

Where Quantum and Fluids entangle

Quantum Computational Fluid Dynamics
OpenFOAM to QCFD Interface
Project Number# 101080085

Call:	HORIZON-CL4-2021-DIGITAL-EMERGING-02
Topic:	HORIZON-CL4-2021-DIGITAL-EMERGING-02-10
Type of action:	HORIZON Research and Innovation Actions
Granting authority:	European Commission-EU
Project starting date: fixed date:	1 November 2022
Project end date:	30 April 2027
Project duration:	54 months
Project Coordinator:	University of Hamburg (UHH)
Deliverable number:	D9.1
WP contributing to the deliverable:	WP9 – Open-Source Integration and Dissemination
Deliverable Type:	Other
Revision:	0
Dissemination level:	Public
Due Submission date:	30/10/2026
Prepared By:	ENG, TUHH, UHH, FZJ, TUC
Internal Reviewers:	UHH, TUHH, FZJ, ENG, TUC, PlanQC, TUM
Final Approval:	Prof. Dr. Dieter Jaksch

Revision History

Version	Date	Who	Changes
0	30/10/2026	ENGYS	1st submission

List of abbreviations

Acronym / Short Name	Meaning
EU	European Union
EC	European Commission
QCFD	Quantum Computational Fluid Dynamics
UHH	University of Hamburg
TUHH	Technical University of Hamburg
FZL	Jülich Research Centre
TUM	Technical University of Munich
ENG	ENGYS SRL
TUC	Technical University of Crete
PlanQc	PlanQc GMBH
CDE	Communication, Dissemination and Exploitation
KPI	Key Performance Indicator
WP	Work Package
PO	Project Officer
SME	Small and Medium Enterprise
WP	Work Package
WPL	Work Package Leader

Executive Summary

The Quantum Computational Fluid Dynamics (QCFD) project is dedicated to advance quantum computing for Computational Fluid Dynamics (CFD) simulations and their use in industry.

The deliverable D9.1 provides the software interface and library as part of Work Package 9 (WP9) ("OpenFOAM-QCFD software interface"). This comprises an OpenFOAM software library that interfaces with the QCFD software framework, making quantum CFD capabilities accessible to a broad user base, along with a tutorial document [1] that demonstrates the use of QCFD algorithms for simple CFD cases.

In accordance with the Data Management Plan (DMP), D9.1 follows the FAIR data principles – Findability, Accessibility, Interoperability, and Reusability – the associated open-source software are provided with a detailed metadata structure, unique identifiers, and comprehensive documentation. This ensures that the software is easily accessible and usable by third parties. Public access is granted via a dedicated research data repository (FDR) hosted at the University of Hamburg (UHH), https://doi.org/10.25592/uhhfdm.22083. To make it easier for users to find and use the software, all QCFD-related code, interfaces, and tutorial material belonging to the project are collected in a QCFD community group on the mentioned data server. The software library is also made available through a public-access git repository in accordance with the project's dissemination strategy.

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

Contents

1 Introduction **6**

2 Software Description **7**
2.1 Software Quality and Accuracy 7

3 Indirect Coupling: Data Mapping via Immersed Boundary Method **8**
3.1 Overview of the Indirect Coupling Workflow 8
3.2 Geometry Extraction from OpenFOAM 8
3.3 Solid Mask Generation on Cartesian Grid 8
3.4 Preparation of QCFD Input Data 9
3.5 Inverse Mapping of Results 9
3.6 Advantages and Limitations 9
3.7 Software Implementation 9

4 Direct Matrix-to-Tensor Network Conversion **10**
4.1 Overview of the Conversion Workflow 10
4.2 Tensor Train Representation of Flow Fields 10

5 Software Validation Cases **12**
5.1 2D Problems - Mapping Conversion 12
5.1.1 V2D-1: Heat conduction in a square with a cylinder 13
5.1.2 V2D-2: Flow around cylinder 20
5.1.3 V2D-3: S-Bend channel 27
5.2 2D Problem - Matrix Coupling 34
5.2.1 V2D-4a: Flow around cylinder 34
5.2.2 V2D-4b: Verification against OpenFOAM 35

6 Conclusions **36**

References **38**

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

1 Introduction

The integration of quantum computing into engineering workflows represents a fundamental shift in how complex simulation problems can be approached. However, bridging the gap between quantum algorithms and established engineering practice requires more than theoretical advances; the long-term needs of the field have been mapped out in strategic studies such as the CFD Vision 2030 report [2]. It demands practical, usable software interfaces that lower the barrier to entry for the broader CFD community. Deliverable D9.1 addresses this need directly by providing an OpenFOAM software library that connects the QCFD software framework to one of the most widely used CFD platforms in both academia and industry. This library, enables users to access quantum CFD capabilities without requiring deep expertise in quantum programming or tensor network methods.

The current landscape of quantum computing is characterized by rapid but uneven progress. Noisy intermediate-scale quantum (NISQ) devices are available today [3], but their practical utility remains limited by qubit counts, coherence times, and gate fidelities. Future fault-tolerant quantum computers promise far greater capabilities, but their arrival is not yet certain, and the transition to early fault-tolerant machines is itself an active area of research [4]. The potential of quantum computation for aeroscience and engineering has been discussed in the literature [5]. The OpenFOAM-QCFD interface developed in this deliverable is designed to operate across this evolving landscape. It does not assume the availability of large-scale, error-corrected quantum hardware. Instead, it provides a flexible coupling mechanism that can function with current quantum devices, quantum emulators, or classical tensor network simulations, allowing users to choose the most appropriate backend for their needs.

Two complementary coupling strategies are implemented within this library, each with its own strengths and trade-offs. The first approach maps conventional OpenFOAM geometry and configuration data-tipically represented on unstructured grids-into a level-set formulation on a hierarchical Cartesian lattice. This level-set is then converted into matrix product state representations suitable for QCFD algorithms. This method prioritizes flexibility and robustness, as it does not require equivalent discretisation infrastructures between the classical and quantum domains. The second approach pursues direct conversion of OpenFOAM solution matrices into tensor network states, offering a more seamless integration path at the cost of imposing certain constraints on the grid structure and matrix connectivity.

The tutorial document [1] accompanying the software library serves multiple purposes. They demonstrate basic usage patterns, validate the correctness of the coupling implementation, and provide reference points for users wishing to extend the library to their own problems. These examples are deliberately kept simple in terms of spatial resolution and physical complexity, reflecting the current practical limits of quantum hardware and emulation. This relative simplicity should not be mistaken for lack of ambition; rather, it represents an honest assessment of where the fields stands today, while laying a foundation for future scalability.

A central principle guiding this deliverable is that quantum CFD methods should not develop in isolation from classical CFD. The scientific community benefits most when new approaches can be directly compared against establieshed ones, using the same input data and yielding comparable output formats. The OpenFOAM-QCFD interface enables exactly this kind of side-by-side comparison, allowing users to benchmark quantum algorithms against classical solvers on identical problems. The benchmark cases used in this deliverable are aligned with the core benchmark CFD set defined in Deliverable D1.1 [6]. This transparency is essential for building trust in quantum methods and for identifying the specific regimes-problem sizes, flow conditions, or optimisation objectives-where quantum approaches may eventually offer advantages.

Recognizing that software usability is as important as technical functionality, the library is released as open source under a permissive licence, with full documentation provided in accordance with the project's Data Management Plan. A public git repository hosts the source code, while stable releases are assigned Digital Object Identifiers (DOIs) to ensure long-term findability and citability. The tutorial document [1] describes the workflow step-by-step for the three indirect-coupling cases, and metadata including versioning is accessible through the QCFD community repository hosted at the University of Hamburg (UHH): www.fdr.uni-hamburg.de/communities/qcfd/.

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

By providing this interface, D9.1 lays the groundwork for broader adoption of quantum methods in industrial CFD applications. It transforms the QCFD project from a collection of algorithms and theoretical results into a usable tool that can be integrated into existing engineering workflows. The following sections detail the design, implementation, and usage of the OpenFOAM-QCFD software library.

2 Software Description

Deliverable D9.1 builds on the foundational work from previous work packages – particularly WP1, WP2, WP3, and WP8 – to provide a functional software library that bridges classical CFD workflows with quantum CFD algorithms. The deliverable comprises an OpenFOAM software library that interfaces with the QCFD software framework, accompanied by the tutorial document [1] that demonstrates its usage for simple CFD cases. Where quantum hardware or emulators have been accessible within the project timeline, validation results obtained using the interface are documented as part of the tutorial examples.

The coupling between OpenFOAM and QCFD is implemented through two distinct pathways, each suited to different use cases and risk profiles. The first pathway employs a level-set-based geometry transfer using a hierarchical Cartesian grid with adaptive mesh refinement, converting unstructured mesh representations into signed distance fields and subsequently into matrix product state penalisation functions. The second pathway pursues direct matrix-to-tensor network conversion, offering tighter integration at the cost of additional constrains on mesh connectivity. The software distinguishes between several key aspects: (a) the spatial dimensionality of the input CFD problem, (b) the representation fidelity of geometry and boundary conditions, (c) single- or multi-physics problem types, (d) linear and nonlinear equation systems, (e) steady-state versus time-dependent simulations, (f) the availability of multiple resolution levels for multigrid-style transfers, (g) compatibility with industrial use cases derived from WP2, (h) validation against benchmark datasets, and (i) the specific coupling strategy employed (level-set mapping or direct matrix conversion).

The remainder of this section provides information on the software architecture, quality and testing requirements, as well as the suggested validation criteria (e.g., pass/fail criteria for interface correctness; see Section 5).

2.1 Software Quality and Accuracy

Reliable software interfaces are fundamental to effective adoption of quantum methods in engineering practice, and are therefore a key component of deliverable D9.1. The quality standards applied to the OpenFOAM-QCFD library are given below:

- Supported file formats for input/output with corresponding precision, accessible without compression:
  - OpenFOAM native format (.foam) for mesh and field data
  - Hierarchical Data Format version 5 (.hdf5) for matrix and tensor data
  - NumPy binary format (.npy) for tensor network interchange
  - Boundary condition specification (.bcs) for the QCFD solver input
  - Configuration parameters (.toml) for the QCFD solver input
  - Julia Data format (.jld) for the tensor network solver output
  - Portable Document Format (.pdf) for the tutorial documentation
  - Plain text format for configuration parameters
- Double-precision floating-point number format for all exchanged field data to maintain consistency with classical CFD solvers
- For complex-valued tensor network states arising from quantum algorithm representations, the 53-bit significant precision of 16 decimal digits is divided between real and imaginary parts
- Version control through public git repository with tagged releases
- Continuous integration testing for interface stability across OpenFOAM versions

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

3 Indirect Coupling: Data Mapping via Immersed Boundary Method

The first coupling strategy implements an indirect approach, where OpenFOAM geometry and configuration data are mapped into a QCFD-compatible model through an immersed boundary solid mask representation [7]. This method has been developed and implemented by ENGYS.

3.1 Overview of the Indirect Coupling Workflow

The indirect coupling strategy serves as a data mapping tool that translates OpenFOAM inputs into a format suitable for the QCFD framework. Rather than performing any CFD simulation within the coupling interface, the method focuses on the faithful representation of the physical domain and boundary conditions in a form compatible with tensor network algorithms. The workflow consists of three main stages: extraction of geometry and boundary conditions from the OpenFOAM case, construction of a solid mask on a Cartesian grid, and encoding of the solid mask together with the problem parameters as matrix product state (MPS) inputs for the QCFD solver.

3.2 Geometry Extraction from OpenFOAM

The indirect coupling workflow begins with the extraction of geometry and boundary condition information from an existing OpenFOAM case. This includes the computational domain geometry defined by the OpenFOAM mesh, the location and shape of solid obstacles within the domain, boundary condition specifications for velocity, pressure, and other fields, and material properties such as fluid viscosity and density. The boundary conditions are encoded in a structured format that the QCFD framework can interpret, preserving the physical meaning of each boundary type, whether inlet, outlet, wall, or symmetry.

3.3 Solid Mask Generation on Cartesian Grid

Once the geometry is extracted, a solid mask is constructed on a hierarchical Cartesian lattice. The solid mask $\mathcal{M}(\mathbf{x})$ is a scalar field defined over the Cartesian grid that distinguishes between fluid and solid regions:

$$\mathcal{M}(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} \in \text{solid region} \\ 0, & \mathbf{x} \in \text{fluid region} \end{cases} \tag{1}$$

The construction of the solid mask involves sampling the OpenFOAM geometry on the Cartesian grid and assigning a value of unity to grid points that lie inside the solid body and zero to points in the fluid region. The grid is refined hierarchically near the solid boundaries to capture the geometry with sufficient resolution.
A critical aspect of the mask generation is the treatment of staggered grid locations. In the QCFD framework, pressure is defined at cell centres while velocity components are defined at staggered face locations, consistent with the finite volume discretisation employed in OpenFOAM [8]. To ensure compatibility, the solid mask is interpolated to these staggered positions:

- The pressure mask $\mathcal{M}_p$ is defined at cell centres, identical to the solid mask.
- The u-velocity mask $\mathcal{M}_u$ is defined at faces normal to the x-direction, computed as the average of adjacent cell-centred mask values.
- The v-velocity mask $\mathcal{M}_v$ is defined at faces normal to the y-direction, computed as the average of adjacent cell-centred mask values.

This staggered approach ensures that the solid mask is correctly aligned with the velocity degrees of freedom in the QCFD solver, preserving the accuracy of the immersed boundary formulation.

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

3.4 Preparation of QCFD Input Data

The solid mask, together with its staggered variants, is stored in HDF5 format along with the grid metadata — including grid dimensions, physical extents, and reference parameters. This forms the geometry input for the QCFD solver, independent of the physical problem to be solved.
In addition to the solid mask, the indirect coupling generates the necessary configuration files for the QCFD solver. The simulation parameters are stored in a TOML configuration file. These parameters include the problem-specific physical settings — such as the Reynolds number for flow problems, the thermal diffusivity for heat conduction problems, or the material properties for conjugate heat transfer — together with the time-stepping controls, the parameters of the immersed boundary forcing, and the MPS-specific parameters such as the bond dimension χ that controls the compression level. The immersed boundary parameters depend on the physical model: for the momentum equation they define the penalisation strength that drives the velocity to zero inside the solid, while for the energy equation they define the thermal conductivity and the fixed temperature imposed within the solid region.
The boundary conditions are stored separately in a BCS file, where each boundary is encoded as a pair consisting of a boundary type and an associated value. The supported boundary types are Dirichlet and Neumann. The encoding is applied per field and per boundary, so the same format covers velocity, pressure, temperature, and any additional scalar fields required by the chosen physical model.

