H2020 FETHPC-1-2014

Enabling Exascale Fluid Dynamics Simulations
Project Number 671571

D1.1 - Internal report on formulation of ExaFLOW algorithms

WP1: Algorithmic improvements towards exascale

DOCUMENT INFORMATION

Deliverable Number	D1.1
Deliverable Name	Internal report on formulation of ExaFLOW algorithms
Due Date	31/05/2016 (PM 9)
Deliverable lead	KTH
Authors	David Moxey (IC) Martin Vymazal (IC) Nicolas Offermans (KTH) Adam Peplinski (KTH) Philipp Schlatter (KTH) Christian Jacobs (SOTON) Neil Sandham (SOTON) Björn Dick (USTUTT) Jing Zhang (USTUTT) Uwe Küster (USTUTT) Patrick Vogler (USTUTT) Ulrich Rist (USTUTT) Jan Hesthaven (EPFL)
Responsible Author	David Moxey (IC) e-mail: `d.moxey@imperial.ac.uk`
Keywords	exascale algorithms, scalability, modelling, input/output
WP	WP1
Nature	R
Dissemination Level	PU
Final Version Date	31/05/2016
Reviewed by	Niclas Jansson (KTH), Julien Hoessler (McLaren Racing Ltd)
MGT Board Approval	31/05/2016

DOCUMENT HISTORY

Partner	**Date**	**Comment**	**Version**
IC	20/04/2016	Initial version with task 1.3	0.1
SOTON	02/05/2016	Add contribution to task 1.2	0.2
KTH	04/05/2016	Add contribution to task 1.1	0.3
USTUTT	05/05/2016	Add contribution to task 1.4	0.4
EPFL	06/05/2016	Add contribution to task 1.5	0.5
KTH/McLaren Racing Ltd	17/05/2016	Corrections	0.6
IC	17/05/2016	Final version	0.07
KTH	31/05/2016	Final version after PMB approval	1.0

Executive Summary

In this deliverable, we aim to summarise the contributions of the ExaFLOW partners that form work package 1 across the first nine months of the project. The goals of this report are to:

- describe our efforts in work package 1 to extend current state-of-the-art methods for large-scale computational fluid dynamics simulations to meet the needs of exascale computing;
- summarise the progress of the ExaFLOW consortium members to date in developing new algorithms to tackle these challenges;
- look ahead to expected developments of these algorithms as the project progresses.

We begin with an outline of the WP1 objectives and how these align with the challenges of exascale computing. Each of the tasks within WP1 is then described in detail with progress reports from each of the responsible partners. We conclude this summary with an overview of progress on each of the tasks:

- **Task 1.1 (KTH):** The focus of this task is in developing exascale parallel adaptive methods in terms of polynomial order, element size and moving mesh methods, alongside developments needed in preconditioners and error estimators that underpins this activity. The focus of KTH over the initial phase of the project has focused around two important aspects of this task. Firstly, a prototype spectral error estimator has been developed which will drive the adaption process, which being local in nature, is highly scalable in parallel and has been examined in the context of h-adaption for a diffusion equation. Secondly, a detailed examination of different preconditioning strategies in large-scale parallel simulations up to 500k computing cores has been performed in the Nek5000 code. Work is now ongoing to develop a highly parallel algebraic multigrid (AMG) solver which is required for the solution of the pressure Poisson equation in flow simulations. This is being coordinated with ICL, who will attempt to evaluate the use of the KTH AMG developments in the Nektar++ code for unstructured grids.
- **Task 1.2 (SOTON):** This task aims 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. Progress on this task has been limited to date due to difficulties and late appointment of staff at the beginning of the project. However, manpower is now in place to begin work on this effort and we anticipate supplying a detailed update on progress in the next report.
- **Task 1.3 (IC):** In this task, IC is developing a new finite element method based on the concept of a joint formulation between the compact, computationally efficient but communication-intense continuous Galerkin method (CG) and the communication

efficient hybridized discontinuous Galerkin (HDG). We aim to use the CG method to perform intra-node computation with the HDG method used intra-node in order to provide a highly scalable and computationally efficient method. Progress on this task has focused around the development of the initial intra-node formulation, which requires the use of a weak Dirichlet boundary condition to impose communication between nodes. We are now working towards testing of this method and furthering the development of the inter-node formulation. In tandem, we are starting to investigate options for coarse space preconditioning. Initial investigations are being undertaken with EPFL regarding the use of multi-level preconditioners for the HDG trace space system.

- **Task 1.4 (USTUTT):** Input and output (I/O) of data is presently a major bottleneck in a number of existing large-scale CFD applications, both from the context of I/O during the simulation and . This issue is expected to become even more pertinent in the context of even larger scale exascale computations. This task is investigating the application of data reduction techniques, which retain the key features of the flow whilst reducing the amount of data that needs to be stored. Progress on this task over the initial phase of the project has focused around two areas. The first is in investigating the applicability of common compression algorithms such as JPEG/JPEG-2000 still image and H.264 motion image compression for fluid simulations. Examination of the compression ratio, time taken for compression and resulting bitrate error was performed across four different discrete cosine and discrete wavelet transformation methods. The second half of progress on this task has investigated the formulation of singular value and dynamic mode decomposition methods, looking to apply these methods in parallel to CFD data. Both of these areas have therefore established a good starting point for further investigation of these data reduction techniques as the project progresses.

- **Task 1.5 (EPFL):** At exascale levels of computing, it is anticipated that faults arising from node failure (hard errors) or data corruption (soft errors) can be expected during the course of an average simulation. The purpose of this task is therefore to design fault tolerant methods that target the common numerical methods being used in CFD simulations. Progress on this task has focused around developing an understanding of how to improve existing algorithms to enhance resilience to both hard and soft errors hard and soft errors, including investigations of future support for fault detection and handling through MPI 4.0. Initially this has focused on the investigation of resilience to parallel-in-time methods, the results of which are available in a preprint [28]. Effort has been expended on the development and analysis of fault detection and recovery in complex iterative solvers, with multiple right hand sides. This will serve as a prototype for the development of resilient linear and nonlinear solvers in complex PDE solvers such as those considered in ExaFLOW. Investigation of sensitivities and critical resilience in complex PDE solvers is also being considered: e.g., some information such as geometry information must clearly be safe while others parts of the algorithm such as the pre-convergence iterates are less critical as they can be regenerated. Having a

thorough understanding of this is critical to propose an efficient strategy for ensuring resilience of a large scale production code. Progress is now being undertaken in collaboration with ICL after a recent meeting in London to discuss the application of these techniques to the Nektar++ code, specifically focusing around the conjugate gradient method that forms a key part of the CFD simulation.

Contents

1 Introduction

Computational fluid dynamics (CFD) is a clear area in which exascale computing can have significant impact in furthering our understanding of fundamental fluid mechanics problems that are of key importance to both industry and academia. Exascale computing will allow researchers to examine these flows at the extremely fine time- and length-scales needed to accurately model highly turbulent flows at high Reynolds numbers. Presently, this capability remains out of reach of current high performance computing platforms. However, the use of exascale resources should overcome this limitation and allow us to obtain a deeper understanding of the fundamental behaviour of these fluid flows.

Exascale computing will impose unique challenges and requirements in terms of designing algorithms that can effectively utilise the expected heterogeneous nature of these machines. These algorithms need to not only be massively parallel, but be capable of exploiting a combination of many-core architectures, vector units, and accelerators. Additionally, input and output (I/O), already a bottleneck in current systems, will likely become an even larger problem in exascale machines. Finally, the likely size of these machines means that there is a very high probability of machine failure during a typical length simulation. Algorithms for resilience and fault-tolerance are therefore required in order to detect and adapt around these failures.

Developing new methods to overcome these challenges and forms the key objectives of the ExaFLOW project. Specifically, the work being undertaken in this work package is designed to overcome these key algorithmic bottlenecks that need to be addressed before 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:

- **Task 1.1:** *Mesh quality and grid adaptivity (KTH)*
  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 (SOTON)*
  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 increase 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 (IC)*
  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 CG-HDG system, 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 (USTUTT)*
  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 (EPFL)*
  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 will outline the importance of each task for the exascale challenge, report the progress undertaken in each task and give an outlook as to how we expect work to progress over the course of the project. Where specific subtasks are noted (e.g. task 1.2.1), we refer the reader to ANNEX 1 of the ExaFLOW project for detailed information for each subtask.

2 Task 1.1: Mesh quality and grid adaptivity (KTH)

2.1 Objective of this task

Realistic flow problems involving turbulence quickly require large-scale simulation capabilities. The most crucial aspect in these situations is the proper representation of small-scale flow structures in time-dependent transport problems. This must be accomplished with minimal dissipation, as errors accumulated at small scales may become dominant when propagated through large computational domains over long integration times, and with minimal cost. Flexibility in mesh topology is instrumental, as simulation accuracy depends strongly on the quality of the mesh, which in turn must be adjusted to the (a priori unknown) flow. This is why mesh generation is considered to be a significant bottleneck in modern and future CFD. As more powerful HPC resources enable the simulation of complex, more realistic and industrially relevant flow problems, reliable mesh generation becomes more problematic, resulting in significant uncertainties in the simulation result. Although numerical uncertainties arise from many sources, including errors due to spatial and temporal discretisations or incomplete convergence, they can be minimised during the simulation by appropriate adaptation of the grid structure to the dynamic flow solution. Such automated mesh adaptivity, combining error estimation with dynamical refinement, is considered an essential feature for large-scale, computationally expensive simulations. It considerably improves the efficient use of resources, simplifies the grid generation and ensures a consistent accuracy (depending on the chosen measure) of the solution. The objectives of this task can be divided into:

- finding an effective criterion for refinement;

- and implementing a mesh refinement method.

The regions of the mesh where refinement is relevant can be located by the combination of local spectral error estimators and the resolution of an adjoint problem that establishes the accumulation of errors in space and time. Therefore, the first objective is the development of a criterion for refinement based on these two approaches.

The implementation of a tool for grid adaptivity will also be investigated. Specifically, refinement techniques (like the r, h and p-refinement methods) need to be provided. This can be achieved only by addressing the several issues raised by these methods. First, a parallel algebraic multigrid solver that will serve as a preconditioner for the resolution of the pressure equation has to be implemented. Presently, a serial implementation of this preconditioner exists, but since a modification of the mesh during the adaptation process implies a modification in the preconditioner, this needs to be taken care of in the most efficient way possible. Then, it is required to elaborate a parallel and dynamical grid partitioner to ensure proper load balancing between the computing nodes. This will ensure that the workload remains constant over the processes after the number of elements or polynomial order has been locally increased (or decreased).

2.2 Background

2.2.1 A posteriori local error estimator

Errors resulting from the discretization and resolution of a system of partial differential equations arise from four different sources. Modeling error occurs when the mathematical model for the equations does not match reality. This kind of error can not be handled by the code and we assume that the model is consistent with the actual physics that each user wants to simulate. Then, roundoff error is due to the finite accuracy of computers. This kind of error is also ignored as we once again assume that numerical parameters have been chosen properly (by using double precision floating point arithmetic for example). Truncation error arises because the solution is approximated by a finite spectral expansion. We will mostly focus on this source of error, which can be estimated by different methods and reduced by adaptive mesh refinement. Finally, there is quadrature error because of the discrete integration, which can also be estimated.

Truncation error. The local error on a spectral element can be estimated by extrapolating the decay of the spectral coefficients as shown in [25, 24]. If we consider $u(x)$, the exact solution of a 1D partial differential equation, then its spectral transform is

$$u(x) = \sum_{k=0}^{\infty} \hat{u}_k \, p_k(x), \tag{1}$$

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

$$\hat{u}_k = \frac{1}{\gamma_k} \int_{-1}^{1} w(x) u(x) p_k(x) \mathrm{d}x, \tag{2}$$

where w is a weight associated to the family of polynomials and $\gamma_k = \|p_k\|^2_{L^2_w}$.

The corresponding discrete expansion is truncated to order N by the truncation operator P_N as

$$P_N u(x) = \sum_{k=0}^{N} \hat{u}_k \, p_k(x). \tag{3}$$

Consequently, the truncation error can be written as the L^2_w-norm of $u - P_N u$. If we assume Legendre polynomials, $w = 1$ and the estimate for the truncation error ϵ_t becomes

$$\begin{aligned}
\epsilon_t = \|u - P_N u\|_{L^2_w} &= \left(\int_{-1}^{1} w(x) \left[\sum_{k=0}^{\infty} \hat{u}_k \, p_k(x) - \sum_{k=0}^{N} \hat{u}_k \, p_k(x) \right]^2 \mathrm{d}x \right)^{\frac{1}{2}} \\
&= \left(\int_{-1}^{1} \sum_{k=N+1}^{\infty} \hat{u}_k^2 \, p_k^2(x) \right)^{\frac{1}{2}} \\
&= \left(\sum_{k=N+1}^{\infty} \frac{\hat{u}_k^2}{\frac{2k+1}{2}} \right)^{\frac{1}{2}}.
\end{aligned} \tag{4}$$

However, the coefficients $\hat{u}_k$ are unknown for $k > N$ and need to be approximated. This is achieved by interpolating a linear least-squares approximation of $\log(\hat{u}_k)$ with respect to k for $k \leq N$ and then extrapolating the coefficients for $k > N$. In practice, this is done by finding the best parameters c and σ such that

$$\hat{u}_k \approx c \exp(\sigma k). \tag{5}$$

This interpolation gives valuable information about the decay rate σ of the coefficients, which is a good indication for convergence and can be used to decide which refinement method to choose. If the solution is smooth and decay is monotonic, this estimator performs well. In [37], it is suggested to shift the linear least-squares interpolation upwards so that no spectral coefficient lies above it.

Quadrature error. In [25], Mavriplis also estimates the quadrature error that needs to be added to the truncation error. We denote by $\bar{u}_k$ the discrete version of the continuous coefficients from equation (2). They are evaluated as

$$\bar{u}_k = \frac{1}{\gamma_k} \sum_{i=0}^{N} \rho_i u(\xi_i) p_k(\xi_i), \tag{6}$$

where ξ_i are the Gaussian quadrature points and ρ_i are the associated quadrature weights. We let $I_N u$ be the Lagrange polynomial interpolation of u

$$I_N u(x) = \sum_{k=0}^{N} \bar{u}_k \, p_k(x). \tag{7}$$

Therefore, the quadrature error ϵ_q is is given by

$$\epsilon_q = \|I_N u - P_N u\|_{L^2_w} = \left(\sum_{k=0}^{N} \frac{(\bar{u}_k - \hat{u}_k)^2}{\frac{2k+1}{2}} \right)^{\frac{1}{2}}. \tag{8}$$

Arguing that the spectral coefficients are obtained exactly for $k \leq N-1$, the quadrature error can be reduced to

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

In practice, we compute

$$\epsilon_q \approx \left(\frac{\bar{u}_N^2}{\frac{2N+1}{2}} \right)^{\frac{1}{2}}, \tag{10}$$

which is a safe over-estimate of the actual error.

Alternative error estimate. In [37], Willyard suggests to use a simpler error estimate. If exponential decay is strong enough, the tuncation error ϵ_t can be estimated by

$$\epsilon_t \approx \bar{u}_N. \tag{11}$$

This solution is much cheaper to compute than (4) but is also supposedly less accurate.

