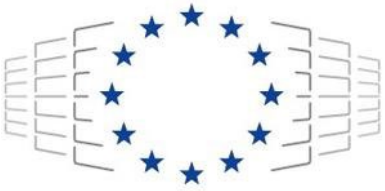
HORIZON-EUROHPC-JU-2021-COE-01

The European Centre of Excellence for Engineering Applications

Project Number: 101092621

D4.1
Workflows for engineering simulations progress report



The EXCELLERAT P2 project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101092621. The JU receives support from the European Union's Horizon Europe research and innovation programme and Germany, Italy, Slovenia, Spain, Sweden and France.

Work Package:	4	Workflow Development
Author(s):	Jonathan Vincent	KTH
	Antoine Dauptain	CERFACS
	Dennis Grieger	USTUTT
Approved by	Executive Centre Management	21.11.2023
Reviewer	Jernej Kovačič	UL
Reviewer	Christopher Röhl	SSC
Dissemination Level	Public	

Date	Author	Comments	Version	Status
2023-10-18	J. Vincent	Initial draft for review by co-authors	V0.1	Draft
2023-10-19	J. Vincent	Final draft for review by co-authors	V0.2	Draft
2023-10-20	J. Vincent	Draft internal review	V0.3	Draft
2023-11-07	J. Vincent	First review corrections	V0.4	Draft
2023-11-16	J. Vincent	Final Version	V1.0	Final

List of abbreviations

ASMR	Automatic Static Mesh Refinement
CFD	Computational Fluid Dynamics
CI	Continuous Integration
HPC	High-Performance Computing
IO	Input-Output
MPI	Message Passing Interface
SME	Small and Medium-sized Enterprises
VTK	Visualization Toolkit
WP	Work Package

Executive Summary

This document summarises the progress in workflows for engineering simulations connected with the use cases defined in EXCELLERAT P2. We present the development of in-situ visualisation which is a requirement for running the use cases at exascale.

We also show developments in improving efficiency of workflows, and work on standardising workflows. The latter is particularly important in simplifying workflows for new users, particularly new industrial users. In addition, we show the initial results for the use of automatic static mesh refinement, as an automatic mesh generation tool, which has been used to generate meshes of 50 million elements to date.

Table of Contents

Table of Figures

1 Introduction

Workflows in High Performance Computing (HPC) can be considered on two levels, firstly the workflow of an individual simulation and secondly that of an entire set of simulations working towards solving a particular problem.

Traditionally the first of these consists of pre-processing, solving and post-processing, which implies a split between obtaining the solution, storing the output data and then visualising the results. Exascale engineering simulations, however, produce a very large amount of data and make this method very challenging to use, so a different approach is needed. A major focus of EXCELLERAT WP4 is to enable in-situ analysis and visualisation, enabling the simulation output to be visualised and analysed in-situ instead of being saved and stored, greatly reducing the storage requirements for exascale simulations, and enabling the workflows to be run at exascale.

EXCELLERAT also focuses on the second part of workflow development, where we consider the efficiency and standardisation of the sets of simulations working towards solving a particular problem. The standardisation of workflows is particularly important for simplifying access to HPC for industrial users, a key task within EXCELLERAT P2. As part of this work the Lemmings tool [1] has been developed during EXCELLERAT P1 which allows workflows to be developed and deployed on different hardware easily, as the tool hides the complexity and differences between the systems from the end user, simplifying the usage of the HPC workflows and reducing the training and expertise needed to run an HPC workflow.

Another important part of Computational Fluid Dynamics (CFD) workflows is the generation of the mesh on which the final simulations are run. Generating this mesh can be a difficult task, both in computer time and human time. For exascale an automatic method is required, and we present progress on the development of the automation of mesh design.

2 In-situ Visualisation

In-situ visualisation is an important part of preparing workflows for exascale, as it removes the very large storage and IO requirements of the traditional pre-processing, solving and post—processing workflow. The amount of data needed for this traditional workflow is no longer an option for exascale, due to the very large amount of data that would be needed, so greatly reducing the storage and IO is a requirement.