3.5 Inverse Mapping of Results

After the QCFD simulation is complete, it produces a solution in JLD format. The purpose of the inverse mapping is to compare this solution with the output of the modified OpenFOAM solver that solves the immersed boundary problem on the Cartesian solid mask grid. The two solvers do not store their fields at the same locations: the modified OpenFOAM solver uses a collocated arrangement, with pressure and velocity at the cell centres, while the QCFD solver uses a staggered arrangement, with the velocity components at the cell faces and the pressure at the cell centres. The reconstruction therefore maps the tensor network solution onto the staggered positions used by the QCFD solver and stores it in the same OpenFOAM Cartesian format as the reference, so that the two solutions are expressed at the same field locations and in the same file format. This reconstruction is the basis for the cell-by-cell comparison and for the error metrics reported in the validation cases.

3.6 Advantages and Limitations

This indirect approach prioritises flexibility and robustness, as it does not require equivalent discretisation infrastructures between the classical and quantum domains. The immersed boundary formulation naturally handles complex geometries and allows for adaptive resolution refinement where needed. The staggered mask generation ensures compatibility with the QCFD solver's grid arrangement.
However, the conversion introduces an additional approximation step through the interpolation between the body-fitted and Cartesian grids, and the hierarchical grid representation may not preserve all the fine-scale features of the original body-fitted mesh.

3.7 Software Implementation

The indirect coupling strategy is implemented as part of the OpenFOAM-QCFD software library, and it has been developed by ENGYS. The geometry extraction and the solid mask generation are performed within the OpenFOAM environment, using the standard OpenFOAM mesh and field utilities. The prepared input data — including the solid mask in HDF5 format, boundary conditions, and simulation parameters — are passed to the QCFD framework as input. The inverse mapping reconstructs the QCFD output on the Cartesian solid mask grid, at the staggered positions used by the QCFD solver, so that it can be compared with the output of the modified OpenFOAM solver on the same grid.

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

4 Direct Matrix-to-Tensor Network Conversion

The second approach pursues direct conversion of the discretized system matrices arising from the finite difference (FD) formulation into tensor network representations, offering a more seamless integration path at the cost of imposing certain constraints on the grid structure and matrix connectivity. This method has been developed and implemented by the University of Hamburg (UHH) and is based on the algorithmic framework introduced in their publication [9]. This framework employs a reduced-order modeling approach based on binary Tensor Trains (TT), which is directly portable to quantum hardware. The method enables tensor network simulations to be carried out solely using highly compressed tensor representations of both the flow fields and the differential operators.

4.1 Overview of the Conversion Workflow

The conversion strategy translates the algebraic system produced by the FD discretisation into a tensor network format. The method operates on the discretised operators — such as the gradient, divergence, and Laplacian — and on the solution fields themselves, converting each into a Matrix Product Operator (MPO) or a Matrix Product State (MPS), respectively.

The workflow consists of four main stages: construction of the discretised operators on the computational grid, encoding of the operators as MPOs, encoding of the solution fields as MPSs, and reconstruction of the fields from the tensor network representation after the solver has been executed. A fractional-step time integration scheme [10], analogous to classical projection methods, is employed to advance the solution in time while preserving the mathematical structure of the governing equations.

4.2 Tensor Train Representation of Flow Fields

In the TT framework, a vector of length p^N representing a discretised field is reshaped into an order-N tensor with indices $r_k \in \{0, \ldots, p-1\}$ and subsequently decomposed into a tensor train of N order-3 tensors [11, 12, 13]:

$$\mathbf{a}_{r_1,\ldots,r_N} = \sum_{\alpha_1,\ldots,\alpha_{N-1}} A[1]^{r_1}_{\alpha_1} A[2]^{r_2}_{\alpha_1,\alpha_2} \cdots A[N]^{r_N}_{\alpha_{N-1}} \tag{2}$$

where the indices α_k are the bond dimensions that control the compression level, and $A[k]$ are the order-3 tensor train cores with mode indices r_k and bond indices α_k. Each velocity component and the pressure field are represented as individual tensor train vectors. The decomposition is illustrated graphically in Figure 1.

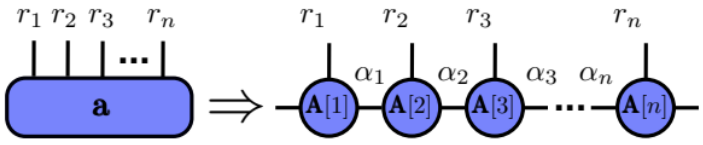


Figure 1: Example of the decomposition of an arbitrary vector $\mathbf{a}$, reshaped as a 2^n-component tensor $\mathbf{a}_{r_1,r_2,\ldots,r_n}$, into the TT format by performing a series of SVDs, where $A[k]$ are order-3 tensors with mode indices $r_k \in \{0, 1\}$ and bond indices α_k. The corresponding mathematical interpretation is given in Eq. (2).

For a matrix of dimension $p^N \times p^N$, the representation generalises to a Matrix Product Operator, in which each core carries two physical legs — one for the row index and one for the column index — in addition to the bond indices:

$$\mathbf{M}_{(r'_1,\ldots,r'_N),(r_1,\ldots,r_N)} = \sum_{\alpha_1,\ldots,\alpha_{N-1}} M[1]^{r'_1 r_1}_{\alpha_1} M[2]^{r'_2 r_2}_{\alpha_1,\alpha_2} \cdots M[N]^{r'_N r_N}_{\alpha_{N-1}} \tag{3}$$

The discretised differential operators — gradient, divergence, and Laplacian — are constructed as Matrix Product Operators and encoded in the tensor train format. The graphical depiction of the TT representation of a matrix is shown in Figure 2.
The number of parameters in such a representation scales as

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

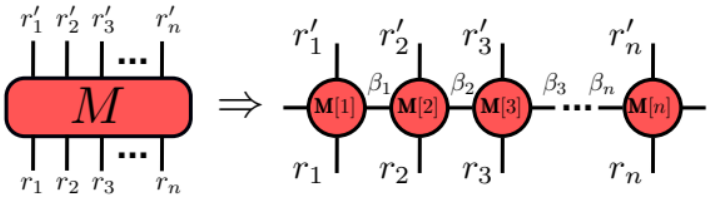


Figure 2: Graphical depiction of the TT representation $\mathbf{M}$ of a matrix M. The red circles with outgoing legs denote TT-cores, where $(r'_1, \ldots, r'_N)$ represents the rows of M and $(r_1, \ldots, r_N)$ enumerates its columns. The corresponding mathematical description is given in Eq. (3).

$$\text{Parameters} \sim N \cdot p \cdot \chi^2 \tag{4}$$

where χ is the maximum bond dimension. This is in contrast to the p^{2N} parameters required for a dense matrix, making the tensor train representation a powerful tool for compression when the bond dimension remains modest.

A central operation in the tensor network solution of the discretised equations is the application of a Matrix Product Operator to a Matrix Product State. In the tensor network formalism, this operation is performed by contracting the TT-matrix and the TT-vector over their common physical indices. The result of the contraction is again a Matrix Product State, whose bond dimensions are given by the product of the bond dimensions of the operator and the state at each site. The graphical representation of this operation is shown in Figure 3.

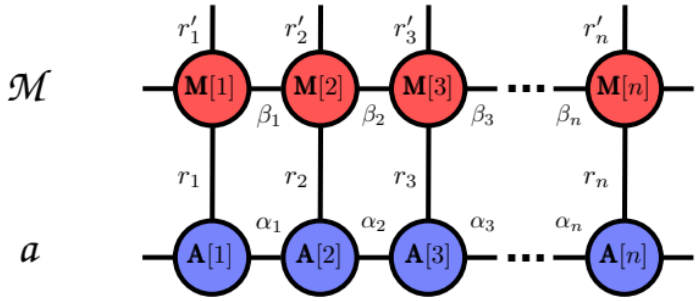


Figure 3: Graphical representation of TT-matrix acting on a TT-vector, which is achieved by contraction over common physical indices.

The ability to apply operators directly in the compressed tensor network format, without reconstructing the full vector or matrix, is one of the principal advantages of the TT representation. It allows the entire time-stepping procedure of the fractional-step algorithm to be carried out in the compressed format, with truncation applied after each operator application to keep the bond dimensions under control.

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

5 Software Validation Cases

This section presents the validation cases developed to test and demonstrate the OpenFOAM-QCFD software library. Unlike conventional CFD benchmark cases that assess physical accuracy or numerical performance, the cases presented here are designed to validate the correctness and fidelity of the software interface itself. Each case examines a specific component of the conversion pipeline — from the extraction of body-fitted grids and boundary conditions in OpenFOAM, through the generation of level-set solid masks and their conversion into matrix product state representations, to the inverse mapping of quantum simulation results back onto the original OpenFOAM grid. The validation strategy is consistent throughout: a reference solution is computed entirely within OpenFOAM using classical methods; the same problem is then passed through the interface (or a selected subset thereof) and the reconstructed solution is compared to the reference. The reconstructed QCFD solution is compared cell-by-cell with the output of the modified OpenFOAM solver on the same Cartesian solid mask grid, so the two fields are expressed at the same locations and in the same format. The comparison is not expected to be exact, because the two solvers use different numerical methods to advance the solution on that grid. In the thermal case (V2D-1) the maximum absolute difference in temperature near the cylinder is of the order of 7×10^{-1} K, in the flow around a cylinder (V2D-2) the maximum difference in velocity magnitude near the cylinder is of the order of 2×10^{-2} m/s, and in the S-Bend channel (V2D-3) the maximum difference in velocity magnitude in the bend region is of the order of 1×10^{-1} m/s. These levels are consistent with the differences between two independent numerical methods operating on the same grid, and they confirm that the reconstructed QCFD solution reproduces the reference solution to the accuracy expected from the interface. The cases are organized by increasing geometric and physical complexity, starting from two-dimensional geometries with simple solid obstacles and progressing through immersed boundary representations of curved and complex shapes. The validation set presented in this deliverable consists of two-dimensional cases only. The industrial benchmark for the project is the three-dimensional battery electric vehicle (BEV) cooling serpentine, which is derived from the geometry of the S-Bend channel used in V2D-3; the 2D S-Bend is the first step of a progression that continues through a serpentine geometry and culminates in the full BEV cooling case defined in Deliverable D2.1 [14]. Each validation case is identified by a unique Digital Object Identifier (DOI) and is accompanied by complete input files, reference solutions, validation scripts, and documentation. All cases are accessible through the public git repository and the QCFD community data server hosted at the University of Hamburg. The benchmark problems used in this section are aligned with the core benchmark CFD set defined in Deliverable D1.1 [6] and extended in Deliverable D2.1 [14]. The three indirect-coupling cases (V2D-1, V2D-2, V2D-3) are accompanied by a step-by-step instruction set in the tutorial document [1], which describes how to prepare each case, generate the solid mask, assemble the QCFD inputs, and run the solver.

5.1 2D Problems - Mapping Conversion

This subsection presents the results obtained from selected two-dimensional benchmark cases relevant to thermo-fluid dynamic computational simulations. The chosen examples serve as a standardised test set for validating the conversion pipeline of the software interface between OpenFOAM and the QCFD framework, focusing on the fidelity of the geometry and boundary condition mapping rather than on physical accuracy. The cases are organised by increasing geometric complexity, starting from a heat conduction problem in a simple square domain, progressing through a flow around an internal body, and culminating in a curved channel geometry representative of industrial cooling applications.

Table 1 summarises the key attributes of all 2D benchmark problems, highlighting their physical modelling focus and engineering relevance. The benchmarks address several critical phenomena, including: (1) boundary conditions, (2) internal solid obstacles, (3) unsteady behaviour, (4) interplay between advection and diffusion, (5) nonlinear flow dynamics, and (6) curved geometries representative of industrial applications. The governing equations encompass a range of problem types, denoted as unsteady diffusion (uD) and unsteady convection-diffusion (uCD). In addition to the Finite Volume (FV) method employed in OpenFOAM, the table also includes references to the modified solvers that incorporate the immersed boundary term and serve as the Cartesian reference for the QCFD solver.

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

Table 1: Summary of 2D software validation cases.

2D Case	Eqn.	Spatial	Temporal	non/linear	Flow	Num.
V2D-1 Thermal Cylinder	uD	2D	unsteady	linear	—	FV
V2D-2 Flow around Cylinder	uCD	2D	unsteady	nonlinear	laminar	FV
V2D-3 S-Bend Channel	uCD	2D	unsteady	nonlinear	laminar	FV

5.1.1 V2D-1: Heat conduction in a square with a cylinder

The 2D heat conduction validation case is designed to test the software interface's capability to handle steady and transient heat conduction problems in a geometry with an internal obstacle. The geometry consists of a two-dimensional square domain spanning $[0, 1] \times [0, 1]$ [m] with a cylinder of diameter $D = 0.2$ [m] placed at its center. The heat is prescribed on each side of the square and diffuses through the domain towards the center, where the cylinder acts as an internal body with a prescribed temperature. This case validates the indirect coupling strategy, where the OpenFOAM geometry is mapped to a Cartesian grid via the immersed boundary solid mask method.

The temperature field is governed by the heat equation:

$$\rho c_p \frac{\partial T}{\partial t} - \nabla \cdot (k \nabla T) = 0 \quad \text{in } \Omega, \tag{5}$$

where ∇ denotes the gradient operator, ρ [kg/m^3] is the density, c_p [J/(kg·K)] is the specific heat capacity, k [W/(m·K)] is the thermal conductivity, and T [K] is the temperature. Introducing the thermal diffusivity $D_T = k/(\rho c_p)$, the heat equation can be written in the equivalent form:

$$\frac{\partial T}{\partial t} - D_T \nabla^2 T = 0 \quad \text{in } \Omega, \tag{6}$$

where $D_T = 0.025$ [m^2/s]. Figure 4 shows the computational domain and a zoom of the body-fitted mesh around the cylinder. The simulation is carried out on a 256×256 grid over the unit square, with grid spacings $\Delta x \approx 3.906 \times 10^{-3}$ [m] and $\Delta y \approx 3.906 \times 10^{-3}$ [m].

The boundary conditions are summarised in Table 2. A fixed temperature of 400 [K] is prescribed on each side of the square, while the cylinder surface is held at a fixed temperature of 350 [K]. The initial condition is a uniform temperature of 300 [K]. The simulation is advanced in time starting from an initial time $t_0 = 0$ [s], with a time step $\Delta t = 0.1$ [s], up to a final time $t_{\text{end}} = 5$ [s].

Table 2: Boundary conditions for the 2D square with cylinder case on the body-fitted grid.