Constrained Legendre coefficients error estimator. Another way of estimating the error developed in [37] is by comparing the solution $I_N u$ with polynomial order N and another estimated with fewer degrees of freedom $I_{N-M} u$. The solution $I_{N-M} u$ is not computed by solving the problem a second time but is obtained by truncating the spectral series of $I_N u$. We denote the spectral coefficients of $I_{N-M} u$ by $\tilde{u}_k$ such that its spectral transform is

$$I_{N-M} u = \sum_{k=0}^{N-M} \tilde{u}_k \, p_k(x). \tag{12}$$

For $k = 0, ..., N-M-2$, we take the same coefficients as for $I_N u$ but the last two Legendre coefficients are chosen such that continuity is enforced at the boundary of the reference

element. This leads to the system

$$\tilde{u}_k = \bar{u}_k, \qquad k = 0, ..., N - M - 2 \tag{13}$$

$$I_{N-M}u(-1) = \sum_{k=0}^{N-M} \tilde{u}_k \, p_k(-1) = \sum_{k=0}^{N-M} \tilde{u}_k \, (-1)^k = I_N u(-1) \tag{14}$$

$$I_{N-M}u(1) = \sum_{k=0}^{N-M} \tilde{u}_k \, p_k(1) = \sum_{k=0}^{N-M} \tilde{u}_k \, 1^k = I_N u(1). \tag{15}$$

This constitutes a second estimate of the local truncation error on a coarser mesh with polynomial order $N - M$

$$\epsilon_t = \|I_N u - I_{N-M} u\|_{L^2_w} = \left(\sum_{k=N-M-1}^{N-M} \frac{(\bar{u}_k - \tilde{u}_k)^2}{\frac{2k+1}{2}} + \sum_{k=N-M+1}^{N} \frac{\bar{u}_k^2}{\frac{2k+1}{2}} \right)^{\frac{1}{2}}. \tag{16}$$

This method follows the idea of the τ-criterion for finite differences and finite volumes methods [6].

2.2.2 Refinement criteria

The knowledge of local error estimates is valuable data but it does not provide direct information on where to refine the mesh. In this section, we present different techniques to locate the best spots where to perform refinement. These methods are either driven by simplicity, like the local approach, or by the minimization of an objective function that would ensure an optimal refinement in some sense.

Local approach. The local approach consists of refining an element of the grid solely based on the knowledge of the local error estimate presented in the previous section. This method is applied by Mavriplis [25, 26] for example. The choice between the p- or h-refinement methods is based on the strength of the decay of the spectral coefficients. While this method does effectively reduce the global error, it is far from being optimal as we miss the influence of a local error to the global one and because the method does not account for the error accumulation in time.

Goal-oriented approach. Rather than decreasing the global error itself, the goal-oriented approach tries to minimize a global output of physical interest (typically stresses, mean fluxes, drag or lift coefficients...) by solving the adjoint equation for the output. One of the first work exploring this approach is [31] (based on ideas developed for design optimization [16]).

We explain the basic concept of the method very briefly for the case of the discrete adjoint-equation. Assume that we want to estimate the value of an integral quantity $f(U)$, where U is the solution of the system of partial differential equations $R(U) = 0$. Consider

a coarse mesh Ω_H and a fine mesh Ω_h, where H and $h < H$ represent typical dimensions of the elements and assume that only the coarse mesh Ω_H actually exists. One would like to estimate the value of $f_h(U_h)$ on the hypothetical fine mesh Ω_h while knowing only the coarse solution U_H. This estimated can be computed as

$$f_h(U_h) \approx f_h(I_h^H U_H) - (L_h^H \Psi_H)^T R_h(I_h^H U_H), \tag{17}$$

where I_h^H is a prolongation operator that maps coarse mesh solution onto the fine mesh, Ψ_H represents the adjoint solution on the coarse mesh and L_h^H is a prologation operator which expresses the coarse mesh adjoint on the fine mesh. The adjoint solution Ψ_H is obtained by solving the discrete adjoint equations on the coarse grid

$$\left[\frac{\partial R_H}{\partial U_H}\right]^T \Psi_H = \left(\frac{f_H}{U_H}\right)^T. \tag{18}$$

The method has been implemented and tested for 2D inviscid incompressible flows in [2, 8] and for the 1D viscous Burger equation in [30]. Continuous adjoint-equation applied to the compressible Euler equations is found in [39]. Automatic mesh adapation for the spectral difference method is developed in [20], where the differences between continuous and discrete adjoint methods are also discussed. An entropy based approach is developed in [15, 14] and applied to the compressible Navier-Stokes equations. This technique makes use of the symmetrization properties of the entropy functions and entropy variables that automatically satisfy an adjoint equation. This saves the cost of solving the adjoint equations and provides a cheap error indicator for mesh adaptation.

Sensitivity to refinement with respect to global error. In [37], Willyard presents an adjoint based error estimate to find the contribution of the local error at every time step to the global error at the final time, for the case of a bilinear operator. The concepts behind the method are based on those presented in [13], where the authors devised an adaptive method, also for nonlinear systems of partial differential equations, in the framework of the finite element method.

2.2.3 Refinement techniques

Once the location where refinement is required has been identified, several refinement techniques are available. One can either adjust the relative sizes of some elements (r-refinement), increase the number of elements (h-refinement) or increase locally the polynomial order (p-refinement).

r-refinement. The r-refinement technique consists of moving the inner nodes of the mesh in order to refine poorly resolved regions. This method offers an advantage in that it does not modify the topology of the mesh. However, while it seems an easy way to proceed, the practical implementation of an efficient algorithm for r-refinement is a complex task. In [38], an algorithm based on a classical steepest-descent method is proposed for the finite element

method using planar B-spline surface. In [27], the nodes are moved toward a center of mass depending on the estimated truncation error. In [9], the spring analogy is used to move the nodes in the case of the finit volume method. The idea is explained as follows in the paper:

In the spring analogy, edges of the mesh are treated as springs and the mesh is a web of springs. The error information is associated to each "spring" in the web via averaging of the element-based error indicator, and a force (via Hooke's law) is created by relating the error indicator to the spring equilibrium length, through a prescribed scaling.

h-refinement. The h-refinement method consists of refining the mesh locally by dividing some of the elements into smaller ones. The effect of this method is the apparition of hanging nodes at the interface that need to be taken care of. One of the solutions to solve this issue is to use mortar elements. The idea behind this method is that the facets of each element do not communicate directly. Instead they communicate with and intermediate element, called a mortar element, where a common flux is computed. Then, this flux is mapped back to the faces of the elements connected to the mortar element. This technique is used for the spectral differences in [20]. Similarly, non-conforming refinement using mortar elements for the spectral element method is presented in [22], where the direct stiffness procedure is extended to nonconforming elements via a mortar basis.

A second method for performing h-refinement is by interpolating the solution between the interfaces as in [19]. This method also extends the direct stiffness procedure by interpolating the solution at the Gauss-Lobatto-Legendre points at the interface.

p-refinement. In [12], the authors apply the goal oriented approach and the continuous adjoint formulation to the high-order discontinuous Galerkin method and increase locally the polynomial order. They ensure the continuity of the fluxes between two adjacent elements having different polynomial order by projecting the spectral coefficient from the high-order interface to an orthogonal basis, so as to remove high order frequencies. This can be done because the coefficients in orthogonal space are not coupled and average solution is not modified.

The p-refinement method is also presented for spectral differences using mortar elements in [20]. The procedure is very similar to the one followed for h-refinement.

h or p-refinement? When refining, one will proably wonder about the choice between h or p-refinement and some criterion is required in order to choose the right method. h-refinement induces a modification of the topology, an increase in the elements count but is better suited for flow discontinuities. p-refinement does not modify the mesh but induces load imbalance between the elements.

In [26], the choice is based on the decay rate of the spectral coefficients (following the developments from section 2.2.1). It is argued that if $\sigma < 1$, where σ is defined as in equation (5), h-refinement should be favored while p-refinement is preferred if $\sigma > 1$.

In [20], it is suggested to apply h-refinement around flow discontinuities and p-refinement when the flow is smooth, via the use of a discontinuity sensor.

2.2.4 Coarse grid solver

When solving the Navier-Stokes equations, the resolution of the pressure equation is a challenging task as it constitutes the main source of numerical stiffness. *Nek5000* [1], a CFD code based on the spectral element method, tackles this issue by using an optimal preconditioner for the pressure equation. Part of this preconditioner requires the resolution of an algebraic system on the coarse mesh, made of the vertices of the spectral elements only. This coarse grid problem can be solved by two different methods: a first one called XXT [35] or a second one called algebraic multigrid (AMG) [21]. We briefly present both methods in this section before comparing their performances.

Algebraic multigrid solver (AMG). The convergence rate of iterative solvers usually stalls after a certain number of iterations because of the slow decay of low frequency errors. In order to tackle this issue, multigrid methods have been developed where a correction is computed on a coarser grid, where convergence rate is faster for low frequencies. This technique is applied recursively until the lowest frequencies have converged. The multigrid method implemented in *Nek5000* is an algebraic multigrid (AMG). This method is supposedly better suited than XXT for simulations with a high count of elements and processes. However, it requires a setup step, which is currently performed by a serial Matlab code. This bottleneck makes AMG an unsuitable candidate to use with an adpative grid tool. Therefore, the parallelization of the setup phase needs to be implemented and we now investigate the basic concepts behind the AMG setup, which has been developed by James Lottes at the Argonne National Laboratory. The very basic explanations presented here focus only on the main ideas behind the setup while all the mathematical details can be found in [21].

As a starting point, assume that we want to use the AMG method to solve iteratively the linear system

$$A\mathbf{x} = \mathbf{b}, \tag{19}$$

where A is a large, symmetric positive-definite $n \times n$ matrix. In our case, n represents the number of discrete nodes. The algorithm for the setup is decomposed into three steps: a coarsening phase divides the gridpoints in two levels (coarse and fine points), an interpolation step links the solution on the two levels and a smoothing operator is built on the fine level. These three steps are repeated recursively until the coarse mesh is composed of a very low number of gridpoints.

First, coarsening consists in selecting a subset of nodes to form a coarser mesh. If we denote by a subscript c these nodes at the coarser level and by a subscript f the nodes that remain at the original level, the matrix A can be decomposed as

$$A = \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix}, \tag{20}$$

where we have assumed that the coarse variables are ordered last. For example, a possible coarsening of a 5×5 square domain ($n = 25$) at the first step of the algorithm is shown in Figure 1. Here, the number of c nodes n_c is 4 and the number of f nodes n_f is 21.

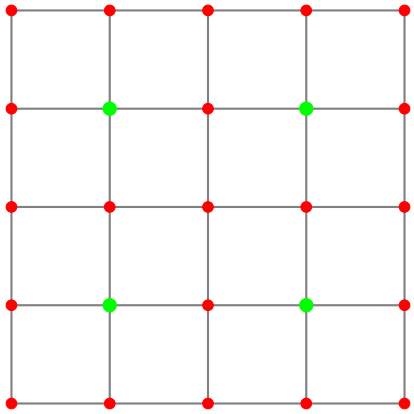


Figure 1: Coarsening of a 5×5 square domain ($n = 25$): f nodes are red and c nodes are green.

Secondly, it is necessary to define an interpolation operator W, having dimensions $n_f \times n_c$ and containing the interpolation weights, to go from the solution on one level to the other. If we denote by vectors $\mathbf{x}_c$ of dimension n_c and $\mathbf{x}_f$ of dimension n_f the solutions on c and f nodes respectively, then the original solution $\mathbf{x}$ is determined by the interpolation matrix

$$P = \begin{bmatrix} W \\ I, \end{bmatrix} \tag{21}$$

where I is an identity matrix of dimensions $n_c \times n_c$, such that the original solution is approximated from the coarse solution by

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_f \\ \mathbf{x}_c \end{bmatrix} \approx P\mathbf{x}_c = \begin{bmatrix} W\mathbf{x}_c \\ I\mathbf{x}_c \end{bmatrix}. \tag{22}$$

The construction of W is decomposed into the computation of the interpolation support and of the numerical weights. We notice that a larger interpolation support provides a better approximation but also a fuller operator W and consequently an increase in communication and computation when it comes to solve the AMG. Therefore, the algorithm is optimized to provide a good trade-off between accuracy and efficiency.

Finally, the last step consists in building a smoother B, which is an operator that will smooth high frequency errors. In practice, B takes the form

$$B = \begin{bmatrix} B_{ff} & \\ & 0 \end{bmatrix}, \tag{23}$$

where B_{ff} is a suitable preconditioner for A_{ff}. In our case, a parameter-free diagonal sparse approximate inverse (SPAI-0) is chosen.

XXT. The coarse grid solver denoted XXT is based on the Cholesky factorization of the matrix A^{-1} into XX^T with a convenient refactoring of the underlying matrix to maximize

the sparsity pattern of X^T. The basic idea behind XXT is to build a sparse basis X such that $XX^T \approx A^{-1}$. The first step for building the basis is to find a set of k_1 unit vector that are A-conjugate. A vector $\mathbf{x}_k$ is A-conjugate if it satisfies

$$\mathbf{x}_k^T A \mathbf{x}_k = \delta_{ij}, \tag{24}$$

where δ_{ij} is the Kronecker operator and A has size $n \times n$. Then, if we denote by $X_{k-1} = (\mathbf{x}_1 \mathbf{x}_2 ... \mathbf{x}_{k-1})$ the $n \times (k-1)$ matrix at iteration k and by $V = (\mathbf{v}_1 \mathbf{v}_2 ... \mathbf{v}_n)$ an appropriate column permutation of the identity matrix, the procedure to compute $\mathbf{x}_k$ is given by

$$\begin{aligned}
&do\ k = 1, ..., n: \\
&\quad \mathbf{w} := \mathbf{v}_k - X_{k-1} X_{k-1}^T A \mathbf{v}_k \\
&\quad \mathbf{x}_k := \mathbf{w} / \|\mathbf{w}\|_A \\
&\quad X_k := (X_{k-1} \mathbf{x}_k) \\
&enddo.
\end{aligned}$$

The algorithm is optimized in order to find an ordering for V that minimizes the fill for X. More details regarding complexity and implementation are available in [35].