During EXCELLERAT P1 the SENSEI [2] in-situ framework was chosen as one of the tools to establish common infrastructure for in-situ capabilities. As part of this an analysis adapter for SENSEI was developed for Vistle [3], and Nek5000 [4] was instrumented with SENSEI. This work is continued in EXCELLERAT P2. Due to changes in the APIs of SENSEI and Vistle the SENSEI – Vistle analysis adapter needed to be updated. The biggest change in this update is that SENSEI's Visualization Toolkit (VTK) bridge is now able to couple simulations and analysis tools built with different VTK versions.

The next step was applying the knowledge from instrumenting Nek5000 to Alya [5]. Since both SENSEI and Alya using the CMake [6] build system as build system generator linking it to SENSEI was relatively straight forward. It was also possible to reuse most of the code used in Nek5000 to bridge from the simulations Fortran code to the SENSEI's C++ interface through C bindings. As an initial step Alya's grid consisting of high-order Lagrange hexahedron cells is converted to VTK which has built-in support for this type of cells.

The velocity data array is exported to the in-situ visualisation every timestep. The grid does not change during the simulation, and so only needs to be exported once. This is done in the first timestep.

Vistle however does not support high-order cells, therefore, a conversion algorithm was implemented to convert the high-order Lagrange hexahedron cells of order x, y, z used in Alya to the transformed to x • y • z linear hexahedrons needed for Vistle.

The functionality of coupling Alya through SENSEI with Vistle was tested using a small simulation case running on four Message Passing Interface (MPI) ranks, and a simple image showing the output of a Taylor-Green vortex simulation (see Figure 1).