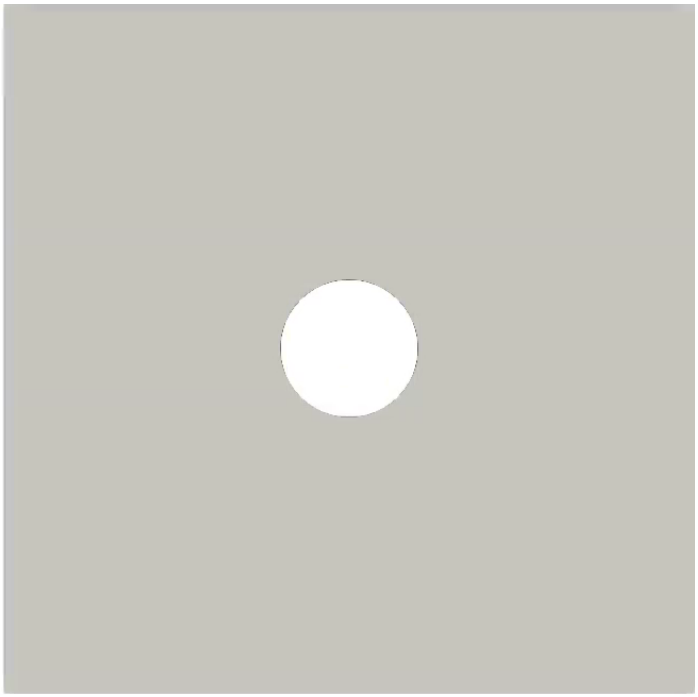
Boundary	Temperature
Left	$T = 400$ [K]
Right	$T = 400$ [K]
Top	$T = 400$ [K]
Bottom	$T = 400$ [K]
Cylinder	$T = 350$ [K]

The discretisation employs a cell-centered finite volume method with second-order accurate schemes for the diffusive fluxes.

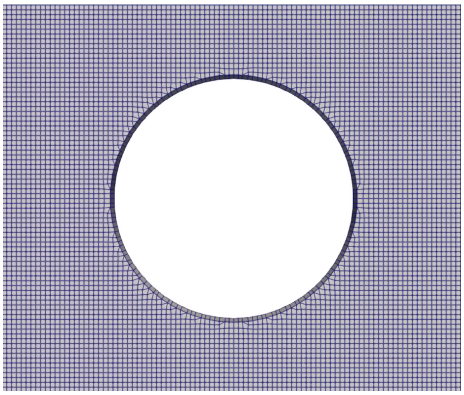
Figure 5 shows the reference temperature field obtained from the OpenFOAM simulation on the body-fitted grid, displayed on a 256×256 grid. This solution serves as the benchmark against which the interface-reconstructed solution will be compared.

For the indirect coupling approach, the body-fitted geometry is mapped onto a Cartesian grid using the immersed boundary method. The square domain remains the same as in the body-fitted case, and the

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085



(a) Full computational domain.



(b) Zoom of the body-fitted mesh around the cylinder.

Figure 4: Body-fitted mesh of the 2D square with the cylinder case. (a) Full computational domain and (b) zoom around the cylinder. The simulation is carried out on a 256×256 grid.

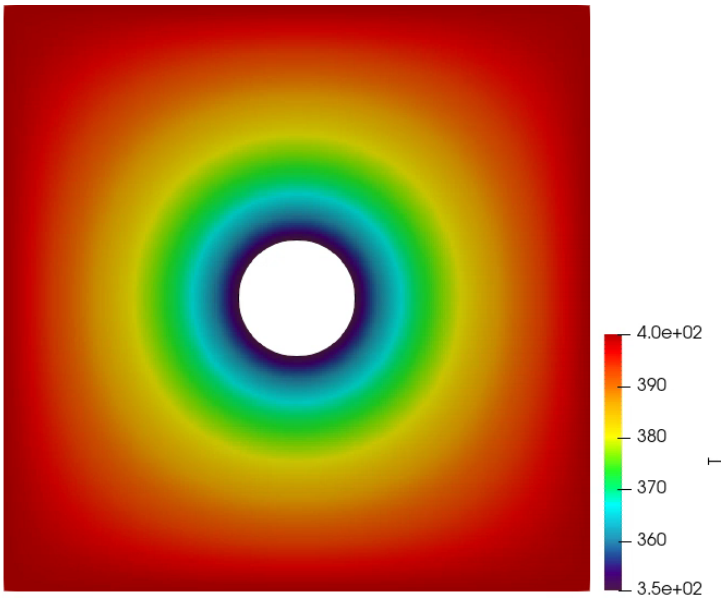


Figure 5: Reference temperature field from the OpenFOAM simulation on the body-fitted grid, displayed on a 256×256 grid.

cylinder is absorbed into the solid mask rather than being represented as a physical boundary. The modified governing equation is:

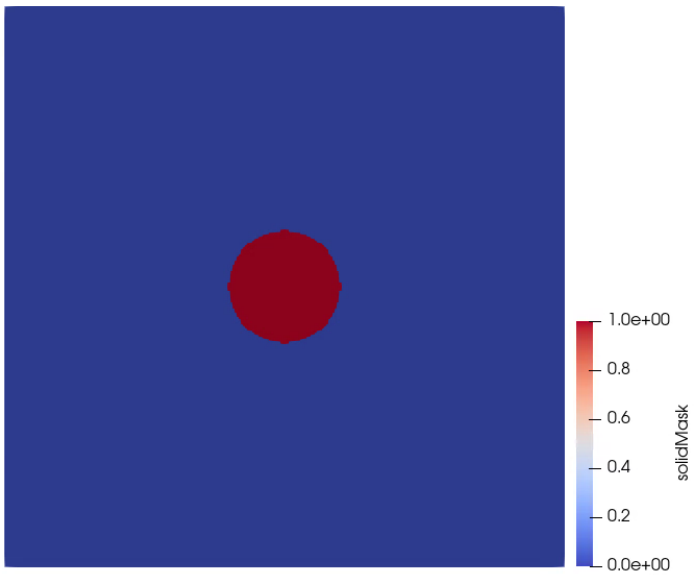
$$\frac{\partial T}{\partial t} - D_T \nabla^2 T + \mathcal{M}(\vec{\mathbf{x}})\, k\, (T - T_{\text{fixed}}) = 0 \quad \text{in } \Omega, \tag{7}$$

where $\mathcal{M}(\vec{\mathbf{x}})$ is the solid mask defined in Eq. (1), $k = 75$ [W/(m·K)] is the thermal conductivity, and $T_{\text{fixed}} = 350$ [K] is the fixed temperature imposed at the cylinder. The term $\mathcal{M}(\vec{\mathbf{x}})k(T - T_{\text{fixed}})$ enforces the fixed-temperature condition within the solid region: it vanishes in the fluid where $\mathcal{M} = 0$, and in

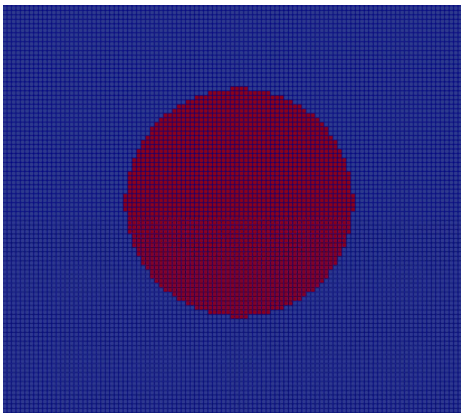
This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

the solid where $\mathcal{M} = 1$ it drives the local temperature towards T_{fixed}.

The solid mask is constructed on a Cartesian grid by sampling the body-fitted geometry, discretised on a 256×256 grid. The square domain remains the same as in the body-fitted case; only the cylinder is absorbed into the solid mask, so that the mask takes the value of unity inside the cylinder and zero in the surrounding domain. Figure 6 shows the Cartesian grid box used for the QCFD simulation, with the full computational domain on the left and a zoom around the cylinder on the right.



(a) Full computational domain.



(b) Zoom of the Cartesian grid around the cylinder.

Figure 6: Cartesian grid box used for the QCFD simulation with the immersed boundary method. (a) Full computational domain and (b) zoom around the cylinder. The simulation is carried out on a 256×256 grid.

The boundary conditions must also be transformed to match the topology of the Cartesian grid, which differs from that of the body-fitted mesh. On the body-fitted grid, the boundaries are associated with the physical geometry, while on the Cartesian grid they are expressed as the four sides of the square. The transformed boundary conditions for the Cartesian grid are summarised in Table 3.

Table 3: Boundary conditions for the 2D square with cylinder case on the Cartesian grid.

Boundary	Temperature
Left	$T = 400$ [K]
Right	$T = 400$ [K]
Top	$T = 400$ [K]
Bottom	$T = 400$ [K]

To provide a benchmark for the QCFD solver output, a reference solution is generated on the Cartesian grid using a modified OpenFOAM solver, `maskedLaplacianFoam`. This solver solves the heat equation with the immersed boundary solid mask term given in Eq. (7). The reference solution is stored in the OpenFOAM Cartesian format and serves as the comparison baseline. The resulting temperature field is shown in Figure 7, displayed on a 256×256 grid. For brevity, this finite-volume solution is denoted T_{FV} in the following.
The solid mask field, together with the boundary conditions and material properties extracted from the OpenFOAM case, is assembled into the input files required by the QCFD solver. These inputs consist of the geometry data, stored in Hierarchical Data Format (HDF5) and containing the solid mask field together with the grid metadata, and the configuration files specifying the boundary conditions and simulation parameters.

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

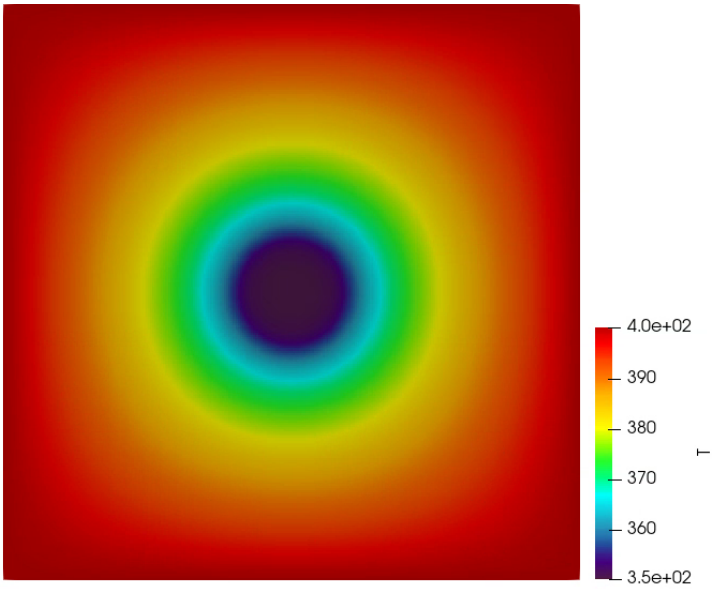


Figure 7: Reference temperature field T_{FV} from the `maskedLaplacianFoam` solution on the Cartesian grid with the immersed boundary solid mask, displayed on a 256×256 grid.

The QCFD solver reads these inputs and encodes the solid mask field into a matrix product state (MPS) representation via tensor train decomposition. The conversion pipeline therefore connects the body-fitted OpenFOAM description of the square with cylinder geometry to the Cartesian immersed boundary representation and finally to the MPS encoding used by the QCFD solver.

Note: The interface component described here is responsible only for preparing these inputs; the execution of the QCFD solver itself is performed externally. Interested users should contact the University of Hamburg (UHH) to obtain access to the QCFD solver and to arrange the execution of the simulations.

For the QCFD solver, the boundary conditions are translated into a compact algebraic form, in which each boundary is encoded as a pair consisting of a boundary type and an associated value. The boundary types are of Dirichlet and Neumann form. The resulting encoding is summarised in Table 4.

Table 4: Boundary condition encoding for the QCFD solver inputs. D denotes Dirichlet and N denotes Neumann.

Boundary	Location	T
x_{left}	Left	D (400)
x_{right}	Right	D (400)
y_{left}	Top	D (400)
y_{right}	Bottom	D (400)

Once the QCFD solver has been executed externally, it produces a solution in JLD format. Figure 8 shows the temperature field T at the final time $t_{\mathrm{end}} = 5$ [s] obtained from four different simulations. Panel (a) shows the DNS solution T_{DNS}. Panels (b), (c), and (d) show the MPS solutions T_{MPS} obtained at bond dimensions $\chi = 64$, $\chi = 48$, and $\chi = 32$, respectively.

To quantify the effect of the bond dimension on the accuracy of the QCFD solver, Figure 9 shows the absolute error $|T_{\mathrm{MPS}} - T_{\mathrm{DNS}}|$ per cell at the final time $t_{\mathrm{end}} = 5$ [s] between the MPS solutions and the DNS reference. The three panels correspond to bond dimensions $\chi = 64$, $\chi = 48$, and $\chi = 32$, respectively.

The error maps show that the MPS solution agrees with the DNS reference to better than 10^{-2} K in every case. At $\chi = 64$ the maximum per-cell error is 1.3×10^{-8} K, at round-off level. At $\chi = 48$ and

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

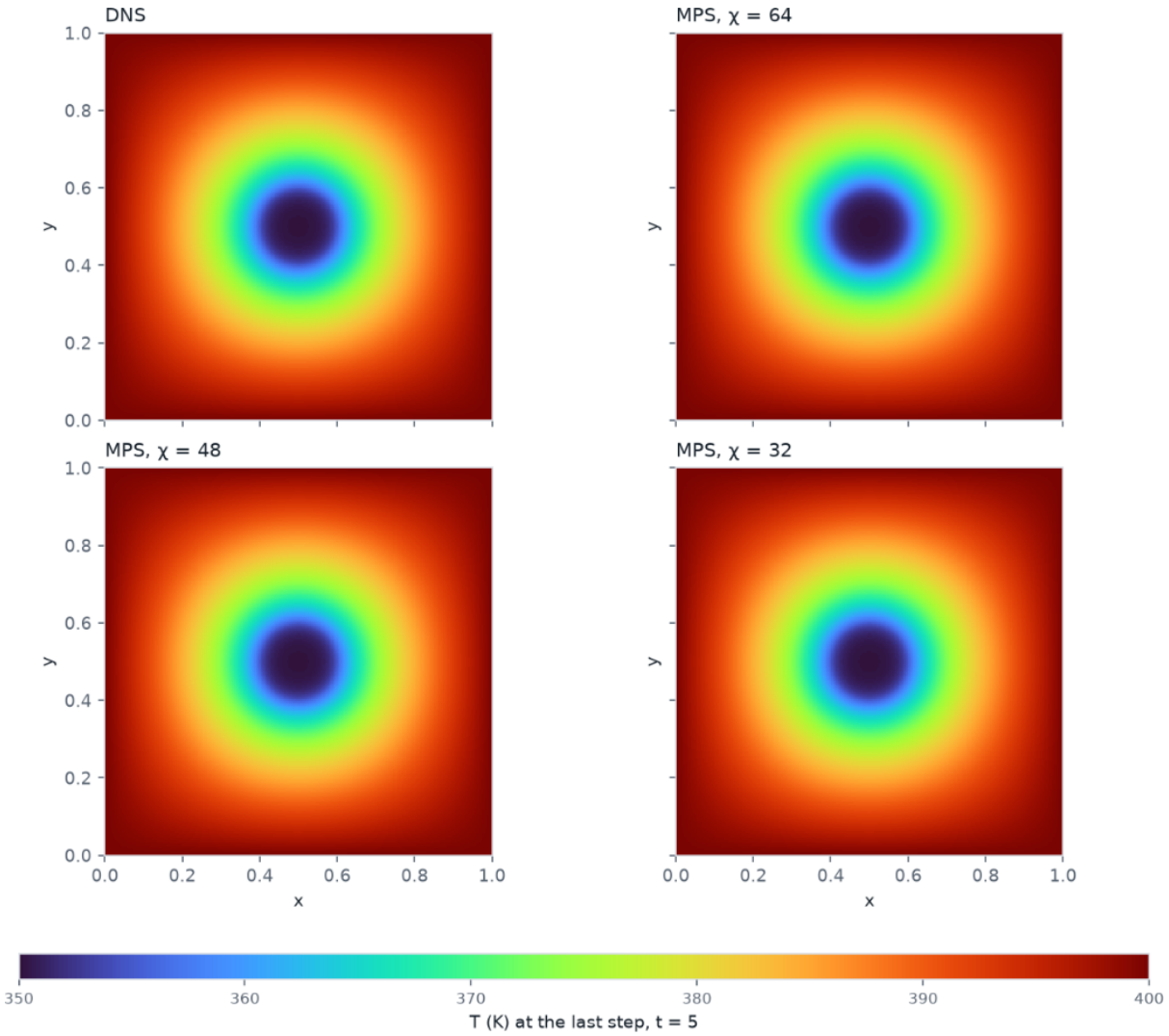


Figure 8: Temperature field T at the final time $t_{\text{end}} = 5$ [s]. Panel (a) shows the DNS solution T_{DNS}, while panels (b), (c), and (d) show the MPS solutions T_{MPS} at bond dimensions $\chi = 64$, 48, and 32, respectively. All fields are shown on the 256×256 Cartesian grid.