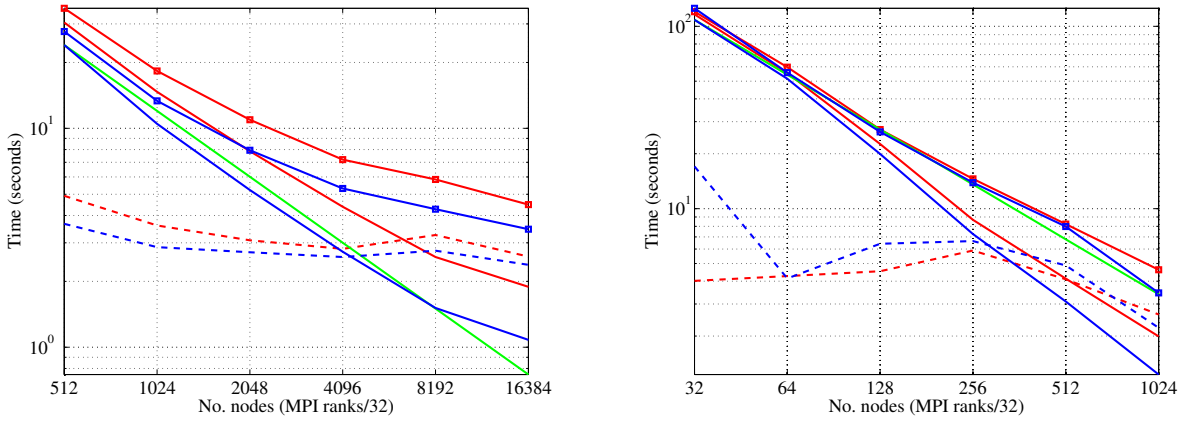
2.3 Summary of progress and outlook

The implementation of error estimators based on the spectral discretization wihtin each element (task 1.1.2.a) has been achieved following the developments presented in section 2.2.1 within *Nek5000*. This tool has already been successfully applied to simple 2D cases.

The implementation within *Nek5000* of the h-refinement technique as part of task 1.1.3.a has been done for the heat equation.

In the future, a new and optimal algorithm for a refinement criterion based on the adjoint method will be devised. This method will minimize a given objective function such as a physical quantity or global error (task 1.1.2.b). The extension of the h-refinement technique to the full Navier-Stokes will be implemented (task 1.1.3.a). Finally, we will enable the use of different polynomial orders among the elements within *Nek5000*. From then on, the p-refinement technique will be implemented (task 1.1.3.b).

In order to compare the two grid solvers, XXT and AMG, we have performed scaling tests with the intention to validate the better performance of AMG over XXT. Details of the test cases, computers used and scaling results can be found in [29]. In this paper, we perform a strong scaling analysis on three parallel machines with different performance characteristics and interconnect networks, namely Mira (IBM Blue Gene/Q), Beskow (Cray XC40) and Titan (Cray XK7). The test cases considered for the simulations correspond to a turbulent flow in a straight pipe at four different friction Reynolds numbers $Re_\tau = 180, 360, 550$ and 1000. For each case on each computer, we measure the computation and communication times over a large range of compute cores. The strong scaling limit is attained for roughly $5000 - 10,000$ degrees of freedom per core on Mira, $30,000 - 50,000$ on Beskow, with only a small impact of the problem size for both machines, and ranges between $10,000$ and $220,000$



(a) Mira (b) Beskow

Figure 2: Total mean time for 20 timesteps: AMG (blue), XXT (red), communication (dashed), computation (solid), total time (□), computational linear scaling (green).

depending on the problem size on Titan. In particular, we also study the effect of the two coarse grid solvers XXT and AMG on the computational time. If we investigate the communication requirements for both solvers, profiling tools have shown that AMG requires the exchanges of more shorter messages while XXT requires fewer but longer messages. In practice however, results show that AMG can lead to a decrease of up to 10% in the total for a high number of elements on a high number of processes. For example, we show in figure 2 the scaling plots for the case $Re_\tau = 550$ on Mira and Beskow.

As we can see, AMG outperforms XXT when the number of cores is high. Overall, AMG should be preferred and this constitutes our main motivation for the development of a parallel version of the setup phase. This task has been started and is still ongoing as part of task 1.1.1.b. The algebraic multigrid setup is decomposed into a coarsening, smoothing and interpolation steps. The first two steps are already parallelized for *Nek5000* and efforts are expended on the latter one. The parallel version of this setup phase will enable the rapid update of the coarse grid solver after the adaptation process.

3 Task 1.2: Error control for heterogeneous modelling (SOTON)

3.1 Overview

Heterogeneous modelling, in which the flow within complex geometries is modelled using different techniques depending on the flow complexity and level of detail required, offers a viable and computationally efficient method for large-scale flow simulations and enables numerical modellers to make the most out of available exascale-capable computing resources. Task 1.2 focuses on addressing the issues arising from extending heterogeneous modelling

to exascale, and in particular 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.

The work will be based around the SBLI code, developed at the University of Southampton, and the task is split into four components:

- **Task 1.2.1:** Error estimation for Finite Difference (FD) codes based on spectral resolution: using small-domain Fourier transforms to form error estimates and following r-refinement strategy.
- **Task 1.2.2:** Dynamic grid modification and load balancing for FD codes: including operation in heterogeneous environment.
- **Task 1.2.3:** Zone interface treatment including modules for generation of turbulence for a wide variety of conditions, combining synthetic eddy modelling with digital filtered white noise approaches. Extension to previous approaches are needed to model acoustic disturbances as well as thermal (entropy) terms. The latter cannot be based on the Reynolds analogy which does not apply to the instantaneous flowfield.
- **Task 1.2.4:** Efficient parallel framework for multi-scale approach, taken to very large scale: In particular the load balancing problem of LES with millions of degrees of freedom, automating the parallelism required for the LES and dealing with simulations where the direct numerical simulations can have different sizes. An additional physical problem at high Reynolds number is the possible need for three layers of simulations to cover the physics of turbulent flow at high Reynolds number. To cater for this the multi-scale approach needs to be generalized for an arbitrary number of tiers of simulations.

3.2 Progress update

Progress on this task during the initial phases of this project has been limited due to a lack of staffing until late. However a new postdoctoral researcher, Christian Jacobs, is now in place (as of 1st February 2016) to start working on this task. Under the general theme of WP1 (i.e. algorithmic improvements), most of the progress to date has been spent replacing the core of the Fortran-based SBLI code with a Python-based code generator. Essentially, this allows users to write the equations they wish to solve as high-level expressions, and the code that performs the finite difference approximations is generated automatically. We anticipate that this will help to future-proof the code as new exascale-capable architectures become available. It also introduces a separation of concerns between domain specialists, numerical modellers, and HPC experts, allowing better maintainability and extendibility of the codebase which will be important when trying out various approaches to heterogeneous modelling and error control.

3.3 Outlook and future work

Our next steps will be to look into developing and evaluating algorithms in SBLI with respect to the computational and energy savings brought about by storing derivatives in memory vs. recomputing them on the device: testing will take place on CPUs, GPUs, and Intel Xeon Phi cards. Code generation allows us to easily switch between these different architectures. This is expected to take place within the next 3-6 months. On the heterogeneous modelling front, a literature review is currently being performed of existing approaches. After this, we will start to investigate/evaluate techniques and extend an existing incompressible code to perform multiple small quasi-direct Navier Stokes (QDNS) simulatinos within a single large eddy simulation (LES), with two-way feedback. This is expected to start in the next 6 months. Finally, we aim to extend the incompressible LES/QDNS code to run a realistic case of compressible air flow past an aerofoil. This is expected to start after the next 12 months.

4 Task 1.3: Mixed CG-HDG formulation (IC)

One of the main challenges facing exascale computing is the communication costs that will be incurred between computing nodes at such large scales. Communication topologies will be inherently heterogeneous, leading to algorithms that must capitalise on both the intra-node communication between cores on each local system, and inter-node communication between systems. Exascale systems will therefore require hierarchical parallelization strategies, essentially differentiating between intra- and inter-node parallelism. Achieving good efficiency on both levels while exploiting existing algorithms is a significant challenge and a barrier to the ambition of using CFD codes on exascale computing platforms.

The purpose of this task is to address this need for hierarchical parallelization by considering a combination of two methods: one that is compact and can exploit intra-node data structures, and one that is more disjoint with simpler communication patterns to enable higher efficiency of strong scalability as the number of compute nodes is increased. The first is a popular method, known as the continuous Galerkin (CG) finite element method, which is well-established and widely used within the computational fluid dynamics community. In the CG method, a domain Ω is split into a mesh of simple elemental shapes such as triangles and tetrahedra, on which an approximate solution to the Navier-Stokes equations can be more readily obtained. These elements are then connected to construct the domain and give an approximate solution to the fluid along intersecting vertices, edges and faces of the elements.

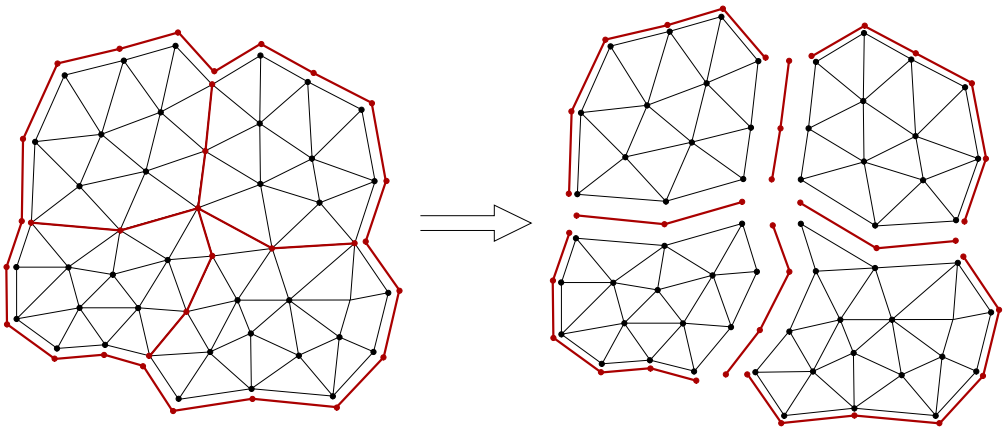
Although the CG method is well-suited for incompressible fluid simulations and is known to be computationally efficient, as the underlying representation on the mesh is compact, the connection between elements is strong in that, wherever elements connect – either through faces, edges, or vertices – communication must occur if these elements lie on different processes. This is a significant challenge, since for the complex geometries, such as Formula 1 cars and aircraft configurations that industry is increasing demanding and exascale comput-

ing can provide, meshes are *unstructured*, meaning that in complex areas, many tetrahedral elements can intersect at a single vertex, vastly increasing communication overheads.

More recently, *discontinuous* Galerkin (DG) methods and their hybridized (HDG) variants for elliptic problems are now gaining traction within the CFD community. The coupling between elements in this setting is far weaker, since they are only connected through faces, with information being transferred between elements via flux terms. From the perspective of exascale computing, this is ideal, since even if a large number of elements are connected through their vertices, the use of faces makes all communication element pairwise. However, this reduction in communication comes at a far higher computational cost, since DG methods typically require far more degrees of freedom to represent the problem.

4.1 Objective of this task

The objective of this task is to create a new method that adopts the best of both of these methods. We will use the CG method on a single node to make best use of the compact and efficient nature of this method. Between nodes we will use the HDG method to improve communication between nodes. The use of both of these methods should therefore lead to an efficient, scalable method, which is capable of realising the potential of exascale-level computing hardware.



The figure above gives an overview of the method in two dimensions. On the left, we can see the domain Ω split into 69 triangles. This left-hand figure essentially represents the CG problem on a single node. Our proposed method is depicted on the right, where four nodes are used to split the domain across the interior red lines and boundary conditions imposed weakly on the exterior red lines, with HDG being used to couple nodes together.

The rest of this section outlines the progress on this task to date. We outline our initial efforts in the formulation in a number of subsections:

- Background formulation of the HDG method;
- Overview of the CG-HDG principle;

- A hybrid formulation of the CG method;
- Summary of progress;
- An outlook for the rest of the course of the project.

4.2 Background formulation of the HDG method

This section is largely based on paper [17], and therefore not all notation is defined in this section. We therefore refer the reader to this article for further clarification, where we note that matrices in this article are typeset in bold face (e.g. $\mathbf{A}$) as opposed to blackboard font ($\mathbb{A}$). We assume that our problem is posed on a domain Ω, and seek the solution of the following prototype elliptic Helmholtz problem with Dirichlet ($\partial\Omega_D$) and Neumann ($\partial\Omega_N$) boundary conditions:

$$-\nabla^2 u(\boldsymbol{x}) = f(\boldsymbol{x}) \qquad \boldsymbol{x} \in \Omega, \tag{25}$$
$$u(\boldsymbol{x}) = g_D(\boldsymbol{x}) \qquad \boldsymbol{x} \in \partial\Omega_D, \tag{26}$$
$$\boldsymbol{n} \cdot \nabla u(\boldsymbol{x}) = g_N(\boldsymbol{x}) \qquad \boldsymbol{x} \in \partial\Omega_N, \tag{27}$$

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 (25) by introducing an auxiliary variable $\boldsymbol{q} = \nabla u$:

$$-\nabla \cdot \boldsymbol{q} = f(\boldsymbol{x}) \qquad \boldsymbol{x} \in \Omega, \tag{28}$$
$$\boldsymbol{q} = \nabla u(\boldsymbol{x}) \qquad \boldsymbol{x} \in \Omega, \tag{29}$$
$$u(\boldsymbol{x}) = g_D(\boldsymbol{x}) \qquad \boldsymbol{x} \in \partial\Omega_D, \tag{30}$$
$$\boldsymbol{q} \cdot \boldsymbol{n} = g_N(\boldsymbol{x}) \qquad \boldsymbol{x} \in \partial\Omega_N. \tag{31}$$

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 (28) and (29)

$$\sum_{\Omega^e \in \mathcal{T}_h} \int_{\Omega^e} \left(\nabla v \cdot \boldsymbol{q}^{DG}\right) d\boldsymbol{x} - \sum_{\Omega^e \in \mathcal{T}_h} \int_{\partial\Omega^e} v(\boldsymbol{n}^e \cdot \tilde{\boldsymbol{q}}^{DG})\, ds = \sum_{\Omega^e \in \mathcal{T}_h} \int_{\Omega^e} v\, f\, d\boldsymbol{x} \tag{32}$$
$$\sum_{\Omega^e \in \mathcal{T}_h} \int_{\Omega^e} (\boldsymbol{w} \cdot \boldsymbol{q}^{DG})\, d\boldsymbol{x} = -\sum_{\Omega^e \in \mathcal{T}_h} \int_{\Omega^e} (\nabla \cdot \boldsymbol{w}) u^{DG}\, d\boldsymbol{x} + \sum_{\Omega^e \in \mathcal{T}_h} \int_{\partial\Omega^e} (\boldsymbol{w} \cdot \boldsymbol{n}^e) \tilde{u}^{DG}\, ds, \tag{33}$$

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 $(u^{DG}, \boldsymbol{q}^{DG})$.