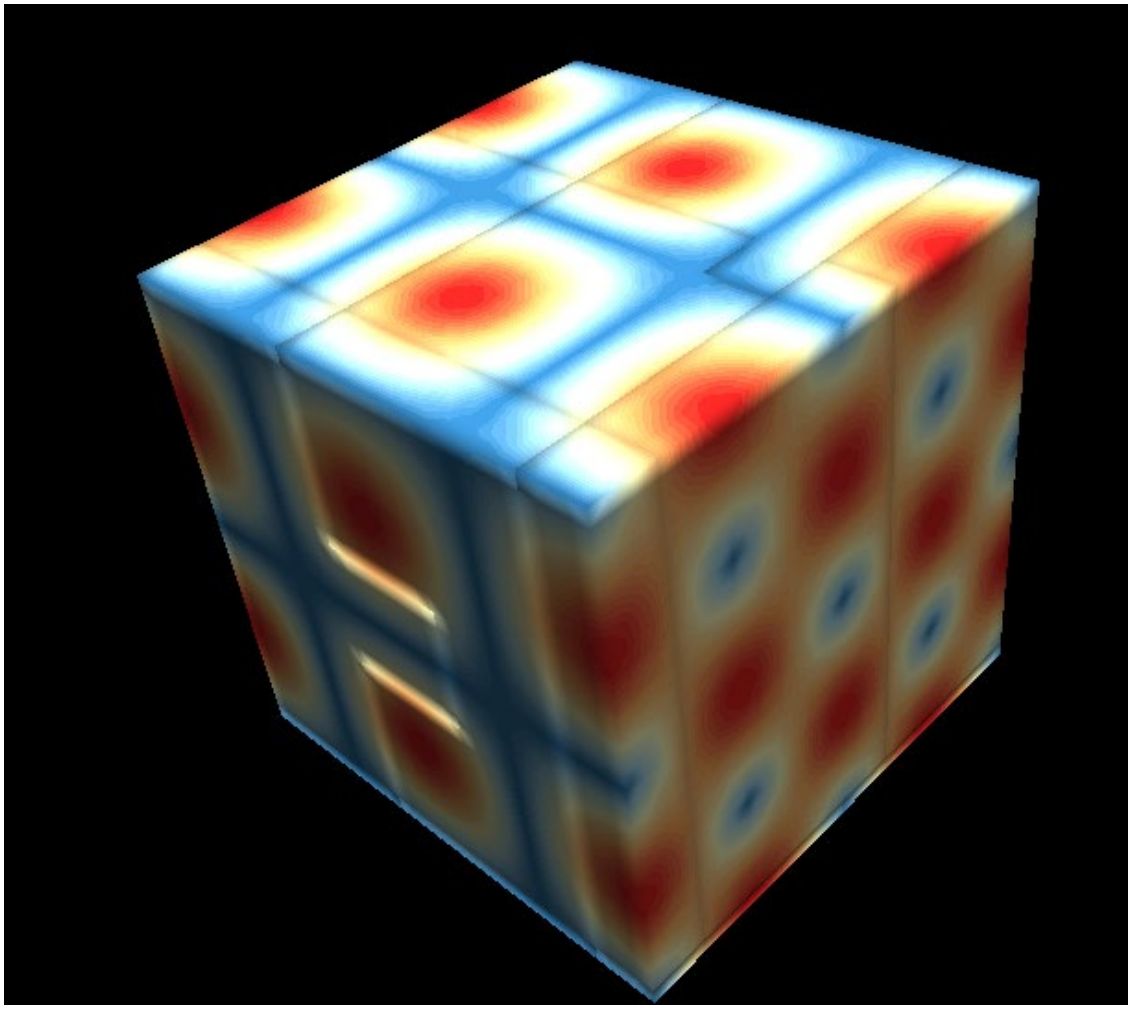


Figure 1: Velocity field of a Taylor Green Vortex obtained using Vistle and the SENSEI interface of Alya.

In addition to the in-situ work on Alya, the initial port of the Nek5000 SENSEI implementation has been completed is also underway on adapting Neko [7], with in-situ tools, however technical problems remain before an in-situ visualisation using Neko can be performed.

3 Homogenisation/Standardisation of HPC Workflows

Using HPC systems is often complicated, and different HPC systems require different commands and tools to use them effectively. To facilitate the industrial use of CFD applications it is important to create a standardised workflow. A standardised workflow is a significant step in improving the tool maturity, replacing several manual actions with an automated workflow,

and simplifying the use of the EXCELLERAT CFD applications as well as reducing training time.

3.1 Tools for Workflow Development

As facilitating the industrial use of CFD though refining the creation of relevant workflows is a key part of EXCELLERAT P2. The initial automated workflows while suitable for research applications, require significant enhancement before they would be suitable for industrial applications. Several tools have been developed to assist this work.

3.1.1 Lemmings

Lemmings is an open-source Python package that is simple to install using python package managers. It was initially developed during EXCELLERAT Part I by CERFACS. This software simplifies the submission of multiple inter-dependent jobs on HPC cluster schedulers. Although originally tailored for CFD applications, Lemmings can be used in many recursive job scenarios. It also incorporates a farming mode that facilitates the replication of recursive jobs for parametric studies. Lemmings offers an efficient solution for automating pre-existing manual workflows while allowing the terminal commands used by conventional HPC to be imported. A typical workflow loop for the Lemmings automation scheme is shown in Figure 2.