$\chi = 32$ the largest per-cell errors are located at the cylinder edge, where the temperature gradient is steepest. A quantitative summary is given in Table 5, which reports the relative deviation of the minimum and mean temperature, the maximum absolute error, and the error in the total thermal energy for each bond dimension.

Table 5: Error metrics for the heat conduction case at the final time $t_{\text{end}} = 5$ [s].

	δT_{min}	δT_{mean}	$\max \lvert T_{\text{MPS}} - T_{\text{DNS}} \rvert$ [K]	E_T
DNS	$T_{\text{min}} = 350.1261$ K	$T_{\text{mean}} = 387.6491$ K	—	—
$\chi = 64$	7.4×10^{-14}	5.2×10^{-12}	1.3×10^{-8}	1.2×10^{-11}
$\chi = 48$	3.9×10^{-10}	1.1×10^{-12}	9.0×10^{-6}	1.4×10^{-9}
$\chi = 32$	2.9×10^{-7}	4.2×10^{-9}	1.3×10^{-3}	2.2×10^{-7}

Figure 10 summarises the dependence of the error on the bond dimension. The three curves show the relative deviation of the minimum and mean temperature and the error in the total thermal energy at the final step, plotted against χ. The dashed horizontal lines mark the 10^{-4} and 10^{-2} thresholds.
The validation of the full conversion pipeline relies on comparing the solutions defined on the Cartesian solid mask grid. The first is the finite-volume solution T_{FV} obtained from `maskedLaplacianFoam`, shown

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

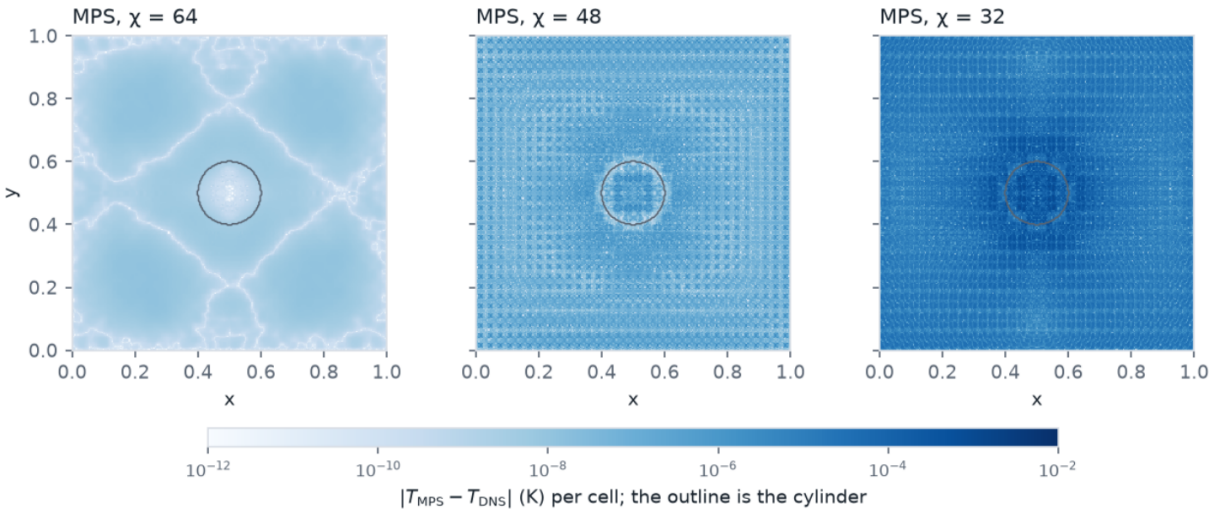


Figure 9: Absolute error $|T_{\mathrm{MPS}} - T_{\mathrm{DNS}}|$ per cell at the final time $t_{\mathrm{end}} = 5$ [s] between the MPS solutions and the DNS reference, for bond dimensions $\chi = 64$ (left), $\chi = 48$ (middle), and $\chi = 32$ (right). The error is shown on a logarithmic scale from 10^{-12} K to 10^{-2} K. The grey circle marks the edge of the cylinder.

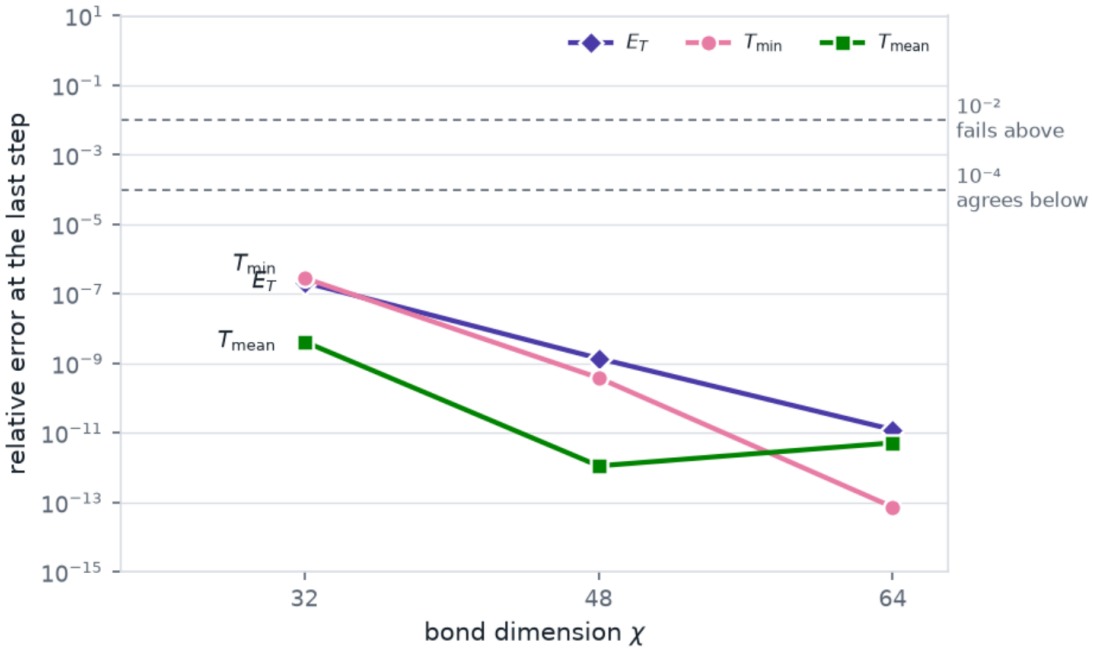


Figure 10: Relative error against bond dimension χ at the final step $t_{\mathrm{end}} = 5$ [s]. The curves show E_T and the relative deviations δT_{min} and δT_{mean}. The dashed horizontal lines mark the 10^{-4} and 10^{-2} thresholds.

in Figure 7 and already available in the OpenFOAM Cartesian format. The second is the reconstructed MPS solution T_{Qrec}, obtained by mapping the MPS output onto the Cartesian solid mask grid.

For this purpose, a reconstruction algorithm has been developed as part of the interface. The reconstruction algorithm reads the JLD output produced by the QCFD solver and converts it back into the OpenFOAM Cartesian solid mask format. The reconstruction maps the tensor network solution onto the grid arrangement used by `maskedLaplacianFoam`, so that the temperature is expressed on the same grid, at the same field locations, and in the same file format as T_{FV}. This ensures that T_{Qrec} and T_{FV} are directly comparable without any further interpolation or post-processing.

The visualisation of Figure 11 shows the absolute error $|T_{\mathrm{Qrec}} - T_{\mathrm{FV}}|$ at the final time $t_{\mathrm{end}} = 5$ [s] on the

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

256×256 Cartesian solid mask grid. The reconstructed MPS solution T_{Qrec} is obtained by mapping the MPS output at bond dimension $\chi = 64$ onto the solid mask grid, and T_{FV} is the finite-volume solution from `maskedLaplacianFoam`.

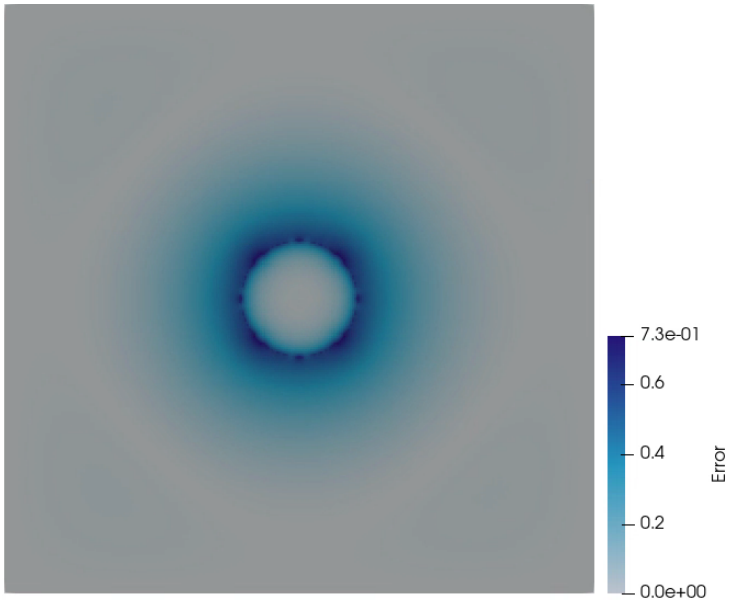


Figure 11: Absolute error $|T_{\mathrm{Qrec}} - T_{\mathrm{FV}}|$ at the final time $t_{\mathrm{end}} = 5$ [s] on the 256×256 Cartesian solid mask grid. The reconstructed MPS solution T_{Qrec} is obtained from the MPS output at bond dimension $\chi = 64$, and T_{FV} is the finite-volume solution from `maskedLaplacianFoam`.

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

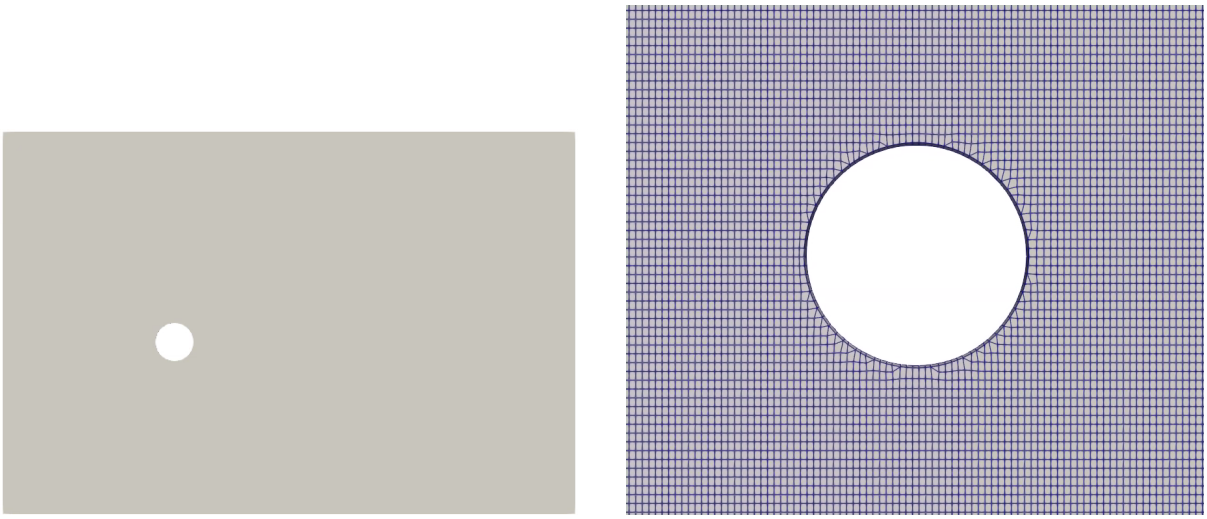
5.1.2 V2D-2: Flow around cylinder

The 2D flow around a cylinder validation case is designed to test the software interface's capability to handle laminar incompressible Navier-Stokes flows around a cylinder. The geometry consists of a two-dimensional cylinder of diameter $D = 1$ [m], at $(4.5, 4.5)$ [m] inside a rectangular channel spanning $[0, 15] \times [0, 10]$ [m], representing a classical benchmark for laminar vortex shedding and wake development. This case validates the indirect coupling strategy, where the OpenFOAM geometry is mapped to a Cartesian grid via the immersed boundary solid mask method. The flow remains laminar throughout the simulation, and no turbulence modelling is employed.

The flow is governed by the incompressible Navier-Stokes equations:

$$\begin{aligned} \rho\frac{\partial \vec{\mathbf{u}}}{\partial t} + \rho(\vec{\mathbf{u}}\cdot\nabla)\vec{\mathbf{u}} - \nabla\cdot(\mu\nabla\vec{\mathbf{u}} + \nabla\vec{\mathbf{u}}^T) + \nabla p = 0 \quad \text{in } \Omega, \\ \nabla\cdot\vec{\mathbf{u}} = 0 \quad \text{in } \Omega, \end{aligned} \tag{8}$$

where ∇ denotes the gradient operator, ρ [kg/m^3] is the density, μ [kg/(m·s)] is the fluid's dynamic viscosity, $\vec{\mathbf{u}} = [u, v]^T$ [m/s] is the velocity vector, and p [m^2/s^2] is the pressure. Figure 20 shows the computational domain and a zoom of the body-fitted mesh around the bend. The simulation is carried out on a body-fitted grid of 512×256 cells, with grid spacings $\Delta x \approx 0.0293$ [m] and $\Delta y = 0.0391$ [m].



