms. This work is split across five tasks, each of which is lead by one of the five work package partners in collaboration with other partners.

- **Task 1.1:** *Mesh quality and grid adaptivity*
  This task focuses 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.

- **Task 1.2:** *Error control for heterogeneous modelling*
  Will investigate the application of heterogeneous modelling to exascale, so that different regions of the flow may be modelled with different approaches, thereby reducing computational cost and increasing scalability. In particular, it will address the challenges that arise when considering 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.

- **Task 1.3:** *Mixed CG-HDG formulation*
  This task will investigate 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.

- **Task 1.4:** *Data reduction*
  The aim of this task is 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.

- **Task 1.5:** *Fault tolerance and resilience*
  This 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 of hardware faults. This will include both the development of suitable error detectors (in collaboration with Task 1.1) and efficient data reduction and model building, partly in collaboration with Task 1.4.

In the following sections, we give a detailed description of the work undertaken across the first half of the project, and outline the current proof-of-concept algorithms being developed by the ExaFLOW partners.

2 Mesh quality and grid adaptivity

One of the major concerns in the numerical solutions of partial differential equations (PDEs) is finding the optimal grid on which the solution can be computed. Unfortunately, in most cases it is not an easy task that can be determined in advance, making space for self-adapting algorithms like Adaptive Mesh Refinement (AMR). Its goal is to control computational error during the simulation and to increase resolution in a particular region of the domain only. It gives the possibility to increase the accuracy of numerical simulations at minimal computational cost, but at the same time it increases solver complexity, which can have negative effect on the code performance. That is why special care has to be taken to develop efficient tools and algorithms that can be used within an AMR framework. In this section we describe progress on developing three tools within WP1:

- adaptation of pressure preconditioners for nonconforming meshes for the Spectral Element Method (SEM),
- spectral error indicators,
- adjoint error estimators.

2.1 Efficient pressure preconditioner for nonconforming meshes

We consider simulations of unsteady incompressible flows for which the sound speed is infinite. In this case, the linear sub-problem associated with the divergence-free constraint can become very ill-conditioned, making its solution the most expensive phase of the simulation when using iterative solvers. Therefore, defining a robust parallel preconditioning strategy has received much attention in past decades. In the context of SEM, two possible approaches based on an additive overlapping Schwarz method [23, 24] and a hybrid Schwarz-multigrid method [26, 43] were proposed and implemented in the *Nek5000* code, giving significant reduction of pressure iterations. In this context, h-type AMR framework can be advantageous, as introducing nonconforming meshes can reduce the aspect ratio of elements, thereby limiting the condition number of the pressure operator (see Sec. 6 in [43]). However, at the same time, it increases complexity of the elements connectivity and makes element overlap evaluation more difficult.

In this section, we concentrate on the additive overlapping Schwarz approach and discuss modifications necessary to adapt this method for h-type AMR framework. In developing a nonconforming solver, we use a conforming-space/nonconforming-mesh approach [25, 37], which instead of employing "mortar" elements enforces function continuity directly at the element faces and edges by interpolation from the low resolution element to the high resolution one. This choice imposes additional restriction on the mesh structure (see Fig. 4 in [25]), but these restrictions are acceptable within our AMR framework.

2.1.1 Navier-Stokes discretisation

We review briefly discretisation of the incompressible Navier-Stokes equations to introduce notation and point algorithm parts that require modification. The more in depth derivation can be found in [24]. The temporal discretisation is based on a semi-implicit scheme, in which the nonlinear term is treated explicitly and the remaining unsteady Stokes problem is solved implicitly. To avoid spurious pressure modes our spatial discretisation is based on the $\mathbb{P}_N - \mathbb{P}_{N-2}$ SEM, where velocity and pressure spaces are spanned by Lagrangian interpolants on the Gauss-Lobatto Legendre (GLL) and Gauss Legendre (GL) quadrature points respectively. Note that the basis for velocity is continuous across element interfaces, whereas the basis for pressure is not. Assuming $\mathbf{f}^n$ incorporates all nonlinear and source terms treated explicitly at time t^n the matrix form of Stokes problem reads:

$$\begin{bmatrix} \mathbf{H} & -\mathbf{D}^T \\ -\mathbf{D} & 0 \end{bmatrix} \begin{pmatrix} \mathbf{u}^n \\ p^n \end{pmatrix} = \begin{pmatrix} \mathbf{f}^n \\ 0 \end{pmatrix}, \tag{1}$$

where $\mathbf{H} = -\frac{1}{Re}\mathbf{A} + \frac{\beta_0}{\delta t}\mathbf{B}$ and $\mathbf{D}$ are discrete Helmholtz and divergence operators respectively. $\mathbf{A}$ and $\mathbf{B}$ denote here discrete Laplacian and the diagonal mass matrix associated with the velocity mesh, and bold indicates matrices that interact with vector fields.

Applying Uzawa decoupling and using inverse mass matrix $\mathbf{B}^{-1}$ as approximation of inverse Helmholtz operator $\mathbf{H}^{-1}$ one ends with

$$\begin{bmatrix} \mathbf{H} & -\frac{\delta t}{\beta_0}\mathbf{H}\mathbf{B}^{-1}\mathbf{D}^T \\ 0 & E \end{bmatrix} \begin{pmatrix} \mathbf{u}^n \\ \delta p \end{pmatrix} = \begin{pmatrix} \mathbf{B}\mathbf{f}^n + \mathbf{D}^T p^{n-1} \\ g \end{pmatrix}, \tag{2}$$

where

$$E = \frac{\delta t}{\beta_0}\mathbf{D}\mathbf{B}^{-1}\mathbf{D}^T \tag{3}$$

is the Stokes Schur complement governing the pressure, $\delta p = p^n - p^{n-1}$ is the pressure update, and g is the inhomogeneity arising from Gaussian elimination. Note that for this splitting method the diagonality of the mass matrix $\mathbf{B}$ is crucial to avoid costly matrix inverse.

All operators $\mathbf{H}$, $\mathbf{A}$, $\mathbf{B}$ and E are symmetric positive definite (SPD) and can be solved with preconditioned conjugate gradient (PCG). Moreover E has properties similar to a Poisson operator, and is often referred as "consistent Poisson operator". The systems involving $\mathbf{H}$ and E are solved iteratively with E being more challenging, and in next section we will present preconditioning strategy for the pressure equation:

$$E\delta p = -\mathbf{D}\mathbf{u}. \tag{4}$$

We close this section presenting shortly SEM operators. SEM introduces globally unstructured and locally structured basis by tessellating the domain into K non-overlapping subdomains (deformed quadrilaterals), $\Omega = \bigcup_{k=1}^{K} \Omega_k$, and representing functions in each subdomain in terms of tensor-product polynomials on a reference subdomain $\hat{\Omega} = [-1, 1]^d$.

In this approach every function or operator is represented by its local counterparts, which in case of functions takes form of the sum over subdomains

$$f(\mathbf{x}) = \sum_{k=1}^{K} \sum_{i} f_i^k h_i(\mathbf{r}).$$

f_i^k and h_i are here function nodal values in Ω_k and the base functions in $\hat{\Omega}$ respectively, with i representing natural ordering of nodes in $\hat{\Omega}$. Combining coefficients f_i^k one can build function global $\underline{f}$ and local $\underline{f}_L$ representations. Each global degree of freedom occurs only once in the global representation and has multiple copies of faces, edges and vertices related to Ω_k in the local one. To enforce function continuity the global-to-local mapping is defined as the matrix-vector product $\underline{f}_L = Q\underline{f}$, where Q is a binary operator duplicating basis coefficients in adjoining subdomains. The action $Q^T\underline{f}_L$ sums any multiple contributions to the global degree of freedom from their local values. The assembled global stiffness matrix A takes a form

$$(\nabla f, \nabla g) = \underline{f}^T A \underline{g} = \underline{f}^T Q^T A_L Q \underline{g},$$

where $A_L = block{-}diag(A^k)$ is the unassembled stiffness matrix comprising the local stiffness matrices A^k. In practise the global stiffness matrix is never formed and the gather-scatter operator QQ^T is used instead. This operator contains all information about element connectivity.

2.1.2 Additive Schwarz method

Efficient solution of Eq. 4 requires finding an SPD preconditioning matrix M^{-1}, which can be inexpensively applied and which reduces the condition number of $M^{-1}E$. Preconditioners based on domain decomposition are a natural choice for SEM as the data is structured within an element but is otherwise unstructured. An overlapping additive Schwarz preconditioner for Eq. 4 was developed in [24] based on linear finite element discretisation of the Poisson operator. It combines 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

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

For the local problems the restriction and prolongation operators, R_k and R_k^T, are Boolean matrices that transfer data to and from the subdomain, and $\hat{A}_k$ is a local stiffness matrix, which can be inversed with fast diagonalisation method (see Sec. 4 in [23]). Note that action of R_k and R_k^T are similar to the gather-scatter operator QQ^T, so in case of *Nek5000* code this data transfer is built on top of QQ^T routines.

The coarse grid problems corresponds to the Poisson problem 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. In our approach we do not follow

[24] and [23], where the coarse grid operator $\hat{A}_0$ is constructed as the finite element Laplacian using triangulation of the linear subdomains whose vertices are coincident with the element vertices. Instead, we define local SEM-based Neumann operators performing projection of local stiffness matrices A^k evaluated on the GLL quadrature points on the set of the coarse base functions b_i, which represent linear finite element base on GLL grid. The coarse base functions are defined in $\hat{\Omega}$ as a tensor-product of the 1D functions $\hat{b}_{01}(r) = 0.5(r+1)$ and $\hat{b}_{10}(r) = 0.5(1-r)$ for $r \in [-1, 1]$, and each of them corresponds to the single elements vertex for which function's value is 1. The local contribution to $\hat{A}_0$ is given by $b_i^T A^k b_j$, and full $\hat{A}_0$ is finally assembled by local-to-global mapping summing contributions to the global degree of freedom from their local counterparts. $\hat{A}_0$ is one of few matrices really formed in *Nek5000*.

2.1.3 Adaptation for nonconforming meshes

An important advantage of SEM in context of AMR is its spatial decomposition into elements that can be easily split into smaller ones, and use of the local representation of the operators which decouples intra- and inter-element operations. As h-type AMR using conforming-space/nonconforming-mesh approaches leaves the approximation spaces unchanged, most of the tensor-product operations evaluated element-by-element are preserved, limiting the number of changes in the algorithm.

The inter-element operations are mostly performed by gather-scatter operator QQ^T, which has to be redefined to include spectral interpolation at the nonconforming faces. Following [25], we consider nonconforming face shared by one low resolution element (parent) and two (in 3D four) high resolution elements (children). We introduce local parent-to-child interpolation operator J^{cp}, which is a simple spectral interpolation operator with entries

$$(J^{cp})_{ij} = h_j(\zeta_i^{cp}),$$

where ζ_j^{cp} represents the mapping of the GLL points from the child face to its parent. This operator is locally applied to give the desired nodal values on the child face, after Q copies data from the parent to the children. Building block-diagonal matrix J_L comprising local matrices J^{cp} one can redefine scatter $J_L Q$ and gather-scatter $J_L Q Q^T J_L^T$ operators. For more discussion see Fig. 6 and Sec. 4 in [25].

The next crucial modification is diagonalisation of the global mass matrix $Q^T B_L Q$ (B_L is a block-diagonal built of local mass matrices), which inverse is required in Eqs 2 and 3. It is non-diagonal due to the fact that the quadrature points in the elements along the nonconforming faces do not coincide. A diagonalisation procedure is given in [25] and consists of building the global vector $\underline{\tilde{b}}$

$$\underline{\tilde{b}} := B\underline{\hat{e}} = Q^T J_L^T B_L \underline{\hat{e}_L},$$

and finally setting the lumped mass matrix $\tilde{B}_{ij} = \delta_{ij}\tilde{\underline{b}}_i$. $\underline{\hat{e}}$ and $\underline{\hat{e}}_L$ denote here the global and local vectors containing all ones.

The additive Schwarz preconditioner requires two significant modifications. The first one is related to the assembly of the coarse grid operator $\hat{A}_0$, which gets more complex for the

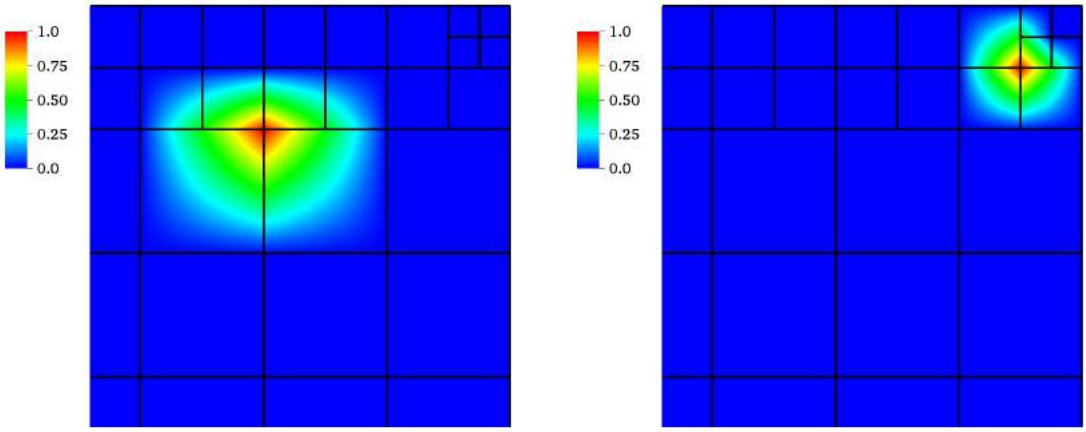


Figure 1: Exemplary shapes of the coarse base functions b_i for the nonconforming meshes. Note, the functions are nonzero in only one vertex corresponding to global degree of freedom but can take any value at the hanging nodes. Element boundaries are marked by black lines.

nonconforming meshes. It is caused by the fact, that the nonconforming mesh introduces hanging vertices located in the middle of faces or edges. These hanging vertices are not global degrees of freedom and cannot be included in $\hat{A}_0$. To remove them from consideration one has to modify set of local coarse base functions b_i, which start to be dependent on the shape of the refined region as well as the position and orientation of the child face with respect to the parent face. Unlike the conforming case, where all b_i could be represented by tensor-product of two simple functions, the nonconforming mesh requires 5 basic components in 2 dimensions and 15 in 3 dimensions to assemble all possible shapes of the coarse base functions. Examples of nonconforming b_i are presented in Fig. 1.

The last missing component are the restriction and prolongation operators, R_k and R_k^T, for the local Poisson problem. Taking into account similarity between these operators with QQ^T and following previous development (Paul Fischer; private communication) we used operator analogous to $J_L QQ^T J_L^T$ replacing J_L with interpolation operator defined on GL quadrature points. This choice seems to be optimal as it preserves properties of the preconditioner and J_L^T is well defined.

This algorithm was implemented and tested within WP2. We describe it in more detail here as the numerical experiment gave an important input for the later algorithm modification. The test case was 2D and 3D unit-size lid driven cavity with Reynolds number $Re = 7500$ and the time step $\delta t = 7.0 \times 10^{-4}$. We compare here performance of the conforming setup with the nonconforming one. At this stage we did not run full AMR simulation and the nonconforming mesh including two refinement levels was fixed in time. The mesh structure and the velocity magnitude of the 2D setups at time $t = 140$ are presented in Fig. 2.

In both cases the mesh was designed to keep minimum amount of elements with resolution

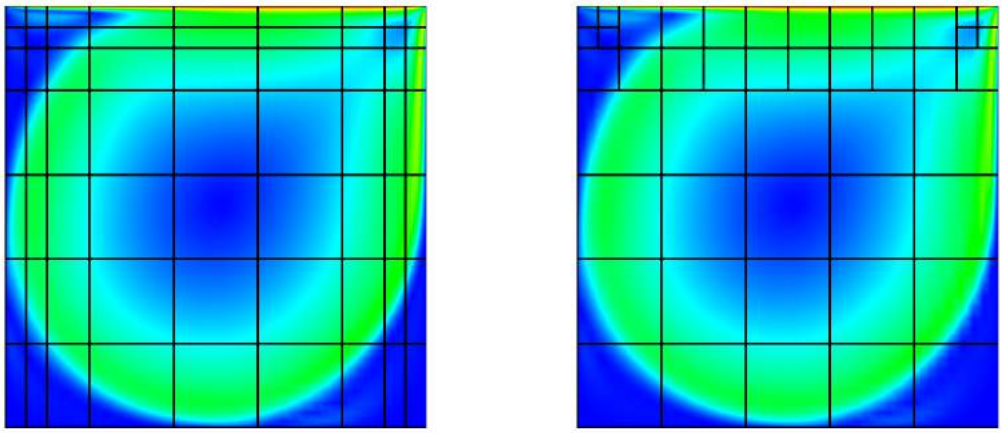


Figure 2: The mesh structure and the velocity magnitude of the 2D lid driven cavity setup at time $t = 140$. Left and right panels present respectively conforming and nonconforming setups. The element boundaries are marked with black lines.

at the cavity centre and at the upper cavity corners equal in both setups. It gave 63 and 46 elements for the conforming and nonconforming setups respectively, which is a 27% reduction in the element number. With these constraints, the CFL number in both simulations was equal 0.5. The generalised minimal residual method (GMRES) was used for iterative solution of Eq. 4.

To validate the setup we compare the velocity profiles at the end of the simulation and the growth of the total energy in the domain presented in Fig. 3. Both the final velocity profiles and the energy growth are very similar in the conforming and nonconforming setups, and the difference in the final total energy value is 1.6% only.

The performance of the pressure solver is presented in Fig. 4 showing the number of

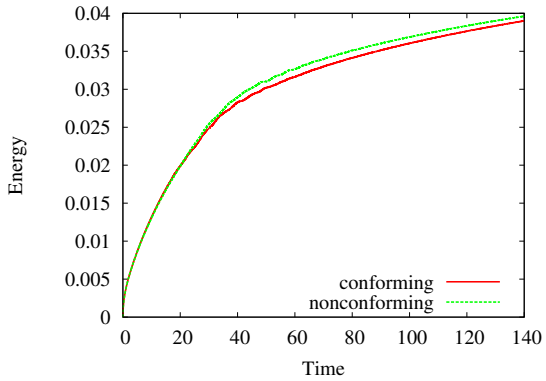


Figure 3: Total energy for the 2D lid driven cavity as a function of time.

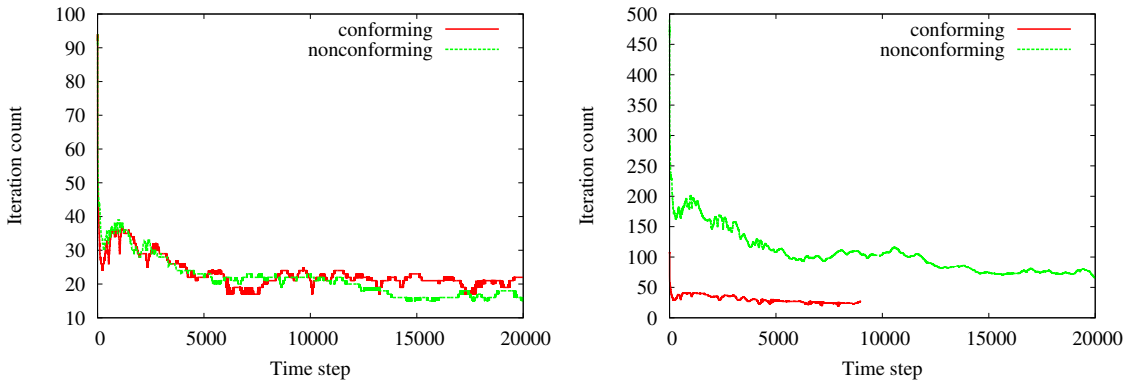


Figure 4: Iteration count of the pressure solver as a function of time step for conforming (red) and nonconforming (green) simulations. Left and right panels present respectively 2D and 3D setups.

iterations as a function of time step. The left plot gives result of the 2D simulation and shows good performance of the nonconforming solver as an increase in solver complexity does not result in visible increase of iteration count. On the contrary, the iteration count at the end of simulation is approximately 20% lower in the nonconforming case, which could be caused by lack of elongated elements in this setup. This effect together with smaller number of elements, gives about 35% reduction in the simulation time, as the conforming and nonconforming simulations took 2.67×10^3 and 1.73×10^3 seconds respectively. We have to stress that it is more than the reduction in the element number.

The 3D setup was designed to preserve 2D nature of this test, so the mesh is periodic in the third dimension. In this case we expect the amount of iterations in the 3D setups to be only slightly higher than in the 2D one, and the 2D results can be used for validation of the 3D run. Right plot in Fig. 4 gives the pressure iteration count for 3D run, showing good performance of conforming solver and significant increase of the iteration number for the nonconforming one. A similar effect is not visible for the velocity iterations.

Closer investigation of the problem showed it to be a result of a way the local restriction and prolongation operators R_k and R_k^T were defined. Although the use of QQ^T in the conforming implementation of R_k and R_k^T suggests use of the operator $J_L Q Q^T J_L^T$ in their nonconforming version, the pressure counterpart of J_L^T was found to introduce significant amount of noise in the third, uniform direction, thereby increasing the iteration count. To reduce the noise, we replaced the transposed interpolation operator with the inverse one, finding significant reduction of iterations. Unfortunately, such preconditioner is no longer SPD and PCG cannot be used as an iterative solver in this case. The other problem is the definition of J_L^{-1}. If J^{cp} is the local parent-to-child interpolation operator the obvious choice is simply J^{cp-1}, however this operation can be performed for the square matrices only excluding p-refinement strategies. To avoid this problem, we define child-to-parent

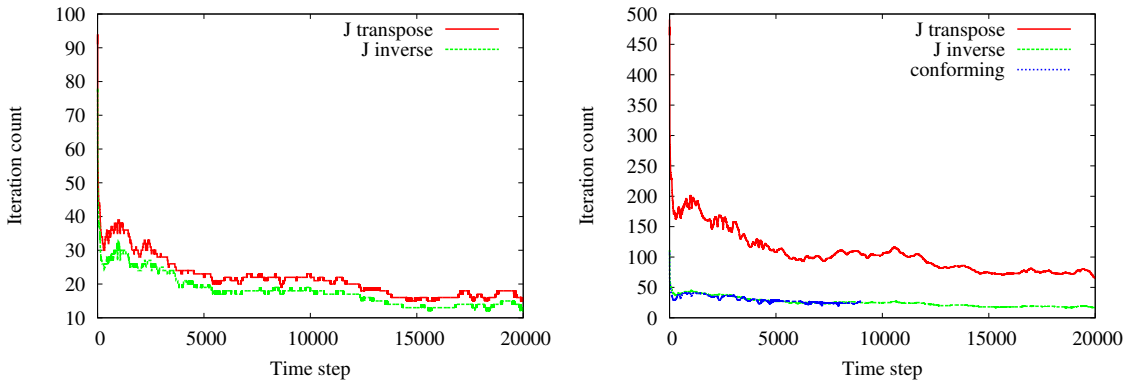


Figure 5: Iteration count of the pressure solver as a function of time step for nonconforming setups using different definitions of the local prolongation operator. Left and right panels present respectively 2D and 3D setups. The left plot shows as well results of the 3D conforming simulation for comparison.

interpolation operator J^{pc} with the entries

$$(J^{pc})_{ij} = \begin{cases} h_j(\zeta_i^{pc}) & \text{if } \zeta_j^p \in \partial\Omega^p \cap \partial\Omega^c \\ 0 & \text{otherwise} \end{cases}$$

where $\partial\Omega^p$ and $\partial\Omega^c$ are the parent and child common faces, ζ_j^p is a parent GLL point at the face $\partial\Omega^p$, and ζ_j^{pc} represents the mapping of ζ_j^p to the child face $\partial\Omega^c$. This operator is locally applied to give the desired nodal values on the child face, before Q^T sums data form the children and the parent. Building block-diagonal matrix J_L^{-1} comprising local matrices J^{pc} one can redefine the gather-scatter $J_L Q Q^T J_L^{-1}$ operator, that is appropriate for pressure preconditioner.

The last approach was implemented in *Nek5000* and tested giving significant improvements in the pressure iteration count. As can be seen on the right plot in Fig. 5 for the 3D case the new preconditioner gives number of pressure iterations similar to the conforming solver. Interestingly this preconditioner gives some reduction of iteration count for 2D case as well (left plot in Fig. 5), even thought there is no uniform direction in this case.

2.2 *A posteriori* spectral error indicators

Spectral error indicators have already been implemented in *Nek5000* as part of the CRESTA project. These indicators follow a method developed by C. Mavriplis [45] and are based on the expected exponential decay of the spectral coefficients with increasing polynomial order. As part of task 1.1 from WP1, we present the algorithm for these indicators in details. Then, they are validated by applying them to the Kovazsnay flow and the flow past a cylinder.

The procedure to compute an estimate of the error committed on the solution of a partial differential equation, called an error indicator, when using the spectral element method is presented in the following steps. The method is described for a one-dimensional flow and

for one element. The computations are easily extended to two- and three-dimensional flows following a similar reasoning and the procedure is repeated on each element of the mesh to get the global map of error indicators.

2.2.1 Step 1: Compute the spectral coefficients

Consider $u(x)$, the exact solution of a 1D partial differential equation, we can compute its spectral transform as

$$u(x) = \sum_{k=0}^{\infty} \hat{u}_k \, p_k(x), \tag{5}$$

where $\hat{u}_k$ are the spectral coefficients and p_k a family of orthogonal polynomials (k denotes the polynomial order).

$$\hat{u}_k = \frac{1}{\gamma_k} \int_{-1}^{1} w(x) u(x) p_k(x) \mathrm{d}x, \tag{6}$$

where w is a weight associated to the family of polynomials and $\gamma_k = \|p_k\|^2_{L^2_w}$. *Nek5000* uses Legendre polynomials and the unknowns in one element are evaluated on the Gauss-Lobatto-Legendre points [18]. In practice, the spectral coefficients are computed via Gauss quadrature using tensor products.

2.2.2 Step 2: Compute the truncation error

In the spectral element method, the expansion from equation (5) does not go to infinity but to some finite polynomial order N, causing a truncation error expressed as

$$\epsilon_t = \left(\sum_{k=N+1}^{\infty} \frac{\hat{u}_k^2}{\frac{2k+1}{2}} \right)^{\frac{1}{2}}. \tag{7}$$

While the spectral coefficients are unknown for $k > N$, an estimate of equation (7) can be computed by assuming that these coefficients follow a decay given by

$$\hat{u}_k \approx c \exp(-\sigma k).$$

The parameters c and σ are computed using a least square best fit with respect to the known coefficients $\hat{u}_k$, $k = 0, ..., N$. In practice, only the four coefficients $N - 3$ to N are considered (if N is higher than or equal 3). Therefore, the truncation error from equation (7) is approximted by the integral

$$\epsilon_t \approx \left(\int_N^{\infty} \frac{(c \exp(-\sigma k))^2}{\frac{2k+1}{2}} \mathrm{d}k \right)^{\frac{1}{2}}.$$

As this integral does not have an analytical solution, we take the upper bound

$$\left(2c^2 \int_N^{\infty} \frac{\exp(-2\sigma k)}{2k+1} \mathrm{d}k \right)^{\frac{1}{2}} \leq \left(\frac{2c^2}{2N+1} \int_N^{\infty} \exp(-2\sigma k) \mathrm{d}k \right)^{\frac{1}{2}},$$

which can be solved and leads to the the following approximate expression of the truncation error

$$\epsilon_t \approx \left(\frac{c^2}{\sigma(2N+1)} \exp(-2\sigma N) \right)^{\frac{1}{2}}. \tag{8}$$

2.2.3 Step 3: Compute the quadrature error

The integral in equation (6) is computed using Gauss quadrature and is exact up to order $2N-1$ only, meaning that coefficient $\hat{u}_N$ is not exact. Consequently, the approximated value of the quadrature error is given by

$$\epsilon_q \approx \left(\frac{\hat{u}_N^2}{\frac{2N+1}{2}} \right)^{\frac{1}{2}}. \tag{9}$$

2.2.4 Step 4: Add both error terms

The final error indicator ϵ_{tot} on one element is computed by adding the contributions from the truncation error (equation 8) and the quadrature error (equation 9)

$$\epsilon_{tot} = \epsilon_t + \epsilon_q.$$

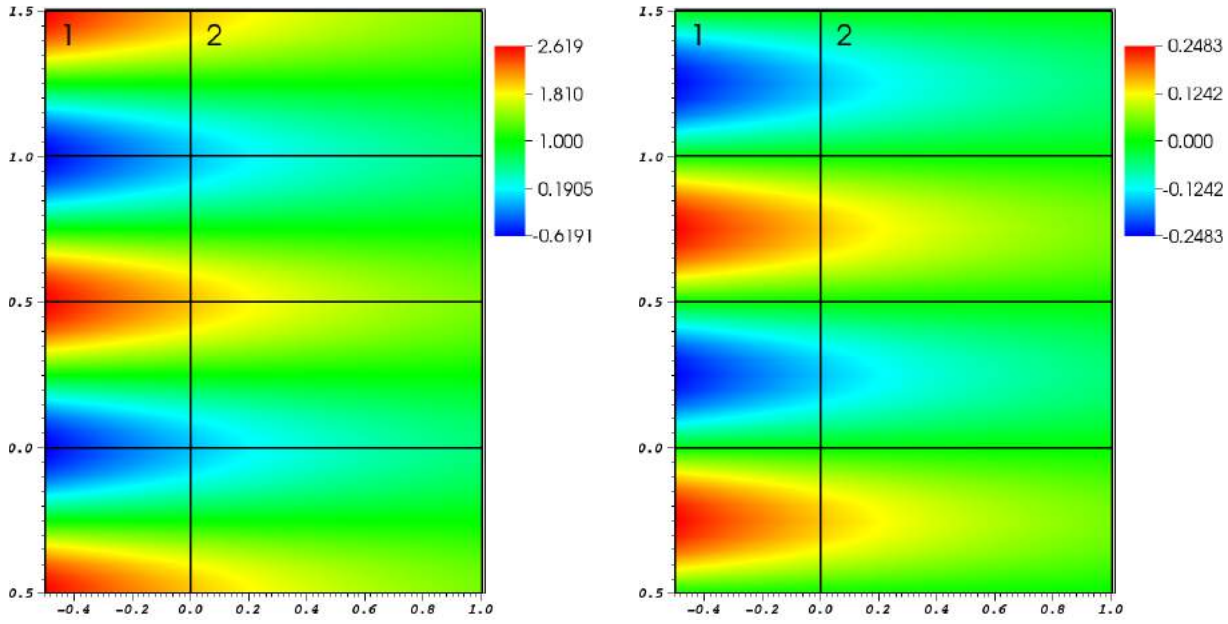
Moreover, the parameter σ is also computed and given as a result. This indication of the strength of the decay of the spectral coefficients might be useful when chosing where to apply refinement.