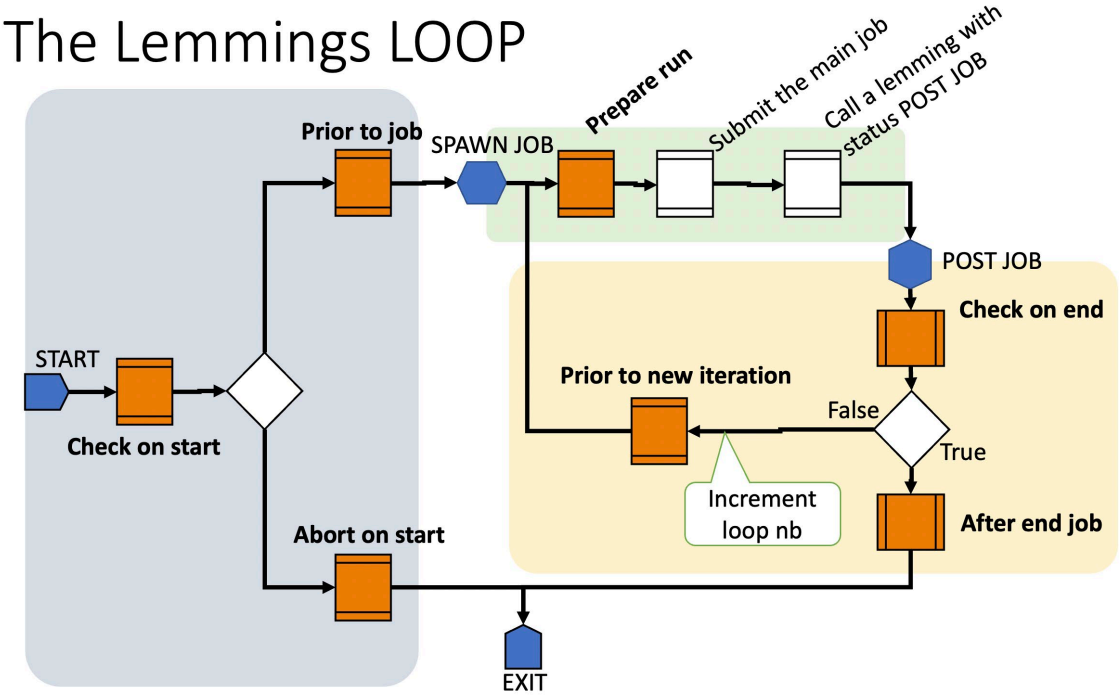


Figure 2: The backbone loop of the Lemmings automation scheme. By injecting custom code into the orange blocks, the user can adapt the tool to most workflows.

3.1.2 Scales

SSC Services GmbH, in alignment with the European project EXCELLERAT, has introduced the Scales solution. Scales was designed to aid small and medium-sized enterprises (SMEs) lacking expertise in HPC and integrates workflows into a web portal. The service records each workflow and can subsequently replay it with diverse data feeds. The user experience provided by Scales resembles that of Gitlab Pipelines, a widely utilised system within the Gitlab community. This approach offers an intuitive learning curve for users, reducing training time and simplifying the use of HPC resources to new users. Scales powers the Data Management

Platform EXCELLERAT service [8]. Figure 3 shows the high-level monitoring dashboard of Scales.

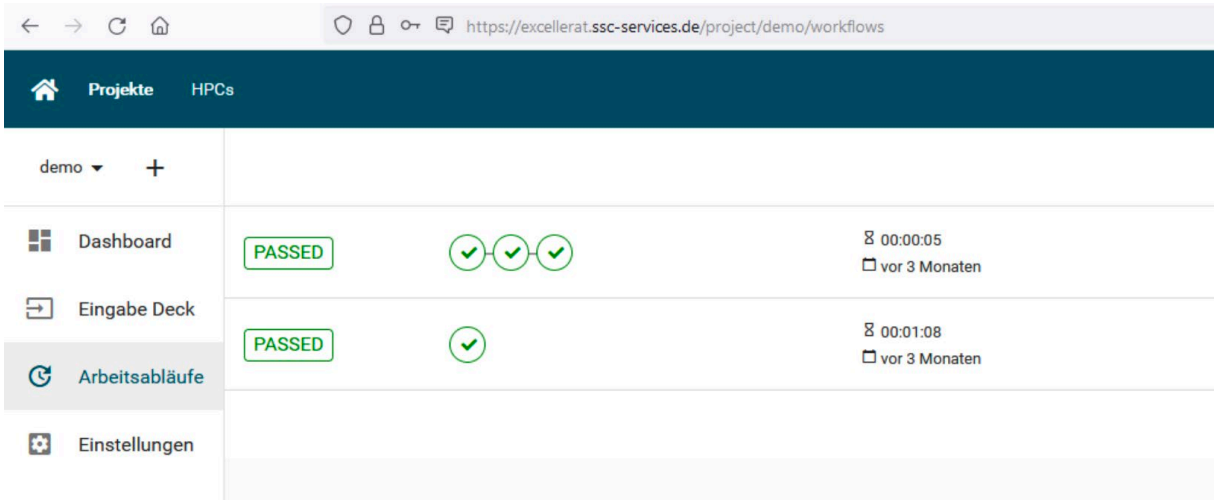


Figure 3: A screen capture of one of the high-level monitoring dashboards of Scales. Workflows of tasks can be resubmitted on the cluster until satisfactory results are obtained. Unlike the Lemmings approach, all connections and expert commands are encapsulated behind this graphical dashboard.

3.2 Recommendations

The initial findings described in the following sections were generated using a test workflow of combustion chamber behaviour analysis provided by the Safran Group [9]. This workflow was analysed using the lightweight open-source tool Lemmings.

While the initial findings were developed using CFD workflows, we believe the insights from studying this workflow are universally applicable across many domains, workflow managers and job schedulers. In the following sections we detail many of these specific insights.