(a) Full computational domain.

(b) Zoom of the body-fitted mesh around the cylinder.

Figure 12: Body-fitted mesh of the 2D cylinder case. (a) Full computational domain and (b) zoom around the cylinder. The simulation is carried out on a 512×256 grid.

The boundary conditions are summarised in Table 6. A uniform velocity profile with reference velocity $U_{\text{in}} = 1$ [m/s] is prescribed at the inlet, no-slip boundary conditions are imposed on the cylinder surface, and free-slip or symmetry conditions are applied on the lateral walls. The outlet is treated as a pressure outlet with zero gauge pressure. The initial conditions are set to zero for all velocity components and the pressure, except at the inlet where the uniform profile is imposed. The material properties are $\rho = 1$ [kg/m^3] and $\nu = \mu/\rho = 0.05$ [m^2/s], corresponding to a Reynolds number of $\text{Re} = 20$ based on the cylinder diameter. The simulation is carried out on a body-fitted grid of 512×256 cells, with grid spacings $\Delta x \approx 0.0293$ [m] and $\Delta y = 0.0391$ [m]. The simulation is advanced in time starting from an initial time $t_0 = 0$ [s], with a time step $\Delta t = 0.01$ [s], up to a final time $t_{\text{end}} = 2.25$ [s].

The discretisation employs a cell-centred finite volume method with second-order accurate schemes for both diffusive and convective fluxes. The pressure-velocity coupling is handled using the PISO algorithm. Figure 13 shows the reference velocity magnitude field obtained from the OpenFOAM simulation on the body-fitted grid, displayed on a 512×256 grid. This solution serves as the benchmark against which the interface-reconstructed solution will be compared.
For the indirect coupling approach, the body-fitted geometry is mapped onto a Cartesian grid using the immersed boundary method. The rectangular domain remains the same as in the body-fitted case, and

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

Table 6: Boundary conditions for the 2D cylinder case on the body-fitted grid.

Boundary	Velocity	Pressure
Inlet	$u = U_{\text{in}},\ v = 0$	$\partial p/\partial n = 0$
Outlet	$\partial u/\partial n = 0$	$p = 0$
Cylinder	$\vec{\mathbf{u}} = 0$	$\partial p/\partial n = 0$
Top	$\partial u/\partial n = 0$	$\partial p/\partial n = 0$
Bottom	$\partial u/\partial n = 0$	$\partial p/\partial n = 0$

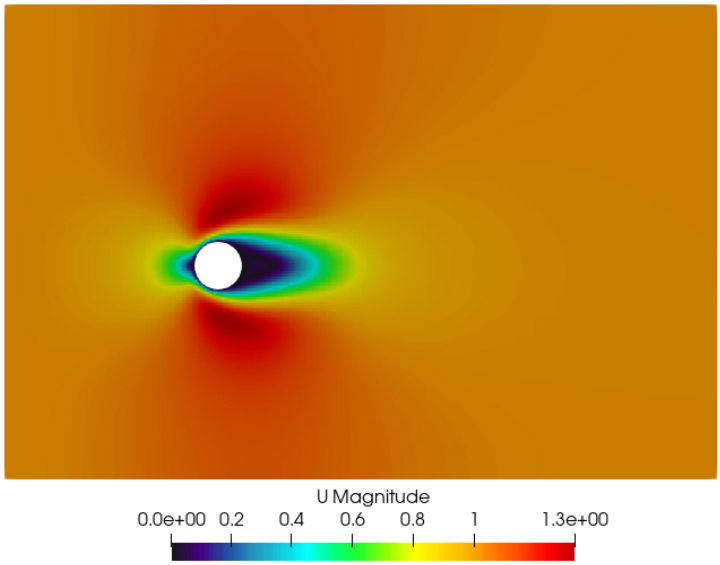


Figure 13: Reference velocity magnitude field from the OpenFOAM simulation on the body-fitted grid, displayed on a 512×256 grid.

the cylinder is absorbed into the solid mask rather than being represented as a physical boundary. The modified governing equations are:

$$\begin{aligned} \rho\frac{\partial \vec{\mathbf{u}}}{\partial t} + \rho(\vec{\mathbf{u}}\cdot\nabla)\vec{\mathbf{u}} - \nabla\cdot(\mu\nabla\vec{\mathbf{u}} + \nabla\vec{\mathbf{u}}^T) + \nabla p + \beta\mathcal{M}(\vec{\mathbf{x}})\vec{\mathbf{u}} = 0 \quad &\text{in } \Omega, \\ \nabla\cdot\vec{\mathbf{u}} = 0 \quad &\text{in } \Omega, \end{aligned} \tag{9}$$

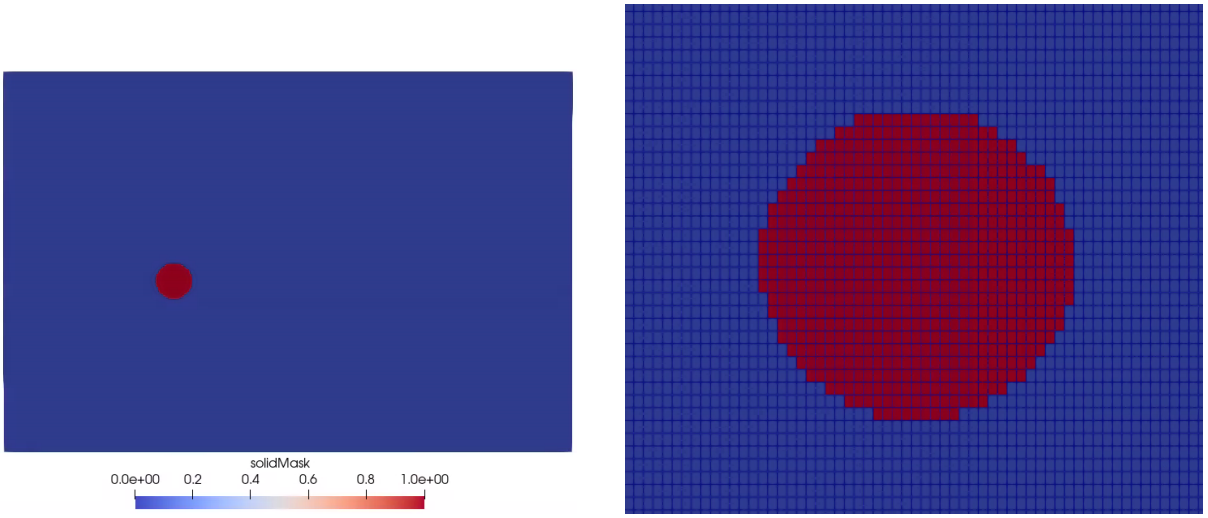
where β [1/s] is the penalisation parameter of the immersed boundary forcing, and $\mathcal{M}(\vec{\mathbf{x}})$ is the solid mask defined in Eq. (1). The penalisation term $\beta\mathcal{M}(\vec{\mathbf{x}})\vec{\mathbf{u}}$ drives the velocity to zero within the cylinder, effectively imposing the no-slip condition on the cylinder surface. The penalisation parameter is computed as

$$\beta = \frac{\eta}{\Delta t}\left(1 + \nu\Delta t\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)\right) \tag{10}$$

where η is a dimensionless parameter controlling the penalisation strength. In the cylinder case, the penalisation constant is set to $\beta = 1.15 \times 10^6$ [1/s], that enforce the no-slip condition at the cylinder surface while remaining stable with the chosen time step. The solid mask is constructed on a Cartesian grid by sampling the body-fitted geometry, discretised on a 512×256 grid. The rectangular domain remains the same as in the body-fitted case; only the cylinder is absorbed into the solid mask, so that the mask takes the value of unity inside the cylinder and zero in the surrounding fluid. Figure 14 shows the Cartesian grid box used for the QCFD simulation, with the full computational domain on the left and a zoom around the cylinder on the right.

Because the QCFD solver employs a staggered grid arrangement, the solid mask is generated separately for each field location: at cell centres for the pressure, and at the staggered face locations for each velocity component, computed as the average of the adjacent cell-centred mask values. The mask takes the value

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085



(a) Full computational domain.

(b) Zoom of the Cartesian grid around the cylinder.

Figure 14: Cartesian grid box used for the QCFD simulation with the immersed boundary method. (a) Full computational domain and (b) zoom around the cylinder. The simulation is carried out on a 512×256 grid.

of unity inside the cylinder and zero in the surrounding fluid.

The boundary conditions must also be transformed to match the topology of the Cartesian grid, which differs from that of the body-fitted mesh. On the body-fitted grid, the boundaries are associated with the physical geometry, while on the Cartesian grid they are expressed as the four sides of the rectangular box. The transformed boundary conditions for the Cartesian grid are summarised in Table 7.

Table 7: Boundary conditions for the 2D cylinder case on the Cartesian grid.

Boundary	Velocity	Pressure
Inlet (left)	$u = U_{\text{in}},\ v = 0$	$\partial p/\partial n = 0$
Outlet (right)	$\partial u/\partial n = 0$	$p = 0$
Top	$\partial u/\partial n = 0$	$\partial p/\partial n = 0$
Bottom	$\partial u/\partial n = 0$	$\partial p/\partial n = 0$

To provide a benchmark for the QCFD solver output, a reference solution is generated on the Cartesian grid using a modified OpenFOAM solver, `maskedIcoFoam`. This solver solves the incompressible laminar Navier-Stokes equations with the immersed boundary penalisation term given in Eq. (9). The simulation is carried out on a uniform Cartesian grid of 512×256 cells, with the same material properties and time-stepping parameters used for the body-fitted case, namely $\rho = 1$ [kg/m^3], $\nu = 0.05$ [m^2/s], $t_0 = 0$ [s], $\Delta t = 0.01$ [s], and $t_{\text{end}} = 2.25$ [s]. The reference solution is stored in the OpenFOAM Cartesian format and serves as the comparison baseline. The resulting velocity magnitude field is shown in Figure 15. For brevity, the velocity magnitude from this finite-volume solution is denoted $\|\vec{\mathbf{u}}\|_{\text{FV}}$ in the following.

The solid mask fields, together with the boundary conditions and material properties extracted from the OpenFOAM case, are assembled into the input files required by the QCFD solver. These inputs consist of the geometry data, stored in Hierarchical Data Format (HDF5) and containing the staggered solid mask fields together with the grid metadata, and the configuration files specifying the boundary conditions and simulation parameters.

The QCFD solver reads these inputs and encodes the solid mask fields into matrix product state (MPS) representations via tensor train decomposition. The conversion pipeline therefore connects the body-fitted OpenFOAM description of the cylinder geometry to the Cartesian immersed boundary representation and finally to the MPS encoding used by the QCFD solver.

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

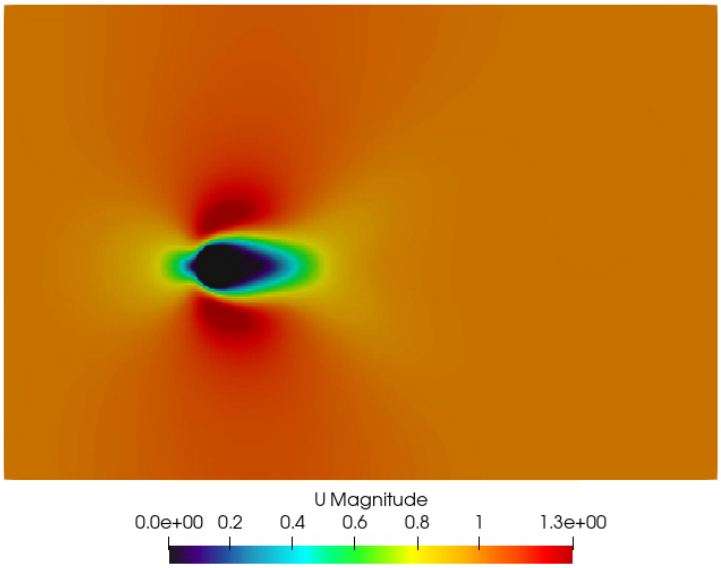


Figure 15: Reference velocity magnitude field $\|\vec{\mathbf{u}}\|_{\mathrm{FV}}$ from the `maskedIcoFoam` solution on the Cartesian grid with the immersed boundary penalisation, displayed on a 512×256 grid.

Note: The interface component described here is responsible only for preparing these inputs; the execution of the MPS solver itself is performed externally. Interested users should contact the University of Hamburg (UHH) to obtain access to the MPS solver and to arrange the execution of the simulations.

For the QCFD solver, the boundary conditions are translated into a compact algebraic form, in which each boundary of each velocity component is encoded as a pair consisting of a boundary type and an associated value. The boundary types are of Dirichlet and Neumann form. The resulting encoding is summarised in Table 8.

Table 8: Boundary condition encoding for the QCFD solver inputs. D denotes Dirichlet and N denotes Neumann.

Boundary	Location	u	v	p
x_{left}	Inlet	D (U_{in})	D (0)	N
x_{right}	Outlet	N	N	D (0)
y_{left}	Top	N	D (0)	N
y_{right}	Bottom	N	D (0)	N

Once the QCFD solver has been executed externally, it produces a solution in JLD format. Figure 16 shows the velocity magnitude field at the final time $t_{\mathrm{end}} = 2.25$ [s] obtained from four different simulations. Panel (a) shows the DNS solution $\|\vec{\mathbf{u}}\|_{\mathrm{DNS}}$. Panels (b), (c), and (d) show the MPS solutions $\|\vec{\mathbf{u}}\|_{\mathrm{MPS}}$ obtained at bond dimensions $\chi = 64$, $\chi = 48$, and $\chi = 32$, respectively.

To quantify the effect of the bond dimension on the accuracy of the QCFD solver, Figure 17 shows the absolute error $|\|\vec{\mathbf{u}}\|_{\mathrm{MPS}} - \|\vec{\mathbf{u}}\|_{\mathrm{DNS}}|$ per cell at the final time $t_{\mathrm{end}} = 2.25$ [s] between the MPS solutions and the DNS reference. The three panels correspond to bond dimensions $\chi = 64$, $\chi = 48$, and $\chi = 32$, respectively.