In order to have suitable numerical properties, the numerical traces must have the form

$$\tilde{u} = \left(\frac{1}{2} - \mathbf{C}_{21} \cdot \boldsymbol{n}^+\right) u^+ + \left(\frac{1}{2} - \mathbf{C}_{21} \cdot \boldsymbol{n}^-\right) u^- - C_{22}(\boldsymbol{q}^+ \cdot \boldsymbol{n}^+ + \boldsymbol{q}^- \cdot \boldsymbol{n}^-)$$
$$\tilde{\boldsymbol{q}} = \left(\frac{1}{2} - \mathbf{C}_{12} \cdot \boldsymbol{n}^+\right) \boldsymbol{q}^+ + \left(\frac{1}{2} - \mathbf{C}_{12} \cdot \boldsymbol{n}^-\right) \boldsymbol{q}^- - C_{11}(u^+ \boldsymbol{n}^+ + u^- \boldsymbol{n}^-),$$

where the scalar parameters C_{11}, C_{22} and the vectors $\mathbf{C}_{12}$ and $\mathbf{C}_{21}$ are chosen to ensure the stability of the method and its optimal convergence properties. The numerical traces on Dirichlet boundary of Ω reduce to

$$\tilde{u} = \left(\frac{1}{2} + \mathbf{C}_{21} \cdot \boldsymbol{n}\right) u + \left(\frac{1}{2} - \mathbf{C}_{21} \cdot \boldsymbol{n}\right) g_D$$
$$\tilde{\boldsymbol{q}} = \boldsymbol{q} - C_{11}(u - g_D)\boldsymbol{n},$$

and to

$$\tilde{u} = u - C_{22}(\boldsymbol{q} \cdot \boldsymbol{n}^+ - g_N),$$
$$\tilde{\boldsymbol{q}} = \left(\frac{1}{2} - \mathbf{C}_{12} \cdot \boldsymbol{n}\right) + \left(\frac{1}{2} + \mathbf{C}_{12} \cdot \boldsymbol{n}\right) g_N \boldsymbol{n}$$

on Neumann boundary $\partial\Omega_N$.

4.2.1 Local formulation of the HDG method

Assume that the function

$$\lambda \coloneqq \tilde{u}^{DG} \in \mathcal{M}_h, \tag{34}$$

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:

$$\int_{\Omega^e} (\nabla v \cdot \boldsymbol{q}^e)\, d\boldsymbol{x} - \int_{\partial\Omega^e} v(\boldsymbol{n}^e \cdot \tilde{\boldsymbol{q}}^e)\, ds = \int_{\Omega^e} v\, f\, d\boldsymbol{x}, \tag{35}$$

$$\int_{\Omega^e} (\boldsymbol{w} \cdot \boldsymbol{q}^e) d\boldsymbol{x} = -\int_{\Omega^e} (\nabla \cdot \boldsymbol{w}) u^e d\boldsymbol{x} + \int_{\partial\Omega^e} (\boldsymbol{w} \cdot \boldsymbol{n}^e)\lambda\, ds, \tag{36}$$

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 $(u^e, \boldsymbol{q}^e)$:

$$\tilde{\boldsymbol{q}}^e(\boldsymbol{x}) = \boldsymbol{q}^e(\boldsymbol{x}) - \tau\big(u^e(\boldsymbol{x}) - \lambda(\boldsymbol{x})\big)\boldsymbol{n}^e \quad \text{on } \partial\Omega^e \tag{37}$$

for some positive function τ.

4.2.2 Global formulation

We denote by $(U_\lambda, \boldsymbol{Q}_\lambda)$ and by $(U_f, \boldsymbol{Q}_f)$ the solution to the local problem (35), (36) when $\lambda = 0$ and $f = 0$, respectively. Due to the linearity of the original problem (25) and its mixed form, the solution satisfies

$$(u^{HDG}, \boldsymbol{q}^{HDG}) = (U_\lambda, \boldsymbol{Q}_\lambda) + (U_f, \boldsymbol{Q}_f). \tag{38}$$

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

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

$$\lambda = P_h(g_D) \quad \text{on } \partial\Omega_D \tag{39}$$

$$\sum_{\Omega_e \in \mathcal{T}_h} \int_{\partial\Omega^e} \mu \tilde{\boldsymbol{q}} \cdot \boldsymbol{n}\, ds = \int_{\partial\Omega_N} \mu g_N\, ds, \tag{40}$$

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 $\mathcal{M}_h$.

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:

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

4.2.3 Matrix form

We first 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 $\boldsymbol{e}_2 = [0, 1]^T$:

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

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

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

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

Inserting the finite expansions of the trial functions into equations (35) and (36) and using the definition of the flux (37) yields the matrix form of *local solvers*

$$\left[(\mathbf{D}_1^e)^T\ (\mathbf{D}_2^e)^T\right] \begin{bmatrix} \underline{\hat{q}}_1^e \\ \underline{\hat{q}}_2^e \end{bmatrix} - \sum_{l=1}^{N_b^e} \left[\widetilde{\mathbf{E}}_{1l}^e\ \widetilde{\mathbf{E}}_{2l}^e\right] \begin{bmatrix} \underline{\hat{q}}_1^e \\ \underline{\hat{q}}_2^e \end{bmatrix} + \sum_{l=1}^{N_b^e} \tau^{e,l} \left[\mathbf{E}_l^e \underline{\hat{u}}^e - \mathbf{F}_l^e \hat{\underline{\lambda}}^{\sigma(e,l)}\right] = \underline{f}^e \tag{41}$$

$$\mathbf{M}^e \underline{\hat{q}}_k^e = -(\mathbf{D}_k^e)^T \underline{\hat{u}}^e + \sum_{l=1}^{N_b^e} \widetilde{\mathbf{F}}_{kl}^e \hat{\underline{\lambda}}^{\sigma(e,l)} \qquad k = 0, 1 \tag{42}$$

The *global equation for* λ can be obtained by discretizing the transmission condition (40). 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 $l = \sigma(e,i) = \sigma(f,j)$. Noting that the following two identities hold

$$\mathbf{F}_l^e = \left(\bar{\mathbf{F}}^{\sigma(e,l),e} \right)^T \qquad \tilde{\mathbf{F}}_{kl}^e = \left(\underset{\sim}{\overline{\mathbf{F}}}{}_k^{\sigma(e,l),e} \right)^T, \tag{43}$$

the transmission condition can be recast as

$$\begin{aligned} &\left\{ \left[\left(\widetilde{\mathbf{F}}_1^e \right)^T \widetilde{\mathbf{F}}_2^e \right)^T \right] \begin{bmatrix} \underline{\hat{q}}_1^e \\ \underline{\hat{q}}_2^e \end{bmatrix} + \tau^{e,i} \bar{\mathbf{G}}^{\sigma(e,i)} \underline{\hat{\lambda}}^{\sigma(e,i)} - \tau^{e,i} \left(\mathbf{F}_j^e \right)^T \underline{u}^e \right\} + \\ &\left\{ \left[\left(\widetilde{\mathbf{F}}_1^f \right)^T \widetilde{\mathbf{F}}_2^f \right)^T \right] \begin{bmatrix} \underline{\hat{q}}_1^f \\ \underline{\hat{q}}_2^f \end{bmatrix} + \tau^{f,j} \bar{\mathbf{G}}^{\sigma(f,j)} \underline{\hat{\lambda}}^{\sigma(f,j)} - \tau^{f,j} \left(\mathbf{F}_j^f \right)^T \underline{u}^f \right\} = \underline{g}_N^l \end{aligned} \tag{44}$$

4.2.4 Combined Continuous-Discontinuous Formulation

To take advantage of the efficiency and lower memory requirements of continuous Galerkin method together with the flexibility and more favourable 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.

4.3 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 with some additional terms, which will be only applied on elements adjacent to partition boundaries, providing weak coupling between each partition and the global trace variable.

Note that in equation (41), 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. (41) then becomes

$$\left((\mathbf{D}_1^e)^T - \sum_{l=1}^{N_b^e} \tilde{\mathbf{E}}_{1l}^e\right) \underline{\hat{q}}_1^e + \left((\mathbf{D}_2^e)^T - \sum_{l=1}^{N_b^e} \tilde{\mathbf{E}}_{2l}^e\right) \underline{\hat{q}}_2^e = \underline{f}^e \tag{45}$$

On affine elements, we can write

$$\left((\mathbf{D}_k^e)^T - \sum_{l=1}^{N_b^e} \tilde{\mathbf{E}}_{kl}^e\right) \underline{\hat{q}}_k^e = -\mathbf{D}_k^e, \quad k = 0, 1.$$

This is a discrete representation of integration by parts:

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

After substituting this identity into the local solver (45), the equation (45) becomes

$$-\mathbf{D}_1^e \underline{\hat{q}}_1^e - \mathbf{D}_2^e \underline{\hat{q}}_2^e = \underline{f}^e, \tag{46}$$

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

The flux in (42) is

$$\underline{\hat{q}}_k^e = \left(\mathbf{M}^e\right)^{-1} \left\{ -(\mathbf{D}_k^e)^T \underline{\hat{u}}^e + \sum_{l=1}^{N_b^e} \widetilde{\mathbf{F}}_{kl}^e \underline{\hat{\lambda}}^{\sigma(e,l)} \right\} \qquad k = 0, 1 \tag{47}$$

and inserting this expression into (46) yields the following form:

$$-\mathbf{D}_1^e \left(\mathbf{M}^e\right)^{-1} \left((\mathbf{D}_1^e)^T \hat{u}^e - \sum_{l=1}^{N_b^e} \tilde{\mathbf{F}}_{1l}^e \underline{\hat{\lambda}}^{\sigma(e,l)} \right) - \mathbf{D}_2^e \left(\mathbf{M}^e\right)^{-1} \left((\mathbf{D}_2^e)^T \hat{u}^e - \sum_{l=1}^{N_b^e} \tilde{\mathbf{F}}_{2l}^e \underline{\hat{\lambda}}^{\sigma(e,l)} \right) = \underline{f}^e \tag{48}$$

The terms $-\mathbf{D}_k^e \left(\mathbf{M}^e\right)^{-1} (\mathbf{D}_k^e)^T, k = 0, 1$ represent Laplace operator, the rest is responsible for weak enforcement of Dirichlet boundary conditions prescribed by λ on the partition boundary.

4.4 Hybrid Formulation for Continuous Galerkin Method

This text is a brief summary of some parts of [7], where again we adapt similar notation to save redefinition in this document. The paper provides a generic framework for hybridization of several methods, among them continous Galerkin. One could thus consider hybridization on a 'local scale' within each partition.

As in the previous sections, we suppose that the solution on element traces $\partial K, K \in \mathcal{T}_h(\Omega)$ is given by $u = \lambda + g$, where

$$\lambda = \begin{cases} u & \text{on} \quad \partial K \backslash \partial\Omega, \\ 0 & \text{on} \quad \partial K \cap \partial\Omega, \end{cases} \qquad \text{and} \qquad g = \begin{cases} 0 & \text{on} \quad \partial K \backslash \partial\Omega, \\ g & \text{on} \quad \partial K \cap \partial\Omega, \end{cases}$$

Because the problem is linear, we can seek the solution as a pair $(u^H, \boldsymbol{q}^H) = (\boldsymbol{Q}_\lambda + \boldsymbol{Q}_g + \boldsymbol{Q}_f, U_\lambda + U_g + U_f)$ in Ω with local solvers $(\boldsymbol{Q}(\cdot), U(\cdot))$ defined in each element $K \in \mathcal{T}_h(\Omega)$:

$$\boldsymbol{Q}_\lambda + \nabla U_\lambda = 0, \qquad \nabla \cdot \boldsymbol{Q}_\lambda + U_\lambda = 0 \text{ on } K, \quad U_\lambda = \lambda \text{ on } \partial K \tag{49}$$

$$\boldsymbol{Q}_f + \nabla U_f = 0, \qquad \nabla \cdot \boldsymbol{Q}_f + U_f = f \text{ on } K, \quad U_f = 0 \text{ on } \partial K \tag{50}$$

Note that *this decomposition of* $(\boldsymbol{q}, u)$ is only possible if the normal component of $(\boldsymbol{q}_\lambda + \boldsymbol{q}_g + \boldsymbol{q}_f)$ is continuous across interelement boundaries, i.e.

$$[\![\boldsymbol{Q}_\lambda + \boldsymbol{Q}_g + \boldsymbol{Q}_f]\!] = 0, \tag{51}$$

where $[\![\cdot]\!]$ denotes the jump of a normal vector across ∂K.

The goal of this section is to identify the bilinear form $a_h(\cdot,\cdot)$ and right-hand side linear form $b_h(\cdot)$ such that the hybridized continuous Galerkin method on element traces can be written as

$$a_h(\lambda_h, \mu) = b_h(\mu) \qquad \forall \mu \in M_h.$$

This global equation for the trace-variable λ_h together with properly defined local solvers should then yield the same solution as the 'standard' weak form of continuous Galerkin method for Laplace equation.

4.4.1 Global Equation for Trace Variable

Without making any further assumption on local solvers other than stated above, it is possible to determine a variational formulation for the trace variable λ. Several auxiliary equalities used in the derivation are stated first. It can be shown that:

$$(\boldsymbol{Q}_m + \nabla U_m, \boldsymbol{v})_{\mathcal{T}_h} = \langle 1, [\![(U_m - m)\boldsymbol{v}]\!] \rangle_{\mathcal{E}_h} \tag{52}$$

$$(\nabla \cdot \boldsymbol{Q}_m + U_m, w)_{\mathcal{T}_h} = -\left\langle 1, [\![w(\widehat{\boldsymbol{Q}}_m - \boldsymbol{Q}_m)]\!] \right\rangle_{\mathcal{E}_h} \tag{53}$$

$$(\boldsymbol{Q}_f + \nabla U_f, \boldsymbol{v})_{\mathcal{T}_h} = \langle 1, [\![U_f\, \boldsymbol{v}]\!] \rangle_{\mathcal{E}_h} \tag{54}$$

$$(\nabla \cdot \boldsymbol{Q}_f + U_f - f, w)_{\mathcal{T}_h} = -\left\langle 1, [\![w(\widehat{\boldsymbol{Q}}_f - \boldsymbol{Q}_f)]\!] \right\rangle_{\mathcal{E}_h} \tag{55}$$

More detailed derivation:

1. The local solver satisfies on each element K

$$\begin{aligned} (\boldsymbol{Q}_m, \boldsymbol{v})_K - (U_m, \nabla \cdot \boldsymbol{v})_K &= -\langle m, \boldsymbol{v} \cdot \boldsymbol{n} \rangle_{\partial K} \\ (\boldsymbol{Q}_m, \boldsymbol{v})_K - \langle U_m, \boldsymbol{v} \cdot \boldsymbol{n} \rangle_{\partial K} + (\nabla U_m, \boldsymbol{v})_K &= -\langle m, \boldsymbol{v} \cdot \boldsymbol{n} \rangle_{\partial K} \\ (\boldsymbol{Q}_m, \boldsymbol{v})_K + (\nabla U_m, \boldsymbol{v})_K &= +\langle U_m - m, \boldsymbol{v} \cdot \boldsymbol{n} \rangle_{\partial K} \end{aligned}$$

And after summing over all elements $K \in \mathcal{T}_h$, we have

$$(\boldsymbol{Q}_m + \nabla U_m, \boldsymbol{v})_{\mathcal{T}_h} = \langle 1, [\![(U_m - m)\boldsymbol{v}]\!] \rangle_{\mathcal{E}_h}$$

2. The second condition on local solver $(U_m, \boldsymbol{Q}_m)$ states that

$$-(\nabla w, \boldsymbol{Q}_m)_K + \langle w, \widehat{\boldsymbol{Q}}_m \cdot \boldsymbol{n} \rangle_{\partial K} + (U_m, w)_K = 0$$

and after integrating by parts

$$-\langle w, \boldsymbol{Q}_m \cdot \boldsymbol{n} \rangle_{\partial K} + (w, \nabla \cdot \boldsymbol{Q}_m)_K + \langle w, \widehat{\boldsymbol{Q}}_m \cdot \boldsymbol{n} \rangle_{\partial K} + (U_m, w)_K = 0$$
$$(\nabla \cdot \boldsymbol{Q}_m + U_m, w)_K = -\left\langle w, (\widehat{\boldsymbol{Q}}_m - \boldsymbol{Q}_m) \cdot \boldsymbol{n} \right\rangle_{\partial K}$$