3.2.1 Unified Chronological logging

Most workflows use tools with different and varying origins. These tools encompass research solvers, shell commands, postprocessing utilities, and analysis software, each with its output format and logging mechanisms. These many and diverse outputs can make identification of the source of any issues challenging. Streamlining these diverse log entries into a single, chronological log file proves invaluable in saving time and enhancing collaboration among developers, users, and support teams.

3.2.2 A Unique Identifier for workflow instances

One common challenge faced by new users when working with workflows is managing multiple instances concurrently. Distinguishing between results generated by different instances can become unexpectedly complex, leading to confusion. To address this, we recommend the generation of unique execution IDs, akin to the unique job IDs provided by the scheduler. These IDs serve multiple purposes: they facilitate monitoring and debugging by naming the unified log, enable customisation of job names for clear identification in scheduler queues, and streamline general control commands, simplifying the management of multiple workflow instances.

This idea is implemented in Lemmings by ensuring that each chain of Lemmings jobs on the cluster uses the same name, a user defined prefix followed by two random syllables, followed by two digits. In the following example, this name is twicer_JEXA93 for all jobs:

```
Status for chain twicer_JEXA93
+------+---------------+-----------------+-----------+--------------+--------+---------+
| Loop | Solution path | Job end status | progress | CPU time (h) | job ID | pjob ID |
+------+---------------+-----------------+-----------+--------------+--------+---------+
|  0   |       ./      | ended, continue |     NA    |    0.001     | 95575  |  95578  |
|  1   |       ./      | ended, continue |     NA    |    0.002     | 95590  |  95591  |
|  2   |   Submitted   |    Submitted    | Submitted |  Submitted   | 95611  |  95612  |
+------+---------------+-----------------+-----------+--------------+--------+---------+
```

3.2.3 No File Movement

In our experience, we encountered a common pitfall where archived simulations involved extensive file movement within the workflow. These moves resulted in broken relative paths, rendering archived runs challenging to reproduce. The solution lies in avoiding any file movement within the workflow. All executions should occur within their designated, definitive folders, ensuring a seamless workflow history for easy re-execution.

3.2.4 Version and Document the Workflow

An effective collaboration between developers and customer beta-testers during workflow iterations requires clear versioning. This extends beyond component versioning to encompass the evolution of the workflow itself. Additionally, documenting the workflow strategy within each job folder, outlining the initial file locations, databases, and mesh, is very important when support is provided by HPC domain experts, who may not be users of the workflow themselves. This approach simplifies future work and enhances communication.

3.2.5 A Sandbox Scheduler

Often HPC facilities offer a “debug” partition for rapid testing of individual runs, workflows that consist of numerous jobs, can result still result in significant waiting times. To reduce feedback delays for developers and support teams, we highly recommend adopting a “sandbox scheduler” i.e., a lightweight emulation of a scheduler, as demonstrated by the Lemmings sandbox, which can significantly accelerate testing and debugging of the automated workflows. A sandbox scheduler can also be easily used within a suite of Continuous Integration (CI) tests, simplifying the automatic testing of changes.

The Lemmings sandbox for example supports four commands start, submit, cancel and qstat, i.e. the commands to start the tool, add a new job, cancel a job and show the current queue respectively. Everything is then synchronised by updating a file on disk. The whole sandbox consists of 400 lines of python lines and is readable at the readable on the Lemmings public repository [10].

3.2.6 A mock-up Solver

Similar to the sandbox scheduler, the concept of a mock-up solver streamlines the testing process. The mock-up solver is a program that takes core HPC solver inputs and rapidly generates dummy outputs, mimicking a real simulation. This testing involving multiple simulations becomes feasible within minutes. This considerably enhances the work of the developers and support teams.

In order to create a mock-up solver for the AVBP [11] case, a script was created called “avbp_mockup”. This script reads the main input file, running parameters and the mesh files. It then creates from these the relevant output files, i.e., the solution at specific time points, averaged solutions and monitoring files.

3.2.7 End User Commands

Workflow commands often contain detailed information essential for engineers overseeing the process, yet they can be overwhelming for end users. To strike a balance, we propose segregating “End User Commands” into a separate script, such as a shell script in our case study, managed by the customer. This straightforward approach has proven highly effective, enabling most initial user requests to be addressed directly within the customer’s organisation by refining the user experience.