The error maps show how the accuracy of the MPS solution depends on the bond dimension, with the largest errors concentrated near the cylinder surface and in the wake region where the velocity gradients are steepest. A quantitative summary is given in Table 9, which reports the relative deviation of the drag coefficient C_D, the pressure difference $\Delta p_{\mathrm{probe}}$, the pressure drop Δp, the absolute difference of the lift coefficient C_L, and the domain maximum $D_{\max}$ for each bond dimension. A second summary, given

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

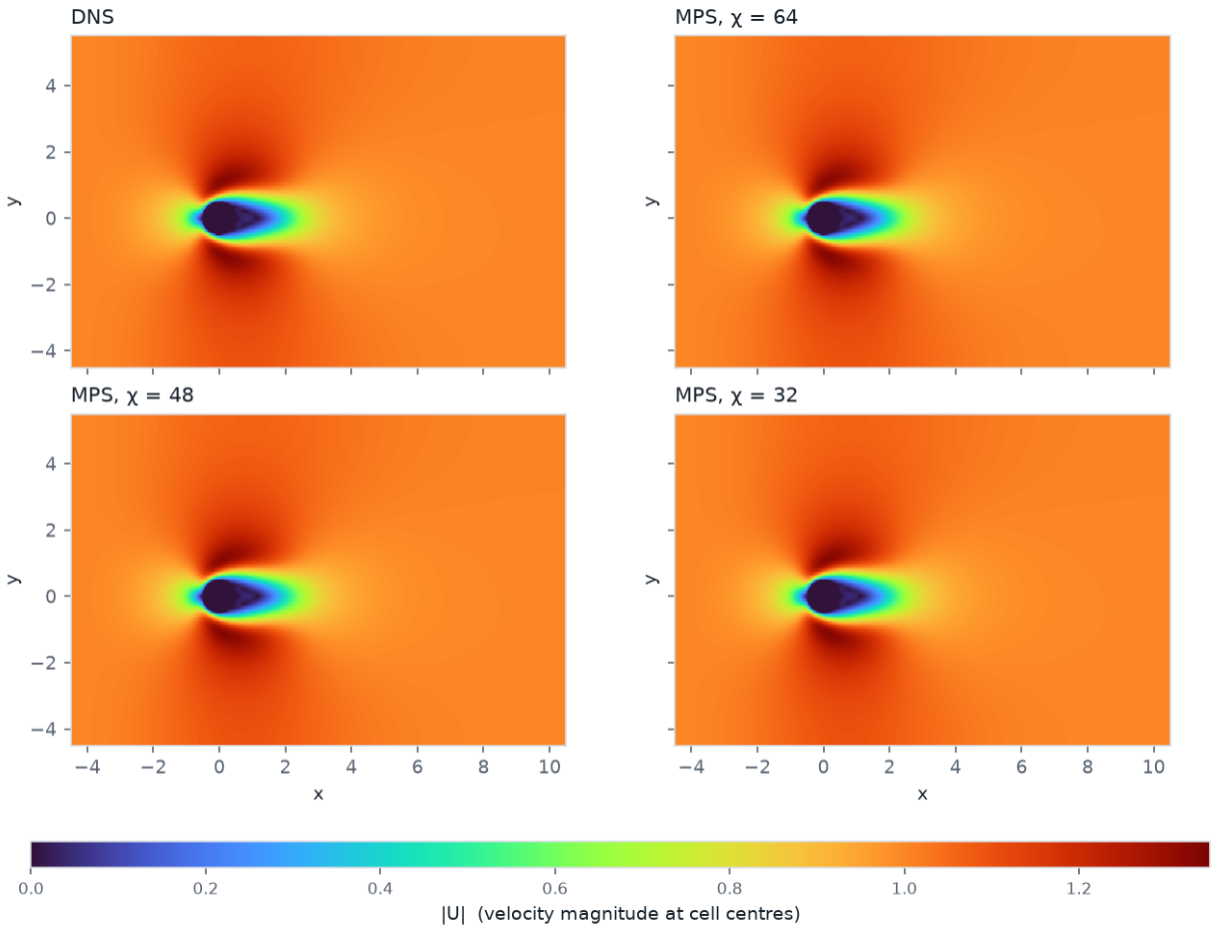


Figure 16: Velocity magnitude field at the final time $t_{\text{end}} = 2.25$ [s]. Panel (a) shows the DNS solution $\|\vec{\mathbf{u}}\|_{\text{DNS}}$, while panels (b), (c), and (d) show the MPS solutions $\|\vec{\mathbf{u}}\|_{\text{MPS}}$ at bond dimensions $\chi = 64$, 48, and 32, respectively. All fields are shown on the 512×256 Cartesian grid.

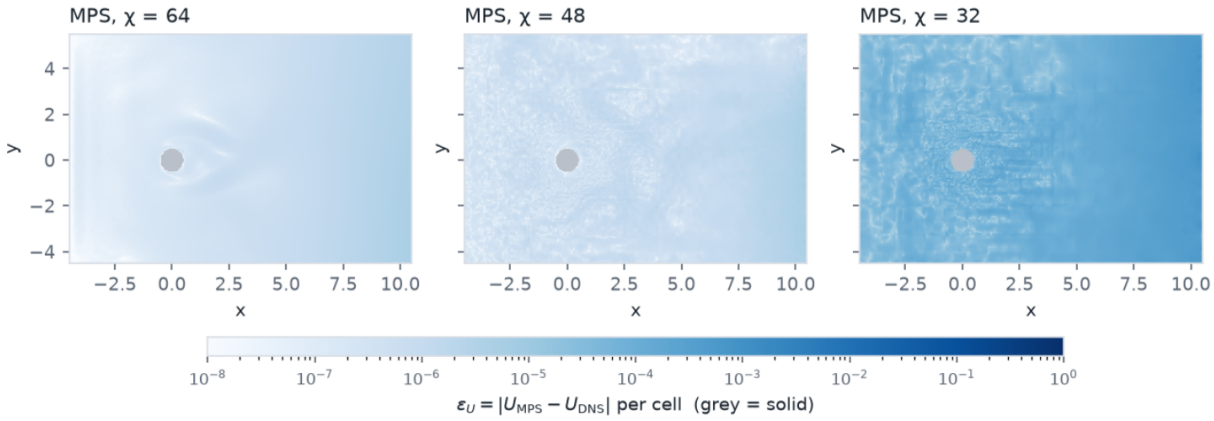


Figure 17: Absolute error $|\|\vec{\mathbf{u}}\|_{\text{MPS}} - \|\vec{\mathbf{u}}\|_{\text{DNS}}|$ per cell at the final time $t_{\text{end}} = 2.25$ [s] between the MPS solutions and the DNS reference, for bond dimensions $\chi = 64$ (left), $\chi = 48$ (middle), and $\chi = 32$ (right). The error is shown on a logarithmic scale. The grey circle marks the edge of the cylinder.

in Table 10, reports the field errors E_u, E_v, and E_p at the last step over the fluid domain.

Figure 18 summarises the dependence of the error on the bond dimension. The curves show the relative deviations of C_D and Δp, together with the field errors E_u and E_p, plotted against χ. The dashed horizontal lines mark the 10^{-4} and 10^{-2} thresholds.
Both $\chi = 64$ and $\chi = 48$ reproduce the DNS velocity, u to within about 10^{-6} and v to within 1.4×10^{-5}.

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

Table 9: Error metrics for the flow around a cylinder at the final time $t_{\text{end}} = 2.25$ [s]. C_D, Δp_{probe}, and Δp are reported as relative deviations δq. $|C_L|$ is the absolute difference. $D_{\max}$ is the run's own value at the last step. The DNS row gives the reference values.

	C_D	Δp_{probe}	Δp	$\lvert C_L \rvert$	$D_{\max}$
DNS	2.395882	1.065662	0.1318334	$C_L = 0.006408$	1.9×10^{-14}
$\chi = 64$	1.5×10^{-6}	9.2×10^{-7}	8.2×10^{-7}	1.5×10^{-5}	6.9×10^{-7}
$\chi = 48$	2.8×10^{-5}	1.4×10^{-5}	8.3×10^{-4}	7.4×10^{-5}	5.8×10^{-5}
$\chi = 32$	2.8×10^{-3}	1.1×10^{-3}	1.1×10^{-1}	2.9×10^{-3}	6.4×10^{-3}

Table 10: Field errors E_ϕ at the last step, over the fluid domain.

MPS χ	E_u	E_v	E_p
$\chi = 64$	9.2×10^{-7}	1.4×10^{-5}	2.9×10^{-4}
$\chi = 48$	8.5×10^{-7}	1.3×10^{-5}	1.3×10^{-3}
$\chi = 32$	1.2×10^{-4}	1.8×10^{-3}	5.0×10^{-2}

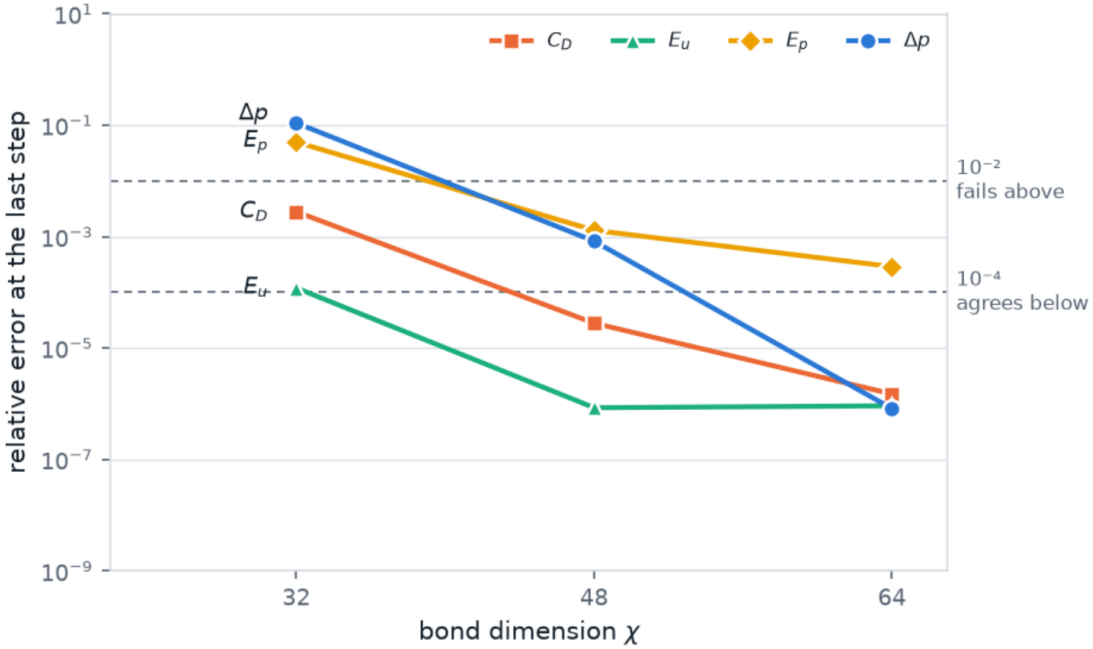


Figure 18: Relative error against χ at the last step: C_D and Δp as δq, E_u and E_p as field errors. The dashed lines are the 10^{-4} and 10^{-2} thresholds of the tables.

The pressure field error reaches 3×10^{-4} only at $\chi = 64$. At $\chi = 48$ it is 1.3×10^{-3}, and the Δp error is marginal. At $\chi = 32$ the error spreads over the whole domain and grows toward the outlet.

The validation of the full conversion pipeline relies on comparing two solutions that are both defined on the Cartesian solid mask grid. The first is the reference solution obtained from the modified OpenFOAM solver `maskedIcoFoam`, which was described in the previous section and is already available in the OpenFOAM Cartesian format. The second is the solution produced by the QCFD solver, which is expressed in the tensor network representation and must therefore be mapped back onto the Cartesian solid mask grid before any comparison can be made.

For this purpose, a reconstruction algorithm has been developed as part of the interface. The reconstruction algorithm reads the JLD output produced by the QCFD solver and converts it back into the OpenFOAM Cartesian solid mask format. The reconstruction maps the tensor network solution onto the staggered grid arrangement used by `maskedIcoFoam`, so that both velocity and pressure are expressed on the same grid, at the same field locations, and in the same file format as the reference solution. This en-

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

sures that the two solutions are directly comparable without any further interpolation or post-processing.

The visualisation of Figure 19 shows the absolute error $|\|\vec{\mathbf{u}}\|_{\text{Qrec}} - \|\vec{\mathbf{u}}\|_{\text{FV}}|$ at the final time $t_{\text{end}} = 2.25$ [s] on the 512×256 Cartesian solid mask grid. The reconstructed MPS solution $\|\vec{\mathbf{u}}\|_{\text{Qrec}}$ is obtained by mapping the MPS output at bond dimension $\chi = 64$ onto the solid mask grid, and $\|\vec{\mathbf{u}}\|_{\text{FV}}$ is the finite-volume solution from `maskedIcoFoam`.

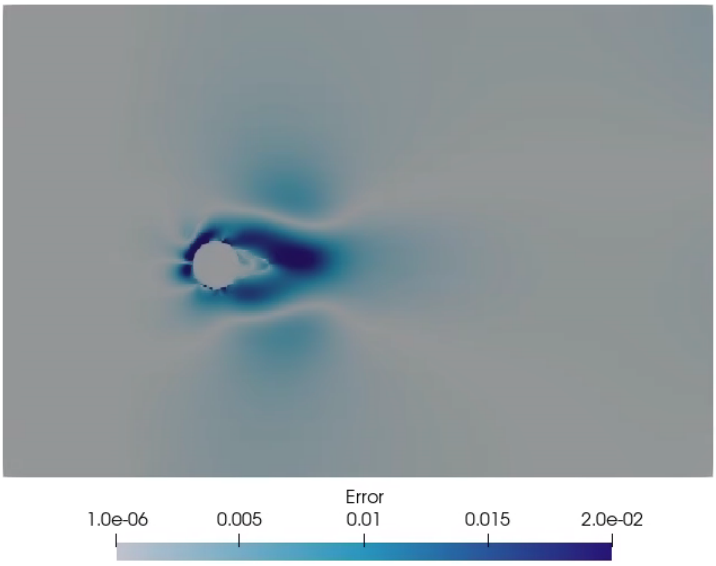


Figure 19: Absolute error $|\|\vec{\mathbf{u}}\|_{\text{Qrec}} - \|\vec{\mathbf{u}}\|_{\text{FV}}|$ at the final time $t_{\text{end}} = 2.25$ [s] on the 512×256 Cartesian solid mask grid. The reconstructed MPS solution $\|\vec{\mathbf{u}}\|_{\text{Qrec}}$ is obtained from the MPS output at bond dimension $\chi = 64$, and $\|\vec{\mathbf{u}}\|_{\text{FV}}$ is the finite-volume solution from `maskedIcoFoam`.

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

5.1.3 V2D-3: S-Bend channel

The 2D S-Bend validation case is designed to test the software interface's capability to handle incompressible Navier-Stokes flows in a curved geometry. The geometry consists of a two-dimensional S-shaped bend channel of uniform width, featuring a straight inlet section transitioning into an S-shaped bend, followed by a straight outlet section. This case represents a simplified example of the Battery Electric Vehicle (BEV) cooling case and validates the indirect coupling strategy, where the OpenFOAM geometry is mapped to a Cartesian grid via the immersed boundary solid mask method. The flow remains laminar throughout the simulation, and no turbulence modelling is employed.