2.2.5 Validation

Kovasznay flow. We first validate the implementation of the error indicators by using the Kovasznay flow [36], which represents the periodic, steady, two-dimensional, laminar flow behind a grid made of equally spaced rods. As this flow possesses an analytical solution, we can compute the exact error and compare it with the spectral error indicator for various polynomial orders. In figure 6, we plot the horizontal and vertical velocity components of the analytical solution of the Kovasznay flow at a Reynolds number $Re = 40$. We also identify two elements of the mesh where we are going to compute the exact and approximated errors: element 1 in the top left corner and element 2 in the top right corner.

The comparisons between the exact L_2 norm of the error and the spectral error indicator in elements 1 and 2 for both the horizontal and vertical components of the velocity are plotted in figure 7.

The spectral error indicators follow closely the exact errors for all velocity compoments on both elements, validating the method for this simple test case.



(a) u (b) v

Figure 6: Solution of the Kovasznay flow. The error is computed in the elements denoted 1 and 2.

2D flow past a cylinder. In order to test the error indicators on a more realistic case, we consider the two-dimensional flow around a cylinder at a Reynolds number $Re = 500$ for polynomial order $N = 8$. The mesh is deliberately kept quite coarse, and it is expected that the indicators will exhibit high values in regions of the domain where the flow physics is more complex: around the cylinder walls and in the wake, due to high velocity gradients and vortex shedding. We plot the error indicators of the solution at an arbitrary given time in figure 8.

As expected, the error is highest in the previously mentioned critical regions. This test confirms the validity of the local spectral error indicators for practical applications.

2.3 Adjoint error estimators

The main advantage of the spectral error indicators is their easy implementation and the low overhead in computational time. Their main drawback is their tendency to over-resolve non significant regions of the flow. In most engineering applications, one is interested in computing accurately some meaningful physical quantities only, such as the drag or the lift in the case of a wing for example. Therefore, adjoint based error estimators will be implemented in a goal-oriented approach. These estimators compute the sensitivity of a given functional with respect to a mesh variation, allowing to refine regions of the flow relevant for the physical quantity of interest only. As a first step towards the implementation of such estimators, we describe the algorithm in the case of a steady flow, based on a paper by Hoffman [30].

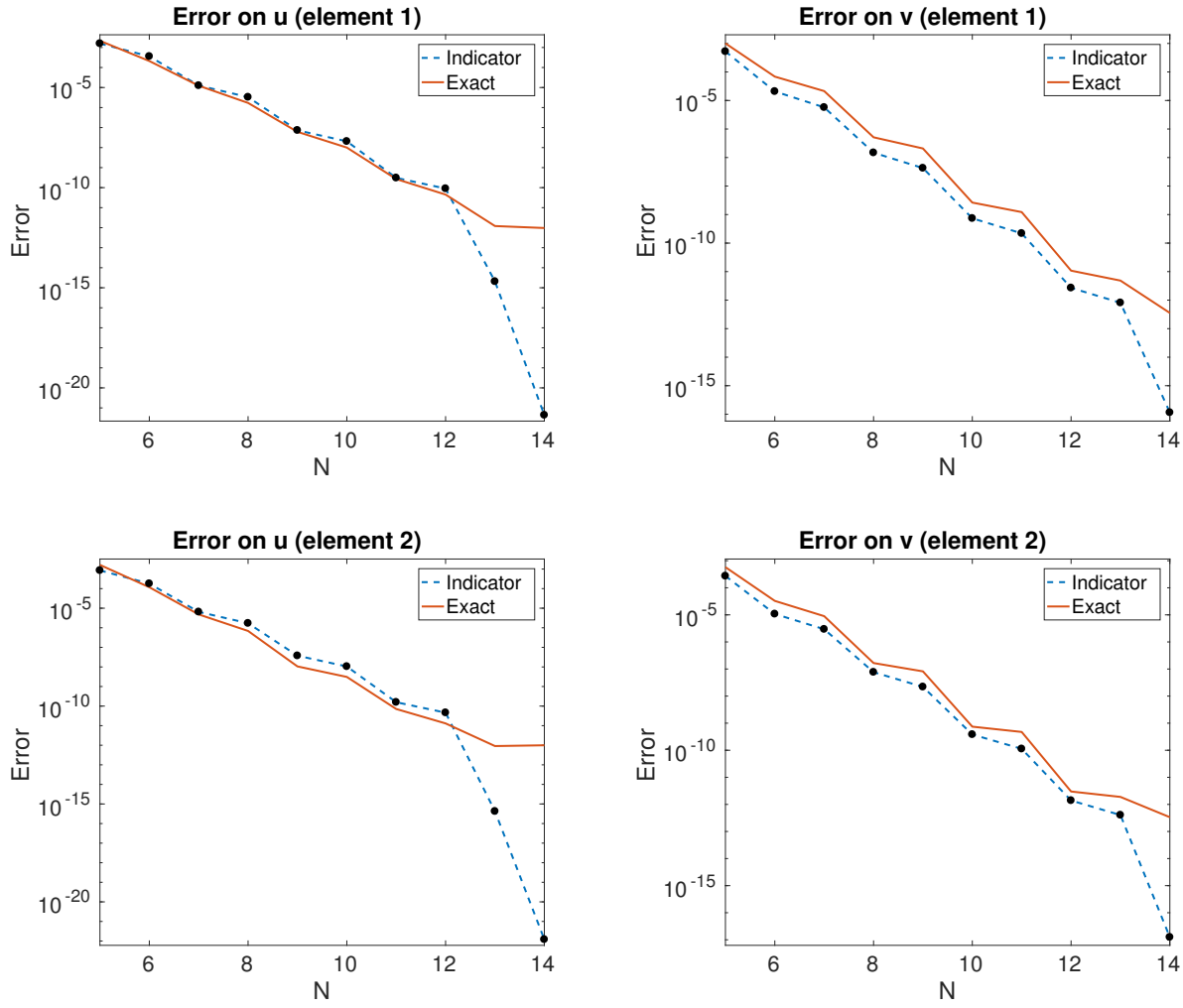


Figure 7: Exact error and error indicator as a function of the polynomial order N for u and v on elements 1 and 2.

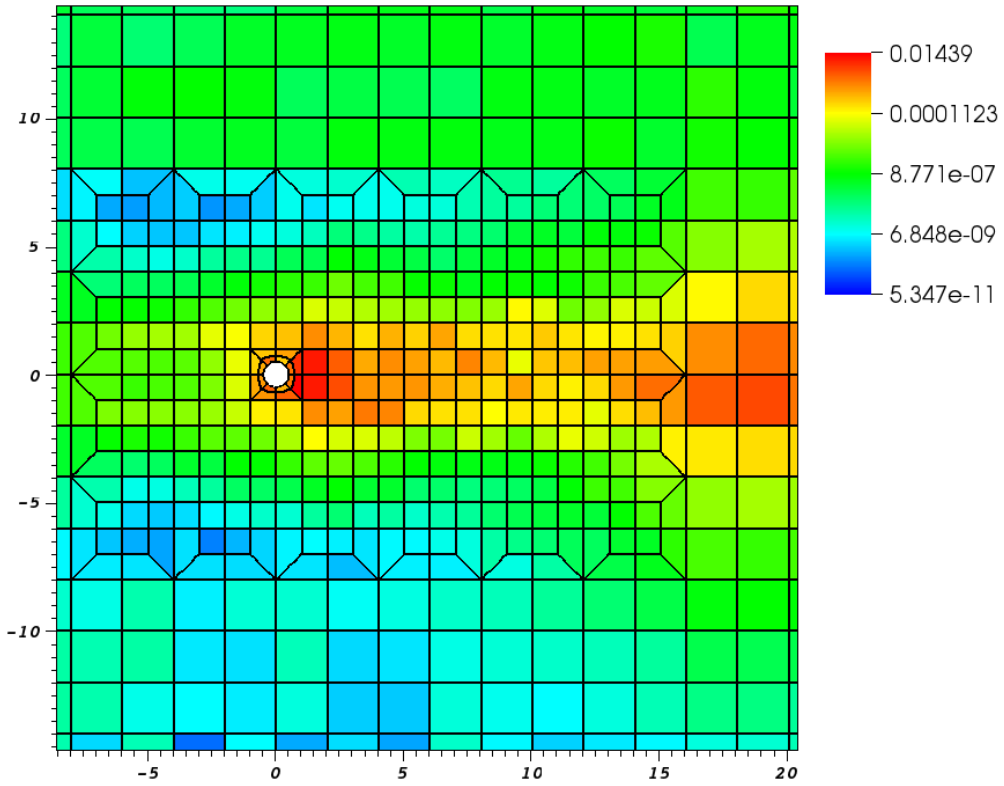


Figure 8: Error indicators for a snapshot of the solution at a given time for the flow past a cylinder at $Re = 500$.

2.3.1 Step 1: Solve the Navier-Stokes equations

First, the steady non-linear Navier-Stokes equations are solved on a domain Ω in $\mathcal{R}^3$, whose boundary is denoted Γ. The non-dimensional formulation of the equations is expressed as

$$\begin{aligned}(\mathbf{u}\cdot\nabla)\mathbf{u} &= -\nabla p + \frac{1}{Re}\Delta\mathbf{u} + \mathbf{f},\\ \nabla\cdot\mathbf{u} &= 0,\\ \mathbf{u}|_{\Gamma_D} &= \mathbf{g}_D,\\ \mathbf{n}\cdot\nabla\mathbf{u}|_{\Gamma_N} &= \mathbf{g}_N,\end{aligned}$$

where Γ_D and Γ_N denote parts of the boundary with Dirichlet and Neumann conditions respectively.

2.3.2 Step 2: Express the functional of interest

The quantity of interest whose sensitivity is to be studied is expressed as a target functional of the form

$$M(\mathbf{q}) = \int_\Omega (\mathbf{u}\cdot\psi + p\chi)\,\mathrm{d}V + \int_\Gamma p\mathbf{n}\cdot\psi_{\mathbf{\Gamma}}\,\mathrm{d}s \tag{10}$$

where $\mathbf{q} = (\mathbf{u}, p)$ is the primal steady solution. The terms ψ, χ and $\psi_{\mathbf{\Gamma}}$ represent some integral weights in the volume and on the border respectively. The expression of the functional is not general but is well suited for a wide range of practical applications. For example, a target functional, which is the sum of the lift and drag on an airfoil, is obtained if $\psi = \chi = 0$ and if $\psi_{\mathbf{\Gamma}}$ is set to the sum of the unit vectors opposite and normal to the flight direction. A more general, for example non-linear, expression for the target functional can be specified if compatibility with the adjoint problem, which is presented immediately, is ensured.

2.3.3 Step 3: Solve the linearized adjoint Navier-Stokes equations

Once a steady solution has been obtained and a target functional defined, the next step is to solve the backward, or adjoint, linearized Navier-Stokes equations. The adjoint equations are expressed as

$$\begin{aligned}-(\mathbf{u}\cdot\nabla)\mathbf{u}^\dagger + (\nabla\mathbf{u})^T\mathbf{u}^\dagger - \nabla p^\dagger - \frac{1}{Re}\Delta\mathbf{u}^\dagger &= \psi,\\ \nabla\cdot\mathbf{u}^\dagger &= \chi,\\ \mathbf{u}^\dagger|_\Gamma &= \psi_{\mathbf{\Gamma}},\end{aligned}$$

where $\mathbf{q}^\dagger = (\mathbf{u}^\dagger, p^\dagger)$ is the adjoint solution and $\mathbf{u}$ is now the constant baseflow. Let us notice that the source terms and boundary conditions for the adjoint equations are directly dependent on (10), the expression of the functional. The solution of this dual problem provides a sensitivity map of the target functional with respect to the primal solution.

2.3.4 Step 4: Adjoint error estimators

The expression for the adjoint error estimators combines the strong residual of the forward solution and adjoint weights based on the interpolation error committed on the dual solution. We start by expressing the error on the output functional between the exact solution of the Navier-Stokes equations $\mathbf{q} = (\mathbf{u}, p)$ and a weak, discrete solution $\hat{\mathbf{q}} = (\hat{\mathbf{u}}, \hat{p})$ obtained via the spectral element method. This is given by

$$\epsilon = \left| M(\mathbf{q}) - M(\hat{\mathbf{q}}) \right|.$$

Using the definition of the adjoint problem and the Galerkin orthogonality, this can be expanded as

$$\begin{aligned}
\epsilon &= \left| M(\mathbf{q}) - M(\hat{\mathbf{q}}) \right|, \\
&= \left| \left(\left(R(\mathbf{q}), \mathbf{q}^\dagger \right) \right) - \left(\left(R(\hat{\mathbf{q}}), \mathbf{q}^\dagger - \hat{\mathbf{q}}^\dagger \right) \right) \right|, \\
&\leq \underbrace{\left| \left(\left(R(\mathbf{q}), \mathbf{q}^\dagger \right) \right) \right|}_{=0} + \left| \left(\left(R(\hat{\mathbf{q}}), \mathbf{q}^\dagger - \hat{\mathbf{q}}^\dagger \right) \right) \right|,
\end{aligned}$$

where $((\cdot, \cdot))$ is a space-time inner product and $R(\mathbf{q})$ denotes the strong residuals

$$R(\mathbf{q}) = \begin{pmatrix} R_1(\mathbf{q}) \\ R_2(\mathbf{u}) \end{pmatrix} = \begin{pmatrix} (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p - \frac{1}{Re} \Delta \mathbf{u} - \mathbf{f} \\ \nabla \cdot \mathbf{u} \end{pmatrix},$$

which are zero only if evaluated on the exact solution. Assuming that the domain Ω is split in K spectral elements, we apply the Cauchy-Schwarz inequality to each element and obtain

$$\begin{aligned}
\epsilon &\leq \left| \left(\left(R_1(\hat{\mathbf{q}}), \mathbf{u}^\dagger - \hat{\mathbf{u}}^\dagger \right) \right) \right| + \left| \left(\left(R_2(\hat{\mathbf{u}}), p^\dagger - p^\dagger \right) \right) \right|, \\
&\leq \sum_{k=1}^{K} \left(\left| R_1\left(\hat{\mathbf{q}}\right) \right|_k \cdot \underbrace{\left| \mathbf{u}^\dagger - \hat{\mathbf{u}}^\dagger \right|_k}_{\omega_{1,k}} + \left| R_2\left(\hat{\mathbf{u}}\right) \right|_k \underbrace{\left| p^\dagger - \hat{p}^\dagger \right|_k}_{\omega_{2,k}} \right),
\end{aligned}$$

where the adjoint weights ω_1 and ω_2 represent interpolation errors between the exact and spectral adjoint solutions. In the case of spectral elements, these errors can be approximated *a priori* by (see [18])

$$\begin{aligned}
\omega_{1,k} &= \left| \mathbf{u}^\dagger - \hat{\mathbf{u}}^\dagger \right|_k \leq C N^{-\frac{1}{2}} N^{-m} \left| \left| \mathbf{u}^\dagger \right| \right|_{H^m(\Omega)}, \\
\omega_{2,k} &= \left| p^\dagger - \hat{p}^\dagger \right|_k \leq C N^{-\frac{1}{2}} N^{-m} \left| \left| p^\dagger \right| \right|_{H^m(\Omega)},
\end{aligned}$$

where C is some constant and N is the polynomial order on each element.

In practice, however, the exact solution is unknown and we use instead the approximation

$$\begin{aligned}
\omega_{1,k} &\lesssim C N^{-\frac{1}{2}} N^{-m} \left| \left| \hat{\mathbf{u}}^\dagger \right| \right|_{H^m(\Omega)}, \\
\omega_{2,k} &\lesssim C N^{-\frac{1}{2}} N^{-m} \left| \left| \hat{p}^\dagger \right| \right|_{H^m(\Omega)}.
\end{aligned}$$

In conclusion, the expression for the adjoint error estimators is obtained on each element by multiplying the norm of the strong residuals by an adjoint weight, which is approximately the interpolation error on the dual solution.

2.4 Summary of progress and outlook

Here we summarise our achievements and future plans:

- We have optimised, implemented in *Nek5000* code and tested the pressure preconditioner based on the additive overlapping Schwarz method within the h-type AMR framework. To achieve this we modified the base functions for the assembly of the coarse-grid operator to neglect hanging nodes. We redefined as well the direct stiffness summation operator QQ^T to include spectral interpolation J at the nonconforming faces and edges. Although the natural replacement of QQ^T based on the Helmholtz operator would be $J_L Q Q^T J_L^T$, we found the operator $J_L Q Q^T J_L^{-1}$ superior for the pressure operator. It gave significant reduction of pressure iterations in the 3D lid driven cavity test, even though it does not preserve the operator symmetry and cannot be evaluated with PCG. Most of 2D and 3D tests showed lower number of pressure iterations for the simulations performed with nonconforming meshes in comparison with simulations using conforming ones. The last crucial modification was diagonalisation of the global mass matrix $Q^T B_L Q$, which is non-diagonal due to the fact that the quadrature points in the elements along the nonconforming faces do not coincide.

  In the future we are going to investigate and adapt for AMR framework restricted additive Schwarz [8, 20] and hybrid Schwarz-multigrid preconditioner. We expect them to be superior over the implemented one.

- *A priori* spectral error indicators based on the decay of the spectral coefficients have been thoroughly described and validated on two-dimensional test cases. These indicators have already been included within the h-type AMR framework and applied to the simulation of a two-dimensional backward-facing step. Results will be presented at the DLES11 conference later this year.

- The algorithm for adjoint error estimators has been described in the case of a steady flow. In the future, it will be extended to unsteady flows and implemented. The capabilities of *Nek5000* already include the resolution of the non-linear Navier-Stokes and linearized adjoint equations. Therefore, the remaining steps towards implementation of the adjoint error estimators are the computation of the strong residuals, the computation of the adjoint weights and the combination of both terms to obtain the final error. It is expected that these error estimators will prevent over-resolution compared to the spectral ones, when only the accurate computation of one or several target quantities is required.

- We expect that both the spectral error indicators and adjoint error estimators will be compared and applied to some of the test cases from work package 3, in particular targeting the jet in crossflow and the NACA4412 airfoil.

3 Error control for heterogeneous modelling

Two new error indicators have recently been developed as part of Task 1.2 in WP1, both based on spectral techniques using small-scale Fourier transforms. The crux of the approach involves quantifying whether the spectral energy $E(k)$ (and therefore the Fourier mode amplitude $Y(k)$) of a solution field decreases at a desired (prescribed) minimal acceptable rate/slope r, such that the smallest scale flow structures have the lowest energy content. If the small scales are not resolved well enough then an increase in $E(k)$ will result and instabilities are likely to occur. Being able to determine and quantify the severity of any deviation away from this acceptable slope facilitates the dynamic focussing of resolution in that area. This is extremely beneficial in the context of exascale scientific computing since accurate simulations can be performed with limited floating-point operations (FLOPS) and memory resources by only placing resolution where it is needed and ensuring that the number of superfluous grid points is minimised.

Examples of the reconstructured Fourier amplitudes against a minimal acceptable slope are plotted in Figure 9, for a poorly-resolved and well-resolved simulation. As expected, the spectral energy deviates away from the slope immediately in the poorly-resolved case, whereas the spectral energy decreases faster than the minimal acceptable slope in the well-resolved case as we would hope.

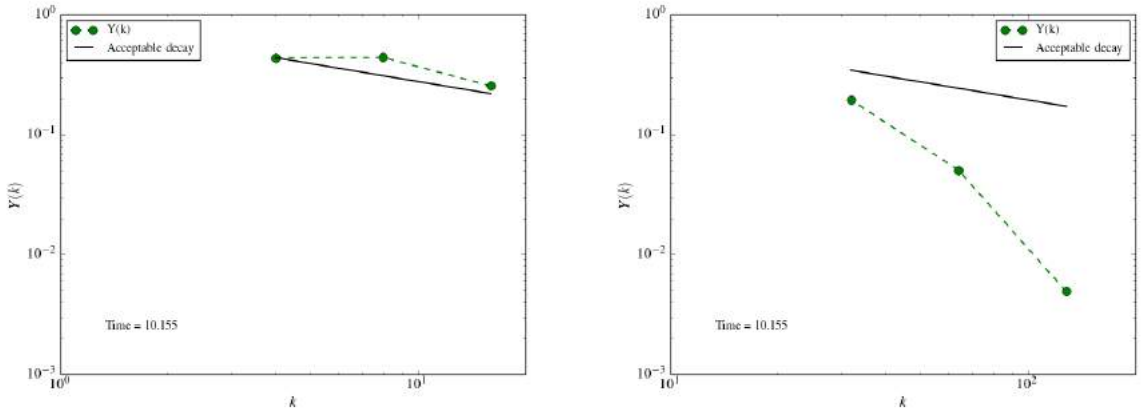


Figure 9: Reconstructed Fourier amplitudes against increasing wavenumber, for a poorly-resolved simulation (left) and a well-resolved simulation (right) of the Taylor-Green vortex problem. The values are the mean amplitudes over all grid points in a representative block in the domain. A minimal acceptable slope of r = -0.5 has also been plotted.

The error indicators were implemented in the OpenSBLI code [32, 33], to take advantage of the ability to target the model's OPS-compliant C code towards a variety of hardware architectures.

3.1 Error indicators

The individual steps of the error indicator algorithm are as follows:

3.1.1 Step 1: Hamming window

Within each N_e^3 block of grid points, loop over each line of grid points in each direction. For each line, apply a Hamming window to the user-specified solution field to prepare it for Fourier analysis:

$$y_j = y_j \frac{\left(0.54 - 0.46\cos\left(\frac{2\pi j}{N_e}\right)\right)}{0.54}, \tag{11}$$

where y_j is the j-th component of the line of solution points y.

3.1.2 Step 2: Fourier amplitude reconstruction

For each line of grid points, the Fourier amplitudes for wavenumbers $N_e/2$, $N_e/4$ and $N_e/8$ of y are computed by using simple summations:

$$S_2 = \sum_{j=1}^{N_e} (-1)^{j-1} y_j, \tag{12}$$

$$S_4 = \sum_{j=1}^{N_e} (-\mathrm{i})^{j-1} y_j, \tag{13}$$

$$S_8 = \sum_{j=1}^{N_e} \exp(-\frac{\pi}{4}\mathrm{i})^{j-1} y_j, \tag{14}$$

where $\mathrm{i} = \sqrt{-1}$.

3.1.3 Step 3: Error severity values

The severity values from the error indicators are then computed. These will be denoted I_i and I_f here, and are integer-valued and floating-point-valued, respectively:

$$I_i = \begin{cases} 1, & \text{if } A_2 > A_4 + \varepsilon \\ 0, & \text{otherwise} \end{cases} + \begin{cases} 1, & \text{if } A_4 > A_8 + \varepsilon \\ 0, & \text{otherwise} \end{cases} + \begin{cases} 1, & \text{if } A_8 > A_2 + \varepsilon \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

$$I_f = \log\left(1 + \lfloor \frac{A_2}{A_4 + \varepsilon} \rfloor + \lfloor \frac{A_4}{A_8 + \varepsilon} \rfloor + \lfloor \frac{A_2}{A_8 + \varepsilon} \rfloor\right), \tag{16}$$

where

$$A_2 = 2^{-2r} \left| \frac{S_2}{N_e} \right|, \tag{17}$$

$$A_4 = 2^{-r} \left| \frac{2S_4}{N_e} \right|, \tag{18}$$

$$A_8 = \left| \frac{2S_8}{N_e} \right|, \tag{19}$$

which are computed in each direction along N_e^2 lines, with the maximum value of A_2, A_4 and A_8 over all lines being used to calculate I_i and I_f. The small value ε is used to avoid division-by-zero problems in the case of uniform flow, and r is the minimal acceptable slope at which the Fourier mode amplitudes should decrease; this is defined by the user and depends on the problem under consideration. We chose a value of $r = 0.5$ here.

3.2 Validation

The 3D Taylor-Green vortex problem [5, 17] was used to evaluate the effectiveness of the error indicators. The domain was a periodic cube of length 2π, and a fourth-order finite difference scheme without additional filtering was used to discretise the domain in space. The grid sizes $N = 32^3$, 64^3, 128^3, 256^3 were considered. A third-order, low-storage Runge-Kutta scheme with a timestep of $\Delta t = 6.77 \times 10^{-3}$ (for the 32^3 grid; this was halved for each successive refinement of the grid) was used to advance the equations forward in time, with each simulation being run until non-dimensional time $T = 20$. All simulations were run on an NVIDIA K40 GPU using the CUDA backend. Further details of the setup and initial conditions are provided in the paper by [32].

The number of 'error indicator blocks' in each direction was set to 4, such that each block contained $(N/4)^3$ solution points. A single value of I_i and I_f were calculated for each block. The solution field y, considered by the error indicators, was chosen to be the z-component of the vorticity field because grid-to-grid point oscillations usually first appear in derivative quantities. It was found that the use of a 32^3 grid resulted in a significant amount of grid-to-grid point oscillation (and thus a considerable amount of solution error), especially at the point of maximum enstrophy. Table 1 and Figure 10 show how I_i and I_f successfully indicate that the error severity decreases as the grid is refined, as expected.

A better visualisation of the effects of refining the grid is given in Figure 11. The total number of high I_i values is greatly reduced throughout time. The point of maximum enstrophy is at $t = 9$–10 where the flow is turbulent. This point in time yields a much greater number of high severity values, which implies that further grid refinement is necessary to adequately resolve the turbulent structures.

The error indicator was improved through the use of overlapping blocks, such that 7^3 blocks were used instead of 4^3 (keeping the number of grid points in each block the same). Thus an error severity value was computed at every $N_e/2$ points in space. Figure 12 shows that the severity values became less uniform (i.e. not all blocks are green or blue like in Figure 11). This overlapping approach therefore gives a more detailed picture of exactly where high-severity errors are in the domain.

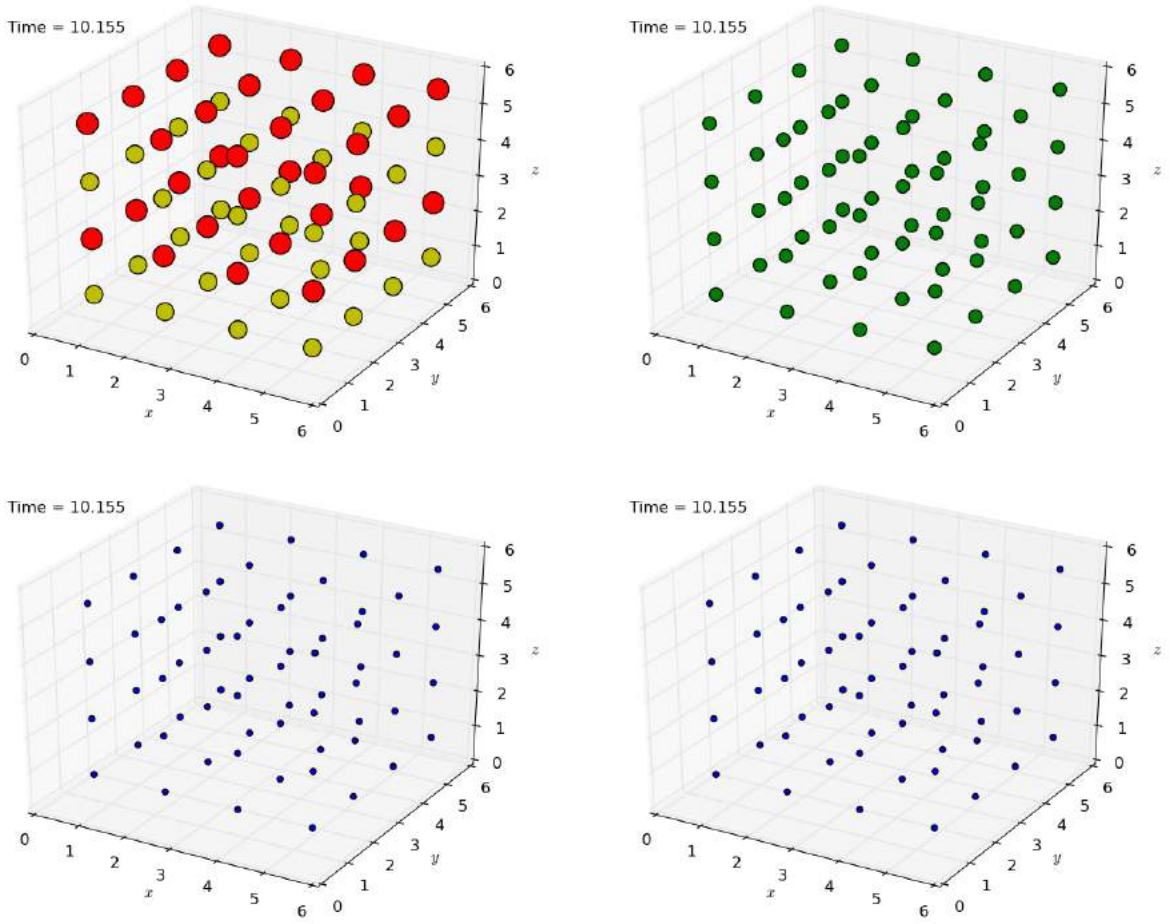


Figure 10: The error severity at all error indicator blocks in a 32^3, 64^3, 128^3 and 256^3 grid (top-left to bottom-right), at $t = 10.155$. Blue, green, yellow and red indicate I_i error severity values of 0, 1, 2 and 3, respectively.