A typical BASH shell script that could be used for this process is shown below.

```
#!/bin/bash

echo "Loading bash commands for WorkFlow FooBar in your terminal."

# Adapt this to your cluster
export LEM_MACHINE_FILE=”mymachine.yml"
export WFLOW_PATH=”(…)/WorkFlow_Lemmings/"

#
export LEM_WORKFLOW="${WFLOW_PATH}workflow_FooBar.py"
export LEM_HELPFILE="${WFLOW_PATH}workflow_FooBar.md"
export LEM_INPUT="${WFLOW_PATH}workflow_FooBar.yml"

echo "You will use workflow  $LEM_WORKFLOW."
echo "Machine configuration is at $LEM_MACHINE_FILE."

function lemfoobar_input () {
    echo "Copy workflow FooBar input file in this directory... "
    \cp $LEM_INPUT .

}
echo " - lemfoobar_input  to get a default input file"

function lemfoobar_run () {
    echo "lemmings run --inputfile $1 --machine-file  $LEM_MACHINE_FILE
$LEM_WORKFLOW "
    lemmings run --inputfile $1 --machine-file  $LEM_MACHINE_FILE
$LEM_WORKFLOW
}
echo " - lemfoobar_run myinput.yml to start your FRT workflow"

function lemfoobar_status  {
    echo "lemmings status --progress"
    lemmings status --progress
}
echo " - lemfoobar_status to get a status of the last workflow, running or
not."

function lemfoobar_help () {
    cat $LEM_HELPFILE
}
echo " - lemfoobar_help  to read about the FRT workflow requirements and
behavior"


function lemfoobar_kill  {
    echo "lemmings kill --machine-file  $LEM_MACHINE_FILE"
    lemmings kill --machine-file  $LEM_MACHINE_FILE
}
echo " - lemfoobar_kill to kill your current workflow."
```

3.2.8 Container issues

Containers, such as Singularity Containers, offer a means to maintain a stable execution environment that can be reproduced across various machines. However, when integrated into workflows, a challenge arises. Containers are typically built on a machine different from that of the workflow creators and lack access to local job submission commands. Addressing this issue requires collaboration with the local IT team to adapt containers, potentially delaying the solution's availability.

3.3 Automatic static mesh refinement

Meshing is an important first step in CFD, where the complex object to be modelled is converted into a mesh of well-defined cells where the governing physical equations can be applied, allowing the solver to simulate the physical behaviour. The traditional method of generating a mesh involves initially generating a final-sized mesh, then refining it iteratively by running simulations using that final-sized mesh. Once the mesh has been sufficiently refined it can be then used in production simulations.

Unfortunately, this method is no longer viable at the exascale level. Using many runs at the final mesh size is too expensive. Therefore, a different method is required. In order to generate a mesh suitable for exascale at an appropriate cost, it is necessary to use an initial mesh that is significantly smaller than the final-sized mesh desired, and iteratively increase the mesh size as more information from simulations are obtained. For each mesh size a simulation will give more information on where the mesh should be refined. When successful this process would provide sufficiently good mesh for the final simulations at a controlled cost.

Note unlike Dynamic Mesh Adaptation, this process discards the simulation history, and retains only the final mesh and solution. Indeed, there are theoretical open issues with high fidelity simulations obtained on a moving mesh, so to stay on the safe side this workflow is an Automated Static Mesh Refinement (ASMR).

The initial goal of this work is to show that it is possible to replace the domain knowledge needed to create a mesh of sufficient quality with mechanical iterative process. A secondary goal is to demonstrate the mesh generation procedure that can reliably produce a stable grid with more than one billon cells. It is also important to understand the most efficient way to produce the grids, i.e., is it better to use many small mesh refinements or aim for fewer but larger refinements?

Initial runs using ASMR, the CFD code AVBP have been completed using EXCELLERAT use case 2 (hydrogen combustion for propulsion). The ASMR runs have to date produced acceptable meshes for industrial configurations. To date meshes of up to 50 million cells have been generated using this method, and these meshes are currently being reviewed to ensure that they are high enough quality.

In order to hit the target of generating a mesh large enough for exascale applications i.e., of around a billion points, the workflow must be adapted to use parallel mesh development. An example of ASMR is shown in Figure 4, where the initial mesh of 2.8 million cells is in several stages into a mesh of 46 million cells.