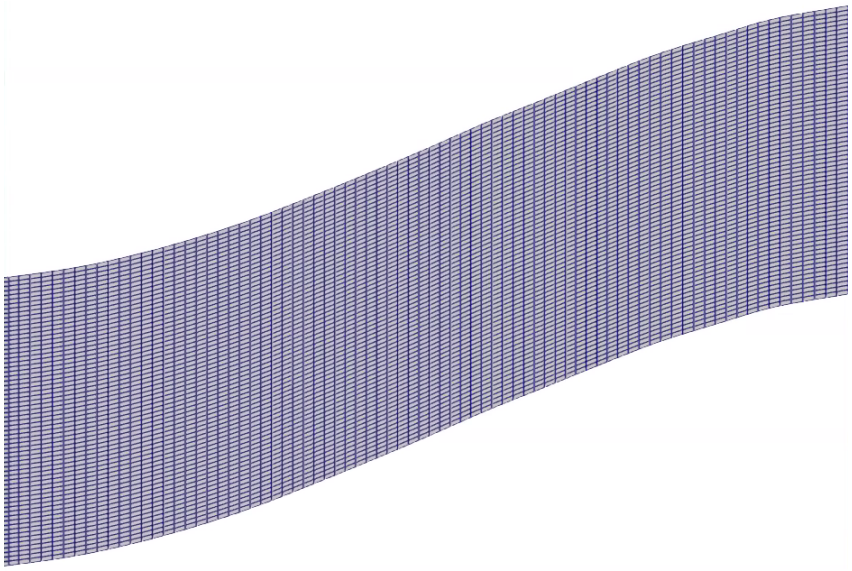
The flow is governed by the incompressible Navier-Stokes equations:

$$\begin{aligned} \rho\frac{\partial \vec{\mathbf{u}}}{\partial t} + \rho(\vec{\mathbf{u}}\cdot\nabla)\vec{\mathbf{u}} - \nabla\cdot(\mu\nabla\vec{\mathbf{u}} + \nabla\vec{\mathbf{u}}^T) + \nabla p = 0 \quad \text{in } \Omega, \\ \nabla\cdot\vec{\mathbf{u}} = 0 \quad \text{in } \Omega, \end{aligned} \tag{11}$$

where ∇ denotes the gradient operator, ρ [kg/m^3] is the density, μ [kg/(m·s)] is the fluid's dynamic viscosity, $\vec{\mathbf{u}} = [u, v]^T$ [m/s] is the velocity vector, and p [m^2/s^2] is the pressure.
Figure 20 shows the computational domain and a zoom of the body-fitted mesh around the bend. The simulation is carried out on a 512×64 grid.



(a) Full computational domain.



(b) Zoom of the body-fitted mesh around the bend.

Figure 20: Body-fitted mesh of the 2D S-Bend case. (a) Full computational domain and (b) zoom around the bend. The simulation is carried out on a 512×64 grid.

The boundary conditions are summarised in Table 11. At the inlet, a parabolic velocity profile is prescribed:

$$u_{\text{in}}(y) = V_{\text{ref}}\left(1 - \left(\frac{2(y - y_c)}{H}\right)^2\right) \quad \text{[m/s]} \tag{12}$$

with the peak velocity $V_{\text{ref}} = 1.5$ [m/s], the channel height $H = 0.05$ [m], and the channel centreline $y_c = 0.025$ [m]. The mean velocity over the channel cross-section is $\bar{u} = (2/3)V_{\text{ref}} = 1$ [m/s]. No-slip boundary conditions are imposed at the solid walls. The initial conditions are set to zero for all velocity

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

components and the pressure. The material properties are $\rho = 1$ [kg/m^3] and $\nu = \mu/\rho = 1.0 \times 10^{-4}$ [m^2/s], corresponding to a Reynolds number of $\mathrm{Re} = \bar{u}H/\nu = 500$. The simulation is advanced in time starting from an initial time $t_0 = 0$ [s], with a time step $\Delta t = 3.335{\times}10^{-4}$ [s], up to a final time $t_{\mathrm{end}} = 2$ [s].

Table 11: Boundary conditions for the 2D Sbend case.

Boundary	Velocity	Pressure
Inlet	$u = u(y),\ v = 0$	$\partial p/\partial n = 0$
Outlet	$\partial u/\partial n = 0,\ \partial v/\partial n = 0$	$p = 0$
Walls	$u = 0,\ v = 0$	$\partial p/\partial n = 0$

The discretisation employs a cell-centred finite volume method with second-order accurate schemes for both diffusive and convective fluxes. The pressure-velocity coupling is handled using the PISO algorithm. Figure 21 shows the reference velocity magnitude field obtained from the OpenFOAM simulation on the body-fitted grid, displayed on a 512×64 grid. This solution serves as the benchmark against which the interface-reconstructed solution will be compared.

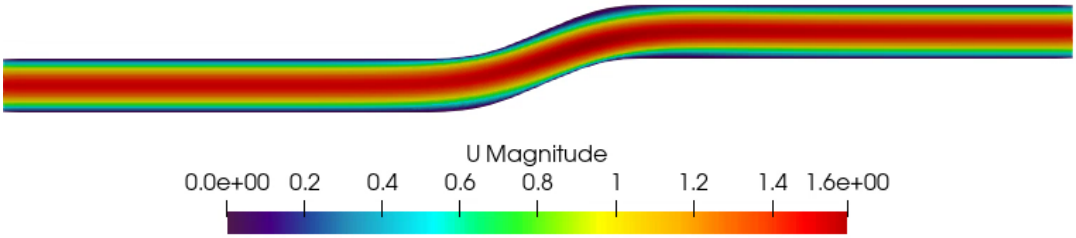


Figure 21: Reference velocity magnitude field from the OpenFOAM simulation on the body-fitted grid, displayed on a 512×64 grid.

For the indirect coupling approach, the body-fitted geometry is mapped onto a Cartesian grid using the immersed boundary method. The geometry is embedded in a rectangular box of length 1 [m] and height 0.2 [m], in which the S-like geometry is realised based on the penalisation approach. The fluid domain is defined by the wetted height $\delta = 0.05$ [m]. The modified governing equations are:

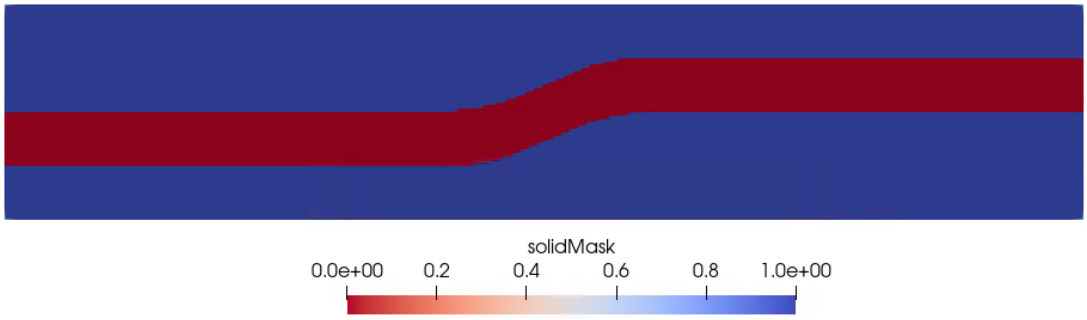
$$\begin{aligned} \rho\frac{\partial \vec{\mathbf{u}}}{\partial t} + \rho(\vec{\mathbf{u}}\cdot\nabla)\vec{\mathbf{u}} - \nabla\cdot(\mu\nabla\vec{\mathbf{u}} + \nabla\vec{\mathbf{u}}^T) + \nabla p + \beta\mathcal{M}(\vec{\mathbf{x}})\vec{\mathbf{u}} = 0 \quad &\text{in } \Omega, \\ \nabla\cdot\vec{\mathbf{u}} = 0 \quad &\text{in } \Omega, \end{aligned} \tag{13}$$

where β [1/s] is the penalisation parameter of the immersed boundary forcing, and $\mathcal{M}(\vec{\mathbf{x}})$ is the solid mask defined in Eq. (1). The penalisation term $\beta\mathcal{M}(\vec{\mathbf{x}})\vec{\mathbf{u}}$ drives the velocity to zero within the solid region, effectively imposing the no-slip condition on the immersed body surface. The penalisation parameter is set to of $\beta = 91$ [1/s] and it is computed from Eq. (10), defined in Section 5.1.2, where η is a dimensionless parameter controlling the penalisation strength. The solid mask is constructed on a Cartesian grid by sampling the body-fitted geometry, discretised on a 512×64 grid. The rectangular domain remains the same as in the body-fitted case; only the S-shaped channel is absorbed into the solid mask, so that the mask takes the value of unity inside the solid region and zero in the fluid. Figure 22 shows the Cartesian grid box used for the QCFD simulation, with the full computational domain on the top and a zoom around the bend on the bottom.

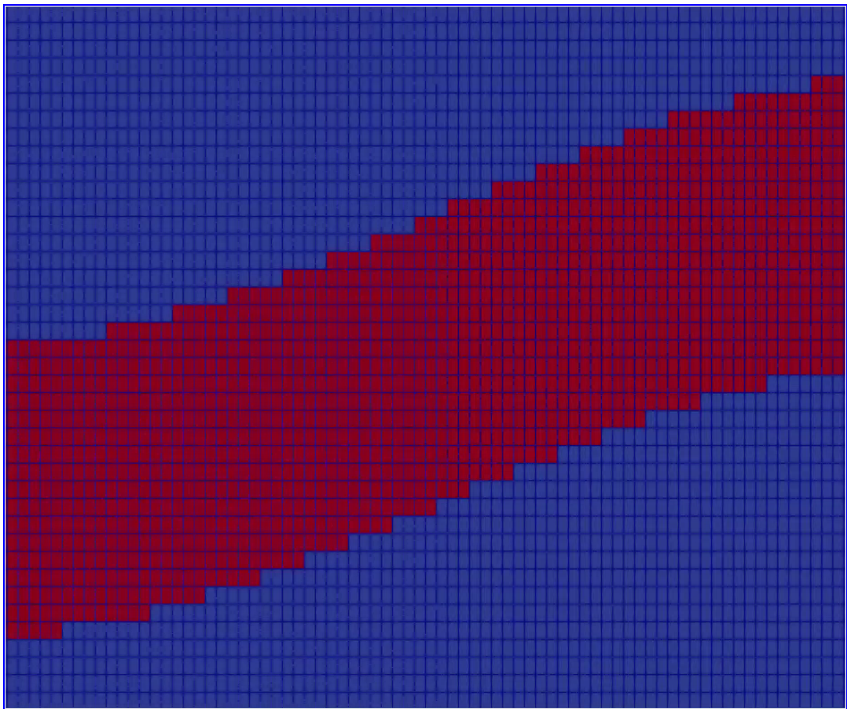
Because the QCFD solver employs a staggered grid arrangement, the solid mask is generated separately for each field location: at cell centres for the pressure, and at the staggered face locations for each velocity component, computed as the average of the adjacent cell-centred mask values.

The boundary conditions must also be transformed to match the topology of the Cartesian grid, which differs from that of the body-fitted mesh. On the body-fitted grid, the boundaries are associated with the physical geometry, while on the Cartesian grid they are expressed as the four sides of the rectangular

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085



(a) Full computational domain.



(b) Zoom of the Cartesian grid around the bend.

Figure 22: Cartesian grid box used for the QCFD simulation with the immersed boundary method. (a) Full computational domain and (b) zoom around the bend. The simulation is carried out on a 512×64 grid.

box together with the inlet and outlet sections of the channel. The transformed boundary conditions for the Cartesian grid are summarised in Table 12.

Table 12: Boundary conditions for the 2D S-Bend case on the Cartesian grid.

Boundary	Velocity	Pressure
Top	$\partial\vec{\mathbf{u}}/\partial n = \vec{\mathbf{0}}$	$\partial p/\partial n = 0$
Bottom	$\partial\vec{\mathbf{u}}/\partial n = \vec{\mathbf{0}}$	$\partial p/\partial n = 0$
Left	$\vec{\mathbf{u}} = \vec{\mathbf{0}}$	$\partial p/\partial n = 0$
Right	$\partial\vec{\mathbf{u}}/\partial n = \vec{\mathbf{0}}$	$\partial p/\partial n = 0$
Inlet	$\vec{\mathbf{u}} = (u_{\text{in}}(y),\, 0)$	$\partial p/\partial n = 0$
Outlet	$\partial\vec{\mathbf{u}}/\partial n = \vec{\mathbf{0}}$	$p = 0$

To provide a benchmark for the QCFD solver output, a reference solution is generated on the Cartesian grid using a modified OpenFOAM solver, `maskedIcoFoam`. This solver solves the incompressible laminar Navier-Stokes equations with the immersed boundary penalisation term given in Eq. (13). The simula-

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

tion is carried out on a uniform Cartesian grid of 512×64 cells, with grid spacings $\Delta x \approx 1.953 \times 10^{-3}$ [m] and $\Delta y = 3.125 \times 10^{-3}$ [m], using the same material properties and time-stepping parameters as the body-fitted case, namely $\rho = 1$ [kg/m^3], $\nu = 1.0 \times 10^{-4}$ [m^2/s], $t_0 = 0$ [s], $\Delta t = 3.335 \times 10^{-4}$ [s], and $t_{\text{end}} = 2$ [s]. The reference solution is stored in the OpenFOAM Cartesian format and serves as the comparison baseline. The resulting velocity magnitude field is shown in Figure 23. For brevity, the velocity magnitude from this finite-volume solution is denoted $\|\vec{\mathbf{u}}\|_{\text{FV}}$ in the following.

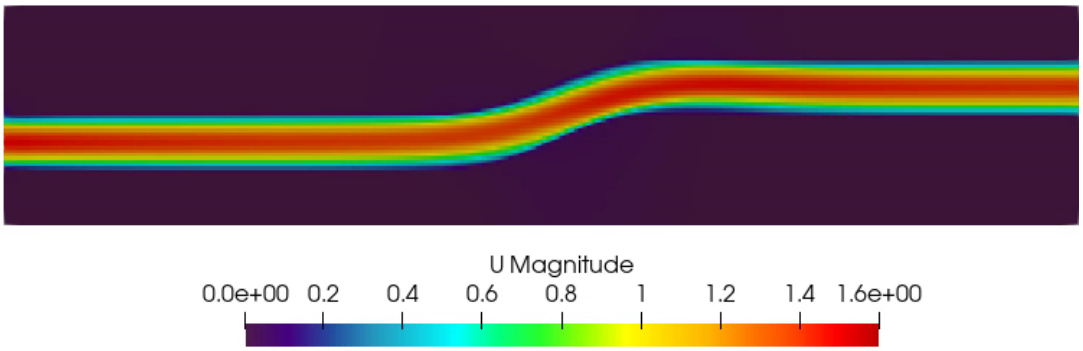


Figure 23: Reference velocity magnitude field from the `maskedIcoFoam` solution on the Cartesian grid with the immersed boundary penalisation.