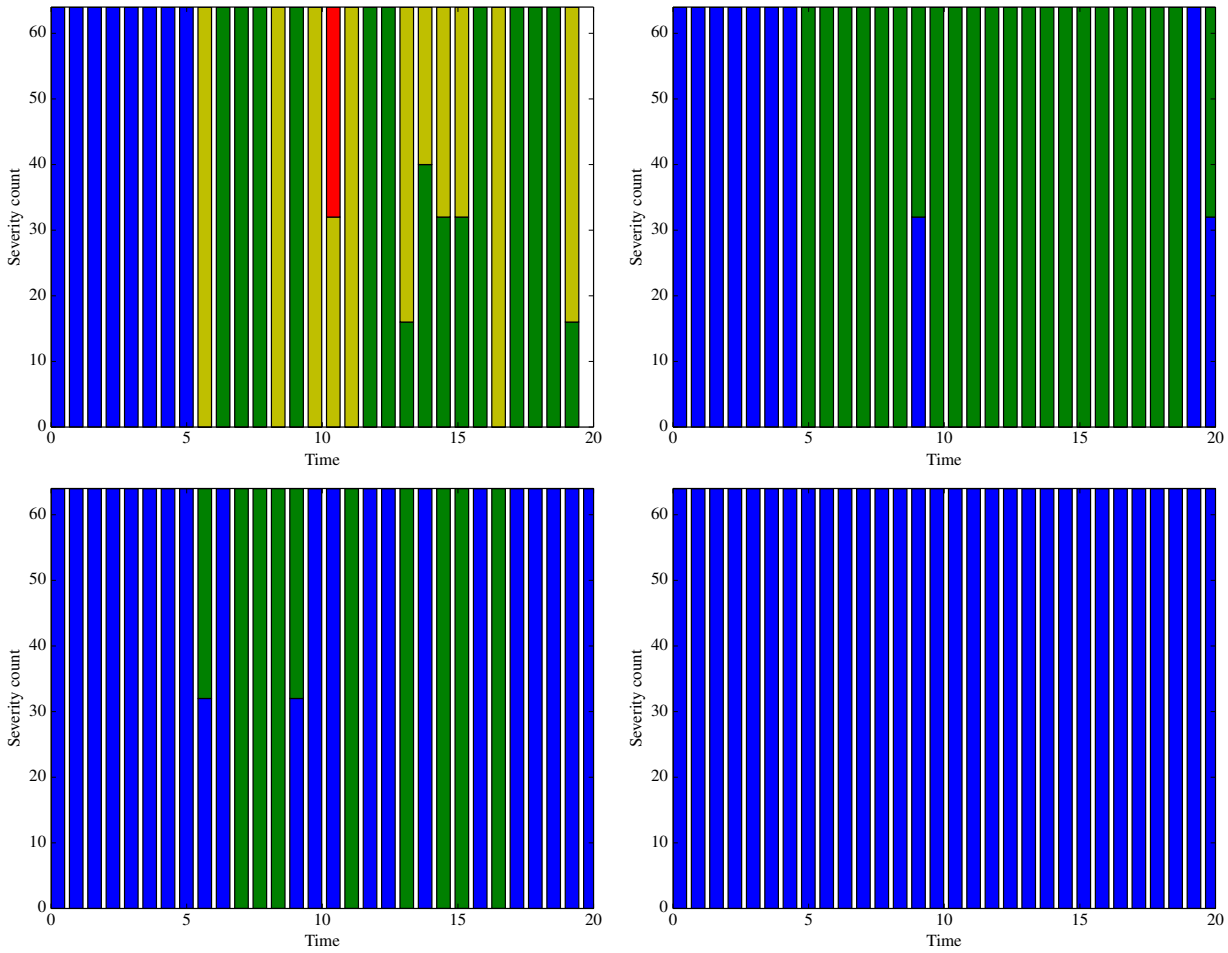


Figure 11: The counts of all the error indicator values across the entire domain for 32^3, 64^3, 128^3 and 256^3 grids (top-left to bottom-right). Blue, green, yellow and red indicate I_i error severity values of 0, 1, 2 and 3, respectively.

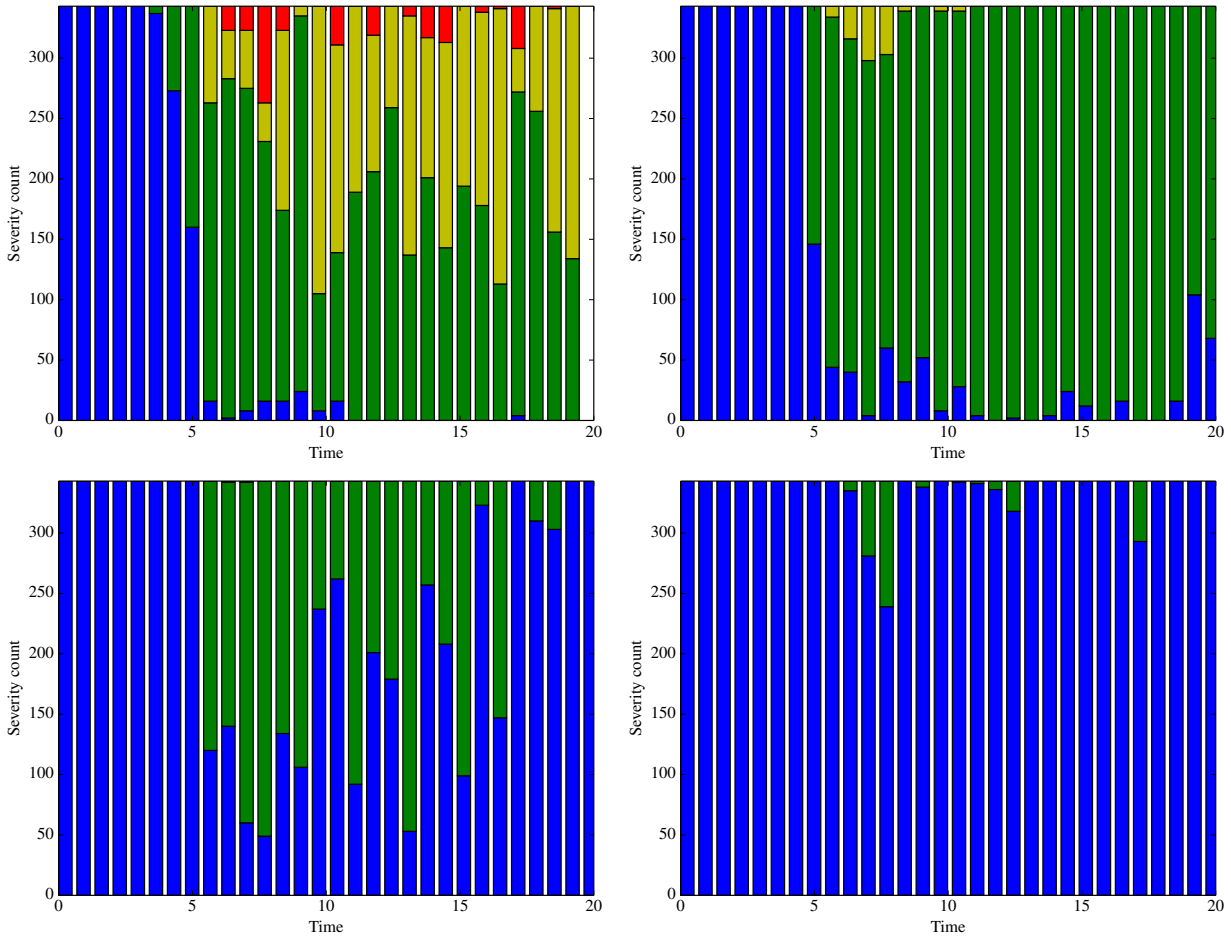


Figure 12: The counts of all the error indicator values across the entire domain for 32^3, 64^3, 128^3 and 256^3 grids (top-left to bottom-right), using overlapping blocks. Blue, green, yellow and red indicate I_i error severity values of 0, 1, 2 and 3, respectively.

3.3 Outlook and future work

An abstract on this work has been accepted for presentation at the ParCFD 2017 conference later in the year. One major limitation is currently the run-time of the error indicator algorithm, which more than doubled the run-time for the 64^3 grid case. The plan for the next 6 months is therefore to (a) improve the efficiency of the implementation as part of WP2, and (b) to apply these error indicators to a simulation of flow past an aerofoil as part of WP3. Results from this will be used to help guide the grid generation process to ensure that a suitably fine grid is produced for the accurate simulation of the highly turbulent flow dynamics.

Grid size	$\max(I_i)$	$\max(I_f)$
32^3	3	1.386294
64^3	1	1.098612
128^3	1	0.693147
256^3	1	0.693147

Table 1: Grid sizes considered in the Taylor-Green vortex simulation, with the maximum severity value of the error indicators over all I/O dumps.

4 Mixed CG-HDG formulation

High-order methods on unstructured grids are now increasingly being used to improve the accuracy of flow simulations since they simultaneously provide geometric flexibility and high fidelity. In ExaFLOW, we are particularly interested in 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 and amongst the most common methods for solving elliptic problems. Compared to its discontinuous counterparts, it involves a smaller number of unknowns (figure 13), 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 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 [67].

Discontinuous Galerkin (DG) methods [2], 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 [14] 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 maximize the potential of each Galerkin variant in a unified setting, the aim of this task is to study a finite element discretization that combines the continuous and discontinuous approach by considering a hybrid discontinuous Galerkin method applied to connected groups of elements supporting a globally continuous polynomial basis. This settings leads naturally to a formulation of weak Dirichlet boundary conditions for the CG method.

In the following sections, we will outline the formulation of this mixed scheme, first concentrating on the weak boundary conditions that are used to weakly impose local Dirichlet

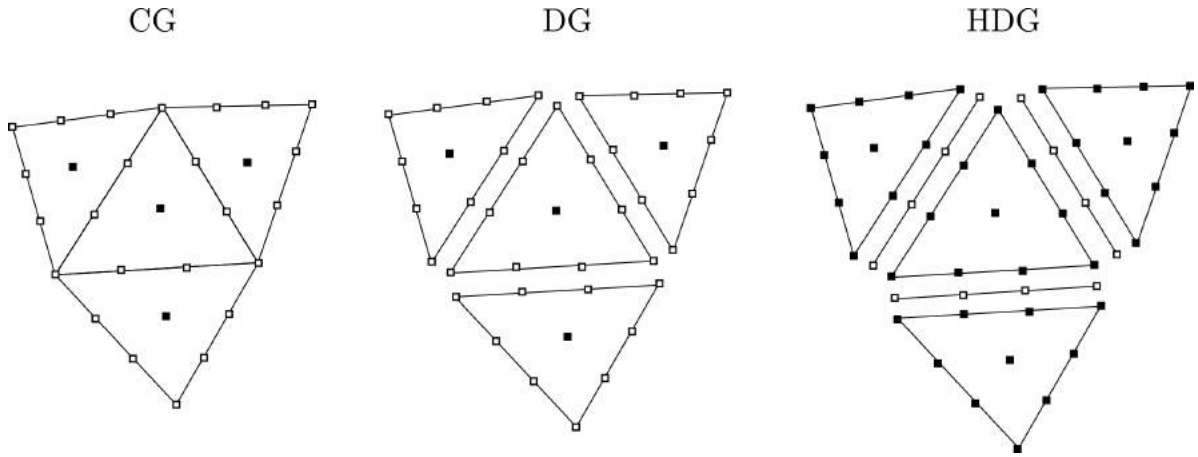


Figure 13: Distribution of unknowns for continuous and discontinuous Galerkin methods.

conditions on each local patch. We will then examine a performance model of the full system to investigate the computational cost associated with solving the problem with either CG, HDG or the mixed CG-HDG scheme.

4.1 Formulation

This section is largely based on paper [34]. We seek solution of the following problem with Dirichlet and Neumann boundary conditions:

$$-\nabla^2 u(\boldsymbol{x}) = f(\boldsymbol{x}) \qquad \boldsymbol{x} \in \Omega, \tag{20}$$

$$u(\boldsymbol{x}) = g_D(\boldsymbol{x}) \qquad \boldsymbol{x} \in \partial\Omega_D, \tag{21}$$

$$\boldsymbol{n} \cdot \nabla u(\boldsymbol{x}) = g_N(\boldsymbol{x}) \qquad \boldsymbol{x} \in \partial\Omega_N, \tag{22}$$

where $\partial\Omega_D \bigcup \partial\Omega_N = \partial\Omega$ and $\partial\Omega_D \bigcap \partial\Omega_N = \emptyset$. To formulate the DG method, we consider a mixed form of (20) by introducing an auxiliary variable $\boldsymbol{q} = \nabla u$:

$$-\nabla \cdot \boldsymbol{q} = f(\boldsymbol{x}) \qquad \boldsymbol{x} \in \Omega, \tag{23}$$

$$\boldsymbol{q} = \nabla u(\boldsymbol{x}) \qquad \boldsymbol{x} \in \Omega, \tag{24}$$

$$u(\boldsymbol{x}) = g_D(\boldsymbol{x}) \qquad \boldsymbol{x} \in \partial\Omega_D, \tag{25}$$

$$\boldsymbol{q} \cdot \boldsymbol{n} = g_N(\boldsymbol{x}) \qquad \boldsymbol{x} \in \partial\Omega_N. \tag{26}$$

The DG methods 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 (23) and (24)

$$\sum_{\Omega^e \in \mathcal{T}_h} \int_{\Omega^e} \left(\nabla v \cdot \boldsymbol{q}^{DG}\right) \, d\boldsymbol{x} - \sum_{\Omega^e \in \mathcal{T}_h} \int_{\partial\Omega^e} v(\boldsymbol{n}^e \cdot \tilde{\boldsymbol{q}}^{DG}) \, ds = \sum_{\Omega^e \in \mathcal{T}_h} \int_{\Omega^e} v \, f \, d\boldsymbol{x} \tag{27}$$

$$\sum_{\Omega^e \in \mathcal{T}_h} \int_{\Omega^e} (\boldsymbol{w} \cdot \boldsymbol{q}^{DG}) \, d\boldsymbol{x} = - \sum_{\Omega^e \in \mathcal{T}_h} \int_{\Omega^e} (\nabla \cdot \boldsymbol{w}) u^{DG} \, d\boldsymbol{x} + \sum_{\Omega^e \in \mathcal{T}_h} \int_{\partial\Omega^e} (\boldsymbol{w} \cdot \boldsymbol{n}^e) \tilde{u}^{DG} \, ds, \tag{28}$$

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 $(u^{DG}, \boldsymbol{q}^{DG})$. Here, $\mathcal{T}_h$ represents an appropriate tesselation of the domain into non-overlapping elements Ω^e so that $\Omega = \bigcup_e \Omega^e$.

4.1.1 Local formulation of the HDG method

Assume that the function

$$\lambda \coloneqq \tilde{u}^{DG} \in \mathcal{M}_h, \tag{29}$$

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

$$\int_{\Omega^e} (\nabla v \cdot \boldsymbol{q}^e)\, d\boldsymbol{x} - \int_{\partial\Omega^e} v(\boldsymbol{n}^e \cdot \tilde{\boldsymbol{q}}^e)\, ds = \int_{\Omega^e} v\, f\, d\boldsymbol{x}, \tag{30}$$

$$\int_{\Omega^e} (\boldsymbol{w} \cdot \boldsymbol{q}^e) d\boldsymbol{x} = -\int_{\Omega^e} (\nabla \cdot \boldsymbol{w}) u^e d\boldsymbol{x} + \int_{\partial\Omega^e} (\boldsymbol{w} \cdot \boldsymbol{n}^e) \lambda\, ds, \tag{31}$$

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

$$\tilde{\boldsymbol{q}}^e(\boldsymbol{x}) = \boldsymbol{q}^e(\boldsymbol{x}) - \tau\big(u^e(\boldsymbol{x}) - \lambda(\boldsymbol{x})\big)\boldsymbol{n}^e \quad \text{on } \partial\Omega^e \tag{32}$$

for some positive function τ.

4.1.2 Global formulation

We denote by $(U_\lambda, \boldsymbol{Q}_\lambda)$ and by $(U_f, \boldsymbol{Q}_f)$ the solution to the local problem (30), (31) when $\lambda = 0$ and $f = 0$, respectively. Due to the linearity of the original problem (20) and its mixed form, the solution satisfies

$$(u^{HDG}, \boldsymbol{q}^{HDG}) = (U_\lambda, \boldsymbol{Q}_\lambda) + (U_f, \boldsymbol{Q}_f). \tag{33}$$

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 (32) is single valued, rendering the numerical trace conservative.

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

$$\lambda = P_h(g_D) \quad \text{on } \partial\Omega_D \tag{34}$$

$$\sum_{\Omega^e \in \mathcal{T}_h} \int_{\partial\Omega^e} \mu \tilde{\boldsymbol{q}} \cdot \boldsymbol{n}\, ds = \int_{\partial\Omega_N} \mu g_N\, ds, \tag{35}$$

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

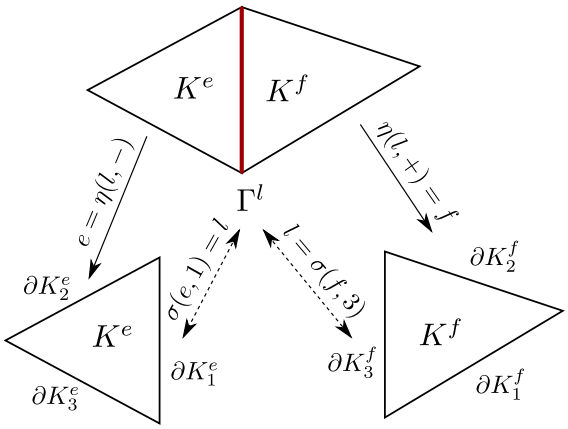


Figure 14: Notation used throughout this section for elements and index mappings.

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 the basis $\phi_j^e(\boldsymbol{x})$ for the expansions over elements and the basis $\psi_j^l(\boldsymbol{x})$ over the traces of the form:

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

4.1.3 Matrix form

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 $i = 1, \ldots, N_u^e$ and vector test functions are represented by $\boldsymbol{e}_k \phi_i$ where $\boldsymbol{e}_1 = [1, 0]^T$ and $\boldsymbol{e}_2 = [0, 1]^T$ in two dimensions:

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

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

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

where σ represents the index mappings of figure 14. 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 (30) and (31) and

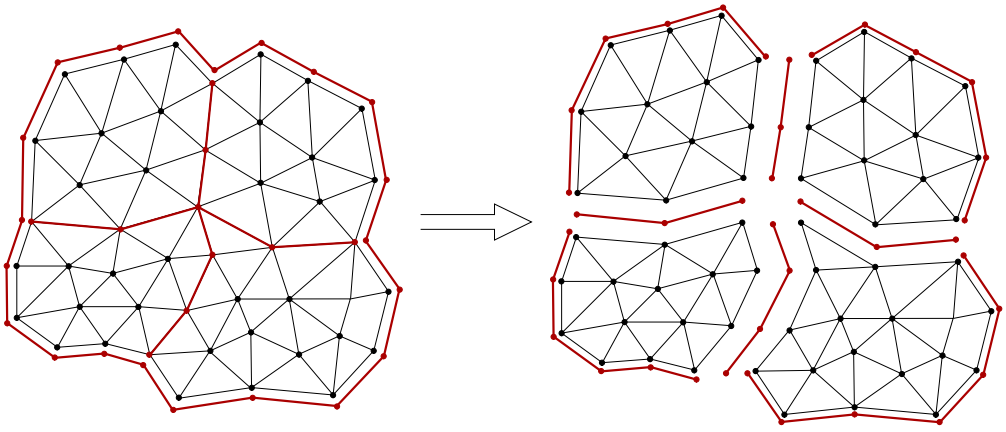


Figure 15: Decomposition of the domain into four macro-elements. Red lines denote the skeleton or trace of the macro-elements on which λ is defined.

using the definition of the flux (32) yields the matrix form of *local solvers*

$$\left[(\mathbf{D}_1^e)^T\ (\mathbf{D}_2^e)^T\right]\begin{bmatrix}\underline{\hat{q}}_1^e\\ \underline{\hat{q}}_2^e\end{bmatrix} - \sum_{l=1}^{N_b^e}\left[\widetilde{\mathbf{E}}_{1l}^e\ \widetilde{\mathbf{E}}_{2l}^e\right]\begin{bmatrix}\underline{\hat{q}}_1^e\\ \underline{\hat{q}}_2^e\end{bmatrix} + \sum_{l=1}^{N_b^e}\tau^{e,l}\left[\mathbf{E}_l^e\underline{\hat{u}}^e - \mathbf{F}_l^e\underline{\hat{\lambda}}^{\sigma(e,l)}\right] = \underline{f}^e \tag{36}$$

$$\mathbf{M}^e\underline{\hat{q}}_k^e = -(\mathbf{D}_k^e)^T\underline{\hat{u}}^e + \sum_{l=1}^{N_b^e}\widetilde{\mathbf{F}}_{kl}^e\underline{\hat{\lambda}}^{\sigma(e,l)} \qquad k=0,1 \tag{37}$$

The *global equation for* λ can be obtained by discretizing the transmission condition (35). We introduce local element-based and edge-based matrices

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

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

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

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

4.2 Combined Continuous-Discontinuous Formulation

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. This means that the scalar flux (hybrid variable) λ is only defined on inter-partition boundaries. A visualisation of this is shown in figure 15. In this section, we will discuss the mathematical formulation of the local problems, and show how this translates into a weak imposition of Dirichlet boundary conditions for each macro-element.

4.2.1 Continuous-Discontinuous Solver

The motivation of this section is to take the matrix form of the HDG solver and apply it in continuous setting. Intuitively, we would expect the discrete weak form to reduce to the 'standard' Laplace operator, accompanied by extra terms, which will be only applied on elements adjacent to partition boundaries, providing weak coupling between each partition and the global trace variable. We will show that this is indeed the case and examine the convergence properties of the local solvers in this setting, compared to the strongly-imposed Dirichlet conditions that are standard to the fully-continuous method.

After expressing the flux variable in equation (37) as

$$\underline{\hat{q}}_k^e = \left(\mathbf{M}^e\right)^{-1} \left\{ -(\mathbf{D}_k^e)^T \underline{\hat{u}}^e + \sum_{l=1}^{N_b^e} \widetilde{\mathbf{F}}_{kl}^e \underline{\hat{\lambda}}^{\sigma(e,l)} \right\} \qquad k = 1, 2 \tag{38}$$

and inserting into (36), the local solver becomes

$$\begin{aligned} &\sum_{k=1}^{2} \left\{ \left(\left(\mathbf{D}_k^e\right)^T - \sum_{l=1}^{N_b^e} \tilde{\mathbf{E}}_{kl}^e \right) \left(\mathbf{M}^e\right)^{-1} \left(-\left(\mathbf{D}_k^e\right)^T \underline{\hat{u}}^e + \sum_{l=1}^{N_b^e} \tilde{\mathbf{F}}_{kl}^e \underline{\hat{\lambda}}^{\sigma(e,l)} \right) \right\} \\ &+ \sum_{l=1}^{N_b^e} \tau^{e,l} \left[\mathbf{E}_l^e \underline{\hat{u}}^e - \mathbf{F}_l^e \underline{\hat{\lambda}}^{\sigma(e,l)} \right] = \underline{f}^e \end{aligned} \tag{39}$$

Note that in equation (36), we can assume that $\mathbf{E}_l^e = \mathbf{F}_l^e$ if the primal variable u and the hybrid variable λ use the same expansion basis on element traces. Equation (36) then becomes

$$\left((\mathbf{D}_1^e)^T - \sum_{l=1}^{N_b^e} \tilde{\mathbf{E}}_{1l}^e \right) \underline{\hat{q}}_1^e + \left((\mathbf{D}_2^e)^T - \sum_{l=1}^{N_b^e} \tilde{\mathbf{E}}_{2l}^e \right) \underline{\hat{q}}_2^e = \underline{f}^e \tag{40}$$

or with $\underline{\hat{q}}^e$ expressed in terms of $\underline{\hat{u}}^e$:

$$\sum_{k=0,1} \left\{ \left(\left(\mathbf{D}_k^e\right)^T - \sum_{l=1}^{N_b^e} \tilde{\mathbf{E}}_{kl}^e \right) \left(\mathbf{M}^e\right)^{-1} \left(-\left(\mathbf{D}_k^e\right)^T \underline{\hat{u}}^e + \sum_{l=1}^{N_b^e} \tilde{\mathbf{F}}_{kl}^e \underline{\hat{\lambda}}^{\sigma(e,l)} \right) \right\} = \underline{f}^e \tag{41}$$

On affine elements, we can write

$$\left((\mathbf{D}_k^e)^T - \sum_{l=1}^{N_b^e} \tilde{\mathbf{E}}_{kl}^e \right) = -\mathbf{D}_k^e, \quad k = 1, 2\,, \tag{42}$$

which is a discrete representation of integration by parts:

$$\int_{\Omega^e} \phi_i \frac{\partial}{\partial x_k} \phi_j d\boldsymbol{x} = -\int_{\Omega^e} \frac{\partial}{\partial x_k} \phi_i \phi_j d\boldsymbol{x} + \int_{\partial\Omega^e} \phi_i \phi_j \boldsymbol{n}_k^e ds. \tag{43}$$

After substituting the discrete identity into the local solver (40), we obtain

$$-\mathbf{D}_1^e \hat{\underline{q}}_1^e - \mathbf{D}_2^e \hat{\underline{q}}_2^e = \underline{f}^e, \tag{44}$$

which is a discrete counterpart of the original mixed form $-\nabla \cdot \boldsymbol{q} = f(\boldsymbol{x})$.

Now suppose that λ is a known value on domain boundary. With the aid of (42), equation (39) can be simplified as:

$$\begin{aligned}
&\sum_{k=1}^{2} \left\{ \left(\left(\mathbf{D}_k^e\right)^T - \sum_{l=1}^{N_b^e} \tilde{\mathbf{E}}_{kl}^e \right) \left(\mathbf{M}_e^{-1}\right) \left(\mathbf{D}_k^e - \sum_{l=1}^{N_b^e} \tilde{\mathbf{E}}_{kl}^e \right) \right\} \hat{u}^e \\
&+ \sum_{k=1}^{2} \left\{ \underbrace{\left(\left(\mathbf{D}_k^e\right)^T - \sum_{l=1}^{N_b^e} \tilde{\mathbf{E}}_{kl}^e \right)}_{-\mathbf{D}_k^e} \left(\mathbf{M}_e^{-1}\right) \sum_{l=1}^{N_b^e} \tilde{\mathbf{F}}_{kl}^e \hat{\underline{\lambda}}^{\sigma(e,l)} \right\} \\
&+ \sum_{l=1}^{N_b^e} \tau^{(e,l)} \left[\mathbf{E}_l^e \hat{\underline{u}}^e - \mathbf{F}_l^e \hat{\underline{\lambda}}^{\sigma(e,l)} \right] = \underline{\boldsymbol{f}}^e
\end{aligned} \tag{45}$$

The terms $\left(\mathbf{D}_k^e\right)^T \left(\mathbf{M}^e\right)^{-1} (\mathbf{D}_k^e)$ for $k = 1, 2$ represent the standard Laplace operator as discretized by the continuous Galerkin method. This is the main reason why the expression (39) was manipulated using (42) - to obtain the product of matrices $\mathbf{D}^T\mathbf{M}\mathbf{D}$ in correct order that is identical to CG discretization. The final discrete form on elements which are adjacent to domain boundary is therefore

$$\begin{aligned}
&\sum_{k=1}^{2} \Bigg\{ (\mathbf{D}_k^e)^T \left(\mathbf{M}^e\right)^{-1} \mathbf{D}_k^e + \left(\sum_{l=1}^{N_b^e} \tilde{\mathbf{E}}_{kl}^e \right) \left(\mathbf{M}^e\right)^{-1} \left(\sum_{l=1}^{N_b^e} \tilde{\mathbf{E}}_{kl}^e \right) \\
&\quad - \left(\sum_{l=1}^{N_b^e} \tilde{\mathbf{E}}_{kl}^e \right) \left(\mathbf{M}^e\right)^{-1} \mathbf{D}_k^e - \left(\mathbf{D}_k^e \right)^T \left(\mathbf{M}^e\right)^{-1} \left(\sum_{l=1}^{N_b^e} \tilde{\mathbf{E}}_{kl}^e \right) \Bigg\} \hat{\underline{u}}^e + \sum_{l=1}^{N_b^e} \tau^{(e,l)} \mathbf{E}_l^e \hat{\underline{u}}^e \\
&\quad = \underline{\boldsymbol{f}}^e + \sum_{k=1}^{2} \left\{ \mathbf{D}_k^e \left(\mathbf{M}^e\right)^{-1} \left(\sum_{l=1}^{N_b^e} \tilde{\mathbf{F}}_{kl}^e \hat{\underline{\lambda}}^{\sigma(e,l)} \right) \right\} + \sum_{l=1}^{N_b^e} \tau^{(e,l)} \mathbf{F}_l^e \hat{\underline{\lambda}}^{\sigma(e,l)} .
\end{aligned} \tag{46}$$

This formulation applied to a single domain enables the *weak imposition of Dirichlet boundary conditions* given by the hybrid variable $\boldsymbol{\lambda}$. A single domain in this setting is no longer one element, but a group of elements supporting a piecewise-continuous basis.