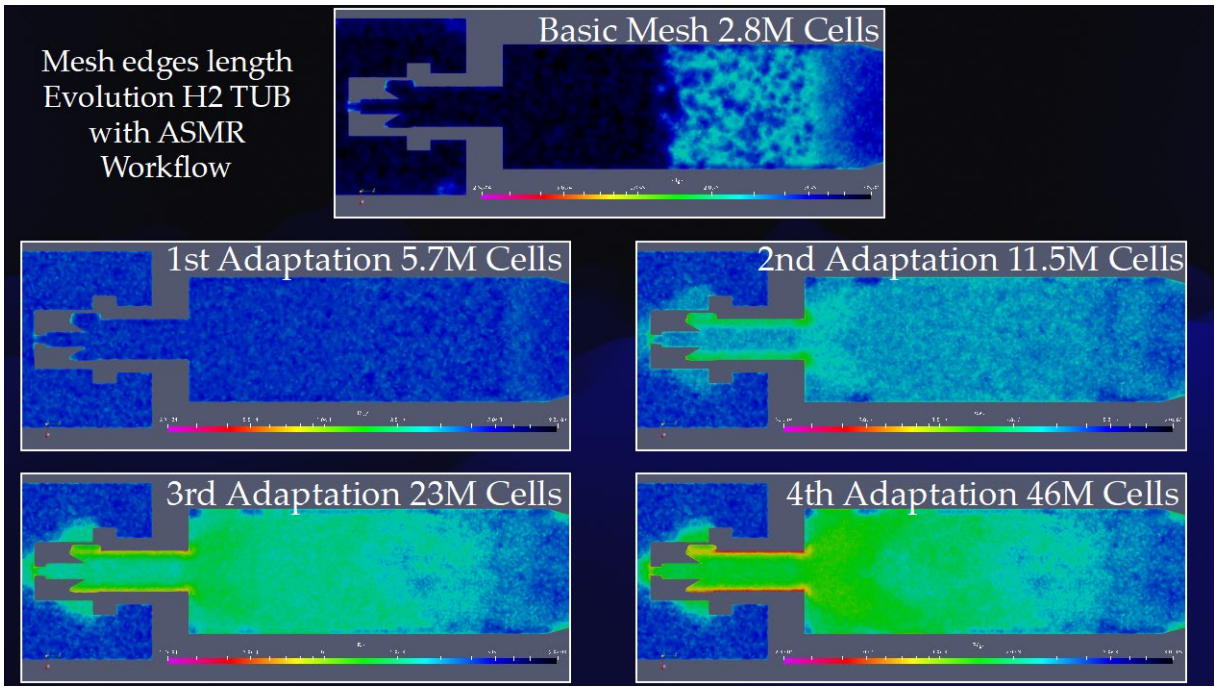


Figure 4: Figure showing how a mesh is automatically refined using mesh adaptation with the ASMR workflow. The configuration is an academic Hydrogen Burner (TUB burner). Our final goal is the generation of a 1 billion cell mesh in a single workflow.

4 Conclusions

In-situ visualisation is required to be able to perform CFD simulations at exascale, as it removes the need for very large amounts of IO and storage that a traditional workflow would require. Initial work on connecting the CFD code Alya with the visualisation software Vistle has been completed, and the first results obtained using a Taylor-Green Vortex as the input. Future work following up this successful start using more scientifically relevant input is planned.

Work on optimising and simplifying workflows has also been completed, with a significant number of recommendations made on how workflows should be organised and optimised. Further analysis will be undertaken as future work.

Initial work on ASMR has also been completed, successfully generating meshes with up to 50 million cells. Work is ongoing to verify the meshes created are of sufficient quality and to produce larger meshes.

5 References

[1] Lemmings, https://lemmings.readthedocs.io/en/latest/
[2] SENSEI, https://sensei-insitu.org/
[3] Vistle, https://vistle.io/
[4] Nek5000, https://nek5000.mcs.anl.gov/
[5] Alya, https://www.bsc.es/research-development/research-areas/engineering-simulations/alya-high-performance-computational
[6] CMAKE, https://cmake.org/
[7] Neko, https://neko.cfd
[8] Data Management Platform, https://services.excellerat.eu/viewcode/9
[9] Safran Group, https://www.safran-group.com/
[10] Lemmings public repository, https://gitlab.com/cerfacs/lemmings//tree/master/src/lemmings/sandbox
[11] AVBP, https://www.cerfacs.fr/avbp7x/