Finally, summation over all $K \in \mathcal{T}_h$ gives

$$(\nabla \cdot \boldsymbol{Q}_m + U_m, w)_{\mathcal{T}_h} = -\left\langle 1, [\![(\widehat{\boldsymbol{Q}}_m - \boldsymbol{Q}_m) w]\!] \right\rangle_{\mathcal{E}_h}.$$

3. For the second local solver $(U_f, \boldsymbol{Q}_f)$, we have

$$\begin{aligned}
(\boldsymbol{Q}_f, \boldsymbol{v})_K - (U_f, \nabla \cdot \boldsymbol{v})_K &= 0 \\
(\boldsymbol{Q}_f, \boldsymbol{v})_K - \langle U_f, \boldsymbol{v} \cdot \boldsymbol{n} \rangle_{\partial K} + (\nabla U_f, \boldsymbol{v})_K &= 0 \\
(\boldsymbol{Q}_f + \nabla U_f, \boldsymbol{v})_K &= \langle U_f, \boldsymbol{v} \cdot \boldsymbol{n} \rangle_{\partial K} \\
(\boldsymbol{Q}_f + \nabla U_f, \boldsymbol{v})_{\mathcal{T}_h} &= \langle 1, [\![U_f \boldsymbol{v}]\!] \rangle_{\mathcal{E}_h}
\end{aligned}$$

4. The second property of the local solver $(U_f, \boldsymbol{Q}_f)$ is $-(\nabla w, \boldsymbol{Q}_f)_K + \langle w, \hat{\boldsymbol{Q}}_f \cdot \boldsymbol{n} \rangle_{\partial K} + (U_f, w)_K = (f, w)_K$ and hence

$$-\langle w, \boldsymbol{Q}_f \cdot \boldsymbol{n} \rangle_{\partial K} + (w, \nabla \cdot \boldsymbol{Q}_f)_f + \langle w, \hat{\boldsymbol{Q}}_f \cdot \boldsymbol{n} \rangle_{\partial K} + (U_f, w)_K = (f, w)_K$$
$$(\nabla \cdot \boldsymbol{Q}_f + U_f - f, w)_K = -\left\langle w, (\widehat{\boldsymbol{Q}}_f - \boldsymbol{Q}_f) \cdot \boldsymbol{n} \right\rangle_{\partial K}$$

To write the global transmission condition, we will use several identities stated in the following lemma.

Lemma 1 (Elementary Identities)**.** *For any $m, \mu \in M_h$ and $f \in L^2(\Omega)$*

$$
\begin{aligned}
\textit{(i)} \qquad -\left\langle \mu, \left[\!\left[\widehat{\boldsymbol{Q}}_m \right]\!\right] \right\rangle_{\mathcal{E}_h} =& (\boldsymbol{Q}_m, \boldsymbol{Q}_\mu)_\Omega + (U_m, U_\mu)_\Omega \\
&+ \left\langle 1, \left[\!\left[(U_\mu - \mu)(\widehat{\boldsymbol{Q}}_m - \boldsymbol{Q}_m) \right]\!\right] \right\rangle_{\mathcal{E}_h}, \\
\textit{(ii)} \qquad -\left\langle \mu, \left[\!\left[\widehat{\boldsymbol{Q}}_{g_h} \right]\!\right] \right\rangle_{\mathcal{E}_h} =& -\left\langle g_h, \left[\!\left[\widehat{\boldsymbol{Q}}_\mu \right]\!\right] \right\rangle_{\mathcal{E}_h} \\
&+ \left\langle 1, \left[\!\left[(U_\mu - \mu)(\widehat{\boldsymbol{Q}}_{g_h} - \boldsymbol{Q}_{g_h}) \right]\!\right] \right\rangle_{\mathcal{E}_h} \\
&- \left\langle 1, \left[\!\left[(U_{g_h} - g_h)(\widehat{\boldsymbol{Q}}_\mu - \boldsymbol{Q}_\mu) \right]\!\right] \right\rangle_{\mathcal{E}_h}, \\
\textit{(iii)} \qquad -\left\langle \mu, \left[\!\left[\widehat{\boldsymbol{Q}}_f \right]\!\right] \right\rangle_{\mathcal{E}_h} =& -(f, U_\mu)_{\mathcal{T}_h} \\
&+ \left\langle 1, \left[\!\left[(U_\mu - \mu)(\widehat{\boldsymbol{Q}}_f - \boldsymbol{Q}_f) \right]\!\right] \right\rangle_{\mathcal{E}_h} \\
&- \left\langle 1, \left[\!\left[U_f(\widehat{\boldsymbol{Q}}_\mu - \boldsymbol{Q}_\mu) \right]\!\right] \right\rangle_{\mathcal{E}_h}.
\end{aligned}
$$

Proof. Recall that the weak form of local solvers is

$$(\boldsymbol{Q}_m, \boldsymbol{v})_K - (U_m, \nabla \cdot \boldsymbol{v})_K = -\langle m, \boldsymbol{v} \cdot \boldsymbol{n} \rangle_{\partial K} \tag{56}$$

$$-(\nabla w, \boldsymbol{Q}_m)_K + \left\langle w, \widehat{\boldsymbol{Q}}_m \cdot \boldsymbol{n} \right\rangle_{\partial K} + (U_m, w)_K = 0 \tag{57}$$

and

$$(\boldsymbol{Q}_f, \boldsymbol{v})_K - (U_f, \nabla \cdot \boldsymbol{v})_K = 0 \tag{58}$$

$$-(\nabla w, \boldsymbol{Q}_f)_K + \left\langle \boldsymbol{v}, \widehat{\boldsymbol{Q}}_f \cdot \boldsymbol{n} \right\rangle_{\partial K} + (U_f, w)_K = (f, w)_K \tag{59}$$

For identity (i) we have

$$
\begin{aligned}
-\left\langle \mu, \left[\!\left[\widehat{\boldsymbol{Q}}_m \right]\!\right] \right\rangle_{\mathcal{E}_h} &= -\langle \mu, [\![\boldsymbol{Q}_m]\!] \rangle_{\mathcal{E}_h} - \left\langle \mu, \left[\!\left[(\widehat{\boldsymbol{Q}}_m - \boldsymbol{Q}) \right]\!\right] \right\rangle_{\mathcal{E}_h} \\
&= \Big[\text{Note that due to (56), } (\boldsymbol{Q}_\mu, \boldsymbol{Q}_m)_K - (U_\mu, \nabla \cdot \boldsymbol{Q}_m)_K = -\langle \mu, \boldsymbol{Q}_m \cdot \boldsymbol{n} \rangle_{\partial K} \Big] \\
&= (\boldsymbol{Q}_\mu, \boldsymbol{Q}_m)_{\mathcal{T}_h} - (U_\mu, \nabla \cdot \boldsymbol{Q}_m)_{\mathcal{T}_h} - \left\langle \mu, \left[\!\left[(\widehat{\boldsymbol{Q}}_m - \boldsymbol{Q}) \right]\!\right] \right\rangle_{\mathcal{E}_h} \\
&= \Big[\text{Use eq. (53) to substitute for } (U_\mu, \nabla \cdot \boldsymbol{Q}_m)_{\mathcal{T}_h} \Big] \\
&= (\boldsymbol{Q}_\mu, \boldsymbol{Q}_m)_{\mathcal{T}_h} + (U_m, U_\mu)_{\mathcal{T}_h} \\
&+ \left\langle 1, \left[\!\left[U_\mu (\widehat{\boldsymbol{Q}}_m - \boldsymbol{Q}) \right]\!\right] \right\rangle_{\mathcal{E}_h} \\
&- \left\langle \mu, \left[\!\left[(\widehat{\boldsymbol{Q}}_m - \boldsymbol{Q}) \right]\!\right] \right\rangle_{\mathcal{E}_h}
\end{aligned}
$$

To prove identity (ii), note that the bilinear form

$$B(m,\mu) = \left\langle \mu, \left[\!\left[\widehat{\boldsymbol{Q}}_m \right]\!\right] \right\rangle_{\mathcal{E}_h} + \left\langle 1, \left[\!\left[(U_\mu - \mu)\big(\widehat{\boldsymbol{Q}}_m - \boldsymbol{Q}_m\big) \right]\!\right] \right\rangle_{\mathcal{E}_h}$$

is symmetric due to the proven result (i) of this lemma. The identity (ii) then follows from $B(\mu, g_h) = B(g_h, \mu)$. The identity (iii) can be written as

$$\begin{aligned}
-\left\langle \mu, \left[\!\left[\widehat{\boldsymbol{Q}}_f \right]\!\right] \right\rangle_{\mathcal{E}_h} &= -\left\langle \mu, [\![\boldsymbol{Q}_f]\!] \right\rangle_{\mathcal{E}_h} - \left\langle \mu, \left[\!\left[(\widehat{\boldsymbol{Q}}_f - \boldsymbol{Q}_f) \right]\!\right] \right\rangle_{\mathcal{E}_h} \\
&= \Big[\text{Note that due to (56), } (\boldsymbol{Q}_\mu, \boldsymbol{Q}_f)_K - (U_\mu, \nabla\cdot\boldsymbol{Q}_f)_K = -\left\langle \mu, \boldsymbol{Q}_f\cdot\boldsymbol{n} \right\rangle_{\partial K}\Big] \\
&= (\boldsymbol{Q}_\mu, \boldsymbol{Q}_f)_{\mathcal{T}_h} - (U_\mu, \nabla\cdot\boldsymbol{Q}_f)_{\mathcal{T}_h} - \left\langle \mu, \left[\!\left[(\widehat{\boldsymbol{Q}}_f - \boldsymbol{Q}_f) \right]\!\right] \right\rangle_{\mathcal{E}_h} \\
&\Big[\text{Use identity (55)}\Big] \\
&= (\boldsymbol{Q}_\mu, \boldsymbol{Q}_f)_{\mathcal{T}_h} - (U_\mu, \nabla\cdot\boldsymbol{Q}_f)_{\mathcal{T}_h} + \Big\{ (\nabla\cdot\boldsymbol{Q}_f, U_\mu)_{\mathcal{T}_h} + (U_f, U_\mu)_{\mathcal{T}_h} - (f, U_\mu)_{\mathcal{T}_h} \Big\} \\
&+ \left\langle 1, \left[\!\left[(U_\mu - \mu)(\widehat{\boldsymbol{Q}}_f - \boldsymbol{Q}_f) \right]\!\right] \right\rangle \\
&\Big[\text{Use local solver (58) for the first term}\Big] \\
&= \underline{(\nabla\cdot\boldsymbol{Q}_\mu, U_f)_{\mathcal{T}_h}} + \underline{(U_f, U_\mu)_{\mathcal{T}_h}} - (f, U_\mu)_{\mathcal{T}_h} \\
&+ \left\langle 1, \left[\!\left[(U_\mu - \mu)(\widehat{\boldsymbol{Q}}_f - \boldsymbol{Q}_f) \right]\!\right] \right\rangle \\
&\Big[\text{Apply identity (53) for the underlined terms}\Big] \\
&= -(f, U_\mu)_{\mathcal{T}_h} + \left\langle 1, \left[\!\left[(U_\mu - \mu)(\widehat{\boldsymbol{Q}}_f - \boldsymbol{Q}_f) \right]\!\right] \right\rangle_{\mathcal{E}_h} - \left\langle 1, \left[\!\left[u_f(\widehat{\boldsymbol{Q}}_m - \boldsymbol{Q}_m) \right]\!\right] \right\rangle_{\mathcal{E}_h}
\end{aligned}$$

□

Theorem 1. *$\lambda_h \in M_h$ satisfies the conservativity condition*

$$\left\langle \mu, \left[\!\left[\widehat{\boldsymbol{Q}}_{\lambda_h} + \widehat{\boldsymbol{Q}}_{g_h} + \widehat{\boldsymbol{Q}}_f \right]\!\right] \right\rangle_{\mathcal{E}_h} = 0 \qquad \forall \mu \in M_h$$

if and only if

$$a_h(\lambda_h, \mu) = b_h(\mu) \quad \forall \mu \in M_h, \ \textit{where}$$

$$\begin{aligned}
a_h(\lambda_h, \mu) &= (\boldsymbol{Q}_\eta, \boldsymbol{Q}_\mu)_{\mathcal{T}_h} + (U_\eta, U_\mu)_{\mathcal{T}_h} + \left\langle 1, \left[\!\left[(U_\mu - \mu)(\widehat{\boldsymbol{Q}}_\eta - \boldsymbol{Q}_\eta) \right]\!\right] \right\rangle_{\mathcal{E}_h} \\
b_h(\mu) &= \left\langle g_h, \left[\!\left[\widehat{\boldsymbol{Q}}_\mu \right]\!\right] \right\rangle_{\mathcal{E}_h} + (f, U_\mu)_{\mathcal{T}_h} - \left\langle 1, \left[\!\left[(U_\mu - \mu)\big(\widehat{\boldsymbol{Q}}_f - \boldsymbol{Q}_f\big) \right]\!\right] \right\rangle_{\mathcal{E}_h} \\
&\quad + \left\langle 1, \left[\!\left[U_f\big(\widehat{\boldsymbol{Q}}_\mu - \boldsymbol{Q}_\mu\big) \right]\!\right] \right\rangle_{\mathcal{E}_h} \\
&\quad - \left\langle 1, \left[\!\left[(U_\mu - \mu)\big(\widehat{\boldsymbol{Q}}_{g_h} - \boldsymbol{Q}_{g_h}\big) \right]\!\right] \right\rangle_{\mathcal{E}_h} \\
&\quad + \left\langle 1, \left[\!\left[(U_{g_h} - g)\big(\widehat{\boldsymbol{Q}}_\mu - \boldsymbol{Q}_\mu\big) \right]\!\right] \right\rangle_{\mathcal{E}_h}
\end{aligned}$$

for all $\mu, \eta \in M_h$.

Proof. The proof is a direct consequence of the previous lemma. □

The weak form for λ in the case of continuous Galerkin method reduces to

$$\left(\nabla U_\eta, \nabla U_\mu\right)_{\mathcal{T}_h} + (U_\eta, U_\mu)_{\mathcal{T}_h} = (f, U_\mu)_{\mathcal{T}_h} + \left\langle g_h, [\![\hat{\boldsymbol{Q}}_\mu]\!] \right\rangle \tag{60}$$

4.5 Summary of progress and outlook

The preceding sections outline our efforts in establishing a fairly complex initial formulation of this joint CG-HDG method, which represents a significant portion of our manpower expended on this task to date. This effort lies solely within tasks 1.3.1 (*development of a mixed CG-HDG formulation within the Nektar++ framework*) and 1.3.2 (*testing and verification of the formulation*) as described in the description of work. From an implementation perspective, the majority of the intra-node CG code is now complete. We have extended our implementation of the CG method, within our spectral/*hp* element framework *Nektar++* [5], to include the implementation of the weak Dirichlet boundary condition imposed by equation (48). Formal testing for convergence of this is now anticipated to be done in the coming weeks. Once this is completed, we then aim to finalise the formulation of the intra-node HDG global solver terms and expedite the implementation of this in order to perform initial tests of the CG-HDG method in two dimensions.