Remark 1. *Using again the identity*

$$\left((\mathbf{D}_k^e)^T - \sum_{l=1}^{N_b^e} \tilde{\mathbf{E}}_{kl}^e \right) = -\mathbf{D}_k^e, \quad k = 1, 2\,, \tag{47}$$

the second and fourth LHS terms in (46) *can be merged and the above variational form becomes:*

$$\begin{aligned}
&\sum_{k=1}^{2} \Bigg\{ \underbrace{(\mathbf{D}_k^e)^T \big(\mathbf{M}^e\big)^{-1} \mathbf{D}_k^e}_{\boxed{1}} - \underbrace{\left(\sum_{l=1}^{N_b^e} \tilde{\mathbf{E}}_{kl}^e \right) \big(\mathbf{M}^e\big)^{-1} \mathbf{D}_k^e}_{\boxed{2}} \Bigg\} \hat{\underline{u}}^e \\
&+ \sum_{k=1}^{2} \underbrace{\left\{ \mathbf{D}_k^e \big(\mathbf{M}^e\big)^{-1} \left(\sum_{l=1}^{N_b^e} \tilde{\mathbf{E}}_{kl}^e \right) \right\}}_{\boxed{3L}} \hat{\underline{u}}^e + \underbrace{\sum_{l=1}^{N_b^e} \tau^{(e,l)} \mathbf{E}_l^e \hat{\underline{u}}^e}_{\boxed{4L}} \\
&= \underbrace{\underline{\boldsymbol{f}}^e}_{\boxed{5}} + \sum_{k=1}^{2} \underbrace{\left\{ \mathbf{D}_k^e \big(\mathbf{M}^e\big)^{-1} \left(\sum_{l=1}^{N_b^e} \tilde{\mathbf{F}}_{kl}^e \hat{\underline{\lambda}}^{\sigma(e,l)} \right) \right\}}_{\boxed{3R}} + \underbrace{\sum_{l=1}^{N_b^e} \tau^{(e,l)} \mathbf{F}_l^e \hat{\underline{\lambda}}^{\sigma(e,l)}}_{\boxed{4R}} .
\end{aligned}$$

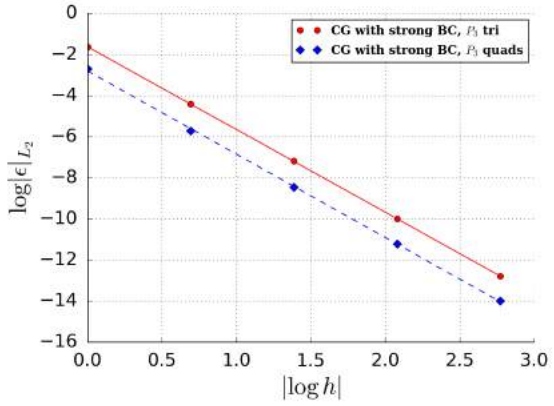
The meaning of the terms is as follows:

- $\boxed{1}$ - *Laplace term*
- $\boxed{2}$ - *Neumann boundary term*
- $\boxed{3}$ *and* $\boxed{4}$ - *weak Dirichlet terms (left- and right-hand side components)*
- $\boxed{5}$ - *source (forcing) term*

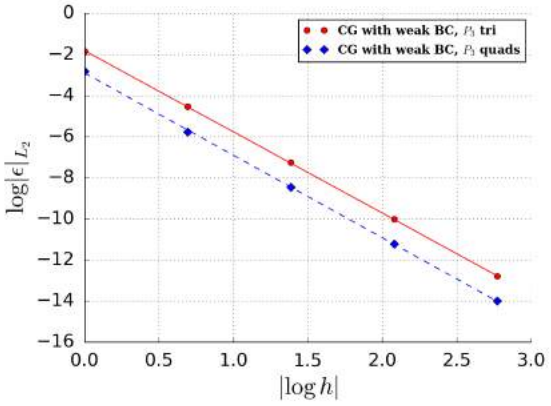
4.2.2 Convergence rates comparison: weak vs. strong boundary conditions

To check the validity of the scheme, we have evaluated the accuracy of weakly imposed Dirichlet boundary conditions on a scalar Helmholtz problem solved in a square domain $(-1, 1)^2$ with zero Dirichlet boundary conditions. To verify the convergence rates, the problem was first solved on a series of meshes with increasing number of unknowns and third-order elements. We then repeated the test on a fixed triangular and quadrilateral grid with polynomial degrees increasing from 2 to 14.

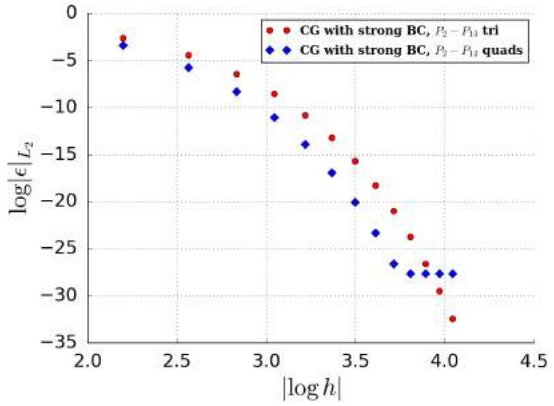
Figure 16 shows that convergence rate in strong and weak settings is virtually identical. The solution of linear system with weak boundary conditions included is slightly slower because the Dirichlet degrees of freedom are not eliminated *a priori* as in the strong case.



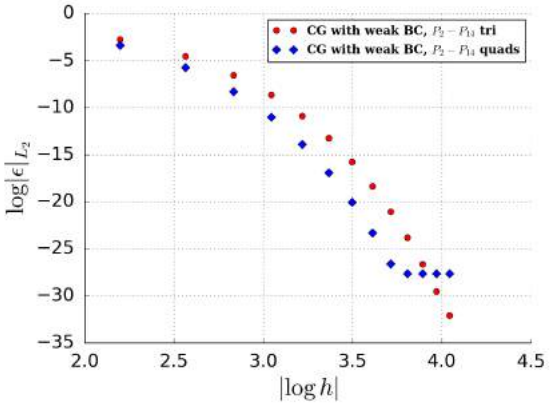
(a) h-convergence, strong boundary conditions.



(b) h-convergence, weak boundary conditions.

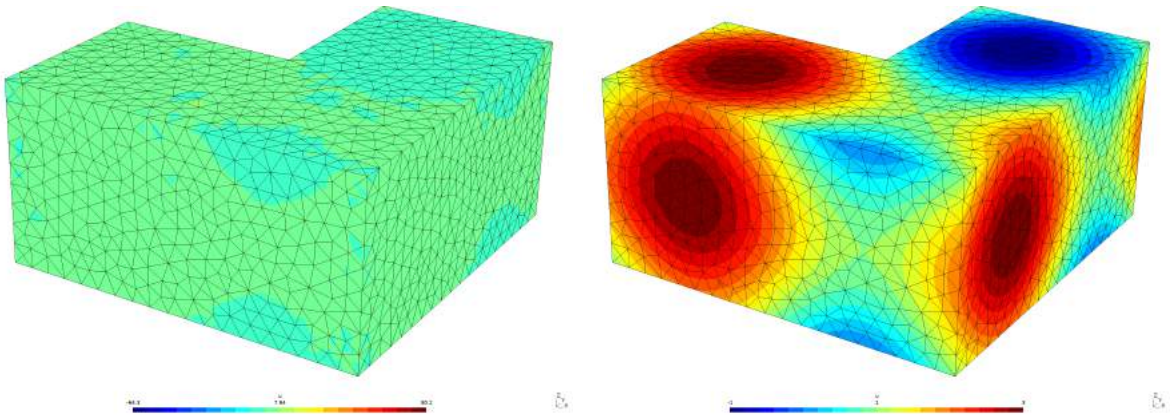


(c) p-convergence, strong boundary conditions.



(d) p-convergence, weak boundary conditions.

Figure 16: Convergence to the exact solution in L_2 norm.



Nitsche, $\eta = 10^4$ Nitsche, $\eta = 10^6$

Figure 17: Comparison of different values of η for the Nitsche method in an L-shaped domain of third-order tetrahedra.

4.3 Weak Dirichlet boundary conditions: benefits and applications

In this section, we briefly describe some of the benefits and applications of the derivation of this weak boundary condition in terms of the extension of existing literature, as well as applications to the ExaFLOW test cases.

4.3.1 Comparison against alternative methods

The HDG method which we departed from in order to derive the formulation of weak boundary conditions is not the only finite element discretization that uses a penalty term (called τ in HDG) in order to enforce constraints. The classical procedure in the continuous case is known as Nitsche's method. Consider a Poisson equation

$$-\Delta u = f \text{ in } \Omega$$

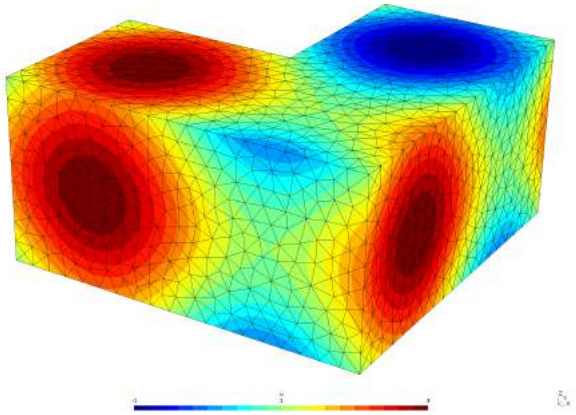
with Dirichlet boundary condition $u|_{\partial\Omega} = g$.

Nitsche's method leads to the following weak form:

$$\int_\Omega uv \, d\boldsymbol{x} - \int_{\partial\Omega} \frac{\partial u}{\partial n} v \, d\boldsymbol{s} - \int_{\partial\Omega} \frac{\partial v}{\partial n} g \, d\boldsymbol{s} + \eta \int_{\partial\Omega} uv \, d\boldsymbol{s} = \int_\Omega fv \, d\boldsymbol{x} - \int_{\partial\Omega} \frac{\partial v}{\partial n} g \, d\boldsymbol{s} + \eta \int_{\partial\Omega} gv \, d\boldsymbol{s},$$

where η is a penalty parameter that enforces the coercivity of the bilinear form on the left-hand side, thus enabling the convergence of the solution. This parameter is case-dependent and generally large, which causes an issue in terms of determining an appropriately sized penalty term in order to achieve convergence and accuracy of the obtained solution.

As an example, consider the Poisson equation with $g = 1 + \sin(\pi x) + \sin(\pi y) + \sin(\pi z)$ in a 3D L-shaped domain discretized by third-order nodal tetrahedra, shown in figure 17.



weak BC, $\tau = 1.0$

Figure 18: The Poisson problem in the L-shaped domain of third-order tetrahedra, but instead using the HDG weak boundary conditions with $\tau = 1$.

We see here that the solution is only correct for a sufficiently large choice of the penalty parameter η. However, if we apply the formulation of the previous section, where weak boundary conditions are obtained by considering the problem in the HDG setting, we obtain the picture in figure 18. Here, simply selecting $\tau = 1$ is sufficient to obtain a correct solution. As seen elsewhere in the HDG literture, we have found that generally the selection of τ does not significantly alter the accuracy of the solutions that are obtained. In this sense, the weak boundary condition formulation is 'almost' parameter free. This would therefore seem to be a more practical alternative to Nitsche's method when the weak imposition of Dirichlet boundary conditions is required.

4.3.2 Applications in the context of ExaFLOW WP3 test cases

We intend to use the weak boundary conditions for the simulation of turbulent flows, particularly in cases when the flow features are not fully resolved and when the imposition of strong Dirichlet conditions can cause numerical convergence issues. Given the fact that the weak enforcement of no-slip condition on wall boundaries allows for the computed velocity to differ from imposed values to larger extent than in strong case, we would like to understand whether this relaxation of constraints would render the solver more stable in applications where stability has been previously been an issue.

In particular, our goal is to understand the application of these weak boundary conditions to the Formula 1 case being investigated in WP3, where the contact patch between moving tyre and stationary road induces a natural discontinuity in the imposition of Dirichlet boundary conditions. This causes large increases in the pressure, which can further lead to numerical instability. We aim to investigate whether the application of the weak boundary conditions derived here will aide in alleviating some of these numerical issues.

As an initial test in three-dimensions, we have run simulations of an incompressible flow in

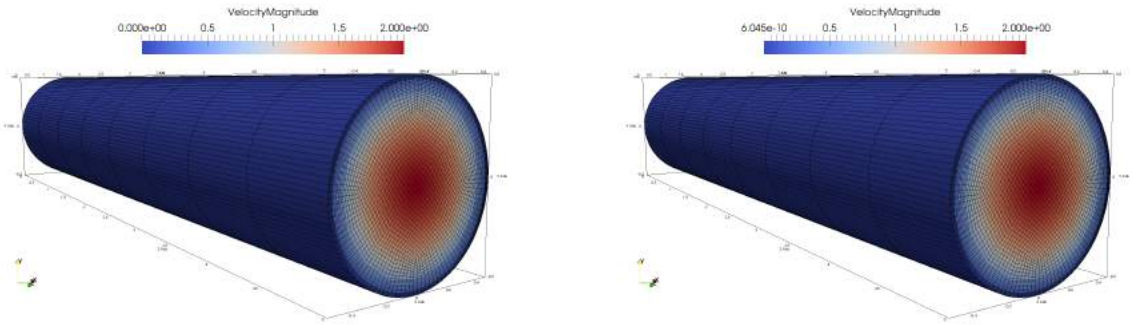


Figure 19: Laminar solution with strong (left) and weak (right) boundary conditions for velocity.

an infinite pipe using a mixed spectral/*hp* element and Fourier pseudospectral discretization available in Nektar++. The Fourier expansion is used in the streamwise direction and the spectral/hp elements are used to capture the circular cross-section, so that the weak boundary conditions are imposed in circular cross-sections at the wall for the velocity field. An example solution of laminar flow with $\mathrm{Re} = 250$ obtained on seventh-order elements with 8 Fourier modes in streamwise direction is presented in figure 19. Note that in the figure depicting the strongly imposed boundary conditions, the velocity field has a minimum of precisely zero, whereas for the weakly imposed conditions this is a small value of around 10^{-10}, which is above zero as expected.

Future work on this topic will involve a more rigorous study of turbulent pipe flow. We will aim to examine the effects of applying these weak boundary conditions when the mesh is under-resolved. In particular, since this case is a well-known benchmark in the fluids community, we will examine known quantities such as the boundary layer velocity profiles, distribution of wall shear stresses and other statistical quantities in order to properly quantify the effects of under-resolution of the mesh in the presence of weak boundary conditions. We note that these efforts align with the work being performed in WP3 under the jet-in-crossflow test case, and so the results of this investigation may guide the use of these boundary conditions for other WP3 cases.

4.4 Expected performance of the CG-DG scheme

Having derived the formulation of the local problems, we now move on to considering the expected performance of the CG-HDG scheme by analysing the likely cost of operator evaluations of a Poisson problem for an idealised mesh. We assume that the discrete Poisson problem is solved in two stages, both of which will significantly contribute to the overall CPU time spent in the solver. The stages are:

1. **Assembly and solution of a statically condensed system.** This step involves processing unknowns on entity boundaries, where 'entity' would be each single element in the context of continuous and hybrid discontinuous Galerkin methods and one mesh

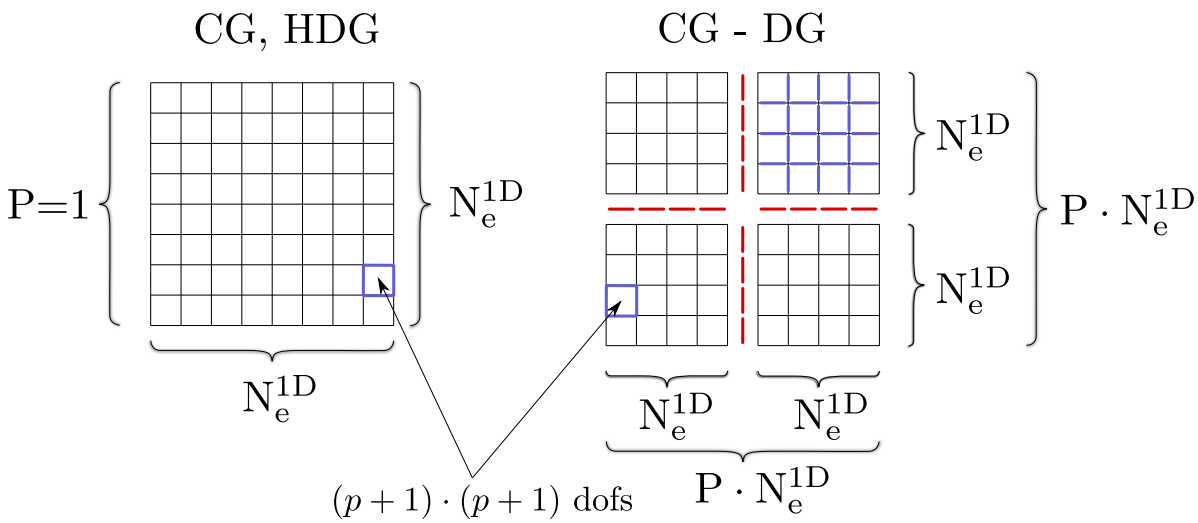


Figure 20: Idealized mesh divided into $P \times P$ patches, each patch containing $N_e^{1D} \times N_e^{1D}$ elements of order p.

patch (a group of elements spanned by a continuous polynomial basis) in the combined continuous-discontinuous Galerkin method.

The main difference between CG and HDG is that in the continuous case, trace variables are identical to variables located on element boundaries and are shared by neighbouring elements. The HDG method, on the other hand, introduces an additional hybrid variable, thus requiring more memory storage. This variable is not globally continuous, hence degrees of freedom on face boundaries are duplicated. As a consequence, the HDG interior solve on each element has to process a slightly larger local system.

2. **Interior solve.** Given solution on entity boundary, the solution in entity interior is reconstructed during this stage. Interior solve involves the inverse of a potentially large matrix.

Since we assume that the Poisson system corresponds to that being solved in, for example, the incompressible Navier-Stokes equations, setup costs (for example precomputing and storing the matrix inverses needed in interior solve above) are not taken into account, since matrices can be stored for use in multiple timesteps.

4.4.1 Domain Description

We first need to define a grid on which CG, HDG and CG-HDG can be examined. We assume a structured grid divided into $P \times P$ patches, each patch consisting of $N_e^{1D} \times N_e^{1D}$ elements (figure 20). Each element has a polynomial basis of degree p, i.e. $(p+1) \times (p+1)$ degrees of freedom. These may or may not be shared with neighbouring elements, depending on the setup (CG vs. DG vs. HDG) and global continuity of the polynomial bases. The number of

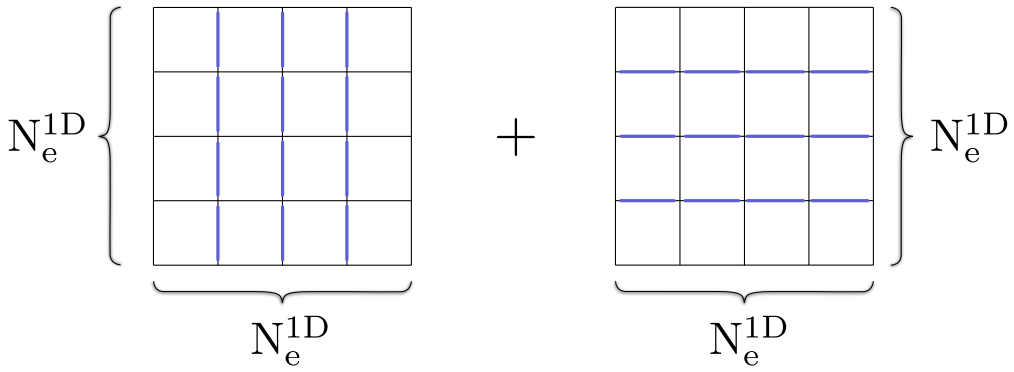


Figure 21: Interior edges (blue) within a patch.

inter-patch edges (red edges in figure 20) is

$$N_{interpatch}^{edges} = 2P \cdot (P-1) \cdot N_e^{1D},$$

and each patch contains N_{patch}^{edges} interior edges, with

$$N_{patch}^{edges} = 2N_e^{1D} \cdot (N_e^{1D} - 1),$$

as shown in figure 21. In the following sections, we analyse the cost of each of the solver stages.

4.4.2 Stage I: Solution of Statically Condensed System

Continuous Galerkin. Since the total number of elements along each side of the mesh is $P \cdot N_e^{1D}$ in 2D, the total number of unknowns before static condensation (assuming Dirichlet boundary condition everywhere) is

$$N_{CG}^{dof} = \left(P \cdot N_e^{1D} \cdot p - 1\right)^2.$$

This is also the rank of global system matrix. In case of one-level static condensation, the global system has the form

$$\begin{bmatrix} \mathbf{M}_b & \mathbf{M}_c \\ \mathbf{M}_c^T & \mathbf{M}_i \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_b \\ \boldsymbol{x}_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_b \\ \boldsymbol{f}_i \end{bmatrix}$$

and the rank of $\mathbf{M}_b$ is *approximately* (counting the boundary modes on the skeleton of the mesh) equal to

$$\begin{aligned} N_{CG}^{\lambda} = (N_{interpatch}^{edges} + P \cdot N_{patch}^{edges}) \cdot p &= (2P \cdot (P-1) \cdot N_e^{1D} + P \cdot 2N_e^{1D} \cdot (N_e^{1D} - 1)) \cdot p \\ &= 2PN_e^{1D}(P + N_e^{1D} - 2)p. \end{aligned}$$

The remaining values of u in element-interior degrees of freedom can be obtained by inverting $(P \cdot N_e^{1\mathrm{D}})^2$ local matrices of rank $p-1$. This means that the total cost of solving the CG problem is

$$C_{CG} = \mathcal{O}\big(cgsolve(PN_e^{1\mathrm{D}}(P+N_e^{1\mathrm{D}})p)\big) + (PN_e^{1\mathrm{D}})^2 \cdot \mathcal{O}\big((p-1)^3\big),$$

where $cgsolve(n)$ is the cost function of solving a sparse system of rank n with conjugate gradients. The cost of the second term is small if the blocks of $\mathbf{M}_i$ are inverted and stored during setup phase. The second term in the estimate assumes that the inverse of each diagonal block of $\mathbf{M}_i$ costs as much as Gauss elimination/LU decomposition of a matrix of rank $p-1$, which has cubic time complexity.

HDG. The discrete transmission condition (35) generates a sparse system of rank

$$\begin{aligned} N_{HDG}^{\lambda} = (N_{interpatch}^{edges} + P \cdot N_{patch}^{edges}) \cdot (p+1) &= (2P(P-1)N_e^{1\mathrm{D}} + P \cdot 2N_e^{1\mathrm{D}}(N_e^{1\mathrm{D}}-1)) \cdot (p+1) \\ &= 2PN_e^{1\mathrm{D}}(P+N_e^{1\mathrm{D}}-2)(p+1). \end{aligned}$$

In addition, we need to invert $(PN_e^{1\mathrm{D}})^2$ local systems lying in $\mathbb{R}^{(p+1)\times(p+1)}$ as in the CG case. The backsolve is more expensive however, because we have d mixed variables $q_1, \dots q_d$ in d dimensions. The element local inversion can be again precomputed and stored during setup.

The overall cost of solving for all unknowns therefore scales as

$$C_{HDG} = \mathcal{O}\big(cgsolve(PN_e^{1\mathrm{D}}(P+N_e^{1\mathrm{D}})(p+1))\big) + (PN_e^{1\mathrm{D}})^2 \cdot \mathcal{O}\big((p+1)^3\big).$$

Combined CG-DG Solver. The number of hybrid degrees of freedom on interfaces between patches is

$$N_{CG-DG}^{\lambda} = N_{interpatch}^{edges}(p+1) = 2P(P-1)N_e^{1\mathrm{D}}(p+1).$$

Each patch contains *approximately* $(N_e^{1\mathrm{D}}p)^2$ interior degrees of freedom, hence the total cost is

$$C_{CG-DG} = \mathcal{O}\big(cgsolve(P^2N_e^{1\mathrm{D}}(p+1))\big) + P^2 \cdot \mathcal{O}\big((N_e^{1\mathrm{D}}p)^3\big).$$

In the limiting case where each patch coincides with one single element (i.e. $P := N_e^{1\mathrm{D}}$ and $N_e^{1\mathrm{D}} = 1$), the three estimates C_{CG}, C_{HDG} and C_{CG-DG} predict the same asymptotic cost.

4.4.3 Cost of Solving the Statically Condensed System

Standard HDG algorithm. The cost of linear solve in the PCG (preconditioned conjugate gradient) solver will mainly depend on the cost of evaluating matrix-vector multiplications. For a matrix of rank n, this cost is $\mathcal{O}(n^2)$. Nektar++ solves the statically condensed system in matrix-free manner by performing the above matrix-vector multiplications *element-wise* and then summing them together. Suppose the (structured) mesh consists of quadrilaterals in 2D and hexahedra in 3D. Furthermore, we will assume that the triangular mesh is obtained by splitting each quadrilateral into 2 triangles and tetrahedral mesh is created by dividing each hexahedron into 6 tetrahedra.

The number of trace degrees of freedom of one element is

- $3 \cdot (p+1)$ for **triangles**
- $4 \cdot (p+1)$ for **quadrilaterals**
- $4 \cdot \frac{(p+1)(p+2)}{2}$ for **tetrahedra**
- $6 \cdot (p+1)^2$ for **hexahedra**

Under this assumption, *one matrix-vector multiplication* for the whole system (but performed on element-wise basis) will take

- $\mathcal{O}\big(2(N_e^{1\mathrm{D}})^2\big[3 \cdot (p+1)\big]^2\big) = \mathcal{O}\big(18(N_e^{1\mathrm{D}})^2(p+1)^2\big)$ operations on **triangles** in 2D
- $\mathcal{O}\big((N_e^{1\mathrm{D}})^2\big[4 \cdot (p+1)\big]^2\big) = \mathcal{O}\big(16(N_e^{1\mathrm{D}})^2(p+1)^2\big)$ operations on **quadrilaterals** in 2D
- $\mathcal{O}\big(6(N_e^{1\mathrm{D}})^3\big[2{\cdot}(p{+}1)(p{+}2)\big]^2\big) = \mathcal{O}\big(24(N_e^{1\mathrm{D}})^3(p{+}1)^2(p{+}2)^2\big)$ operations on **tetrahedra** in 3D
- $\mathcal{O}\big((N_e^{1\mathrm{D}})^3\big[6 \cdot (p+1)^2\big]^2\big) = \mathcal{O}\big(36(N_e^{1\mathrm{D}})^3(p+1)^4\big)$ operations on **hexahedra** in 3D

HDG applied to patches. Now suppose that the trace system is built between patches and each patch has $N_e^{1\mathrm{D}} \times N_e^{1\mathrm{D}}$ quadrilaterals in 2D and $N_e^{1\mathrm{D}} \times N_e^{1\mathrm{D}} \times N_e^{1\mathrm{D}}$ hexahedra in 3D. The number of unknowns on the trace of one patch now becomes

- $4 \cdot N_e^{1\mathrm{D}} \cdot p$ in 2D (**triangles** and **quadrilaterals**) and
- $6 \cdot (N_e^{1\mathrm{D}})^2 \cdot p^2$ in 3D (**tetrahedra** and **hexahedra**),

which will require

- $\mathcal{O}\big(16(N_e^{1\mathrm{D}})^2 \cdot p^2\big)$ operations per matrix-vector multiplication in 2D and
- $\mathcal{O}\big(36(N_e^{1\mathrm{D}})^4 \cdot p^4\big)$ operations in 3D

This means that the PCG algorithm in CG-DG case scales **one order worse** when measured in terms of number of elements along patch face $\big(\mathcal{O}\big((N_e^{1\mathrm{D}})^4\big)\big)$ than the standard HDG algorithm $\big(\mathcal{O}\big((N_e^{1\mathrm{D}})^3\big)\big)$. This can be seen in figure 22, where the asymptotic quantities are visualised for the CG and mixed CG-DG method.

Remark 2. *Note that the number on the surface of the patch is the same for triangles and quadrilaterals and for tetrahedra and hexahedra, respectively. For a continuous expansion, the number of DOFs on one quadrilateral face of a hexahedron is* $(p+1)^2$*, and* $2 \cdot \frac{(p+1)(p+2)}{2} - (p+1) = (p+1)^2$ *for two triangles covering the same quadrilateral face.*

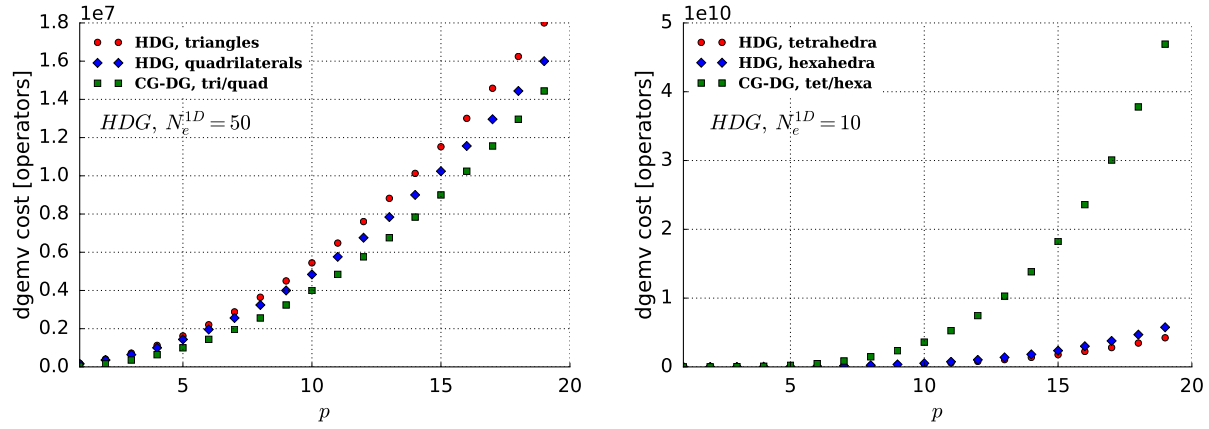


Figure 22: Asymptotic of matrix-vector multiplication measured by operation counts for HDG and combined CG-DG methods.

4.4.4 Stage II: Interior Solve

The reconstruction of interior degrees of freedom involves the solution of a linear system with the matrix

$$\mathbf{A}^e = \begin{pmatrix} \sum_{l=1}^{N_b^e} \tau^{(e,l)} \mathbf{E}_l^e & -\mathbf{D}_1^e & -\mathbf{D}_2^e \\ \left(\mathbf{D}_1^e\right)^T & \mathbf{M}^e & \mathbf{0} \\ \left(\mathbf{D}_2^e\right)^T & \mathbf{0} & \mathbf{M}^e \end{pmatrix}$$

The superscript ^{e} no longer refers to a single element as was the case of HDG. For CG-DG method, all the blocks in $\mathbf{A}$ are a result of a continuous Galerkin discretization in the whole partition/patch. The sparse matrix $\mathbf{A}^e$ is potentially large. Moreover however, presently there does not appear to be a way to exploit the elemental decomposition of the patch in inverting this matrix. We therefore expect that the explicit inverse of the system will be dense, thereby requiring significant storage and very large initial setup costs.

4.5 Summary and outlook

In this section we have described our efforts in developing a mixed continuous-discontinuous method in the context of solving simple linear elliptic problems. The derivation of weakly-imposed Dirichlet boundary conditions has been extensively documented, and the convergence properties have been examined and shown to be identical to those obtained with a strongly-imposed condition. Moreover, the examination of existing methods for CG demonstrates that these schemes are more practical in the sense that they are nearly parameter free. Moreover, we envision that the use of these conditions will significantly aide in the WP3 test cases, particularly in considering the investigation of a Formula 1 configuration in the region of the contact patch.

The results of our initial performance analysis indicate that, in its current form, the CG-DG setting may not be a computationally tractable formulation when applied to elliptic problems. Although our initial examinations in two dimensions proved promising, in that the operator costs are comparable to either CG or HDG, in three dimensions we have shown that the operator count and memory requirements increase significantly when compared to the standard CG or HDG methods. We note that although the communications costs will be significantly reduced in this setting, the increased operator counts will presently yield too much additional work on each node to make the method computationally efficient, when compared to CG or HDG.

However in the context of exascale hardware, in which FLOPS and memory bandwidth are expected to significantly increase and be readily available, we feel that a more detailed performance analysis is needed. We plan to conduct a further study where we impact the effects of communication versus hardware with different FLOPS and memory bandwidth characteristics. This will guide whether the further development of the algorithm will be useful at this stage. If these studies demonstrate that the CG-HDG formulation is unlikely to yield performance benefits, we have identified a number of areas that align with both the goals of the project and the objective of increasing strong scaling in this task:

- Coarse space preconditioners for the pressure Poisson equation, which we note remains as one of the original aims of this task (task 1.3.3). This is presently a barrier to strong scaling. We intend to investigate techniques being developed as part of task 1.1 (AMR), external sparse multigrid solvers such as the Hypre library, as well as other preconditioners based on reduced-order modelling presently being developed by EPFL.

- In combination with task 1.1, we have developed strategies for p-adaptive AMR simulations, and are developing coarse- and fine-scale preconditioners to accompany this work. We will therefore look to evaluate these alongside the h-adaptive methods, with a specific focus on load-balancing strategies to improve strong scalability of these simulations. A summary of our work on this topic thus far can be found in the recently accepted conference proceeding of reference [48].

- We will evaluate the use of the ExaGS library currently being developed as part of WP2 in order to reduce the communication overhead that can be found towards the strong scaling limit.

The improvements that these techniques yield will then be assessed by examining the industrial McLaren and ASCS test cases outlined in D3.2 as part of WP3.

5 Data reduction

5.1 Introduction

The steady increase of available computer resources has enabled engineers and scientists to use progressively more complex models to simulate a myriad of fluid flow problems. Yet, whereas modern high performance computers (HPC) have seen a steady growth in computing power, the same trend has not been mirrored by a significant gain in data transfer rates. Current systems are capable of producing and processing high amounts of data quickly, while the overall performance is oftentimes hampered by how fast a system can transfer and store the computed data. Considering that CFD researchers invariably seek to study simulations with increasingly higher temporal resolution on fine grained computational grids, the imminent move to exascale performance will consequently only exacerbate this problem [55]. The aim of this task is to minimize the impact of the I/O bottleneck on the overall computing performance by reducing the amount of data transferred from memory to disk. For this, the 'raw' data produced by a flow field solver will be transformed to spectral space and compressed by means of quantization and entropy encoding.

Figure 23: Stream-wise velocity field from a numerical simulation of a plate flow at $Re_{\theta,0} = 300$.

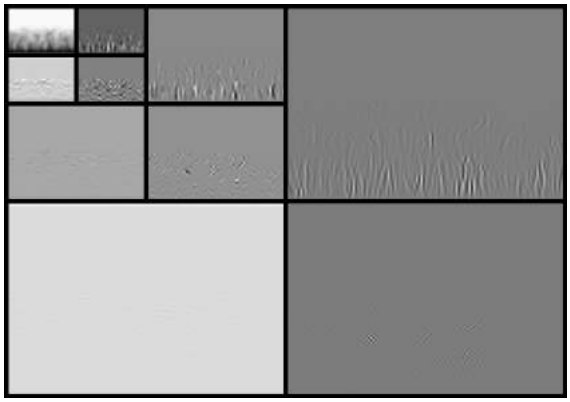


Figure 24: Dyadic decomposition into subbands for the streamwise velocity field of a plate flow.

One way to alleviate the I/O bottleneck would be to reduce the number of time steps which are written to the file system. While this trivial data reduction method may be tolerable for simulations that reach a steady state solution after a minuscule amount of time, the same approach would be fatal for highly transient physical phenomena. Considering that most fluid flow problems are subject to diffusion, however, we can conclude that our numerical datasets will typically be smooth and continuous, resulting in a frequency spectrum that is dominated by lower modes (see Figure 23 and 24) [55]. Thus our best way forward should be to use the otherwise wasted compute cycles by exploiting these inherent statistical redundancies to create a more compact form of the information content. Since effective data storage is a pervasive problem in information technology, much effort has already been spent

on developing newer and better compression algorithms. Most of the prevelent compression techniques, however, are dictionary encoders (i.e. Lempel-Ziv encoding) that merely act upon the statistical redundancies in the underlying binary data structure, unable to exploit the spacial redundancies present in numerical datasets. Furthermore, these lossless compression schemes are limited to a size reduction of only 10-20%, not allowing for a more efficient compression by neglecting parts of the original data that contribute little to the overall information content [42].

A prominent compression standard that allows for both lossy and lossless compression in one code-stream, however, can be found in the world of entertainment technology. JPEG-2000 is an image compression standard which is typically utilized to store natural and computer generated images of any bit depth and color space (i.e. 16-bit grey scale images) [16]. Unlike the original JPEG standard, which is based on the discrete cosine transform (DCT), the JPEG-2000 codec takes advantage of the the discrete wavelet transform (DWT) to reduce spacial redundancies during its compression stage [1]. While Fourier-based transforms are simple and efficient in exploiting the low frequency nature of most numerical datasets, their major disadvantage lies in the non-locality of their basis functions. Thus, if a DCT coefficient is quantized, the effect of a lossy compression stage will be felt throughout the entire flow field [13]. Discrete wavelet transforms, on the other hand, allow for the definition of Regions of Interest (ROI) which are to be coded and transmitted with better quality and less distortion than the rest of the flow field (see Figure 25)[1]. Furthermore, the dyadic decomposition into subbands (see Figure 24) and JPEG-2000s entropy encoder (Embedded Block Coding with Optimized Truncation) allows for the numercal dataset to be transmitted with increasing sample accuracy or spatial resolution[16]. Finally, JPEG-2000s volumetric extension (JP3D) translates these same capabilities to multi-dimensional datasets by applying the one-dimensional discrete wavelet transform along the axis of each subsequent dimension (see Figure 26) [6].

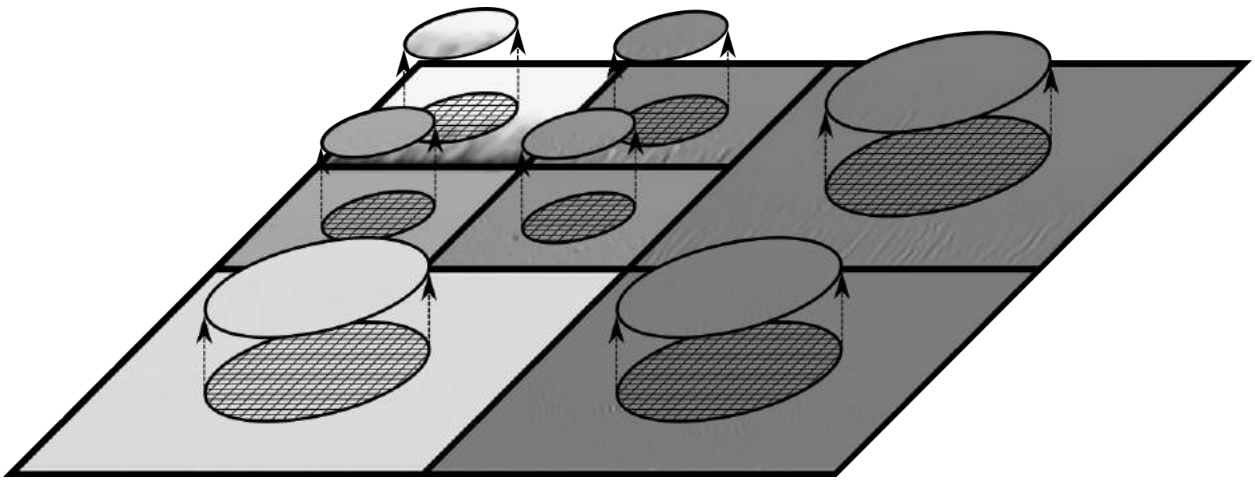


Figure 25: ROI mask generation in the wavelet domain.

In this report we seek to provide an overview of the fundamental building blocks of the

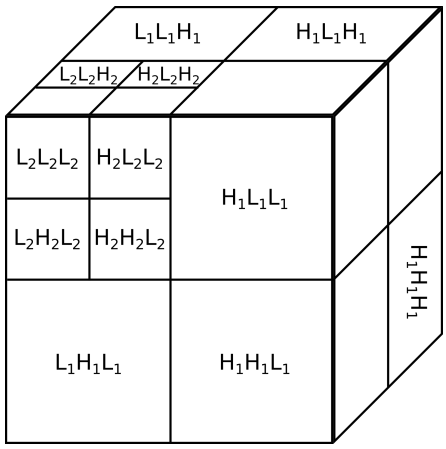


Figure 26: 3-Dimensional dyadic decomposition into subbands[6].

JPEG-2000 Part 1 codec and its volumetric extension JP3D (section 5.2). Our focus will lie on the transformation and quantization stage, since most of our efforts to adapt the JPEG-2000 algorithm to floating point numbers have been spent on this part of the codec. The aforementioned extensions, which expand JP3D's capabilities to IEEE-754 floating point datasets, will be explored in section 5.2.3. Preliminary compression results for a numerical simulation of a plate flow will be shown. Finally, we conclude with some closing remarks in section 5.5.

In a second part, we will evaluate existing methods, like proper-orthogonal decomposition (POD), singular value decomposition (SVD), dynamic-mode decomposition (DMD) and emerging new ideas for the present task. Special attention will be paid to algorithms that identify, extract and preserve physical structures in the flow field, like vortices and shear layers, for instance. This alone has the potential of reducing the initial raw data by more than an order of magnitude. SVD and DMD have already been surveyed in the last months. We will hence give a summary of our results in this fields in sections 5.3 and 5.4.

5.2 JPEG-2000

The JPEG-2000 standard is divided into 14 parts, with Part 1 defining the core coding system of the compression standard. Each subsequent Part, such as the volumetric extension JP3D, translates the capabilities of the baseline codec to different fields of application [16]. Since we are, for the most part, only interested in compressing three-dimensional numerical datasets, we will limit our discussion to the baseline codec and its volumetric extension.

As depicted in Fig. 27, the first step in our compression stage is to generate a time-frequency representation of the original data samples, which enables relatively simple quantization and coding operations. It is worth noting that, given an invertible Transform T, this step will not introduce any distortion in the decompressed dataset[16]. In the second step the transformed samples are represented using a sequence of quantization indices. This mapping operation introduces distortion in our decompressed data, since the set of possible outcomes

for each quantization index is generally smaller than for the transformed samples[1]. Finally, the quantization indices are losslessly entropy coded to form the final bit-stream [52]. In the following, the transform (sect. 5.2.1) and quantization (sect. 5.2.2) components are discussed in more detail to allow for a more thorough discussion of the IEEE-754 extensions in section 5.2.3.

5.2.1 Wavelet Transform

The transform is responsible for massaging the data samples into a more amenable representation for compression. It should capture the statistical dependencies among the original samples and separate relevant from irrelevant information for an optimal quantization stage. As opposed to the baseline JPEG codec, JPEG-2000 employs a lifting-based discrete wavelet transform, which allows for the full frame decorrelation of large scale numerical datasets. This eliminates blocking artifacts at high compression ratios, commonly associated with the JPEG standard. Furthermore, the dyadic decomposition into multi-resolution subbands enables the compression standard to assemble a resolution scalable codestream and the definiton of Regions of Interests (ROI). Finally, the use of integer DWT filters allows for both lossy and lossless compression in a single code-stream [52].

The lifting-based one-dimensional discrete wavelet transform is best understood as a pair of low- and high-pass filters, commonly known as the analysis filter-bank. Successive application of the analysis filter pair is followed by a down-sampling operation by a factor of two, discarding odd indexed samples. The analysis filter-bank is designed in such a manner that perfect reconstruction, barring any quantization error, is still possible after the downsampling step. The low pass filter attenuates high frequency components in a one-dimensional signal, resulting in a blurred version of the original dataset. This low-pass output is typically highly correlated and thus can be subjected to further stages of the analysis filter-bank. The high-pass filter, on the other hand, preservers the high frequency components, which usually results in a sufficiently decorrelated high-pass signal. Consequently, most DWT decompositions only further decompose the low-pass output, to produce what is known as a dyadic decomposition [52].

The filtered samples, which are output from the transform operation, are referred to as wavelet coefficients. To ensure the efficiency of the compression algorithm, these coefficients are critically sampled by virtue of the downsampling operation. This means, that the total

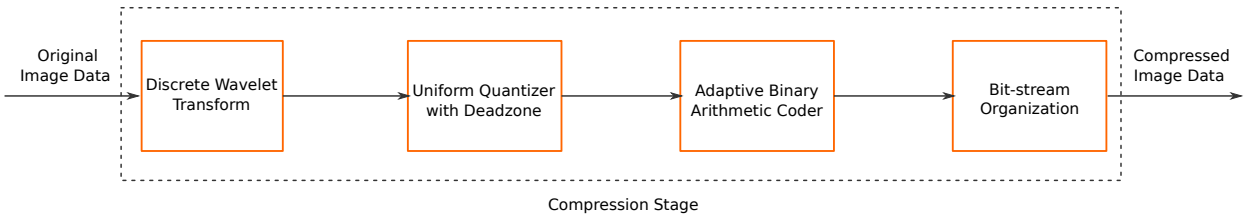


Figure 27: Fundamental building blocks of the JPEG-2000 compression stage[52].

number of wavelet coefficients needs to be equal to the number of original signal samples. Thus, when the DWT decomposition is applied to an odd-length signal, either the low- or high-pass sequence will have one additional sample. This choice is dictated by the position of the odd-length signal in relation to the global coordinate system[44].

Having described the general ideas behind the transform step, we now introduce the specific wavelet transformed described by the JPEG-2000 standard. The reversible transform option is implemented by means of the 5-tap/3-tap filter-bank described in equation (2.1). It is a nonlinear approximation of linear lifting steps which efficiently map integers to integers. The 5/3 filter allows for repetitive en- and decoding without information loss, barring any distortion that arrises due to the decompressed image values being clipped, should they fall outside their full dynamic range[16].

$$\begin{aligned} y(2n+1) &= x(2n+1) - \left\lfloor \frac{x(2n) + x(2n+2)}{2} \right\rfloor, \\ y(2n) \qquad &= x(2n) - \left\lfloor \frac{x(2n-1) + x(2n+1) + 2}{4} \right\rfloor. \end{aligned} \tag{48}$$

While the 5/3 bi-orthogonal filter-bank is a prime example for a reversible integer-to-integer transform, its energy compaction, due to its nonlinearity, usually falls short of most floating point filter-banks. The most prominent real-to-real transform is the irreversible 9-tap/7-tap filter-bank described in equation (2.2)[52].

$$\begin{aligned} y(2n+1) &\leftarrow x(2n+1) + (-1.586 \times |x(2n) + x(2n+2)|), \\ y(2n) \qquad &\leftarrow x(2n) + (-0.052 \times |y(2n-1) + y(2n+1)|), \\ y(2n+1) &\leftarrow y(2n+1) + (0.883 \times |y(2n) + y(2n+2)|), \\ y(2n) \qquad &\leftarrow y(2n) + (0.443 \times |y(2n-1) + y(2n+1)|), \\ y(2n+1) &\leftarrow -1.230 \times y(2n+1), \\ y(2n) \qquad &\leftarrow (1/1.230) \times y(2n), \end{aligned} \tag{49}$$

To ensure the perfect reconstruction property of the wavelet transform, the undefined samples outside of the finite-length signal segment need to be filled with values related to the samples inside the signal segment. When using odd-tap filters, the signal is symmetrically and periodically extended as shown in Figure 28[40].

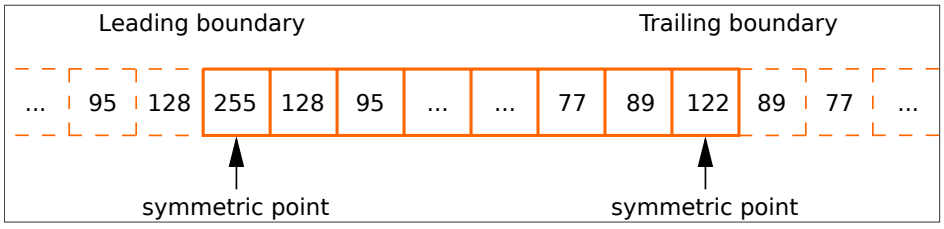


Figure 28: Symmetric extension at the leading and trailing boundaries of a signal segment.

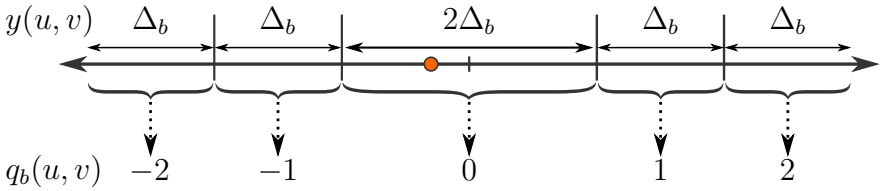


Figure 29: Uniform scalar quantizer with deadzone

5.2.2 Quantization

Unlike its predecessor, the JPEG-2000 algorithm employs a central deadzone quantizer (Fig. 29) to reduce the inherent entropy in the wavelet coefficients. This reduction in precision is lossy, unless the quantization stepsize is set to 1 and the subband samples are integers. Each of the wavelet coefficients y_b of the subband b is mapped to the quantization value q_b according to the formula [16]:

$$q_b = sign\left(y_b\left[\mathbf{n}\right]\right)\left\lfloor\frac{\left|y_b\left[\mathbf{n}\right]\right|}{\Delta_b}\right\rfloor. \tag{50}$$

the stepsize Δ_b is calculated as follows:

$$\begin{aligned}\Delta_b &= 2^{R_b-\epsilon_b}\left(1+\frac{\mu_b}{2^{26}}\right),\\ 0 \le \epsilon_b < 2^6, &\quad 0 \le \mu_b < 2^{26}.\end{aligned} \tag{51}$$

where ϵ_b is the exponent and μ_b is the mantissa of the corresponding stepsize and R_b represents the dynamic range of the subband b. This limits the largest possible stepsize to twice the dynamic range of the subband. In case of a reversible coding path, Δ_b is set to one by choosing $\mu_b = 0$ and $R_b = \epsilon_b$[16].

5.2.3 Extensions

One of the downsides of using image compression codecs is that they have been designed for integer datasets and do not allow for the lossy and lossless encoding of IEEE 754 double precision floating point numbers. The following section will give an overview of the proposed extension that have thus far been tested to alleviate this problem, with section 5.2.4 describing the fixed point number format Q and section 5.2.5 giving an overview over the Shape-Adaptive Discrete Wavelet Transform.

For testing the different algorithms we used the dataset from a numerical simulation of a flat plate flow at $Ma_\infty = 0.3$ and $Ma_\infty = 2.5$ (Fig. 30). The spatial resolution of the numerical grid was set to $n_x \times n_y \times n_z = 3300 \times 240 \times 512$ nodes in streamwise, wall-normal and spanwise directions respectively. The file size for one time step and the conservative variables $\rho, \rho u, \rho v, \rho w, E$ amounted to 16,220,192,238 bytes. Each file was divided into 1664 'tiles', with each tile being transformed and encoded separately. To evaluate the overall

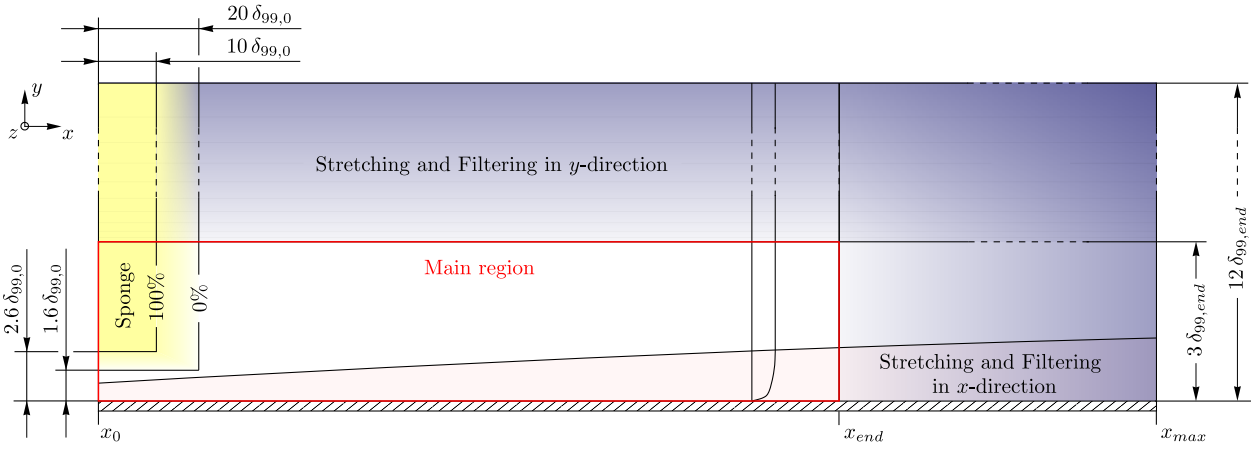


Figure 30: Numerical setup for the simulation of a flat plate flow at $Ma_\infty = 0.3$ and $Ma_\infty = 2.5$[65].

quality of the decompressed file we used the peak signal-to-noise ratio metric (PSNR), which is evaluated as

$$MSE = \frac{1}{ijk} \sum_{x=1}^{i} \sum_{y=1}^{j} \sum_{z=1}^{k} |I(x,y,z) - I'(x,y,z)|^2 \,, \qquad (52)$$
$$PSNR = 20 \cdot \log_{10}(\frac{1}{\sqrt{MSE}}).$$

where $I(x,y,z)$ is the original, $I'(x,y,z)$ the decompressed image and i,j,k the dimensions of the volumetric dataset. The PSNR is expressed in dB (decibels). Good reconstructed datasets typically have PSNR values of 30 dB or more. We ran our experiments on a single core of an Intel Core i7-6700 processor with 3.40GHz and 32 GB of 2133 MHz DDR4 RAM.

5.2.4 Fixed Point Number Format

Our first approach to handle floating point values was to use the fixed point number format Q, which maps the floating point values onto the dynamic range of a specified integer type (i.e. 64bit) [41]. To this end, the conservative variables $\rho, \rho u, \rho v, \rho w, E$ were first aligned with regards to their largest floating point exponent, which are stored uncompressed in the header of the codestream. We used a Q8.23 two's compliment format that allows numbers in the range $[-255, 255)$ to be represented. Although the normalized floating point values lie in a smaller range $(-1, +1)$, the additional dynamic range was added to prevent an overflow of the variables during the transform stage.

The fixed point number format provides a good first step on the way to efficiently store large three-dimensional datasets. Figure 31 shows an overview of the original (a) and compressed (b) direct numerical simulation (DNS) for a turbulent flat plate flow at $Ma_\infty = 0.3$, Figure 33 for a turbulent flat plate flow at $Ma_\infty = 2.5$. A closer look at the flow structures identified by the λ_2-criterion can be found in Figure 32 and Figure 34. The compression ratio for the simulation at $Ma_\infty = 0.3$ measured $17:1$ with a PSNR of 37.5, while the

compression ratio for the simulation at $Ma_\infty = 2.5$ measured $15:1$ with a PSNR of 42.1. The average compression time amounted to $936s$ with each tile taking up roughly $0.6s$ of processing time. In comparison, the compression ratio for the lossless LZMA Algorithm (7-Zip) measured only 1.2-1.3:1, with an average compression time of $1743s$.

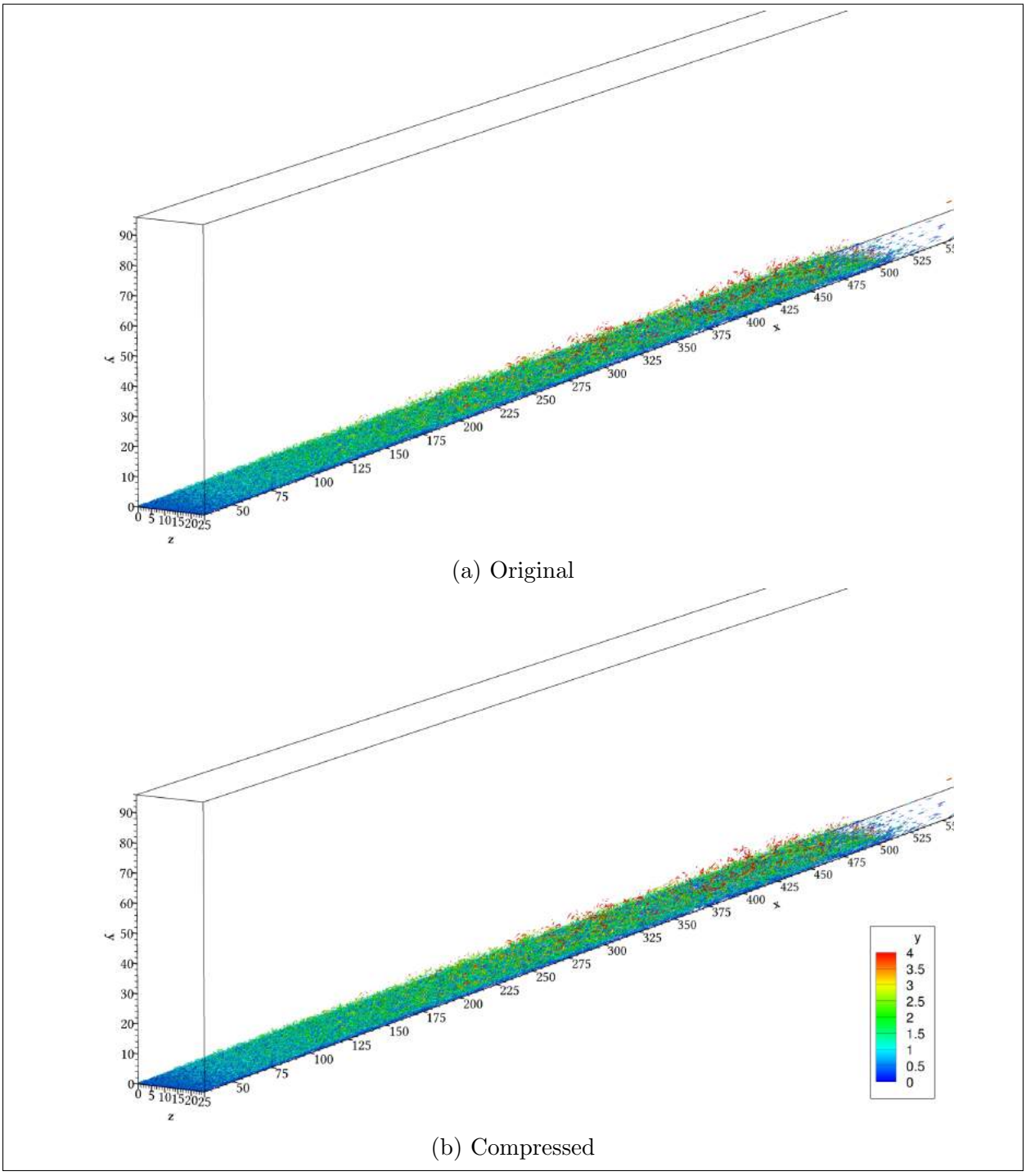
Overall the Q number format offers good compression ratios with reasonably well reconstruction of the numerical datasets. Due to the floating point arithmetic and the many-to-one mapping, however, information will be irreversibly lost during the preprocessing of the data samples and thus true lossless compression cannot be achieved. Furthermore, the entropy encoder is unable to take full advantage of its optimal truncation algorithm since a large part of the quantization (fixed point number transformation) falls outside of its purview. To circumvent this problem we planned to split the floating point datasets into its sign, bit and mantissa integer fields and compress them separately with a Shape-Adaptive Discrete Wavelet Transform.

5.2.5 Shape-Adaptive Discrete Wavelet Transform

In order to motivate the Shape-Adaptive Discrete Wavelet Transform (SA-DWT) extension, we begin this section by studying the evolution of the biased exponent and mantissa for a sequence of single precision floating point numbers $x_i = i/32$. Figure 35a shows the biased exponent and 35b the mantissa for the set $\{x_i : |i| < 128, i \in \mathbb{Z}\}$. While the biased exponent in Figure 35a exhibits discontinuities between two intervals of a constant exponent, with a large discontinuity at $x_0 = 0$, the biased exponent only varies slowly between values of 122 and 128. The result is a frequency spectrum that is dominated by lower modes, which in turn should allow the wavelet transform to decorrelate much of the signal. The mantissa, on the other hand, exhibits linear change within intervals of a constant exponent. Similar to the biased exponent, the wavelet transform should be able to decorrelate much of the signal were it not for the large discontinuities at the intervals between two different exponents. The IEEE-754 representation, however, is highly non-linear. The distance between consecutive floating point numbers depends on the value of their exponent, inevitably introducing high-frequency signals into the mantissa field. This in turn results in large coefficients for the detail subbands of the wavelet decomposition, thus degrading the overall compression efficiency [27].

This could be addressed by applying a SA-DWT inside the smooth regions of the biased exponent and mantissa field (Figure 36 and Figure 37). For the sign field, the wavelet transform can be applied everywhere since it features no discontinuities. The biased exponent field, however, is divided into regions where all sample values are either equal to or different from zero. Lastly, the mantissa field is split into regions that are characterized by having associated sign and exponent values which are constant. Each region is treated as its own independent image, which is symmetrically extended and downsampled accordingly. Furthermore, the zero regions of the exponent field are to be signaled in the codestream header in order to allow a JP3D reader to fully decode the sign, biased exponent and mantissa fields[27].