Looking ahead, once tests of the CG-HDG method have been completed, the solution on both the coarse HDG space and the CG nodes still relies on an iterative method such as the conjugate gradient method. We will therefore be investigating methods for the efficient preconditioning of these systems, such as those based on an additive Schwarz technique, as part of task 1.3.3 (*coarse- and fine-scale preconditioning strategies*). Part of this work has already started in investigating multi-level preconditioners developed by EPFL on our existing HDG formulation. We additionally hope to investigate the use of algebraic multigrid solvers being developed by KTH as part of task 1.1. Finally, we will explore the mapping of the solver technique to the hardware heterogeneity as part of task 1.3.4 (*investigate HDG-CG partitioning onto interconnect topology*), which will explore the range of discretization strategies and their performance on multi-cpu clusters with hierarchical dragonfly topology in conjunction with tasks 1.1.1.a and 1.2.4.

5 Task 1.4: Data reduction (USTUTT)

5.1 Introduction

The steady increase of available computer resources has enabled engineers and scientists to use more and more complex models to simulate a myriad of fluid flow problems. While modern high performance computers (HPC) have seen a steady growth in computing power, this trend, however, 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, but the overall simulations are limited by how fast the system can read and write data.

Considering Computational Fluid Dynamics (CFD), simulations discretize the flow field by a large number of data points and represent the flow by a collection of scalar and vector fields. The move to exascale performance and finer mesh resolutions will consequently only exacerbate this problem.

One of the major pitfalls of storing 'raw' simulation results lies in the implicit and redundant manner in which it represents the flow physics. Thus transforming the large 'raw' to compact feature- or structure-based data could help to overcome this bottleneck and significantly boost the performance and efficiency of HPC systems today and in the near future [32, 33].

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. The basic approach is similar to techniques utilized for still (JPEG, JPEG-2000) and motion image (H.264) compression. Similar algorithms have been used in the medical field to significantly reduce the amount of unsteady data produced by modern, three-dimensional diagnostic imaging systems [3]. We will present first results regarding these techniques in section 5.2.

In a second part, we will evaluate existing methods, like wavelets, 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 Comparative study on compression techniques

Our first objective was to compare the performance of two of the most widely used compression techniques – namely the discrete cosine and wavelet transform – in terms of compression-induced error at a prescribed bit rate. For testing the different algorithms we used the data set from a numerical simulation of a flat plate flow at $Re_{\theta,0} = 300$, where θ denotes the boundary layer momentum thickness and the index 0 indicates flow properties at the inlet of the simulation domain. The spatial resolution of the numerical grid was set to 97x123x400 nodes in wall-normal, spanwise and streamwise directions respectively.

5.2.1 Methodology

To prepare the three-dimensional floating point arrays for the compression algorithms, each variable was normalized in such a way that its range is between 0 and 255. Each array is then rounded up to the nearest integer and mapped into an array of unsigned 8-bit integers. To conduct the comparative study, the three-dimensional data sets were then processed at a prescribed rate of 0.7 bits per pixel using the still and motion image codecs described

in Section 5.2.2. The three metrics used to analyze the image compression schemes are the maximum (err_{max}), the average (err_{ave}) and the mean square error (mse), which are evaluated as follows:

$$err_{max} = \max |I(x,y,z) - I'(x,y,z)| \tag{61}$$

$$err_{ave} = \frac{1}{ijk} \sum_{x=1}^{i} \sum_{y=1}^{j} \sum_{z=1}^{k} |I(x,y,z) - I'(x,y,z)| \tag{62}$$

$$mse = \frac{1}{ijk} \sum_{x=1}^{i} \sum_{y=1}^{j} \sum_{z=1}^{k} |I(x,y,z) - I'(x,y,z)|^2 , \tag{63}$$

where $I(x,y,z)$ is the original, $I'(x,y,z)$ the decompressed image and i,j,k the dimensions of the volumetric data set.

5.2.2 Compression Algorithms

In this study we used the JPEG and JPEG-2000 codecs to process streamwise cross-sections of the simulation data as a group of 400 two-dimensional images. Additionally, the JP3D standard was applied to compress the three-dimensional arrays in a singe file. Finally, the Motion JPEG-2000 scheme was employed to interpret the cross-sections of the computational grid as a temporal sequence of video frames.

The JPEG Algorithm. The JPEG (Joint Photographic Experts Group) standard is a popular method used for lossy compression of 8-bit color and greyscale images. In its more common form, the algorithm uses a discrete cosine transform (DCT) to transform the signal information to spectral space. Depending on the given degree of compression, the algorithm then truncates DCT coefficients associated with high frequency image components. Finally, the compression procedure stores the remaining coefficients using a Huffman-Coding algorithm that exploits redundancies in the image information [33].

The JPEG-2000 algorithm. JPEG 2000 (JP2) is a compression standard used to store images of any bit depth and colour space (i.e. 16-bit greyscale images). In contrast to the JPEG standard it is based on the discrete wavelet transformed (DWT) that can be performed by either the reversible Le Gall-(5,3) taps filter for lossless or the non reversible Daubechies-(9,7) tabs filter for lossy coding. Furthermore, JPEG2000's volumetric extension (JP3D) extents the same capabilities to three-dimensional data sets. Following the transformation to spectral space the DWT coefficients are then truncated during a quantization step and further compressed by means of Embedded Block Coding with Optimized Truncation (EBCOT). [3, 33]

The Motion JPEG-2000 Algorithm. Motion JPEG-2000 (MJ2) is a digital video sequence that is made up of a series of still, complete images compressed with the JPEG-2000

standard. Unlike the more commonly used MPEG-4 codec, which employs inter-frame coding techniques to compress video sequences temporally as well as spatially, MJ2 is more resilient to error propagation, more scalable, better suited for network environments and allows for random access of single images. Furthermore the intra-frame-only compression is insensitive to complex motion, at the expense of increased storage and bandwidth requirements. [23]

5.2.3 Results

Figure 3 shows the compression ratio, Figure 4 the maximum-, average and mean square error for the JPEG, JPEG-2000, JP3D and Motion JPEG-2000 compression codecs. In general we can conclude that JP2 offers a superior compression rate to error ratio when compared to the more common JPEG standard. Exploiting the three-dimensional

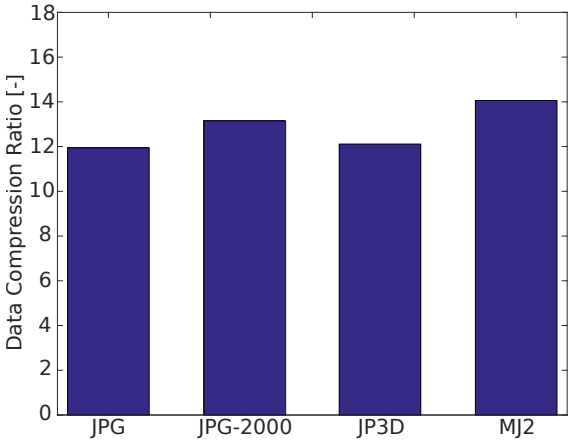


Figure 3: Data Compression Ratio for different still and motion image compression schemes.

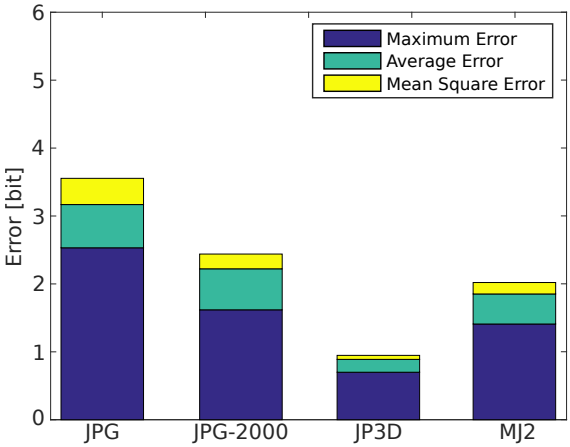


Figure 4: Maximum-, Average- and Mean Square Error for different still and motion image compression schemes.

redundancies in the image signal with JP3D proved to be the best compression technique with only minimal introduction of pixel error at a comparable bit rate. The processor time spent for compression was found to be moderate with 4 seconds for the JPEG codec, while JP2 took almost four times the processing time. JP3D and MJ2, on the other hand, exhibited only a marginal increase over JPEG's compression time.

5.2.4 Summary of progress and outlook

In this section we described a comparative study of lossy compression schemes based on the Discrete Cosine (DCT) and the Discrete Wavelet Transform (DWT). Comparison between these two compression techniques was done using a three-dimensional data set from a numerical simulation of a flat plate flow. Streamwise cross-sections of the plate flow were processed as two-dimensional still images using the JPEG and JPEG-2000 codecs as well as a temporal sequence of video frames using the Motion JPEG-2000 scheme. Furthermore, the JP3D standard was employed to compress the entire vector field into a single file. Based on the results

of this study we found that JPEG-2000, with its volumetric extension (JP3D), enables us to compress three-dimensional CFD data with minimal error and moderate compression time.

Looking ahead, the tested JP3D standard is still only defined to operate on integer arrays and the required truncation of the simulation data is unacceptable due to the irreversible data loss. We will therefore investigate extensions to the JPEG-2000 standard which allow for the compression of floating point data. We additionally hope to investigate the use of Contourlets [10] that allow for the efficient transformation of images with smooth regions separated by smooth boundaries. Finally, we will explore modern inter-frame coding techniques [3] to exploit structural flow field redundancies.

5.3 SVD

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 k in the following form:

$$A = U\Sigma V^T, \Sigma = diag(\sigma_1, \sigma_2, \ldots, \sigma_k) \tag{64}$$

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 $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k \geq 0$. An equivalent way of writing is:

$$A = \sum u_k \sigma_k v_k^T. \tag{65}$$

We can also find the fundamental theorem of SVD as shown in Figure 5 . The vectors u_k of an orthonormal U, called the left singular vectors, are the eigenvectors of AA^T with the associated eigenvalues σ_k. The vectors v_k of orthonormal V, called the right singular vectors, are the eigenvectors of A^TA with the same associated eigenvalues σ_k.

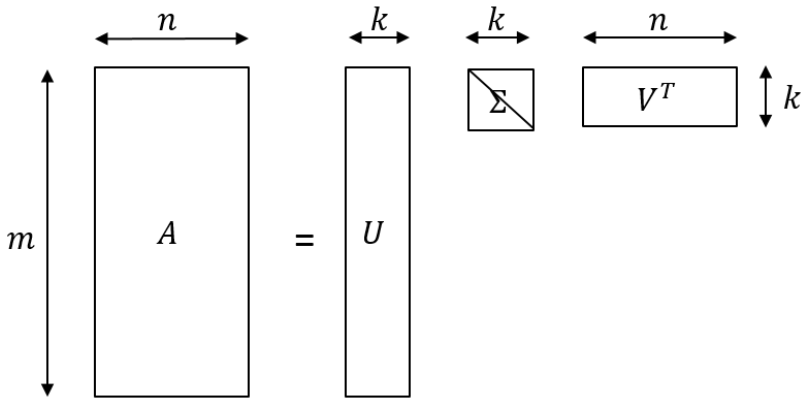


Figure 5: The form of SVD

Assume that we want to represent an extremely large 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

$$A^T = V\Sigma^T U^T. \tag{66}$$

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

$$A^TA = V\Sigma^2 V^T. \tag{67}$$

The formulation is shown in Figure 6. 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.

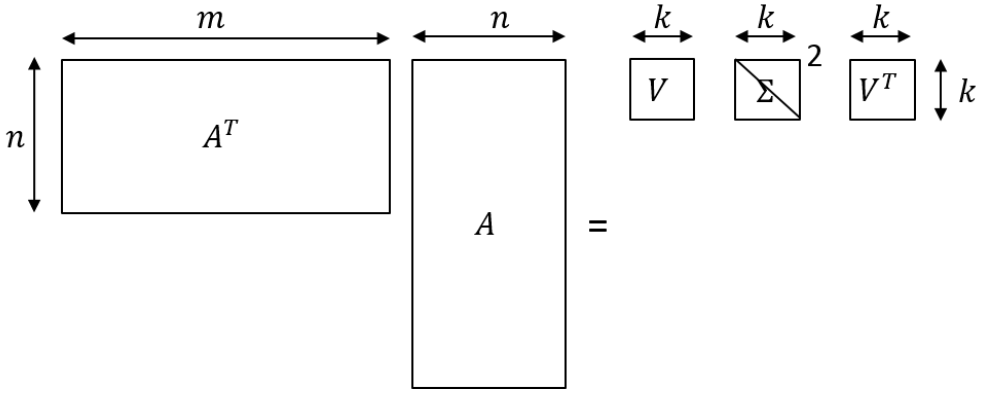


Figure 6: The SVD of matrices A^TA

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.

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 ExaFLOW's use cases.

5.4 Dynamic Mode Decomposition

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 so called Dynamic Mode Decomposition (DMD) of Peter Schmid. This DMD turned out to decompose a time signal into a linear combination of different modes multiplied by the k-th power of a complex value for getting the k-th time step. Williams, Kevrekides and Rowley [36] generalized the approach to Extended DMD and gave a relation to the Koopman operator [18], which again gives the opportunity to apply techniques of the functional analytic ergodic theory (see [11] and [4]). 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 a 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 is a compact topological space and φ is a continuous nonlinear operator

$$\varphi : K \longrightarrow K \tag{68}$$

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 $\mathcal{F} \subset C(K)$ is a linear subspace of "**observables**" with the stability property

$$f \in \mathcal{F} \Rightarrow f \circ \varphi \in \mathcal{F} \tag{69}$$

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

$$T_\varphi : \mathcal{F} \longrightarrow \mathcal{F} \tag{70}$$

$$f \mapsto T_\varphi f = f \circ \varphi \tag{71}$$

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**

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

for some constant λ. It might be a problem to interpret this equation in terms of a physical phenomenon.