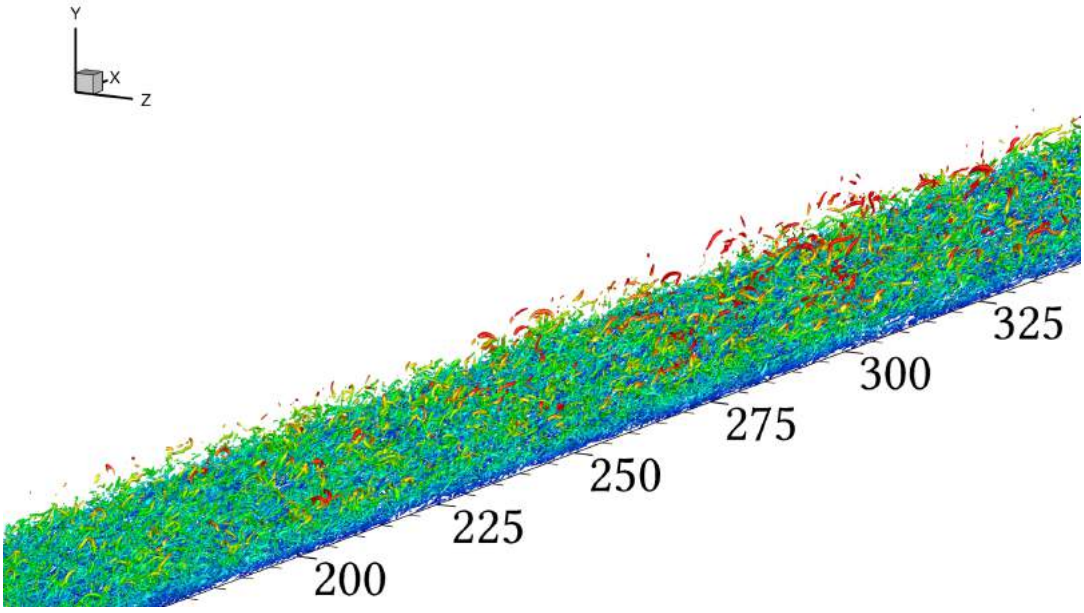
During our investigation, however, it transpired that this approach only works as long as



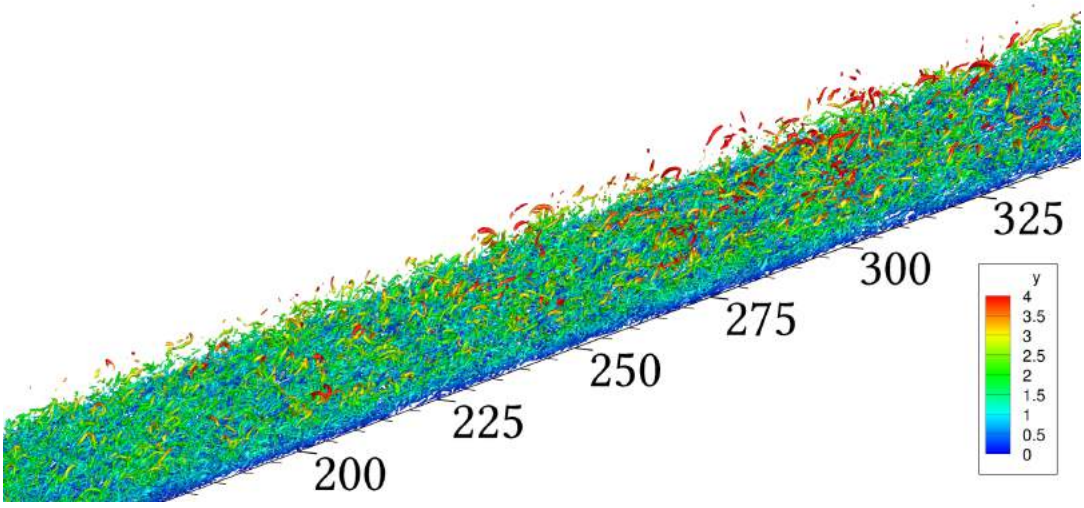
(a) Original

(b) Compressed

Figure 31: Original (a) and Compressed (b) DNS of a turbulent flat plate flow at $Ma_\infty = 0.3$. Flow structures identified by the λ_2-criterion for $\lambda_2 = -0.15$. Coloration of the isosurface according to the wall normal distance y[65].

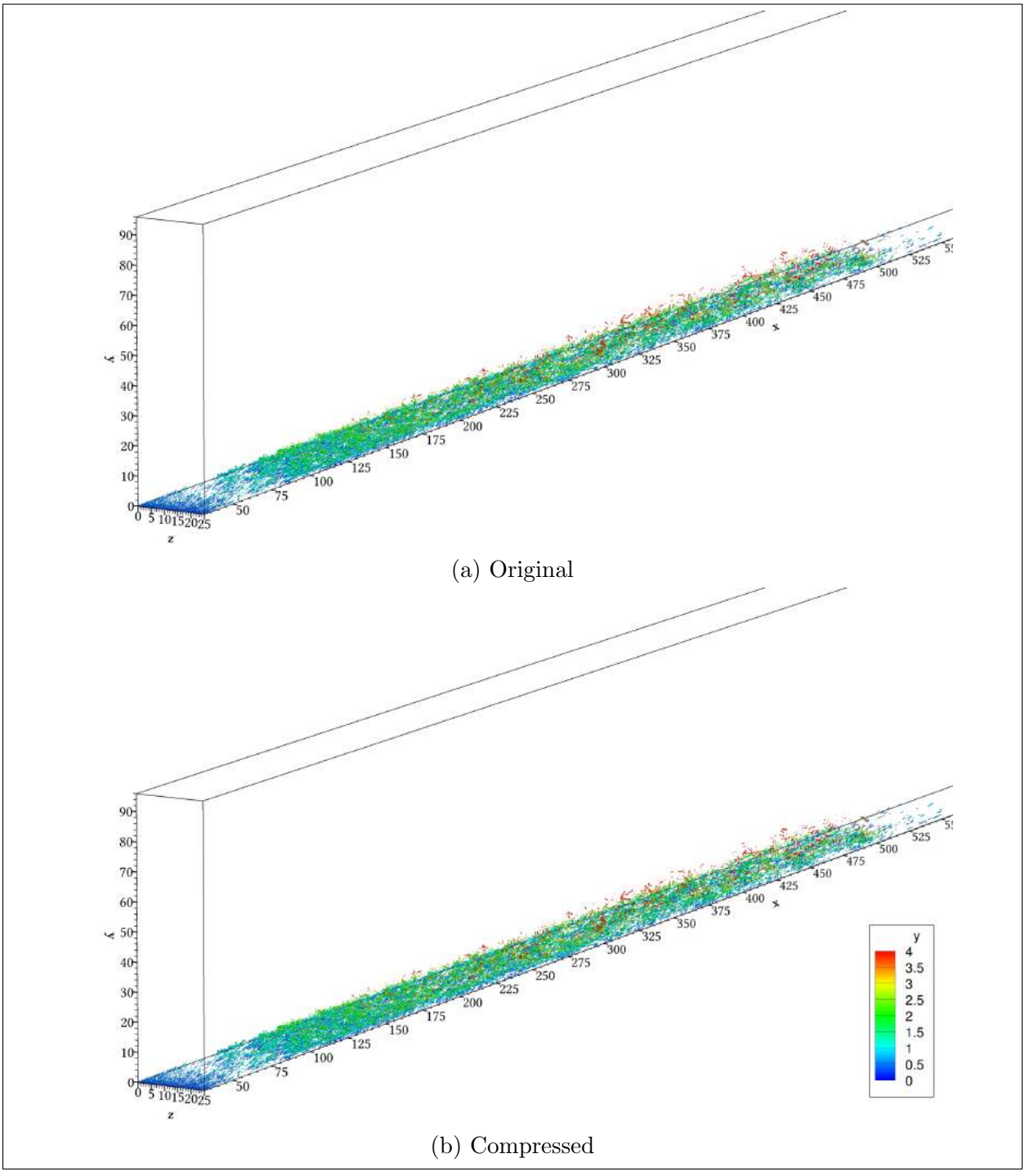


(a) Original



(b) Compressed

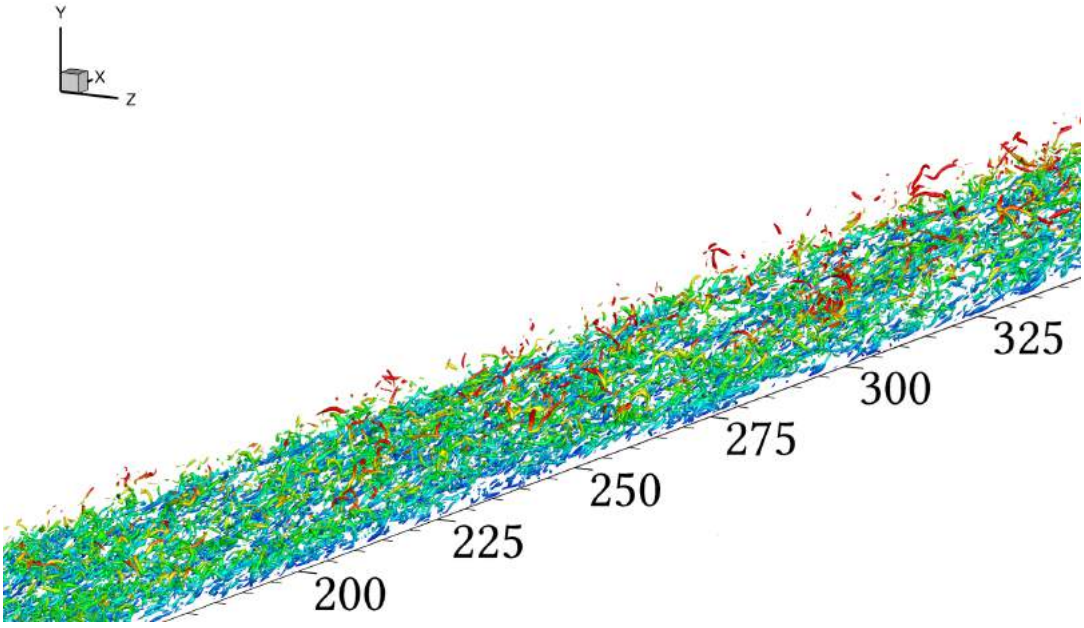
Figure 32: Close-up of the Original (a) and Compressed (b) DNS of a turbulent flat plate flow at $Ma_\infty = 0.3$. Flow structures identified by the λ_2-criterion for $\lambda_2 = -0.15$. Coloration of the isosurface according to the wall normal distance y[65].



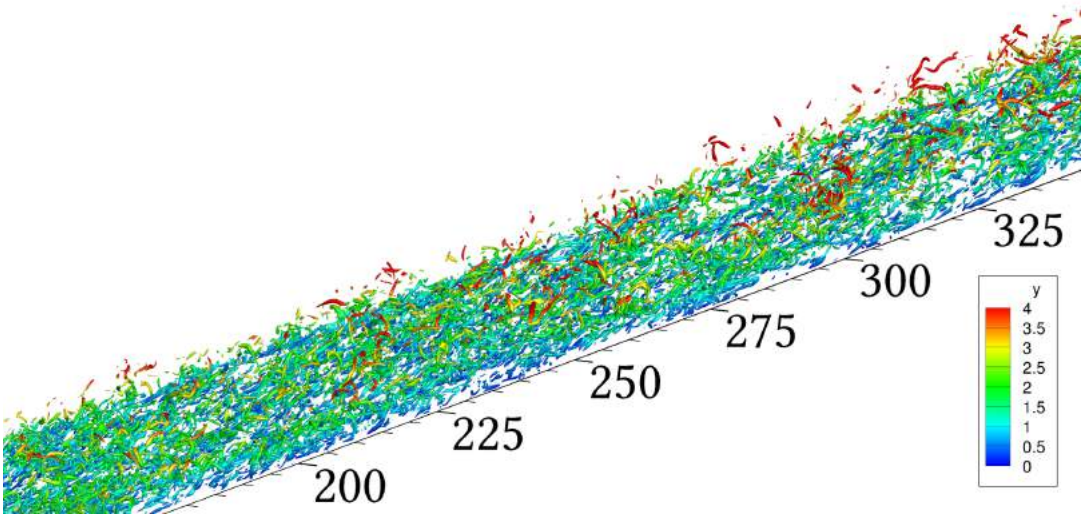
(a) Original

(b) Compressed

Figure 33: Original (a) and Compressed (b) DNS of a turbulent flat plate flow at $Ma_\infty = 2.5$. Flow structures identified by the λ_2-criterion for $\lambda_2 = -0.15$. Coloration of the isosurface according to the wall normal distance y[65].

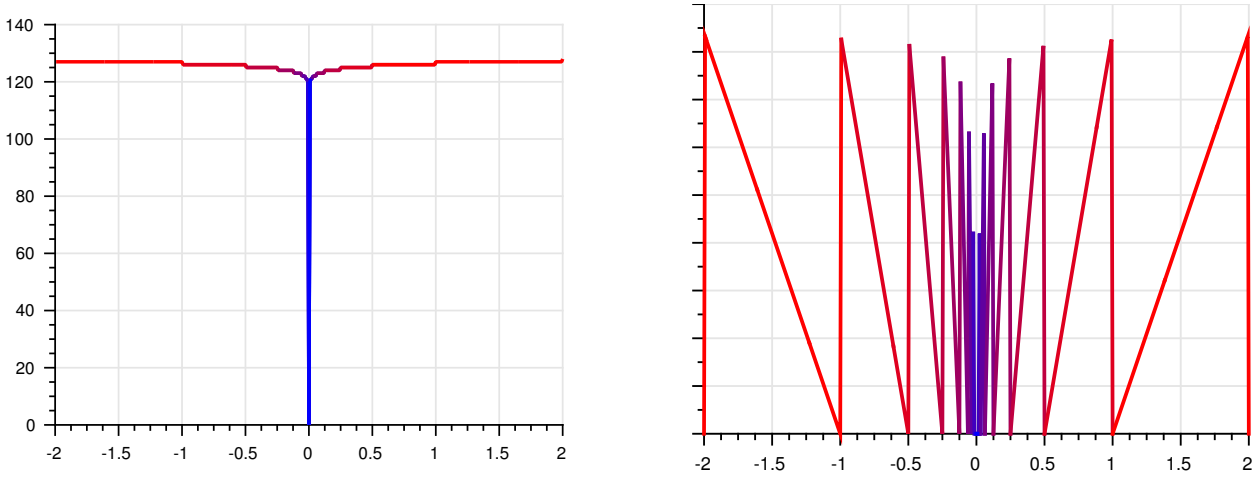


(a) Original



(b) Compressed

Figure 34: Close-up of the Original (a) and Compressed (b) DNS of a turbulent flat plate flow at $Ma_\infty = 2.5$. Flow structures identified by the λ_2-criterion for $\lambda_2 = -0.15$. Coloration of the isosurface according to the wall normal distance y[65].



(a) Biased Exponent

(b) Mantissa

Figure 35: Evolution of the biased exponent and mantissa fields for a uniform sequence of single precision IEEE 754 numbers

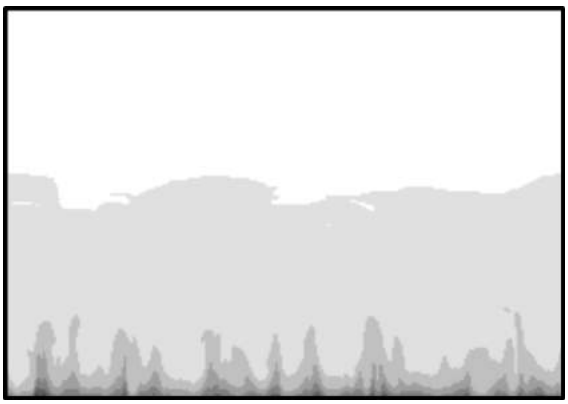


Figure 36: Exponent field of the streamwise velocity from a numerical simulation of a plate flow at $Re_{\theta,0} = 300$.

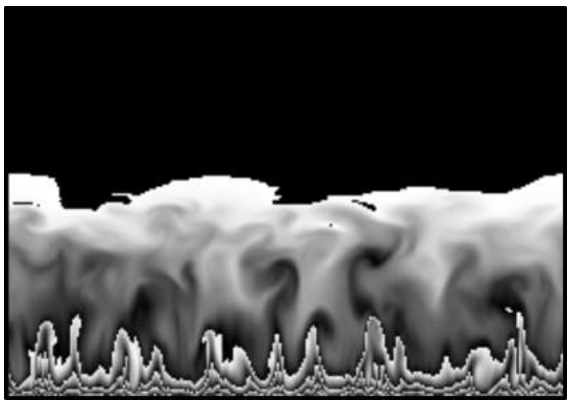


Figure 37: Mantissa field of the streamwise velocity from a numerical simulation of a plate flow at $Re_{\theta,0} = 300$.

in the smooth regions SA-DWT is applied to remain larger than one pixel and the signal segment starts at an even numbered position. If this is not the case the shape-adaptive wavelet transform will introduce a phase shift in the wavelet coefficients and subsequently distort the shape of the subband images. Since the compression algorithm applies the wavelet decomposition on multiple lower resolution approximations of the original dataset, the likelihood of this happening every time a compression is attempted is very high. This in turn will result in a faulty entropy encoding stage since the Embedded Code Block-Encoder expects the sub-bands to be rectangular.

5.3 SVD

We note that this section is an extended version of the details provided in D1.1.

5.3.1 Description of the method

SVD is one of the most useful tools in matrix algebra and includes the concept of the eigenvalue/eigenvector decomposition as prerequisite for data/dimension reduction. We start with the definition of SVD. The SVD is the expression of any $m \times n$ matrix A of rank r ($r \leq min(m, n)$) in the following form [22]

$$A = U\Sigma V^T, \Sigma = diag(\sigma_1, \sigma_2, \ldots, \sigma_r), \tag{53}$$

where the columns of U and V are orthonormal with $U^TU = I = V^TV$ as well as Σ is a diagonal matrix of positive numbers $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0$. An equivalent way of writing is:

$$A = \sum u_r \sigma_r v_r^T. \tag{54}$$

We can also find the fundamental theorem of SVD as shown in Figure 38 . The vectors u_r of an orthonormal U, called the left singular vectors, are the eigenvectors of AA^T with the associated eigenvalues σ_r. The vectors v_r of orthonormal V, called the right singular vectors, are the eigenvectors of A^TA with the same associated eigenvalues σ_r.

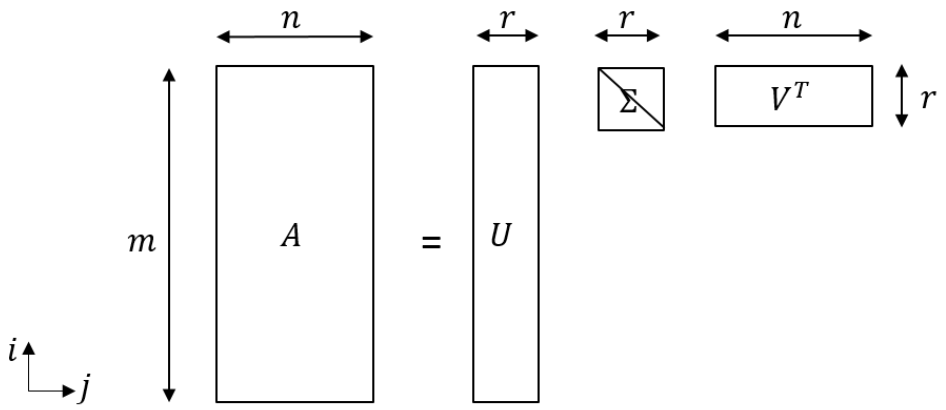


Figure 38: The form of SVD

Assume that we want to represent a very huge matrix A by its SVD components U, Σ and V and that these matrices are also too large to store. The best way to reduce the dimensionality of the three matrices is to set the smallest of the singular values to zero.

There are some excellent software packages available for obtaining the SVD in a numerically accurate manner. In particular, the LAPACK (Linear Algebra Package) library provides much of the functionality needed for dense matrices. A parallel version of the LAPACK functionality is available in the ScaLAPACK (Scalable Linear Algebra Package)

library, which is designed for message passing parallel computers and can be used on system that supports MPI. In the field of dimension reduction, the LAPACK and ScaLAPACK libraries provide high-performance implementations of several versions of SVD for *dense* matrices. Cray provides libraries for scientific computing in its `libsci` library, which includes LAPACK as well as ScaLAPACK and is loaded by default. The SVD has been tested with these implementations of LAPACK and ScaLAPACK on our Cray XC40 (Hazel Hen).

The SVD is strongly connected to the eigenvalues of symmetric matrices A^TA and AA^T, where

$$A^T = V\Sigma^T U^T. \tag{55}$$

Because Σ is a diagonal matrix, transposing it has no effect. Thus

$$A^TA = V\Sigma^2 V^T. \tag{56}$$

The formulation is shown in Figure 39. In this case only right singular vectors V and eigenvalues Σ remain to be computed. Since the rank of A is k, all other eigenvalues will be zero, so that the data could be reduced.

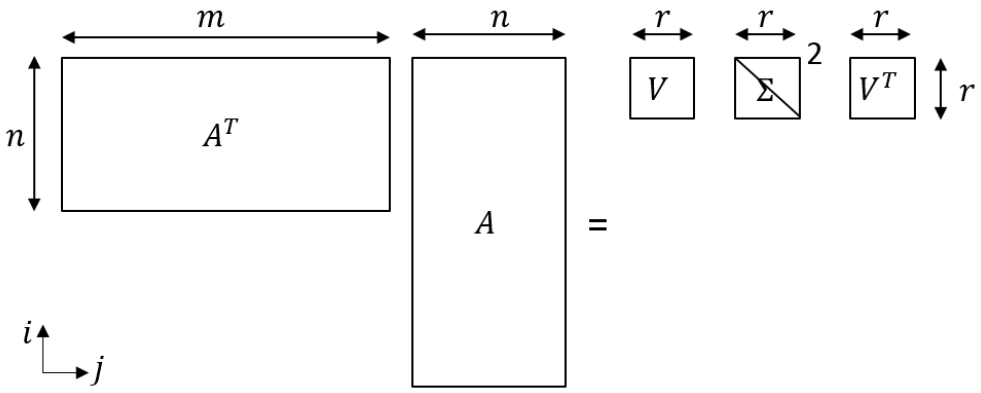


Figure 39: The SVD of matrices A^TA

An implementation of the SVD for large *sparse* matrices is available through ARPACK (Arnoldi Package), which has also been used to test the SVD algorithms on Hazel Hen.

In many CFD simulations, data structures commonly have more than two dimensions, and are usually represented by multidimensional tensors. The different "dimensions" of tensor could also be called modes. So far, the above described 2-mode method SVD is widespread and of great significance in development of mathematics and statistics. However, it does not take the multidimensionality of data into account. In the last years, the mathematical theory and several new algorithms have been fast developed for the multidimensional tensors. Here gives an introduction to one of such tensor methods, i.e. the higher-order singular value decomposition (HOSVD). For simplicity, we present HOSVD of a 3-mode tensors at first. The HOSVD of the $(m \times n \times p)$-tensor $\mathcal{A}$ is visualized in Figure 40 and can be written as

$$\mathcal{A} = \mathcal{S} \times_1 U^{(1)} \times_2 U^{(2)} \times_3 U^{(3)}, \tag{57}$$

where $\mathcal{S}$ is a smaller 3-mode $(r_1 \times r_2 \times r_3)$-tensor and usually referred to as the core tensor. It has the property of all-orthogonality: any two slices of $\mathcal{S}$ are orthogonal in the sense of the scalar product. r_1, r_2 and r_3 are the ranks in each direction i, j and k. $U^{(1)} \in \mathbb{R}^{(m \times r_1)}$, $U^{(2)} \in \mathbb{R}^{(m \times r_2)}$, and $U^{(3)} \in \mathbb{R}^{(m \times r_3)}$ are orthogonal matrices. And $\times_1$ is 1-mode product of tensor.

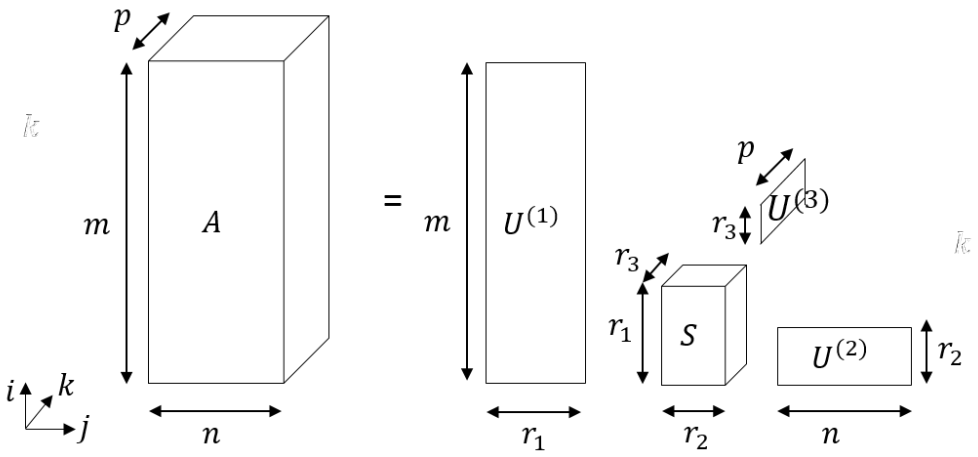


Figure 40: HOSVD expansion of the 3-mode tensor

Then we extend the HOSVD to the general multidimensional tensor. For every n-mode tensor $\mathcal{A}$ the HOSVD was proposed by Lathauwer et al. [39] as follows:

$$\mathcal{A} = \mathcal{S} \times_1 U^{(1)} \times_2 U^{(2)} \times_3 \cdots \times_n U^{(n)}, \tag{58}$$

in which $\mathcal{S}$ is a new n-mode tensor, $\times_n$ is n-mode product of tensor, and $U^{(n)}$ is a unitary matrix obtained by applying SVD to n-mode unfolding matrix.

5.3.2 Outlook

In the next step, other similar mechanisms will be studied and the data-reduction strategies will be implemented and tested in three-dimensional ExaFLOW's use cases.

5.4 Dynamic Mode Decomposition

We note that this section is an extended version of the details provided in D1.1.

The simulation of unsteady fluid flows is essential to predict expected and unexpected features of the systems to be analysed. It is not clear how to extract all the special features of the simulated flow in terms of (quasi-)periodicity or invariance or dominant modes. This applies in different ways to all unsteady processes in nature, technology and economy. Their are several algorithmical approaches for analysis as defining averages, extracting dominant frequences by Fourier Analysis of the signal, Principal Component Analysis (PCA), Proper Orthogonal Decomposition (POD) or Empirical Orthogonal Functions (EOF) in climatology

(see http://brunnur.vedur.is/pub/halldor/PICKUP/eof.pdf). The mathematical kernel of these is the same. Somes years ago a new approach has been given to analyze a large set of time dependent signals especially related to fluid flows by the Dynamic Mode Decomposition (DMD) method of Peter Schmid. This DMD turned out to decompose a time signal a linear combination of different modes multiplied by the k-the power of an complex value for getting the k-th time step. Williams, Kevrekides and Rowley [66] generalized the approach to Extended DMD and gave a relation to the Koopman operator [35], which again gives the opportunity to apply techniques of the functional analytic ergodic theory, see [21] and [7]. The Koopman operator is directly related to the the nonlinear equation of interest, here the Navier-Stokes equations. The Koopman operator acts on an infinite dimensional space of observables. As linear and bounded operator it has a spectrum and may have eigenfunctions in the space of observables, which can be interpreted in terms of (here) the fluid flow. Under some circumstances DMD might give approximations of some eigenvalues and eigenfunctions of the Koopman operator.

5.4.1 Analysis

Assume K a compact topological space and a continuous nonlinear operator

$$\varphi : K \longrightarrow K \tag{59}$$

In this context K is part of the discrete function space containing the discrete time steps of the iterations given by the discrete Navier-Stokes operator φ. Assume $\mathcal{F} \subset C(K)$ being a linear subspace of "**observables**" with the stability property

$$f \in \mathcal{F} \Rightarrow f \circ \varphi \in \mathcal{F} \tag{60}$$

Observables are e.g. the mean pressure of a fluid domain or the evaluation operators δ_x at all points $x \in \Omega$ for continuous functions defined on the domain Ω. The operator T_φ defined by

$$T_\varphi : \mathcal{F} \longrightarrow \mathcal{F} \tag{61}$$

$$f \mapsto T_\varphi f = f \circ \varphi \tag{62}$$

is named the **Koopman-Operator of φ on $\mathcal{F}$** (B.O. Koopman 1931) T_φ is linear and continuous and has a spectrum, but acts on an infinite dimensional space. It may have eigenvalues and eigenfuctions in $\mathcal{F}$ (not in K!). For two eigenvalues also their product is an eigenvalue if the product of both eigenfunctions is also element of $\mathcal{F}$ and is not disappearing.

The eigenfunctions fulfill **Schröders functional equation**

$$f(\varphi q) = \lambda f(q) \quad \forall\, q \in K$$

It might be a problem to interpret this equation in terms of a physical phenomenon. Interesting enough is the idea, that isosurfaces of $|\lambda|$ remain isosurfaces after application of the operator φ [7].

To make the operator T_φ manageable for numerical purposes, it is important to find a small space of observables $\mathcal{F}$. The smallest reasonable numerical setting is to investigate the finite sequence

$$G^f(q) = \left[g_k^f(q)\right]_{k=0,\cdots,n} = \left[f\left(\varphi^k q\right)\right]_{k=0,\cdots,n} \tag{63}$$

for a single observable f (this might be even a vector of observables) starting with an arbitrary state $q \in K$, a first finite part of a trajectory. Starting with $q^{'} = \varphi^j q$ is also a reasonable option enforcing the significance of a shifted sequence on the same trajectory. A finite number of linearly independent observables S can be combined as a vector of observables. Explicit knowlegde of the operator φ is not needed for numerical handling; to know the effect of the operator on the state space as measured by the observables is sufficient.