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

$$G^f(q) = \left[g_k^f(q)\right]_{k=0,\cdots,n} = \left[f\left(\varphi^k q\right)\right]_{k=0,\cdots,n} \tag{72}$$

for a single observable f 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

$$c * b(\lambda) = c(\lambda)\, b(\lambda) \quad \forall\, \lambda \in \mathbb{C}. \tag{73}$$

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

$$\mathfrak{A}(c) = \mathfrak{A}_n(c) = \begin{array}{c} \\ 0 \\ 1 \\ . \\ . \\ p \\ . \\ n \end{array} \begin{array}{c} \begin{array}{ccc} 0 & & n-p \end{array} \\ \begin{pmatrix} c_0 & & \\ c_1 & \ddots & \\ . & & c_0 \\ . & & c_1 \\ c_p & & \vdots \\ & \ddots & \\ & & c_p \end{pmatrix} \end{array} \tag{74}$$

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 ($\deg b = n - p$)

$$\mathfrak{A}(c)\; b = c * b \tag{75}$$

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 be a polynom coefficient vector with with a degree deg $c = p$ not larger than the sequence. We multiply G with the convolution matrix

$$G \quad \mathfrak{A}_n(c) \;=\; R \tag{76}$$

$$\begin{bmatrix} g_0 & g_1 & g_2 & \cdots & g_n \end{bmatrix} \; \mathfrak{A}_n(c) \;=\; \begin{bmatrix} r_0 & r_1 & \cdots & r_{n-p} \end{bmatrix} \tag{77}$$

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. To analyse this, we decompose G in two parts related to c

$$G = G_{modes} \;+\; \Delta G \tag{78}$$

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

$$G_{modes}\; \mathfrak{A}(c) = 0 \tag{79}$$

It will described later. The second part defines the defect in equation (76) given by

$$\Delta G\; \mathfrak{A}(c) = R \tag{80}$$

There are some not unique but reasonable requirements in selecting ΔG. We restrict ΔG to be

$$\Delta G = R\;\; (\mathfrak{A}(c)^* \mathfrak{A}(c))^{-1}\, \mathfrak{A}(c)^* \tag{81}$$

With these assumptions we get by (76)

$$\Delta G = G\; Q \tag{82}$$

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

$$Q \;\; = \;\; \mathfrak{A}(c) \;\; (\mathfrak{A}(c)^* \mathfrak{A}(c))^{-1} \;\; \mathfrak{A}(c)^* \tag{83}$$

with the property

$$(I - Q) \;\; \mathfrak{A}(c) \;\; = \;\; 0. \tag{84}$$

For G_{modes} we have by (84)

$$G_{modes} = G - \Delta G = G - GQ = G\,(I - Q) \tag{85}$$

and therefore

$$0 = G_{modes}\; \mathfrak{A}(c) = \begin{bmatrix} \tilde{g}_0 & \tilde{g}_1 & \ldots & \tilde{g}_n \end{bmatrix} \; \mathfrak{A}(c) \tag{86}$$

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

$$G_{modes} = \begin{bmatrix} \tilde{g}_0 & \tilde{g}_1 & \dots & \tilde{g}_n \end{bmatrix} = \sum_{l=1}^{\mathbf{p}} v_l \left[1, \lambda_l, \lambda_l^2, \dots, \lambda_l^{\mathbf{n}}\right] \tag{87}$$

with the modes

$$v_l = G_{modes} \, \frac{1}{w_l\left(\lambda_l\right)} \, \begin{bmatrix} w_l \\ 0 \end{bmatrix} \tag{88}$$

given by the polynom

$$c\left(\lambda\right) = \left(\lambda - \lambda_l\right) w_l\left(\lambda\right) \quad \forall \, \lambda \in \mathbb{C} \quad \text{or} \quad c = w_l * \begin{bmatrix} -\lambda_l \\ 1 \end{bmatrix} \tag{89}$$

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

$$\Delta G^* \Delta G \leq \mu \; I \tag{90}$$

or by (82)

$$Q^* \; H \; Q \leq \mu I \tag{91}$$

with the covariance matrix $H = G^T G$ and the projection Q in (83). Because $\mathfrak{A}(c)$ has full rank, this is equivalent to find a minimum $\mu > 0$ fulfilling

$$\mathfrak{A}(c)^* \; H \; \mathfrak{A}(c) \leq \mu \; \mathfrak{A}(c)^* \; \mathfrak{A}(c) \tag{92}$$

we summarize

Theorem 2. *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*

$$G = G_{modes} + \Delta G \tag{93}$$

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 (81) we have $\|\Delta G\|_2 \leq \sqrt{\mu}$ iff

$$\mathfrak{A}(c)^* \; H \; \mathfrak{A}(c) \leq \mu \; \mathfrak{A}(c)^* \; \mathfrak{A}(c) \tag{94}$$

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

$$c = w_l * \begin{bmatrix} -\lambda_l \\ 1 \end{bmatrix} \tag{95}$$

and the part of modes

$$G_{modes} = \sum_{l=1}^{p} v_l \left[1, \lambda_l, \lambda_l^2, \ldots, \lambda_l^n\right] \tag{96}$$

The complex vectors $v_l(q) = \left(v_l^f(q)\right)_{f \in S}$ *are named Koopman modes ([4]).*

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 [34] 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 $G^j = \begin{bmatrix} g_{0+j} & g_{1+j} & \ldots & g_{n+j} \end{bmatrix}$

$$\frac{1}{n-p+1} \sum_{j=0}^{n-p} \left\| G^j c \right\|^2 \leq \frac{\mu}{n-p+1} \sum_{j=0}^{n-p} \|c\|^2 \;=\; \mu \;\|c\|^2 \tag{97}$$

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

$$H^{n-p} = \frac{1}{n-p+1} \sum_{j=0}^{n-p} \left(G^j\right)^T G^j \tag{98}$$

we have

$$< H^{n-p}\, c\,,\; c > \leq \;\mu\; \| \; c \; \|^2 \tag{99}$$

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}

$$H^{n-p}\, c \;=\; \mu_{min}\, c \tag{100}$$

μ_{min} underestimates μ in (94) 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 (94). It might introduce additional unwanted eigenvalues.

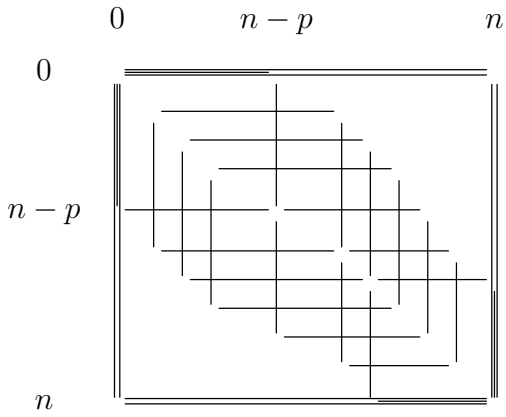


Figure 7: shifted submatrices as part of the total matrix

5.4.3 Ensembles

Summing up matrices of type H in (94) allows to handle all matrices together by a common polynom coefficient vector c.

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

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 (96)

$$\mathbb{N}_0 \ni k \mapsto g_k(q) = \sum_{l=1}^{p} v_l(q) \;\; {\lambda_l}^k \tag{101}$$

allows us to calculate Koopman eigenfunctions for this approximative sequence. Remember $g^f_k(q) = f\left(\varphi^k q\right)$ in (72) 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 (72) by stacking $(0:p)$ subsequent elements leads to

$$\begin{bmatrix} g_k(q) \\ g_{k+1}(q) \\ \vdots \\ g_{k+p}(q) \end{bmatrix} = \sum_{l=1}^{p} \begin{bmatrix} v_l(q) & \lambda_l^0 \\ v_l(q) & \lambda_l^1 \\ & \vdots \\ v_l(q) & \lambda_l^p \end{bmatrix} {\lambda_l}^k = \sum_{l=1}^{p} v_l(q) \begin{bmatrix} \lambda_l^0 \\ \lambda_l^1 \\ \vdots \\ \lambda_l^p \end{bmatrix} {\lambda_l}^k \tag{102}$$

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 (95) 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

$$u_i^* \begin{bmatrix} g_k(q) \\ g_{k+1}(q) \\ \vdots \\ g_{k+p}(q) \end{bmatrix} = \sum_{l=1}^{p} d_i^* v_l(q) \frac{w_i(\lambda_l)}{w_i(\lambda_i)} {\lambda_l}^k = d_i^* v_i(q) {\lambda_i}^k \tag{103}$$

Returning back to the definition (72) of $g_k^f(q) = f\left(\varphi^k q\right)$

$$u_i^* \begin{bmatrix} f\left(\varphi^k \circ \varphi\ q\right) \\ f\left(\varphi^{k+1} \circ \varphi\ q\right) \\ \vdots \\ f\left(\varphi^{k+p} \circ \varphi\ q\right) \end{bmatrix} = d_i^* v_i(q) {\lambda_i}^{k+1} = \lambda_i u_i^* \begin{bmatrix} f\left(\varphi^k q\right) \\ f\left(\varphi^{k+1} q\right) \\ \vdots \\ f\left(\varphi^{k+p} q\right) \end{bmatrix} \tag{104}$$

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. Remarkably, 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 too 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 Summary

We have described algorithms and their mathematical background generalizing the Dynamic Modes Decomposition of ([34]) 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.

6 Task 1.5: Fault tolerance and resilience (EPFL)

6.1 Overview

Resilience to faults has been identified as being critical for future exascale HPC systems. The techniques needed to achieve a thousand fold increase in computational capacity expected over the next decade, are predicted to also increase the rate of faults on large systems. This poses substantial new challenges in terms of how to effectively use the machines, and how to assess and assure the correctness of numerical simulation results.

Solving time dependent PDEs is often done in a methods-of-line approach where spatial derivatives are discretized in some appropriate manner to create a system of coupled ODEs to be integrated in time. The approach extends trivially to distributed memory machines by application of domain decomposition, letting adjacent nodes communicate boundary information between time-steps. A key limitation of this approach lies in the strong scaling limit. As spatial sub-domains decrease in size, nodes will increasingly be spending time on communicating boundary information rather than computing. With the large machines comprising thousands of nodes available to research today, this is a substantial bottleneck in scaling applications efficiently, clearly new algorithmic developments are required.

A potential path to increased parallelism in the solution procedure is to attempt parallel time-integration. Through the methods-of-lines approach this problem is traditionally viewed as a sequential process. However, various attempts of extracting parallelism in this otherwise traditionally sequential procedure do exist. These algorithms borrows ideas from spatial domain decomposition to construct an iterative approach for solving the temporal problem in a parallel global-in-time approach.

In the context of parallel integration techniques, the issue of algorithmic resilience is of particular relevance since methods of parallel integration are developed primarily with the focus of extracting parallelism in the solution of PDEs beyond what is possible using standard domain decomposition techniques. Challenges in addressing faults at exascale computing include faults caused by malfunctioning hardware. These are typically placed into two overall categories, soft and hard node errors. A significant source of soft errors arises from energetic particles interacting with the silicon subtract, causing either flipped states of a storage element or disruption of the operation of a combinational 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. Hard errors are faults that lead to the complete failure of a node. For current parallel applications based

on MPI, the approach for dealing with the loss of a process is to kill all remaining processes and restart the application at nearest check-point. This approach is costly as many modern clusters now scale to thousands of nodes, the I/O cost of a check-point/restart procedure may be prohibitively costly. Ideally, a local failure should permit local recovery. Unlike hard errors, soft errors have the potential to corrupt computer simulations in ways that may not be immediately obvious to the domain scientist or engineer relying on them as part of their work. The typically attitude towards SDC resilience is to assume that 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 questionable. This approach raises questions on the trustworthiness of numerical simulations performed. In addition it is worth noting that both the cost of an SDC induced re-run and the probability of needing such a re-run scales linearly with the size of the machine, therefore, this simple approach may not be acceptable on future exascale systems.

The goals of this effort withing the ExaFLOW effort can separated into a few categories:

1. Development an understanding of how to improve existing algorithms to enhance resilience to both hard and soft errors, including investigations of future support for fault detection and handling through MPI 4.0.

2. Development and analysis of fault detection and recovery in complex iterative solvers, with multiple right hand sides. This will serve as a prototype for the development of resilient linear and nonlinear solvers in complex PDE solvers such as those considered in ExaFLOW.

3. Investigation of sensitivities and critical resilience in complex PDE solvers, e.g., some information such as geometry information must clearly be safe while others parts of the algorithm such as the pre-convergence iterates are less critical as they can be re-generated. Having a thorough understanding of this is critical to propose an efficient strategy for ensuring resilience of a large scale production code.

6.2 Progress update

We have investigated fault tolerance and techniques for resilience within the context of parallel-in-time methods.

To present the method, consider the problem

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} + \mathcal{A}\left(t, \mathbf{u}\right) = 0 \\ \mathbf{u}\left(T_0\right) = \mathbf{u}_0 \quad t \in [T_0, T] \end{cases} \tag{105}$$

where $\mathcal{A} : \mathrm{R} \times V \to V'$ is a general operator depending on $\mathbf{u} : \Omega \times \mathrm{R}^+ \to V$ with V being a Hilbert space and V' its dual. Now, assume there exists a unique solution $\mathbf{u}\left(t\right)$ to (105) and decompose the time domain of interest into N individual time slices

$$T_0 < T_1 < \cdots < T_{N-1} < T_N = T. \tag{106}$$

Let $T_n = \Delta T n$ with $n \in \mathrm{N}$. We now define a numerically accurate solution operator $\mathcal{F}_{\Delta T}$ which for any $t > T_0$ advances the solution as

$$\mathcal{F}_{\Delta T}\left(T_n, \mathbf{u}\left(T_n\right)\right) = \mathbf{U}_{T_n+\Delta T} \approx \mathbf{u}\left(T_n + \Delta T\right) \tag{107}$$

To solve (105) on $[T_0, T_0 + N \cdot \Delta T]$ we define $M_{\mathcal{F}}$, $\bar{\mathrm{U}}$ and $\bar{\mathrm{U}}_0$ as operators on the form

$$M_{\mathcal{F}} = \begin{bmatrix} 1 & & & \\ -\mathcal{F}_{\Delta T}^{T_0} & \ddots & & \\ & \ddots & \ddots & \\ & & -\mathcal{F}_{\Delta T}^{T_{N-1}} & 1 \end{bmatrix} \tag{108}$$

with $\bar{\mathrm{U}} = [\mathrm{U}_0, \ldots, \mathrm{U}_N]$ and $\bar{\mathrm{U}}_0 = [u\left(T_0\right), 0, \ldots, 0]$. The sequential solution procedure is then equivalent to solving $M_{\mathcal{F}}\bar{\mathrm{U}} = \bar{\mathrm{U}}_0$ for $\bar{\mathrm{U}}$ to recover $\mathbf{U}_0 \cdots \mathbf{U}_N$ as approximations to $\mathbf{u}(T_0) \cdots \mathbf{u}(T_N)$ by forward substitution. The lower bi-diagonal nature of (108) express the explicit and local nature of the approach. If we instead seek to solve the system using a point-iterative approach i.e., we seek the solution on form $\bar{U}^{k+1} = \bar{U}^k + \left(\bar{\mathrm{U}}_0 - M_{\mathcal{F}}\bar{\mathbf{U}}^k\right)$, we observe that at the beginning of each iteration $\bar{\mathbf{U}}^k$ is known, allowing that in each iteration we may compute $\mathcal{F}_{\Delta T}^{T_1} \cdots \mathcal{F}_{\Delta T}^{T_N}$ on all intervals in parallel. Note that the computational complexity of every iteration is strictly larger than that of the sequential solution procedure, so reduced time to solution is possible only if the number of iterations k_{conv} needed for convergence is much smaller than the number of time sub-domains N.