The **convolution product** of two polynom-coefficient vectors c with $\deg c = p$ and b with $\deg b = q$ is the given by the product polynom

$$c * b(\lambda) = c(\lambda)\, b(\lambda) \quad \forall\ \lambda \in \mathbb{C}. \tag{64}$$

The **convolution-matrix** $\mathfrak{A}_n(c)$ for a coefficient vector c of $\deg c = p$ is

$$\mathfrak{A}(c) = \mathfrak{A}_n(c) = \begin{array}{c} \\ 0 \\ 1 \\ . \\ . \\ p \\ . \\ n \end{array} \begin{array}{c} \begin{array}{ccc} 0 & & n-p \end{array} \\ \begin{pmatrix} c_0 & & \\ c_1 & \ddots & \\ . & & c_0 \\ . & & c_1 \\ c_p & & \vdots \\ & \ddots & \\ & & c_p \end{pmatrix} \end{array} \tag{65}$$

Let c be the coefficient-vector of a polynom with $\deg c = p$ ($c_p \neq 0$). The convolution matrix acts as the convolution product ($\deg b = n - p$)

$$\mathfrak{A}(c)\ b = c * b \tag{66}$$

Let G the matrix of series of measurements or a contiguous finite part of observables applied to the sequence of states in K. Let c a polynom coefficient vector with a degree $\deg c = p$ not larger than the sequence. We multiply G with the convolution matrix

$$G \quad \mathfrak{A}_n(c) \ = \ R \tag{67}$$

$$\begin{bmatrix} g_0 & g_1 & g_2 & \cdots & g_n \end{bmatrix} \ \mathfrak{A}_n(c) \ = \ \begin{bmatrix} r_0 & r_1 & \cdots & r_{n-p} \end{bmatrix} \tag{68}$$

That is the same as multiplying all possible $n - p$ contiguous fractions of G by c from the right and building a new matrix with $n - p$ vectors. We expect c selected in a way, that R

is small in some sense, best would be $R = 0$. To analyse this, we decompose G in two parts related to c

$$G = G_{modes} \; + \; \Delta G \tag{69}$$

The first part G_{modes} will be given by a linear decomposition in modes defined by the roots of c with the property

$$G_{modes} \; \mathfrak{A}(c) = 0 \tag{70}$$

It will described later. The second part defines the defect in equation (67) given by

$$\Delta G \; \mathfrak{A}(c) = R \tag{71}$$

We assume that $\operatorname{Im} \Delta G \subset \operatorname{Im} R$. There are some not unique but reasonable requirements in selecting ΔG. We restrict ΔG to be

$$\Delta G = R \; \left(\mathfrak{A}(c)^* \mathfrak{A}(c)\right)^{-1} \mathfrak{A}(c)^* \tag{72}$$

With these assumptions we get by (67)

$$\Delta G = G \; Q \tag{73}$$

for the selfadjungated projection Q ($Q^* = Q = Q \; Q$)

$$Q \;\; = \;\; \mathfrak{A}(c) \;\; \left(\mathfrak{A}(c)^* \mathfrak{A}(c)\right)^{-1} \;\; \mathfrak{A}(c)^* \tag{74}$$

with the property

$$(I - Q) \;\; \mathfrak{A}(c) \;\; = \;\; 0 \tag{75}$$

for G_{modes} we have by (75)

$$G_{modes} = G - \Delta G = G - GQ = G \, (I - Q) \tag{76}$$

and therefore

$$0 = G_{modes} \; \mathfrak{A}(c) = \begin{bmatrix} \tilde{g}_0 & \tilde{g}_1 & \dots & \tilde{g}_n \end{bmatrix} \; \mathfrak{A}(c) \tag{77}$$

We can show that this allows a decomposition in p Koopman modes v_l

$$G_{modes} = \begin{bmatrix} \tilde{g}_0 & \tilde{g}_1 & \dots & \tilde{g}_n \end{bmatrix} = \sum_{l=1}^{\mathbf{p}} v_l \left[1, \lambda_l, \lambda_l^2, \dots, \lambda_l^{\mathbf{n}}\right] \tag{78}$$

with the modes

$$v_l = G_{modes} \; \frac{1}{w_l \left(\lambda_l\right)} \; \begin{bmatrix} w_l \\ 0 \end{bmatrix} \tag{79}$$

given by the polynom

$$c\left(\lambda\right) = \left(\lambda - \lambda_l\right) w_l\left(\lambda\right) \quad \forall\ \lambda \in \mathbb{C} \quad \text{or} \quad c = w_l * \begin{bmatrix} -\lambda_l \\ 1 \end{bmatrix} \tag{80}$$

We quantify now the l_2-norm $\|\Delta G\|_2$ of the defect operator ΔG. Taking $\mu = \|\Delta G\|_2^2$ we have to analyse the operator inequality

$$\Delta G^* \Delta G \leq \mu\ I \tag{81}$$

or by (73)

$$Q^*\ H\ Q \leq \mu I \tag{82}$$

with the covariance matrix $H = G^T\, G$ and the projection Q in (74). Because $\mathfrak{A}(c)$ has full rank, this is equivalent to find a minimum $\mu > 0$ fulfilling

$$\mathfrak{A}(c)^*\ H\ \mathfrak{A}(c) \leq \mu\ \mathfrak{A}(c)^*\ \mathfrak{A}(c) \tag{83}$$

we summarize

Theorem 1. *Given is an arbitrary coefficient vector c with* $\deg c = p$. *Assume that the polynom c has no multiple roots. We can decompose G in two parts*

$$G = G_{modes} + \Delta G \tag{84}$$

where for $\Delta G = G\ Q$ with $Q = \mathfrak{A}(c)\ \left(\mathfrak{A}(c)^* \mathfrak{A}(c)\right)^{-1}\ \mathfrak{A}(c)^*$ fulfilling the requirements (72) we have $\|\Delta G\|_2 \leq \sqrt{\mu}$ iff

$$\mathfrak{A}(c)^*\ H\ \mathfrak{A}(c) \leq \mu\ \mathfrak{A}(c)^*\ \mathfrak{A}(c) \tag{85}$$

For the roots λ_l of c and $v_l = \frac{1}{w_l(\lambda_l)} G_{modes} \begin{bmatrix} w_l \\ 0 \end{bmatrix}$ with

$$c = w_l * \begin{bmatrix} -\lambda_l \\ 1 \end{bmatrix} \tag{86}$$

and the part of modes

$$G_{modes} = \sum_{l=1}^{p} v_l \left[1, \lambda_l, \lambda_l^2, \ldots, \lambda_l^n\right] \tag{87}$$

The complex vectors $v_l\left(q\right) = \left(v_l^f\left(q\right)\right)_{f \in S}$ are named Koopman modes [7].

The p roots provide different behaviour: $|\lambda_l| = 1$ for unsteady but stable modes (typical); $|\lambda_l| < 1$ for disappearing modes; $|\lambda_l| > 1$ for unstable modes. A system with such a mode cannot be stable.

It is possible to calculate a provisional c, delete any unwanted root as long as μ remains small. The degree of c should be small to limit the number of modes; on the other hand a small degree enlarges the approximation error μ. We have an algorithm minimizing μ and calculating c for a given degree p. This algorithm is still inefficient. Remark, that the classical DMD formulation of [56] is a special case for $p = n$.

Important for calculations for discretized partial differential equations is, that whereas G is a very large matrix with many rows, H is a quadratic matrix having the number of time steps as dimension.

5.4.2 Simplified approach

Applying the **trace** on both sides of this operator inequality, we get by definition of $H = G^T\ G$ for the j-th shifted row $G^j = \begin{bmatrix} g_{0+j} & g_{1+j} & \dots & g_{n+j} \end{bmatrix}$

$$\frac{1}{n-p+1} \sum_{j=0}^{n-p} \left\| G^j\ c \right\|^2 \leq \ \frac{\mu}{n-p+1} \sum_{j=0}^{n-p} \| \ c \ \|^2 \ = \ \mu \ \| \ c \ \|^2 \tag{88}$$

or with the collapsed matrix H^{n-p} which is composed by a sum of shifted submatrices of H

$$H^{n-p} = \frac{1}{n-p+1} \ \sum_{j=0}^{n-p} \left(G^j \right)^T G^j \tag{89}$$

we have

$$< H^{n-p}\ c\ ,\ c > \leq \ \mu \ \| \ c \ \|^2 \tag{90}$$

μ is not smaller than the largest eigenvalue of H^{n-p}

The coefficient vector c can be defined as the eigenvector of the minimal eigenvalue of the positive semidefinite matrix H^{n-p}

$$H^{n-p}\ c \ = \ \mu_{min}\ c \tag{91}$$

μ_{min} underestimates μ in (85) The matrix H^{n-p} can simply derived from the matrix H. The computational effort is relatively small. This procedure is an alternative to the procedure described before but has shortcomings. The vector c generated by this procedure might be not the best in sense of (85). It might introduce additional unwanted eigenvalues. Important is a small eigenvalue μ.

5.4.3 Ensembles

Summing up matrices of type H in (85) allows to handle all matrices together by a common polynom coefficient vector c.

$$< \frac{1}{i_{max}} \sum_{i=1}^{i_{max}} H_i\ c\ ,\ c > \overset{?}{\approx} \ 0$$

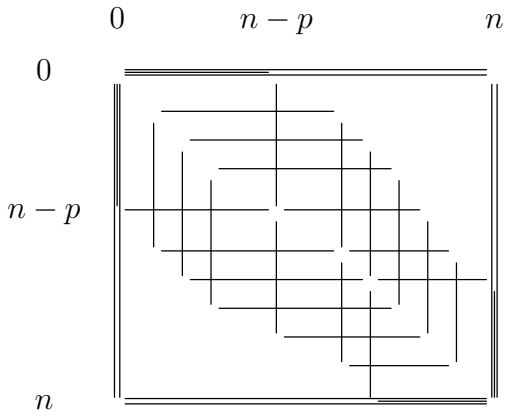


Figure 41: shifted submatrices as part of the total matrix

If such a vector with the involved roots exist, the **ensemble** starting with different start vectors or parameters can be compared with respect to a common decomposition. The approximative spectrum is common to all sets. the respective eigenvectors follow from the common spectrum by multiplication of the $\frac{1}{w_l(\lambda_l)} \begin{bmatrix} w_l \\ 0 \end{bmatrix}$ with the individual approximative measurements G^i_{modes}. This enables for extraction of common relevant features of ensemble calculations.

5.4.4 Koopman eigenfunctions

The finite approximative decomposition (87)

$$\mathbb{N}_0 \ni k \mapsto g_k(q) = \sum_{l=1}^{p} v_l(q) \ {\lambda_l}^k \tag{92}$$

allows to calculate Koopman eigenfunctions for this approximative sequence. Remember $g_k^f(q) = f\left(\varphi^k q\right)$ in (63) and the number of essential roots p. This describes the iterative development with respect to index k of all observables in S by modes λ_l (Ritz values), which are common for different starting values q and observables f. We have given an computable estimation of the error.

Rewriting equation (63) by stacking $(0:p)$ subsequent elements leads to

$$\begin{bmatrix} g_k(q) \\ g_{k+1}(q) \\ \vdots \\ g_{k+p}(q) \end{bmatrix} = \sum_{l=1}^{p} \begin{bmatrix} v_l(q) & \lambda_l^0 \\ v_l(q) & \lambda_l^1 \\ & \vdots \\ v_l(q) & \lambda_l^p \end{bmatrix} {\lambda_l}^k = \sum_{l=1}^{p} v_l(q) \begin{bmatrix} \lambda_l^0 \\ \lambda_l^1 \\ \vdots \\ \lambda_l^p \end{bmatrix} {\lambda_l}^k \tag{93}$$

Multiplying from the left by a vector $u^* = \frac{w_i^*}{w_i(\lambda_i)} d^*$, where w_i is the same polynom coefficient vector of degree $p-1$ as in (86) with $w_i(\lambda_l) = 0 \quad \forall\, l \neq i$ and d_i is an arbitrary vector, this transforms the decomposition to the action on the single mode i

$$u_i^* \begin{bmatrix} g_k(q) \\ g_{k+1}(q) \\ \vdots \\ g_{k+p}(q) \end{bmatrix} = \sum_{l=1}^{p} d_i^* v_l(q) \frac{w_i(\lambda_l)}{w_i(\lambda_i)} {\lambda_l}^k = d_i^* v_i(q) {\lambda_i}^k \tag{94}$$

Returning back to the definition (63) of $g_k^f(q) = f\left(\varphi^k q\right)$

$$u_i^* \begin{bmatrix} f\left(\varphi^k \circ \varphi\ q\right) \\ f\left(\varphi^{k+1} \circ \varphi\ q\right) \\ \vdots \\ f\left(\varphi^{k+p} \circ \varphi\ q\right) \end{bmatrix} = d_i^* v_i(q) {\lambda_i}^{k+1} = \lambda_i u_i^* \begin{bmatrix} f\left(\varphi^k q\right) \\ f\left(\varphi^{k+1} q\right) \\ \vdots \\ f\left(\varphi^{k+p} q\right) \end{bmatrix} \tag{95}$$

showing, that $u_i^* \left[f \circ \varphi^{k+j}\right]_{j=0,\cdots,p}$ is a Koopman eigenfunction for the eigenvalue λ_i on the trajectories starting with $q \in Q$. Given eigenvalues λ_i this does not dependent on $q \in Q$ nor on f. Remarkable is, that the eigenfunction is composed by the values only on the specific trajectory belonging to $q \in Q$. Because d_i is an arbitrary vector, the eigenspace belonging to λ_i is as large as the dimension of the linear space generated by the observables $f \in S$.

5.4.5 How to realize the Koopman related Dynamic Modes approach?

The covariance matrix $H = G^T G$ has to be calculated together with its spectrum and partially also with the eigenvectors. They may also serve as decomposition vectors for POD. The matrix is relatively small, as the diagonal is given by the number of analysed time steps. But the calculation might be very time consuming and expensive. Because the stiffness of the product matrix is much higher than the stiffness of the singular values of G, it is reasonable to calculate the singular value decomposition of G as for the original DMD or to calculate the QR-decomposition of G. For very large problems with many grid points this might be to time consuming. In these cases approaches by iterative techniques as Arnoldi procedures could be investigated. In any case parallel input and output in combination with the algorithms to get and to use the data is important and will be investigated.

5.4.6 Implementation and Experiences

The spectral theory of the Koopman operator expects eigenvalues (even the total spectrum) to be part of the unit disk. Eigenvalues outside of the disk would imply, that the operator φ would not reside on a compact domain or with other words, the discretization operator used would not be stable. Approaches as DMD do not ensure this property in any case. We developed a mechanism to get only eigenvalues with modulus not exceeding 1. It turns

out, that also the other calculated eigenvalues essentially dependent on this property. We found empirically, that for some of the eigenvalues λ also the multiples λ^l for $l = 1, 2, 3, \cdots$ are eigenvalues as predicted by the Koopman-operator theory. This is only true in an approximative sense and only for a few, but shows an interesting property for the underlying nonlinear systems as these will have harmonic properties. Each frequency implies its multiple frequencies related to their Koopman-modes.

The Koopman-modes as calculated from fluid flow simulation experiments look as smooth as the flow itself at least for smaller frequencies. Under certain conditions they earn linear properties from the fluid flow. For example the modes are underlying linear boundary conditions or are divergence free if they are calculated from iterated velocity fields of an incompressible flow.

We implemented VTKoutput for animated visualization of the generated modes by Paraview. Data may be given in a simple neutral format. Data given in VTKformat may also be analysed.

We have done first tests. The code is not yet parallelized for analysis of large data sets. First approach will be the (simple) parallelization of dense matrix-matrix products which need a large amount of computing time.

5.5 Conclusions and Future Work

In the preceding sections, we described two extensions to the JPEG-2000 compression algorithm, which we hoped would allow for the lossy and lossless encoding of IEEE-754 double precision floating point numbers. Based on the results of our study we found, that neither of these two extension are satisfactory for our ultimate goal of a true IEEE-754 compression algorithm. While the Q number format offers good compression ratios with reasonably low signal distortion, it is unable to compress the numerical dataset losslessly and therefore might limit the ultimate usability of the codec. The Shape-Adaptive Discrete Wavelet transform, on the other hand, proved to be unsuitable for decorrelating arbitrarily shaped regions within our numerical domain and is thus unsuitable for the task at hand.

Our hope is that intraband prediction methods, which are used in the High Efficiency Video Coding Standard (HEVC), could instead be used to overcome the efficiency problem when compressing the sign, biased exponent and mantissa fields [63]. We will also investigate higher order signal transforms which allow for the efficient transformation of images with smooth regions separated by smooth boundaries.

We have further described algorithms and their mathematical background generalizing the Dynamic Modes Decomposition of [56] with a clear relation to eigenfunctions of an appropriate Koopman operator and showed how to handle ensembles and have taken first steps for implementing the necessary procedures. Efforts to interpret the mathematical and physical properties of the Koopman-modes are ongoing.

6 Fault tolerance and resilience

Faults caused by malfunctioning hardware are placed into two overall categories, soft and hard node errors. Hard errors are faults that lead to the complete failure of a node. Common reasons being a bad power supply, a failed network interface card, or simply unexplained reboots. Typically a hard error involves only a single node, but in particular if critical infrastructure is shared among multiple nodes, such as a power supply units, occasionally hard errors do affect more than one node at a time. Soft errors on the other hand are all related to storage elements in the form of spurious bit-flips. A significant source of soft errors arise from energetic particles interacting with the silicon subtract that either flip the state of a storage element or disrupt the operation of a logic circuit. Such events may lead to a silent data corruption (SDC), i.e., no warning or exception is raised but data has been corrupted. Depending on the location of the SDC, it may lead to an event that over the course of many compute cycles turns into a hard error. An extensive overview of challenges in addressing fault-tolerance for future exascale computing systems is given in [60] and [9] along with an overview of sources and potential ways if mitigating the problems.

Faults that begin as soft-errors but eventually propagate into hard-errors have been a major issue in the past, in particular during the early days of building clusters based on commodity hardware. In a number of notable cases, radiation sources were shown to be the culprit in rendering, at the time being, very large scale clusters useless [28]. On todays clusters, all components from memory to CPUs to networks have some form of error correcting code build in. This does however not mean that soft-errors and SDCs have become non-issues. Some logic units, particularly with-in CPUs are prohibitively expensive to protect with error-correcting code.

There exists a large body of published research on handling SDC type errors from an application perspective. Many fault tolerant versions of commonly used algorithms in computational science, such algorithm based fault tolerance for general matrix operations [31], or more specific algorithms designed for certain iterative solvers such as those presented in [58, 54] among many others have been proposed. However, few of these algorithms have made it into practical applications, the typically attitude towards SDC resilience appears to be to simply assume that such errors are so rare that they may as well be ignored, favoring the simple solution of doing a re-run if the computational output looks dodgy. This approach raises questions on the trustworthiness of numerical simulations performed. In addition it is worth noting that not only the cost of an SDC induced re-run, but also the probability of needing such a re-run both scales linearly with the size of the machine, therefore, this naive approach may not be acceptable on future exascale systems.

Whilst the work on algorithm based fault-tolerance is quite extensive, the amount of publications on quantifying the rate at which SDC type errors occur on modern day clusters is somewhat limited in comparison. In [62, 61] the authors present a study on soft-errors occurring in the DRAM measuring error rate ECC, reporting rates of correctable and not correctable errors. As for CPUs, in [11] the authors present a study on the occurrence of soft-errors when irradiating a Power BQC 16C chip with high-energy particles during execution. They use the measurements to project actual long term failure rate for larger-scale HPC

systems. They project using the irradiation experiments the mean time between errors at sea level of the SRAM-based register files and Level-1 caches for a system similar to the scale of Sequoia system with roughly 1.6 million cores to be approximately 1.5 days. The numbers suggest that the extensive hardware based error correction in modern systems is doing a good job making SDCs somewhat infrequent even on Petascale systems. When SDCs do happen, they may have no measurable impact, depending on the location of the fault, as many iterative algorithms used widely in computational science are inherently soft error resilient [57]. Whilst the impact of SDCs on production code remains somewhat speculative, hard errors already present a very real challenge on large compute clusters[60]. Having at least some form of rudimentary fault-tolerance is essential for running code that scales to the full size of peta-scale systems. Simplistic approaches are unlikely to scale exascale systems. Even if the compute nodes in a potential exa-scale system would have an individual MTBF (Mean Time Between Feature) of a century, a machine with 100.000 such nodes would encounter a failure every 9 hours on average[19]. This being shorter than the execution time of many HPC applications. For the reasons mentioned we have decided to focus our efforts on hard errors, applying current state-of-the-art to Nektar++, improving upon techniques.

6.1 Check-pointing for Resilience

For current distributed memory applications, fault tolerance is most commonly achieved by simply periodically saving a solution state to a checkpoint file. These check-points are written to reliable storage, typically a parallel file system. Upon failure, an application may restart from a prior state by reading the checkpoint. An estimate for the optimal length of the time-interval between checkpoints can be computed using the formulas derived by Young [68] or Daly [15] under various assumptions. Youngs 1st order approximation is particularly simple

$$T_{FO} = \sqrt{2\mu C} \tag{96}$$

Where μ is the mean time between failure and C the time to create a checkpoint. The checkpoint-to-filesystem approach works well when using a comparatively small number of nodes, but does not scale well. The mean time between failure μ scales linearly with the number of nodes used, so does the associated lost computational work done between the point of failure and the most recent check-point. The problem is further exacerbated by the fact that the compute capabilities at large-scale facilities have increased much quicker than I/O bandwidth over the last decade, a trend that is expected to continue.

A promising approach to mitigate the scaling issues is to use multiple levels of checkpoints [29, 64]. It has been observed that most hard errors only effect a single node, and when more nodes fail at the same time they typically do so in a predictable manor, i.e. a power supply unit serving multiple nodes fail or something similar. Ideally, a local failure should permit local recovery. Multi-level check-pointing address the problem using different types of checkpoints, each of which have their own level of resilience and associated cost. Slower and more resilient levels could be made by writing to parallel file-system, thus allowing

recovery from many nodes failures. Cheaper and less resilient checkpoint levels may be constructed utilizing node-local storage such as RAM or Disk along with some form of cross-node redundancy or erasure code. If the cheaper checkpoints are able to recover from are comparatively large number of failures, the expensive parallel-file-system checkpoints will not have to be made as often. This leads to higher overall system efficiency as less time is spent creating checkpoint, and recomputing lost work.

6.1.1 SCR: Scalable Checkpoint/Restart for MPI

A number of libraries for fault-tolerance in HPC are in development, notable the Fault Tolerance Interface [3], Global View Resilience [12] and Scalable Checkpoint/Restart [47]. Among these options we have decided that SCR is the most promising option. It is well tested and has been used in early versions in production code at LLNL since 2008, additionally there is a fairly comprehensive manual on how to use SCR. They've documented that their large-scale jobs run more efficiently, recover more work upon failure and that the load on shared resources such as the parallel file system and network infrastructure are reduced when using SCR. Finally, the SCR Library is actively being developed on with planned improvements to prepare for exascale scale computing. Well in line with the goals of ExaFLOW. For a code to be able to utilize the SCR library, the code must satisfy a list of 11 criteria. The most limiting of these being

- The code must take globally-coordinated checkpoints written primarily as a file per process.
- On some systems, checkpoints are cached in RAM disk. This restricts usage of SCR on those machines to applications whose memory footprint leaves sufficient room to store checkpoint file data in memory simultaneously with the running application.

As for the first requirement that a globally-coordinated checkpoint must be written as a file per MPI rank, this may be limiting in context Nektar++. The code is MPI only, so having separate files for each process per checkpoint will create a very large number of files which may conflict with file system quotas. To mitigate this issue, Nektar++ supports creating checkpoints using HDF5, but this does not satisfy the first criteria for using SCR. The second requirement that could potentially cause issues is that of system memory. SCR uses a two-level checkpoint scheme. The more resilient level is a complete check-point to the parallel file system whereas the cheap, less resilient, checkpoint level is constructed using smaller groups of processors that save a check-point in local memory whilst applying some redundancy scheme across the processors in the group. The authors of SCR find that on their systems, 85 percent of all node failures, may require from the cheaper local checkpoint when an XOR redundancy scheme is applied[46]. The amount of additional RAM disk needed depends on what redundancy scheme is used, but the total memory footprint of the application will increase by atleast a factor of two. The added memory footprint of the application is less of a concern than the requirement that one must write checkpoints as a file per process since in the strong scaling limit applications tend not to be memory bound.

6.1.2 ULFM-MPI: User Level Failure Mitigation

An alternative to using the SCR library is to implement a multi level checksum checkpoint scheme directly into Nektar++ using ULFM-MPI[4]. User Level Failure Mitigation (ULFM) is a proposed extension to MPI developed by the MPI Forums Fault Tolerance Working Group. It is not a fault-tolerance library, rather it is an API that allows developers to implement fault-tolerant algorithms in MPI. ULFM was designed to manage failures following three fundamental concepts: 1) simplicity, the API should be easy to understand and use in most common scenarios; 2) flexibility, the API should allow varied fault tolerant models to be built as external libraries and; 3) absence of deadlock, no MPI call (point-to-point or collective) can block indefinitely after a failure, but must either succeed or raise an MPI error. A prototype of ULFM is available to be used with the OpenMPI compiler. As an alternative to using the multi-level checkpoint system that is the core of the SCR library, we may implement our own multi-layered recovery system specifically tailored to an application using ULFM-MPI. This has the advantage of control. We may experiment with different numbers of layers and various checksum checkpoint approaches as we see fit. The cost of the approach being that the implementation aspect is much more challenging than using the SCR library.

6.1.3 Multi-level check-sum check-pointing in Nektar++

As a first step towards adding resilience to hard errors, a prototype solver has been developed in Nektar++ which leverages the ULFM functionality developed within OpenMPI. This is combined with traditional disk-based check-pointing at relatively infrequent intervals.

Fluid dynamics simulations can be characterised by three phases: mesh partitioning, initialisation of the linear systems in memory, and time integration. The mesh partitioning phase takes non-negligible amount of time to complete on the very large meshes which would be expected to justify the use of exascale computing resources. However, the resulting partitions are written to the filesystem and are therefore inherently resilient. In comparison, the initialisation of the linear systems in memory is primarily a local operation with limited inter-node communication. By far the most considerable cost of the simulation is the time-integration phase, which typically takes $> 99.9\%$ of the walltime.

The complete ULFM recovery checkpoint is partitioned into two components: a *static* component and a *dynamic* component. The *static* component comprises all information necessary to reconstruct the linear systems in memory, which do not evolve during the time-integration phase. The *dynamic* component comprises the solution state variables which do evolve during time integration. The *static* checkpointing is performed once at the end of the initialisation phase, while the *dynamic* checkpointing is performed at regular intervals during the time-integration phase.

A *transaction log* approach is used for *static* checkpointing. In this approach, the output of all MPI transactions on a given process is logged throughout the initialisation phase. At the end of this phase, the log is transmitted once to the partner node, which retains the data in memory. This strategy has a number of advantages:

- Given the number of communication calls is relatively small (and of small size), the *static* checkpoint occupies much less memory than a copy of the completed linear system, as well as less interconnect bandwidth when transmitted to the partner node.
- A local process can recover in complete isolation from other processes by constructing the local portion of the linear systems as normal and replying the transaction log. The assumption is made that the recovering process will follow the exact same program pathway as the original process.
- Since almost all the modifications to the code are to intercept the MPI calls, this procedure can be cleanly applied with limited modifications outside of the communications layer.

A potential disadvantage is that the local recovering process must perform all the computation to reconstruct the linear systems while surviving processes wait idle. For large local systems this may be wasteful of resources but it is anticipated that at exascale computation will be highly parallel and local problem sizes will be small with linear systems that can be rapidly recomputed.

The first prototype implementation which provides inter-node in-memory check-pointing, thereby providing resilience against node failure, has been developed in Nektar++ for a diffusion problem. This has been tested to correctly recover the computation when a node fails at any point during the time-integration phase.

6.2 Improving upon State-of-the-art

Some form of redundancy scheme is needed when using disk-less checkpoint groups as detailed in the previous sections to create fast, lightly resilient, checkpoint levels. One approach is neighbor based check-pointing. In neighbor-based check-pointing, neighbor groups are assigned among the many checkpoint groups so that each group has at least one neighbor. In addition to keeping a local checkpoint in memory, at least one other checkpoint from a neighbor is stored. In this way the recovery process will always be localized to only involve the neighbor to a failed processor and the replacement processor, and no global communication and encoding/decoding calculations are needed. Various neighbor-hood based checkpoint schemes have been proposed such as mirroring [51] and ring neighbor [59].

An alternative approach is to apply some form of erasure code across a group of processors. Erasure codes 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 advantage compared to using a neighbor-based checkpoint scheme is that one can achieve a more resilient checkpoint at a much lower memory footprint. The disadvantage being the introduction of a potentially complicated encoder-decoder scheme as well as non-local communications during both the

check-pointing and recovery procedure. Many types of erasure codes exists, the simplest being the XOR scheme as used in SCR [50, 46]. If bit-wise exact recovery to multiple failures within a group is required, then some variant of the Reed-Solomon error correction code can be used [49].

In scientific applications, what needs to be check-pointed is often some form of numerical data. The data may be treated as bit-streams using the algorithms mentioned to create a checksum, but a checksum may also be generated from the floating point numbers directly. In [38] the authors list a number of advantages in doing so. A main point being that one avoids the trouble of introducing Galois Field arithmetic in the encoding and decoding procedure, instead using standard matrix operations on floating point numbers directly. The disadvantage of doing so is the introduction of round-off errors during the recovery process due to limited precision of representation of floating point numbers. As shown in [10, 38], the loss of accuracy may however be quite limited even for large data sets with many checksums with a proper choice of checksum encoding matrix. No matter what kind of erasure code is used for redundancy across groups of processors, if more processors are lost than checksums available in local memory within the group, it is not possible to recover the original data using the associated decoder. This is unfortunate as this means one must revert to a more resilient checkpoint such as one written to the parallel-file-system. Recovery from the parallel-file-system is not only slower to restart, but one also loses all computational progress since the last parallel file-system checkpoint was made.

When the number of lost processors is larger than $n-k$, infinitely many solutions satisfy the linear encoding scheme, and it is in general not possible to find the original data lost. However, often we do have some knowledge of the data encoded and its structure. In computational science, the data to be protected might very well be floating point numbers representing some smooth 2D or 3D surface. Can we somehow formulate the prior knowledge of the data, as a form of constraint to enforce uniqueness? The idea being to find the solution among infinite many possible solutions that best satisfy our regularity assumptions whilst also satisfying the checksum. Such a scheme does not need to be perfect in recovering the data. If the lost data can be recovered with an accuracy smaller than that that of numerical scheme used to solve the system of PDEs, it may be sufficient to avoid reverting to a higher level of resilience. Our preliminary findings is that the answer is yes. It is possible to recover information from under-determined systems when having knowledge of the underlying data structure and what it is to represent. In the next three short sections, an introduction to some algorithms for doing is presented for encoding floating point numbers using a Gaussian random matrix. These algorithms are comparatively simple to derive and illustrate in 1D as only standard matrix operations are involved unlike the case for incomplete data recovery in erasure codes on Galois fields.