A question that remains open is what would be an appropriate preconditioner to accelerate the iteration. A typical approach is to create an approximation $M_{\mathcal{G}} \approx M_{\mathcal{F}}$, where $M_{\mathcal{G}}$ is cheap to apply, hence allowing us to solve the preconditioned system $\left(M_{\mathcal{G}}\right)^{-1} M_{\mathcal{F}}\bar{\mathrm{U}} = \left(M_{\mathcal{G}}\right)^{-1} \bar{\mathrm{U}}_0$. In the case considered here, we can readily create such an $M_{\mathcal{G}}$ by defining a new operator $\mathcal{G}_{\Delta T}$ as

$$\mathcal{G}_{\Delta T}\left(T_n, \mathbf{u}\left(T_n\right)\right) = \mathbf{U}_{T_n+\Delta T} \approx \mathbf{u}\left(T_n + \Delta T\right) \tag{109}$$

and relax the requirements on the accuracy of $\mathcal{G}_{\Delta T}$, by using a coarser grid or a different numerical model. Solving the system using a standard preconditioned Richardson iterations, one recovers

$$\bar{\mathbf{U}}^{k+1} = \bar{\mathbf{U}}^k + \left(M_{\mathcal{G}}\right)^{-1}\left(\bar{\mathrm{U}}_0 - M_{\mathcal{F}}\bar{\mathbf{U}}^k\right) \tag{110}$$

We can write this as

$$\begin{bmatrix} 1 & & & \\ -\mathcal{G}_{\Delta T}^{T_0} & \ddots & & \\ & \ddots & \ddots & \\ & & -\mathcal{G}_{\Delta T}^{T_{N-1}} & 1 \end{bmatrix} \begin{bmatrix} \mathrm{U}_0^{k+1} \\ \mathrm{U}_1^{k+1} \\ \vdots \\ \mathrm{U}_N^{k+1} \end{bmatrix} = \begin{bmatrix} 1 & & & \\ \mathcal{F}_{\Delta T}^{T_0} - \mathcal{G}_{\Delta T}^{T_0} & \ddots & & \\ & \ddots & \ddots & \\ & & \mathcal{F}_{\Delta T}^{T_{N-1}} - \mathcal{G}_{\Delta T}^{T_{N-1}} & 1 \end{bmatrix} \tag{111}$$

to recover the Parareal algorithm in the form that it is typically presented

$$\mathbf{U}_{n+1}^{k+1} = \mathcal{G}_{\Delta T}^{T_n}\mathbf{U}_n^{k+1} + \mathcal{F}_{\Delta T}^{T_n}\mathbf{U}_n^{k} - \mathcal{G}_{\Delta T}^{T_n}\mathbf{U}_n^{k} \tag{112}$$

with $\mathbf{U}_{n+1}^0 = \mathcal{G}_{\Delta T}^{T_n}\mathbf{U}_n^0$ and $\mathbf{U}_0^k = \mathbf{u}(T_0)$. Other parallel-in-time methods may be derived by simply constructing a new preconditioner for the system (108). Unlike parallel RK methods, this class of time parallel methods have no inherent upper limits to the parallelism that may be extracted. In this paper, we focus the analysis and implementation on the Parareal algorithm where the preconditioner have the same lower bi-diagonal structure as the matrix $M_{\mathcal{F}}$. However, similar fault-tolerant implementations may be constructed for other fixed-point iteration type time-parallel domain-decomposition methods.

To enhance the resilience of the parareal method, we have pursued two different approached, aiming at hard and soft errors, respectively.

The Parareal correction (112) may be implemented in different ways. The simplest approach is to divide work into two phases; a purely sequential phase, computing $\mathbf{U}_{n+1}^{k+1}$ from $\mathcal{G}_{\Delta T}^{T_n}\mathbf{U}_n^{k+1}$ with the correction (112), and a parallel phase where $\mathcal{F}_{\Delta T}^{T_n}\mathbf{U}_n^{k}$ is computed in parallel on $n \in N$ nodes. Ideally, the wall-time $T_{\mathcal{G}}$ for a node group to compute $\mathcal{G}_{\Delta T}^{T_n}\mathbf{U}_n^{k+1}$ is much smaller than the wall-time $T_{\mathcal{F}}$ to compute $\mathcal{F}_{\Delta T}^{T_n}\mathbf{U}_n^{k}$, and the limiting factor in obtainable speed-up will be the number of iterations $k_{conv} < N$ needed for convergence. In practice however, it is seldom possible to construct a coarse operator $\mathcal{G}_{\Delta T}^{T_n}$ so computationally cheap that its cost may be ignored. Fortunately, there exists many other ways for scheduling the computational work than having two strictly separated phases, switching between computing $\mathcal{G}_{\Delta T}^{T_n}\mathbf{U}$ sequentially and $\mathcal{F}_{\Delta T}^{T_n}\mathbf{U}$ in parallel. For example, $\mathcal{G}_{\Delta T}^{T_0}\mathbf{U}_0^0$ and $\mathcal{F}_{\Delta T}^{T_0}\mathbf{U}_0^0$ may be computed concurrently. By exploiting such independencies, it is possible, to some extend, to mitigate the effects of a relatively expensive coarse operator $\mathcal{G}_{\Delta T}^{T_n}$.

We have developed a new scheduler that uses features of the UFLM MPI framework to build a fault tolerant algorithm. Here idle node-groups may be used as spares for the event that an active node-group fails.

We have extended the Parareal algorithm to make SDC resilience an integral part of the algorithm, regardless of the SDC resilience properties of the underlying operators $\mathcal{F}_{\triangle T}$ and $\mathcal{G}_{\triangle T}$. The new algorithm preserves the nice feature, that the algorithm may be wrapped around previous code, only with the possible need for the addition of an interpolation or projection to map between the spaces on which $\mathcal{F}_{\triangle T}$ and $\mathcal{G}_{\triangle T}$ operates.

In building an SDC resilient algorithm for iterative methods, it is natural to look at the difference between consecutive iterations to detect whether or not an SDC-type error occurred. As presented in the introduction, Parareal is in essence also a fixed point iteration, but with a non-normal iteration matrix, the elements of which are potentially non-linear operators. Since the upper bound on parallel efficiency of the algorithm scales as $1/k_{conv}$, we find it reasonable to assume that for any practical application, $\mathcal{G}_{\Delta T}^{T_n}$ is constructed sufficiently close to $\mathcal{F}_{\Delta T}^{T_n}$ so that the iteration matrix will remain a contraction on $\mathbf{U}_n^k$ from $k = 0$ and onwards. Due to the special structure of the iteration matrix (110), we may construct a local approach without the need for a synchronization between iterations. First, define the

residual between two consecutive iterations on the node-group local time sub-domain n as

$$e_n^{k+1} = \left\| \mathbf{U}_n^{k+1} - \mathbf{U}_n^k \right\|_\infty \tag{113}$$

For an SDC resilient model, the above e_n^{k+1} must be computed at iteration $k+1$ on each time sub-domain n, and communicated along with *converge*, so that the node-group responsible for the n'th time sub-domain at the $k+1$'th iteration can access $e_i^{k+1} \forall i \in 1 \dots n$. Then, if at any iteration for any time sub-domain

$$\max_{i=0\dots n} e_n^{k+1} \geq \beta \max_{i=0\dots n} e_n^k \tag{114}$$

is true, we reject $\mathbf{U}_n^{k+1}$ and replace it with

$$\mathbf{U}_n^{k+1} = \mathbf{U}_n^{k-1}, \quad e_n^{k+1} = e_n^{k-1} \tag{115}$$

where $\beta \leq 1$ is an upper bound to the contraction factor. If no upper bound is available, using $\beta = 1$ appears to work well. To avoid stagnation due to false rejection, we reject the previous two local iterates.

Extensive numerical examples confirm that the developed parareal approach, combining careful scheduling and algorithmic innovation, achieves substantial improvements in fault tolerance and resilience towards simulated error rates.

We refer to [28] for further details and numerical validation.

6.3 Outlook and future work

With the improved understanding of the MPI 4.0 supported features and ways to alter existing algorithms to gain resilience, we are now ready to pursue the development of resilient techniques for more complex iterative solvers, e.g., conjugate gradient solvers used within Nektar++.

We will focus on the solvers for the Poisson equation but with a right hand side that is simulated to be time-dependent - this mimics the situation in a time-dependent PDE solver. Several developments are required to ensure resilience in such a scenario. On one hand, we need a strategy to detect SDC errors - for this we will rely on local error estimators as an indicated of corruption. At the same time, we need an understanding of which pieces of the algorithm is most sensitive to corruption and guard these as well as establish a strategy for 'on-the-fly' regeneration of critical parts, e.g., through the development of reduced complexity models of the operators, solution and right hand side as needed.

This development will set the stage for transitioning the technology into Nektar++ and evaluate the resilience of the large scale production model to simulated faults and, ultimately, investigate fault tolerance on a leading computational platform.

7 Summary

The initial algorithmic developments made over the first part of the ExaFLOW project have made a good start in understanding the key parts of each task. Collaboration between partners is also now underway between various tasks, as described in the description of work:

- Coarse space preconditioners (tasks 1.1.1b and 1.3.3): KTH and ICL will attempt to evaluate the use of the KTH AMG developments in Nektar++ for unstructured grids. ICL and EPFL have also been communicating regarding the use of multi-level preconditioners for the HDG trace space system.

- Error estimation (tasks 1.1.3c and 1.2.1): SOTON will investigate the applicability of the spectral error estimation method used by KTH in their developments.

- p and r-refinement: ICL will investigate potential for using this technology in unstructured meshes alongside KTH's development.

- Resilience (task 1.5.3): ICL/EPFL will now start to look at development of algorithms for conjugate gradient solve, as mentioned in the task 1.5 summary. Representatives from EPFL and McLaren Racing Ltd recently visited ICL to discuss this topic in detail and devise potential strategies for the recovery from hard errors in the conjugate gradient solve, which is now undergoing development.

The focus of each of the partners is to now move towards more concrete implementations that can be undertaken in WP2 and then evaluated using the challenging test cases in WP3.

References

[1] Nek5000. `http://nek5000.mcs.anl.gov/`.

[2] Ravishankar Balasubramanian and James C. Newman. Adjoint-based error estimation and grid adaptation for functional outputs: Application to two-dimensional, inviscid, incompressible flows. *Computers & Fluids*, 2009.

[3] T. Bruylants, A. Munteanua, and P. Schelkensa. Wavelet based volumetric medical image compression. *Signal Processing: Image Communication*, 31:112–133, 2015.

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

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

[6] Vincenzo Citro. *Unsteady and three-dimensional fluid dynamic instabilities*. PhD thesis, University of Salerno, 2015.

[7] 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.

[8] David L. Darmofal and David A. Venditti. Grid adaptation for functional outputs: Application to two-dimensional inviscid flows. *Journal of Computational Physics*, 2002.

[9] Kaihua Ding, Krzysztof J. Fidkowski, and Philip L. Roe. Continuous adjoint based error estimation and r-refinement for the active-flux method. In *54th AIAA Aerospace Sciences Meeting*, 2016.

[10] M. N. Do and M. Vetterli. The contourlet transform: an efficient directional multiresolution image representation. *IEEE Transactions on Image Processing*, 14:2091–2106, 2005.

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

[12] D. Ekelschot, D. Moxey, S.J. Sherwin, and J. Peiro. A p-adaptation method for compressible flow problems using a goal-based error indicator. *Special Issue Computers & Structures (submitted)*, 2016.

[13] Kenneth Eriksson, Don Estep, Peter Hansbo, and Johnsonm Claes. Introduction to adaptive method for diffential equations. *Acta Numerica*, pages 1–54, 1995.

[14] K. J. Fidkowski and P. L. Roe. An entropy adjoint approach to mesh refinement. *SIAM Journal on Scientific Computing*, 2010.

[15] Krzysztof J. Fidkowski and Philip L. Roe. Entropy-based mesh refinement, i: The entropy adjoint approach. In *19th AIAA Computational Fluid Dynamics*, 2009.

[16] Michael B. Giles and Niles A. Pierce. An introduction to the adjoint approach to design. *Flow, Turbulence and Combustion*, 2000.

[17] 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.

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

[19] Gerald W. Kruse. *Parallel Nonconforming Spectral Element Solution of the Incompressible Navier-Stokes Equations in Three Dimensions*. PhD thesis, Brown University, 1997.

[20] Yi Li. *Automatic Mesh Adaptation Using the Continuous Adjoint Approach and the Spectral Difference Method*. PhD thesis, Stanford University, 2013.

[21] James William Lottes. *Towards Robust Algebraic Multigrid Methods for Nonsymmetric Problems*. PhD thesis, University of Oxford, 2015.

[22] Yvon Maday, Catherine Mavriplis, and Anthony Patera. Nonconforming mortar element methods: Application to spectral discretizations. In *Proceedings of the 2nd International Conference on Domain Decomposition Methods*, pages 392–418, 1988.

[23] D. Marpe, V. George, H. L. Cycon, and K. U. Barthel. Performance evaluation of Motion JPEG-2000 in comparison with H.264/AVC operated in pure intra coding mode. In *Wavelet Applications in Industrial Processing*, pages 129–137. SPIE Press, October 2003.

[24] Catherine Mavriplis. *Nonconforming Discretizations and a Posteriori Error Estimators for Adaptive Spectral Element Techniques*. PhD thesis, Massachusetts Institute of Technology, 1989.

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

[26] Catherine Mavriplis. Adaptive mesh strategies for the spectral element method. *Computer methods in applied mechanics and engineering*, 116:77–86, 1994.

[27] D. Scott McRae. r-refinement grid adaptation algorithms and issues. *Comput. Methods Appl. Mech. Engrg.*, 2000.

[28] A. S. Nielsen and J. S. Hesthaven. Fault tolerance in the parareal method. In *25th ACM Parallel and Distributed High Performance Computing*, June 2016.

[29] Nicolas Offermans, Oana Marin, Michel Schanen, Jing Gong, Paul Fischer, Philipp Schlatter, Aleks Obabko, Adam Peplinski, Maxwell Hutchinson, and Elia Merzari. On the strong scaling of the spectral element solver nek5000 on petascale systems. In *International Conference on Exascale Applications and Software, EASC 2016, Stockholm, Sweden*, 2016.

[30] K. Ou and A. Jameson. Unsteady adjoint method for the optimal control of advection and burgers equations using high order spectral difference method. In *In 49th AIAA Aerospace Sciences Meeting*, 2011.

[31] Niles A. Pierce and Michael B. Giles. Adjoint recovery of superconvergent functionals from pde approximations. *SIAM REVIEW*, 42(2):247–264, 2000.

[32] R. Sakai, H. Onda, D. Sasaki, and K. Nakahashi. Data compression of large-scale flow computation for aerodynamic/aeroacoustic analysis. In *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, pages 112–133. AIAA, January 2011.

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

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

[35] H. M. Tufo and P. F. Fischer. Fast parallel direct solvers for coarse grid problems. *J. Parallel and Distributed Computing*, 61:151–177, 1997.

[36] 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.

[37] Matthew Willyard. *Adaptive Spectral Element Methods To Price American Options*. PhD thesis, The Florida State University, 2011.

[38] Gang Xu, Bernard Mourrain, Regis Duvigneau, and Andre Galligo. Parameterization of computational domain in isogeometric analysis: Methods and comparison. *Comput. Methods Appl. Mech. Engrg.*, 2011.

[39] Lala Yi Li, Yves Allaneau, and Anthony Jameson. Continuous adjoint approach for adaptive mesh refinement. In *20th AIAA Computational Fluid Dynamics Conference*, 2011.