6.2.1 Incomplete information recovery in under-determined check-sums

Define a vector $\bar{x}$ of length m times n. Now, assume that this vector is partially stored in n equal parts on n independent compute nodes. Then the "sub-vector" of each node contains a total of m elements. We can access the i'th out of m elements stored on the

jth node as x_{jm+i}, where $i \in \{0, m-1\}$ and $j \in \{0, n-1\}$. Say that we are at risk of losing the information stored on some of these nodes. One way of guarding against this is to compute an element wise checksum. That is, imagine that we have an additional spare node, somewhere that we can store another vector denoted $\bar{c}$ of length m. And say that for this vector $\bar{c}$, we compute each element as

$$c_i = \sum_{j=0}^{n-1} x_{jm+i}, \quad \forall\, i \in \{0, m-1\} \tag{97}$$

Then, if at some point we lose some node k. No matter which node it is, we may use the vector $\bar{c}$ to recover the data on this node by doing m summations of the form

$$x_{n_l m+i} = c_m - \sum_{j=0, j\neq k}^{n-1} x_{jm+i} \tag{98}$$

In other words, if we lose just one node, we may recover the data exactly - no matter what node was lost. We denote the vector $\bar{c}$ the checksum. But what if we lose more than one node? Well, it turns out that the above idea is extendable to what is called a weighted checksum that we will elaborated below. For ease of notation, let $\bar{x}^j$ denote the j'th sub-vector. Suppose that we can afford to store n_c checksum vectors $\bar{c}$, and that we compute each sub-vector on the form

$$\begin{cases} a_{11}\bar{x}^1 + \ldots + a_{1n}\bar{x}^n & = \bar{c}_0 \\ & \vdots \\ a_{n_c 1}\bar{x}^1 + \ldots + a_{n_c n}\bar{x}^n & = \bar{c}_{n_c - 1} \end{cases} \tag{99}$$

where a_{ij}, $i = 1, 2, \ldots n_c$, $j = 1, 2, \ldots n$ are some weights to be chosen. The matrix $A = (a_{ij})_{n_c n}$ is called the checkpoint matrix for the weighted checksum scheme. It turns out, that by choosing the weights in a clever way, one can recover lost information on up and including n_c lost nodes. How is the recovery done in practice? Assume, without loss of generality, that the computational processors $j_1, j_2, \ldots, j_k$ has failed and the $j_{k+1}, j_{k+2}, \ldots, j_n$ computational processors has survived, then the subvectors $\bar{x}^{j_1}, \bar{x}^{j_2} \ldots, \bar{x}^{j_k}$ are the unknowns that we would like to recover. By restructuring Eq. (99), we find that we have a system on the form Eq. (100) to solve

$$\begin{cases} a_{11}\bar{x}^1 + \ldots + a_{1j_k}\bar{x}^{j_k} & = \bar{c}_0 - \sum_{t=k+1}^{n} a_{0j_t}\bar{x}^{j_t} \\ & \vdots \\ a_{n_c 1}\bar{x}^1 + \ldots + a_{n_c j_k}\bar{x}^{j_k} & = \bar{c}_{n_c - 1} - \sum_{t=k+1}^{n} a_{n_c j_t}\bar{x}^{j_t} \end{cases} \tag{100}$$

Let A_r denote the coefficient matrix of the above linear system. If the number of checksums is equal to the number of lost nodes,$n_c = k$, then the above system has a single unique solution. If $n_c > k$, a unique solution exists as long as the elements in the checkpoint matrix A is chosen so that any sub-matrix of A is non-singular as this guarantees that A_r will have full rank. If the number of nodes lost k is larger than n_c , there is no unique solution to

the problem of recovering the lost sub-vectors $\bar{x}^n$. Of course, one could simply let n_c be very large to avoid such a situation, but increasing the number of checksum vectors increases the amount of resources that needs to be used both in terms of storage, computation and commutation. Ideally we'd like to keep the overhead resources low. So let's go back to the case of having only a single checksum vector. If we lose say $n_l > 1$ nodes, the solution space of every element $i \in [0, m-1]$ in each lost sub-vector $\bar{x}^j$ is spanned by an n_l dimensional affine hyperplane.

The solution space is infinitely large, and we can not recover the lost sub-vectors of the lost nodes directly. Let us now introduce the idea that the data vector stored in a distributed manor on many nodes has some structure to it that we have some *a priori* knowledge of. Say, we might know that the data points are sampled from a $\mathcal{C}^\infty$ functional. Now, given this knowledge, what if, among the infinite number of solutions to (100), for each m when $r > 1$, we choose a solution that, in some yet to be defined sense, makes the function that we imagine $\bar{x}^j$ to approximate as smooth as possible? In section 6.2.2, one possible approach of doing so is presented. We show that the ill-posed problem of recovering information when $r > t$ can be transformed into a well-posed convex optimization problem. Similarly, imagine that we have access to a reduced model, or any other form of inexact compression of the original data, we may then formulate our problem as to search for whatever data that is as close as possible to our reduced model, but satisfy checksums (99). This approach is outlined in section 6.2.2. All examples are given in 1D but may be extended to higher dimensions.

6.2.2 Uniqueness by minimizing the 1st order derivative

In order to define an objective function to be minimized that somehow expresses the "smoothness" of function represented by the data in the vector $\bar{x}$, we define a new variable $\bar{y} \in \mathbb{R}^{n_l m}$ where each element in the vector is an approximation of the 1st order derivative at the corresponding element in $\bar{x}$. For simplicity we use a simple centered finite difference scheme

$$y_{in_l+k} = \sum_{i_s=0}^{s} (-i)^{i_s} \begin{pmatrix} s \\ i_s \end{pmatrix} x_{L_k m+i+\frac{s}{2}-i_s} \quad , \; k = 0 \dots n_l - 1, \; i = 0 \dots m-1 \tag{101}$$

Where $L \in \mathbb{N}^{n_l}$ is the set of lost nodes, each lost node indicated with an integer. Note here that given $\bar{y}$, we may directly recover $\bar{x}^j$ for all the lost nodes assuming the boundaries are provided. One idea for an objective function to minimized could be the total variance, i.e.

$$V^{n_l} = \sum_{i=0}^{mn_l} |y_i| \tag{102}$$

Alternatively we could also seek to minimize the inner product of $\bar{y}$ with itself.

$$\bar{y}^T \bar{y} \tag{103}$$

We wish to minimize our objective function under the constraint that the checksum (99) is satisfied for all m after the sub-vectors $\bar{x}^j$ has been reconstructed. We must somehow

transform the constraints (99) to be expressed in terms of our new variable $\bar{y}$ to be minimized. The procedure of doing so is fairly technical, so we just give the result

$$\bar{y}\begin{cases} \sum_{k=0}^{n_l-1} y_k & = \beta_0 + \sum_{i=0}^{s}(-1)^i \binom{s}{i} c_{0+\frac{s}{2}-i} - \sum_{j=0,j\notin L_k}^{n-1}\sum_{i=0}^{s}(-1)^i\binom{s}{i}x_{jm+\frac{s}{2}-i} \\ \sum_{k=0}^{n_l-1} y_{n_l+k} & = \beta_1 + \sum_{i=0}^{s}(-1)^i \binom{s}{i} c_{1+\frac{s}{2}-i} - \sum_{j=0,j\notin L_k}^{n-1}\sum_{i=0}^{s}(-1)^i\binom{s}{i}x_{jm+1+\frac{s}{2}-i} \\ & \vdots \\ \sum_{k=0}^{n_l-1} y_{(m-2)n_l+k} & = \beta_{m-2} + \sum_{i=0}^{s}(-1)^i \binom{s}{i} c_{m-2+\frac{s}{2}-i} - \sum_{j=0,j\notin L_k}^{n-1}\sum_{i=0}^{s}(-1)^i\binom{s}{i}x_{(j+1)m-2+\frac{s}{2}-i} \\ \sum_{k=0}^{n_l-1} y_{(m-1)n_l+k} & = \beta_{m-1} + \sum_{i=0}^{s}(-1)^i \binom{s}{i} c_{m-1+\frac{s}{2}-i} - \sum_{j=0,j\notin L_k}^{n-1}\sum_{i=0}^{s}(-1)^i\binom{s}{i}x_{(j+1)m-1+\frac{s}{2}-i} \end{cases} \tag{104}$$

where $\beta_0 \dots \beta_{m-1}$ denotes the terms that take into account the left and right boundary on the data $\bar{x}$. For example, if in (101) we had chosen a 4th order centered stencil to approximate the derivatives that we are aiming to minimize, that we would have the stencil

$$y_{in_l+k} = x_{L_k m-2+i} - 4x_{L_k m-1+i} + 6x_{L_k m+i} - 4x_{L_k m+1+i} + x_{L_k m+2+i} \quad , k = 0 \dots n_l - 1, i = 0 \dots m-1 \tag{105}$$

If the data $\bar{x}$ is assumed symmetric around the boundaries, then one may show that the terms $\beta_0 \dots \beta_{m-1}$ become

$$\bar{\beta}\begin{cases} \beta_0 & = x_{-2} + 4x_{nm-1} - 4x_{-1} - x_{nm-2} \\ \beta_1 & = x_{-1} - x_{nm-1} \\ \beta_2 & = 0 \\ & \vdots \\ \beta_{m-3} & = 0 \\ \beta_{m-2} & = x_{nm} - x_0 \\ \beta_{m-1} & = x_{nm+1} + 4x_0 - 4x_{nm} - x_1 \end{cases} \tag{106}$$

The optimization problem of minimize (103) subject to the equality constraint (104) falls into a well studied branch of mathematical optimization called quadratic programming. Our problem is particularly nice since if we wrote (103) in the following form

$$\bar{y}^T Q \bar{y} \tag{107}$$

Then Q is just the identity matrix, i.e., it is positive definite. That the objective function is positive definite makes the optimization procedure much simpler. Let's write our equality constraint on matrix form as

$$E\bar{y} = \bar{d} \tag{108}$$

with $E \in \mathbb{R}^{n_l m} \times \mathbb{R}^m$ and $\bar{d} \in \mathbb{R}^m$ being defined directly from (103). For a positive definite quadratic optimization problem with only equality constraints, the solution process is linear. By using Lagrange multipliers, and seeking the extrema of the Lagrangian, it may be shown that the solution to our optimization problem is given by the solution to a linear system of the form

$$\begin{bmatrix} I & E^T \\ E & 0 \end{bmatrix} \begin{bmatrix} \bar{y} \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ \bar{d} \end{bmatrix} \tag{109}$$

where λ is a set of Lagrange multipliers which come out of the solution alongside $\bar{y}$. From 101, it is evident that $\bar{y}$ with boundaries uniquely defines $\bar{x}^j \forall j \in L$. In figure 42 a small

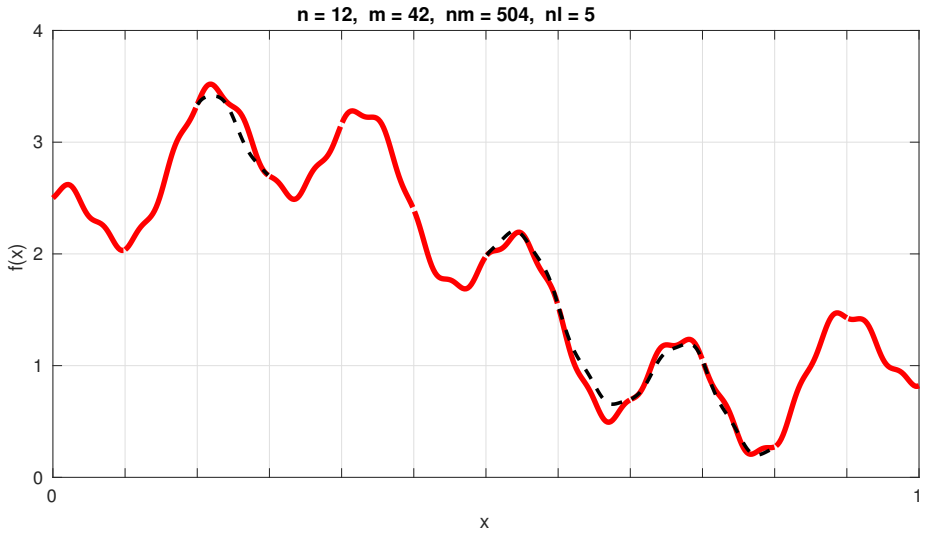


Figure 42: A single checksum is created for red-line data stores on $n = 12$ nodes. The data for $b_l = 5$ nodes is removed. Exact recovery from the checkpoint is not possible, but incomplete information recovery by solving the system (109) is and yields the output as indicated by the black lines.

example of the recovery is presented for a single checksum across $n = 12$ nodes, finding an approximation for the lost data when losing $n_l = 5$ nodes. The method as derived above may be extended two higher dimensions, and multiple checksums. How computationally efficient is the recovery procedure? The linear system (109) is of dimensionality $(n_l + 1)\, m$, that is, potentially very large. However, (104) is a system of m, n_l dimensional affine hyperplanes. Each hyperplane independent from the other. Since our objective function (107) is also positive semi definite, one can show that we may reduce our optimization problem from one $n_l m$ dimensional problem, to m small n_l dimensional problems, for each of which an analytical solution exists.

6.2.3 Uniqueness by minimizing distance with respect to inexact data

Imagine that we have some inexact approximation to the data that we wish to recover from the under-determined checksum equation 99 in the event that we have lost more nodes than we have checksums. In essence, we need a way to find whatever data x that minimize the distance to the inexact data $\bar{x}$, under the constraint that all checksums must be satisfied. That is, we wish to minimize the elements in $\bar{y}$

$$y_{mN_l+n_l} = x_{L_k(M+1)+m} - \bar{x}_{L_k(M+1)+m} \quad , \, n_l = 0 \ldots N_l - 1, \; m = 0 \ldots M \tag{110}$$

As before we must transform the constraints (99) to be expressed in terms of our new variable $\bar{y}$ to be minimized. Again, the procedure is a bit tedious, so we just skip to the result

$$\bar{y} \begin{cases} \sum_{n_l=0}^{N_l-1} a^k_{L_{n_l}} y_{n_l} & = c^k_0 - \sum_{n_l=0}^{N_l-1} a^k_{L_{n_l}} \bar{x}_{L_{n_l}M} - \sum_{n=0,n\notin L}^{N-1} a^k_n x_{nM} \\ & \vdots \\ \sum_{n_l=0}^{N_l-1} a^k_{L_{n_l}} y_{(M-1)N_l+n_l} & = c^k_{M-1} - \sum_{n_l=0}^{N_l-1} a^k_{L_{n_l}} \bar{x}_{\left(L_{n_l}+1\right)M-1} - \sum_{n=0,n\notin L}^{N-1} a^k_n x_{(n+1)M-1} \end{cases} \tag{111}$$

The above must be satisfied for all $k = 0 \dots K-1$. If we minimize the inner product of $\bar{y}$ with itself as before eq (107), then we are again faced with a positive definite quadratic optimization problem with equality constraints that have well studied solution methods. Once $\bar{y}$ has been found, an approximation to the lost data may be found by solving (110). As before we provide a small example, see figure 44. The advantage of this approach is that there is no need to save data at the data interfaces between nodes, an obvious disadvantage is the need for some reduced model or compressed data.

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 minimization problem (109) with the constraint (111), followed by the application of some regularization function that modifies the data towards some property, smoothness for example. Figure 43 shows an example of doing so on a very large system with, $n_c = 500$ checksums and $n_l = 1000$ lost nodes. Only the recovered data is depicted. This approach works surprisingly well, but is also very costly since the whole reconstruction procedure happens during each iteration.

6.3 Work-in-progress

The extension to 2D and higher of the algorithms derived in section 6.2 is conceptually no different than what was presented already, albeit the derivation of the constraints is substantially more involved. The data recovered when $n_l > n_c$ is not of machine accuracy, so for the approach to be of practical use in multi-level checksum checkpointing, we need some way of quantifying the accuracy expected of the data recovered. Otherwise we will not know whether the recovered data is sufficiently accurate to be used, so that one wouldn't need to revert to another checkpoint with a higher level resilience. We have experimentally observed a relation between n, n_l and n_c and the average accuracy of the recovered solution. We have however yet to find a theoretical foundation for this relation, and therefore can not claim a complete understanding of how the accuracy of the recovered data depends on parameters.

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 bit-wise representation is not necessary upon recovery. Accuracy below some given bound is considered sufficient. A much more widely used method is the class of Reed-Solomon codes [53]. Reed-Solomon codes have found wide-spread use for error-correction in bistreams and for resilience in storage centers such as RAID 6. The Reed-Solomon class of erasure codes are, at their core, also linear checksum schemes. The difference being that their checksum

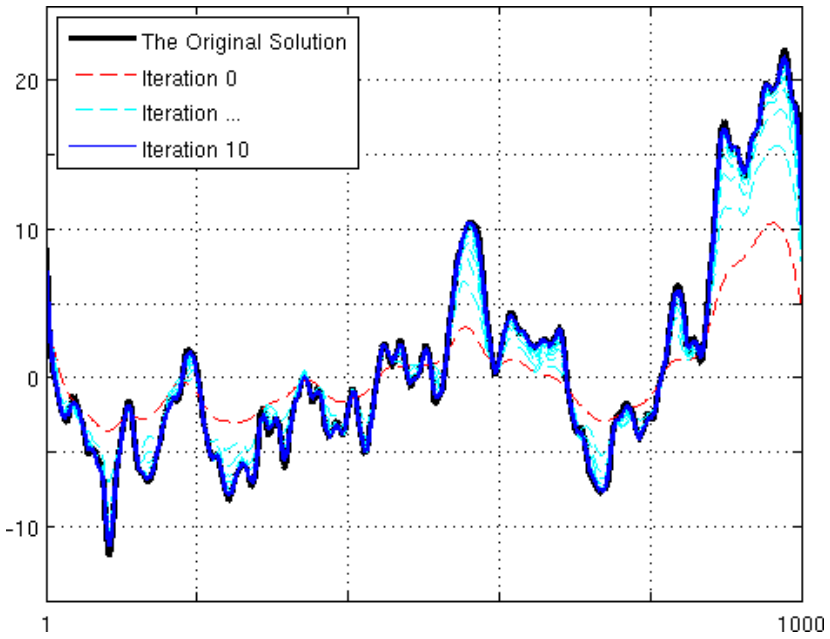


Figure 43: Incomplete information recovery using the iterative approach briefly outlined at the end of section 6.2.3. Only the data lost, and to be recovered is depicted. The black line indicates the original solution, and the blue line the recoveered solution after 10 iterations. Here $n_c = 500$ and $n_l = 1000$.

matrices are vandermonde matrices with coefficients in a Galois field. We believe that our approach of iteratively switching between enforcing some regularity condition, and solving a minimization problem to satisfy a checksum is agnostic to the underlying checksum matrix, and that the algorithm is extendable. If successful, the method would potentially have a wide number of applications in fault tolerance and error correction also outside the context of HPC.

Finally, as outlined in section 6.1 there are a number of limitations associated with using the SCR library for multi-level diskless checksum checkpointing in Nektar++. We are therefore moving in the direction of creating our own multi-level checksum checkpointing scheme and implementing it in Nektar++ using UFLM MPI. There are many design trade-offs that must be considered when creating a multi-level checkpointing scheme, but to a large extent we may be guided in our choices trough the extensive body of work already published on fault-tolerance using checksum-checkpointing.

Figure 44: $n_c = 10$ checksums have been created for the black line data stored on $n = 100$ nodes. The data for $n_l = 20$ nodes is removed. Exact recovery from the checkpoint is not possible, but incomplete information recovery by solving the optimization problem with constraint 111 and approximate data as indicated by the red line. The output from the solution procedure is marked in green. (a) full view. (b-c) zoomed in.

7 Summary

Significant progress has been made in the development of algorithms within all of the tasks in this work package, which are now being implemented as part of WP2 as detailed in D2.2, and examined in the use cases of WP3 as shown in D3.2. Several areas of overlap and collaboration between the different work package partners, as well as feedback from other work packages, are under active development. This includes, for example:

- Coarse space AMG preconditioners developed as part of task 1.1 by KTH are now being investigated by IC in task 1.3 in order to improve strong scalability of simulations, as identified by simulations performed in WP3 by ASCS.
- Significant progress has been made in collaborations between IC and EPFL have resulted in a prototype fault-tolerant solver for the diffusion equation. This prototype will now be extended to the Navier-Stokes equations. This will prove useful both as a platform for further algorithmic developments, as well as in examining node failures for the WP3 cases.
- SOTON and KTH have been in communication regarding the error estimators outlined in this deliverable, resulting in a number of developments on this front.

As outlined in each of the task summaries, there is a clear plan for the development of each task across the final phase of the project. We plan to continue the development of the algorithms along these lines by incorporating feedback and implementation from efforts in WP2, as well as their performance in the WP3 test cases.

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] 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.

[3] Leonardo Bautista-Gomez, Seiji Tsuboi, Dimitri Komatitsch, Franck Cappello, Naoya Maruyama, and Satoshi Matsuoka. Fti: high performance fault tolerance interface for hybrid systems. In *Proceedings of 2011 international conference for high performance computing, networking, storage and analysis*, page 32. ACM, 2011.

[4] Wesley Bland, Aurelien Bouteiller, Thomas Herault, Joshua Hursey, George Bosilca, and Jack J Dongarra. An evaluation of user-level failure mitigation support in mpi. In *European MPI Users' Group Meeting*, pages 193–203. Springer, 2012.

[5] M. E. Brachet, D. I. Meiron, S. A. Orszag, B. G. Nickel, R. H. Morf, and U. Frisch. Small-scale structure of the Taylor-Green vortex. *Journal of Fluid Mechanics*, 130:411–452, 5 1983.

[6] Tim Bruylants, Adrian Munteanu, and Peter Schelkens. Wavelet based volumetric medical image compression. *Signal Processing: Image Communication*, 31:112 – 133, 2015.

[7] M. Budišić, R. Mohr, and I. Mezić. Applied Koopmanisma). *Chaos*, 22(4):047510, December 2012.

[8] 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.

[9] Franck Cappello, Al Geist, William Gropp, Sanjay Kale, Bill Kramer, and Marc Snir. Toward exascale resilience: 2014 update. *Supercomputing frontiers and innovations*, 1(1):5–28, 2014.

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

[11] Chen-Yong Cher, Meeta S Gupta, Pradip Bose, and K Paul Muller. Understanding soft error resiliency of bluegene/q compute chip through hardware proton irradiation and software fault injection. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 587–596. IEEE Press, 2014.

[12] A Chien, Pavan Balaji, Nan Dun, Aiman Fang, Hajime Fujita, Kamil Iskra, Zachary Rubenstein, Ziming Zheng, Jeff Hammond, Ignacio Laguna, et al. Exploring versioned distributed arrays for resilience in scientific applications: global view resilience. *International Journal of High Performance Computing Applications*, page 1094342016664796, 2016.

[13] C. Christopoulos, A. Skodras, and T. Ebrahimi. The jpeg2000 still image coding system: an overview. *IEEE Transactions on Consumer Electronics*, 46(4):1103–1127, Nov 2000.

[14] 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.

[15] John T Daly. A higher order estimate of the optimum checkpoint interval for restart dumps. *Future generation computer systems*, 22(3):303–312, 2006.

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

[17] J. DeBonis. Solutions of the Taylor-Green Vortex Problem Using High-Resolution Explicit Finite Difference Methods. In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, Aerospace Sciences Meetings*, 2013.

[18] M. O. Deville, P. F. Fischer, and E. H. Mund. *High-Order Methods for Incompressible Fluid Flow*. Cambridge University Press, 2003.

[19] Jack Dongarra, Thomas Herault, and Yves Robert. Fault tolerance techniques for high-performance computing. In *Fault-Tolerance Techniques for High-Performance Computing*, pages 3–85. Springer, 2015.

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

[21] T. Eisner, B. Farkas, M. Haase, and R. Nagel. *Operator Theoretic Aspects of Ergodic Theory*. Berlin, Springer, 2015.

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

[23] 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.

[24] P. 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.

[25] 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.

[26] 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.

[27] 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.

[28] Al Geist. Supercomputings monster in the closet. *IEEE Spectrum*, (3), 2016.

[29] Erol Gelenbe. A model of roll-back recovery with multiple checkpoints. In *Proceedings of the 2nd international conference on Software engineering*, pages 251–255. IEEE Computer Society Press, 1976.

[30] Johan Hoffman, Johan Jansson, Niclas Jansson, and Rodrigo Vilela De Abreu. Towards a parameter-free method for high Reynolds number turbulent flow simulation based on adaptive finite element approximation. *Computer Methods in Applied Mechanics and Engineering*, 288:60–74, 2015. Error Estimation and Adaptivity for Nonlinear and Time-Dependent Problems.

[31] Kuang-Hua Huang et al. Algorithm-based fault tolerance for matrix operations. *IEEE transactions on computers*, 100(6):518–528, 1984.

[32] 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.

[33] S. P. Jammy, C. T. Jacobs, and N. D. Sandham. Performance evaluation of explicit finite difference algorithms with varying amounts of computational and memory intensity. *Journal of Computational Science*, In Press.

[34] 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.

[35] B. O. Koopman. Hamiltonian Systems and Transformations in Hilbert Space. *Proceedings of the National Academy of Science*, 17:315–318, May 1931.

[36] L. I. G. Kovasznay. Laminar flow behind a two-dimensional grid. *Proc. Cambridge Phil. Soc.*, 4:58–62, 1948.

[37] 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.

[38] 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.

[39] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21(4):1253–1278, March 2000.

[40] Shipeng Li and Weiping Li. Shape-adaptive discrete wavelet transforms for arbitrarily shaped visual object coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(5):725–743, Aug 2000.

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

[42] 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.

[43] 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.

[44] Y. Lu and M. N. Do. Crisp-contourlets: A critically-sampled directional multiresultion image representation. In *Proc. of SPIE Conference on Wavelet Applications in Signal and Image Processing X*, pages 655–665, San Diego, USA, Aug. 2003.

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

[46] Kathryn Mohror, Adam Moody, Greg Bronevetsky, and Bronis R de Supinski. Detailed modeling and evaluation of a scalable multilevel checkpointing system. *IEEE Transactions on Parallel and Distributed Systems*, 25(9):2255–2263, 2014.

[47] Adam Moody, Greg Bronevetsky, Kathryn Mohror, and Bronis R de Supinski. Design, modeling, and evaluation of a scalable multi-level checkpointing system. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–11. IEEE Computer Society, 2010.

[48] 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. to appear in proceedings of ICOSAHOM 2016, January 2017.

[49] James S Plank et al. A tutorial on reed-solomon coding for fault-tolerance in raid-like systems. *Softw., Pract. Exper.*, 27(9):995–1012, 1997.

[50] James S Plank and Kai Li. Faster checkpointing with n+ 1 parity. In *Fault-Tolerant Computing, 1994. FTCS-24. Digest of Papers., Twenty-Fourth International Symposium on*, pages 288–297. IEEE, 1994.

[51] James S Plank, Kai Li, and Michael A Puening. Diskless checkpointing. *IEEE Transactions on Parallel and Distributed Systems*, 9(10):972–986, 1998.

[52] 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.

[53] 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.

[54] Piyush Sao and Richard Vuduc. Self-stabilizing iterative solvers. In *Proceedings of the Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems*, page 4. ACM, 2013.

[55] J. Schmalzl. Using standard image compression algorithms to store data from computational fluid dynamics. *Computers and Geosciences*, 29:1021–1031, October 2003.

[56] P. J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, July 2010.

[57] Manu Shantharam, Sowmyalatha Srinivasmurthy, and Padma Raghavan. Characterizing the impact of soft errors on iterative methods in scientific computing. In *Proceedings of the international conference on Supercomputing*, pages 152–161. ACM, 2011.

[58] Manu Shantharam, Sowmyalatha Srinivasmurthy, and Padma Raghavan. Fault tolerant preconditioned conjugate gradient for sparse linear system solution. In *Proceedings of the 26th ACM international conference on Supercomputing*, pages 69–78. ACM, 2012.

[59] Luís Moura Silva and Joao Gabriel Silva. An experimental study about diskless checkpointing. In *Euromicro Conference, 1998. Proceedings. 24th*, volume 1, pages 395–402. IEEE, 1998.

[60] Marc Snir, Robert W Wisniewski, Jacob A Abraham, Sarita V Adve, Saurabh Bagchi, Pavan Balaji, Jim Belak, Pradip Bose, Franck Cappello, Bill Carlson, et al. Addressing failures in exascale computing. *The International Journal of High Performance Computing Applications*, 28(2):129–173, 2014.

[61] Vilas Sridharan, Nathan DeBardeleben, Sean Blanchard, Kurt B Ferreira, Jon Stearley, John Shalf, and Sudhanva Gurumurthi. Memory errors in modern systems: The good, the bad, and the ugly. In *ACM SIGPLAN Notices*, volume 50, pages 297–310. ACM, 2015.

[62] Vilas Sridharan, Jon Stearley, Nathan DeBardeleben, Sean Blanchard, and Sudhanva Gurumurthi. Feng shui of supercomputer memory positional effects in dram and sram faults. In *High Performance Computing, Networking, Storage and Analysis (SC), 2013 International Conference for*, pages 1–11. IEEE, 2013.

[63] Vivienne Sze, Madhukar Budagavi, and Gary J. Sullivan. *High Efficiency Video Coding (HEVC)*. Springer International Publishing, Switzerland, 2014.

[64] Nitin H Vaidya. A case for two-level distributed recovery schemes. In *ACM SIGMETRICS Performance Evaluation Review*, volume 23, pages 64–73. ACM, 1995.

[65] Christoph Wenzel, Björn Selent, Markus Kloker, and Ulrich Rist. Direct numerical simulation of compressible turbulent boundary layers at various subsonic and supersonic mach numbers. *Under consideration for publication in J. Fluid Mech.*

[66] Matthew O. Williams, Ioannis G. Kevrekidis, and Clarence W. Rowley. A data–driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.

[67] 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.

[68] John W Young. A first order approximation to the optimum checkpoint interval. *Communications of the ACM*, 17(9):530–531, 1974.