The solid mask fields, together with the boundary conditions and material properties extracted from the OpenFOAM case, are assembled into the input files required by the QCFD solver. These inputs consist of the geometry data, stored in Hierarchical Data Format (HDF5) and containing the staggered solid mask fields together with the grid metadata, and the configuration files specifying the boundary conditions and simulation parameters.

The QCFD solver reads these inputs and encodes the solid mask fields into matrix product state (MPS) representations via tensor train decomposition. The conversion pipeline therefore connects the body-fitted OpenFOAM description of the S-Bend geometry to the Cartesian immersed boundary representation and finally to the MPS encoding used by the QCFD solver.

Note: The interface component described here is responsible only for preparing these inputs; the execution of the MPS solver itself is performed externally. Interested users should contact the University of Hamburg (UHH) to obtain access to the MPS solver and to arrange the execution of the simulations.

For the QCFD solver, the boundary conditions are translated into a compact algebraic form, in which each boundary of each velocity component is encoded as a pair consisting of a boundary type and an associated value. The boundary types are of Dirichlet, Neumann, and Poiseuille form. The inlet assumes the Poiseuille form given by Eq. (12), that is a specialised Dirichlet condition in which the prescribed profile is a parabolic function of the wall-normal coordinate. The resulting encoding is summarised in Table 13.

Table 13: Boundary condition encoding for the QCFD solver inputs. *D* denotes Dirichlet and *N* denotes Neumann.

Boundary	Location	u	v	p
x_{left}	Inlet	D (Poiseuille)	D (0)	N
x_{right}	Outlet	N	N	D (0)
y_{left}	Top	N	N	N
y_{right}	Bottom	N	N	N

Once the QCFD solver has been executed externally, it produces a solution in JLD format. Figure 24 shows the velocity magnitude field at the final time $t_{\text{end}} = 2$ [s] obtained from four different simulations. Panel (a) shows the DNS solution $\|\vec{\mathbf{u}}\|_{\text{DNS}}$. Panels (b), (c), and (d) show the MPS solutions $\|\vec{\mathbf{u}}\|_{\text{MPS}}$

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

obtained at bond dimensions $\chi = 64$, $\chi = 48$, and $\chi = 32$, respectively.

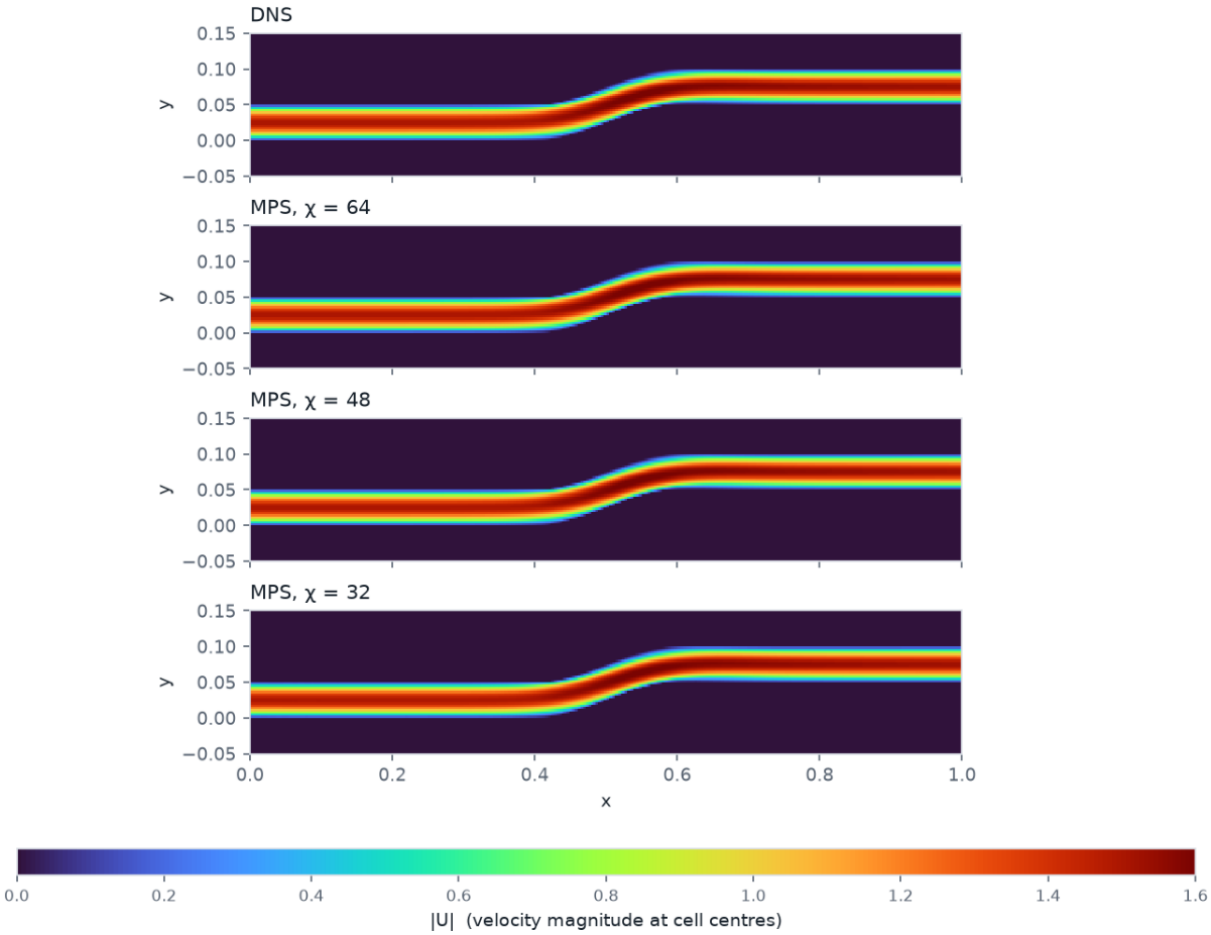


Figure 24: Velocity magnitude field at the final time $t_{\text{end}} = 2$ [s]. Panel (a) shows the DNS solution $\|\vec{\mathbf{u}}\|_{\text{DNS}}$, while panels (b), (c), and (d) show the MPS solutions $\|\vec{\mathbf{u}}\|_{\text{MPS}}$ at bond dimensions $\chi = 64$, 48, and 32, respectively. All fields are shown on the 512×64 Cartesian grid.

To quantify the effect of the bond dimension on the accuracy of the QCFD solver, Figure 25 shows the absolute error $|\|\vec{\mathbf{u}}\|_{\text{MPS}} - \|\vec{\mathbf{u}}\|_{\text{DNS}}|$ per cell at the final time $t_{\text{end}} = 2$ [s] between the MPS solutions and the DNS reference. The three panels correspond to bond dimensions $\chi = 64$, $\chi = 48$, and $\chi = 32$, respectively.

The error maps show how the accuracy of the MPS solution depends on the bond dimension, with the largest errors concentrated in the bend region where the velocity gradients are steepest. A quantitative summary is given in Table 14, which reports the relative deviation of the pressure drop Δp and the domain maximum $D_{\max}$ for each bond dimension.

Table 14: Error metrics for the S-Bend case at the final time $t_{\text{end}} = 2$ [s]. Δp is reported as a relative deviation δq. $D_{\max}$ is the run's own value at the last step. The wall flux is the run's own value at the last step. The DNS row gives the reference values.

	Δp	$D_{\max}$	Wall flux
DNS	0.548	1.4×10^{-13}	0
$\chi = 64$	5.1×10^{-8}	1.9×10^{-8}	3.1×10^{-12}
$\chi = 48$	1.0×10^{-2}	8.6×10^{-3}	1.3×10^{-6}
$\chi = 32$	2.3	7.9×10^{-1}	4.8×10^{-4}

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

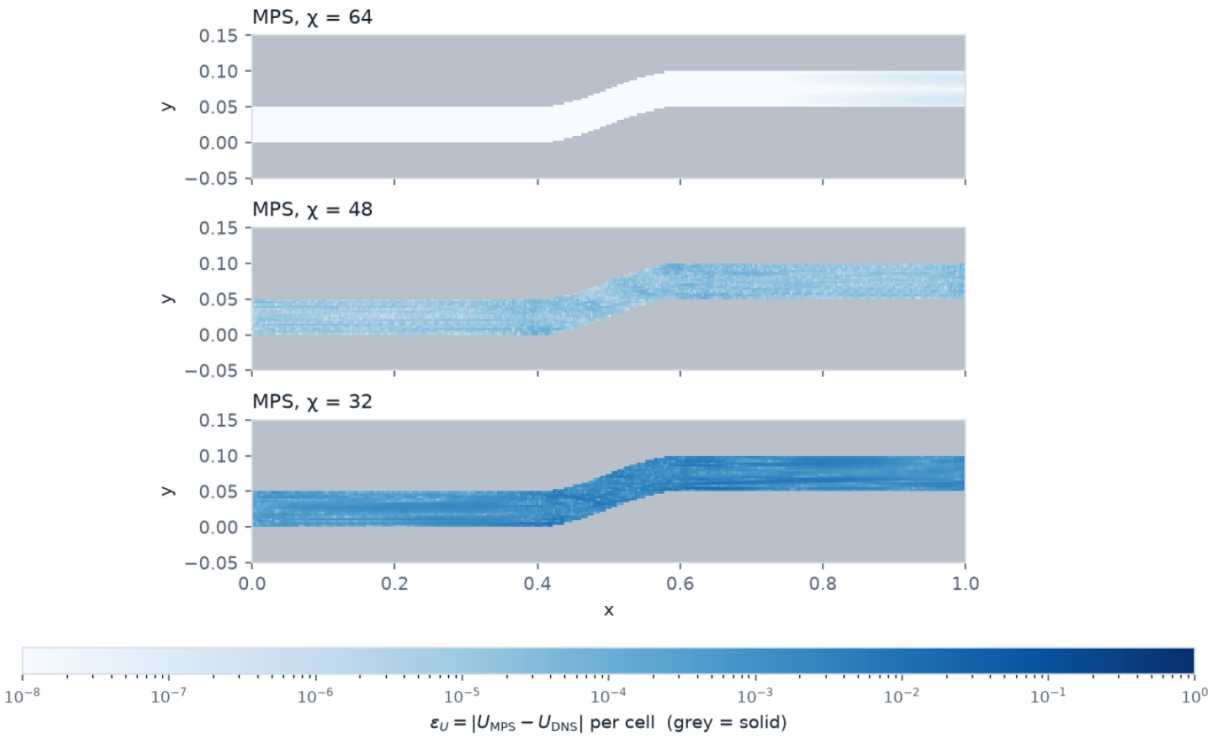


Figure 25: Absolute error $|\|\vec{\mathbf{u}}\|_{\mathrm{MPS}} - \|\vec{\mathbf{u}}\|_{\mathrm{DNS}}|$ per cell at the final time $t_{\mathrm{end}} = 2$ [s] between the MPS solutions and the DNS reference, for bond dimensions $\chi = 64$ (left), $\chi = 48$ (middle), and $\chi = 32$ (right). The error is shown on a logarithmic scale.

A second summary, given in Table 15, reports the field errors E_u, E_v, and E_p at the last step over the fluid domain.

Table 15: Field errors E_ϕ at the last step, over the fluid domain.

MPS χ	E_u	E_v	E_p
$\chi = 64$	2.6×10^{-8}	2.3×10^{-8}	5.1×10^{-8}
$\chi = 48$	1.8×10^{-5}	1.3×10^{-4}	1.1×10^{-2}
$\chi = 32$	1.9×10^{-3}	5.7×10^{-3}	2.2

Figure 26 summarises the dependence of the error on the bond dimension. The curves show the relative deviation of Δp and the field errors E_u and E_p, plotted against χ. The dashed horizontal lines mark the 10^{-4} and 10^{-2} thresholds.
The two tables show a clear separation between the behaviour of the velocity and the pressure. At $\chi = 64$ the MPS solution matches the DNS reference to about 5×10^{-8} in Δp and in all three fields. At $\chi = 48$ the velocity remains within 1.3×10^{-4}, but the pressure drop and the pressure field miss the DNS values by about 1%. At $\chi = 32$ the velocity error stays within 6×10^{-3}, but the pressure degrades badly: Δp reaches 1.78 against the DNS value 0.548, and $D_{\max}$ reaches 0.79.

The validation of the full conversion pipeline relies on comparing two solutions that are both defined on the Cartesian solid mask grid. The first is the reference solution obtained from the modified OpenFOAM solver `maskedIcoFoam`, which was described in the previous section and is already available in the OpenFOAM Cartesian format. The second is the solution produced by the QCFD solver, which is expressed in the tensor network representation and must therefore be mapped back onto the Cartesian solid mask grid before any comparison can be made.

For this purpose, a reconstruction algorithm has been developed as part of the interface. The reconstruction algorithm reads the JLD output produced by the QCFD solver and converts it back into the

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

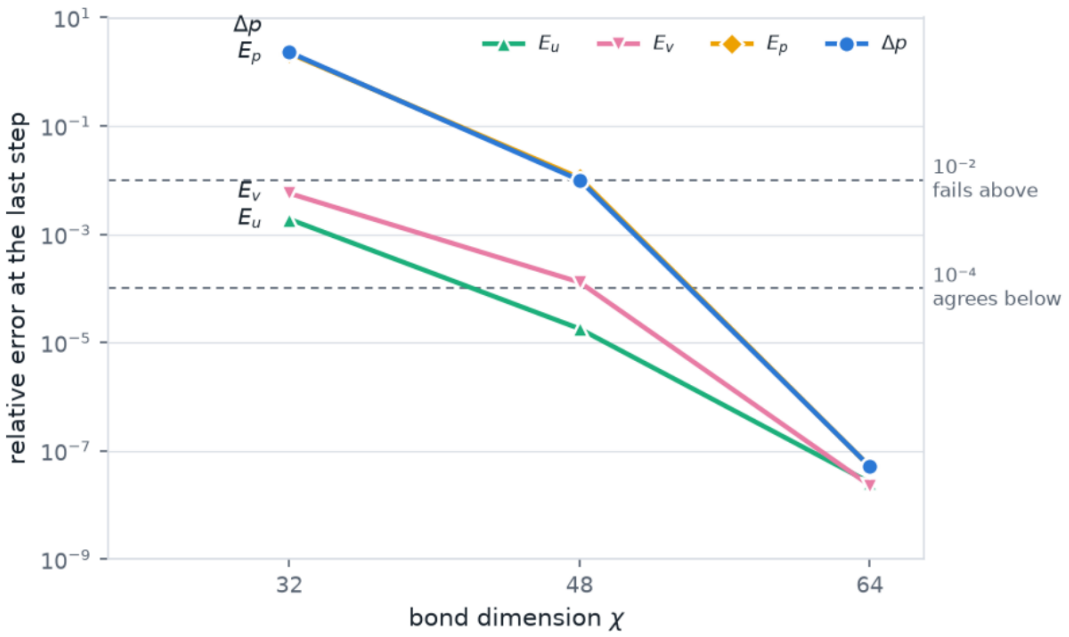


Figure 26: Relative error against χ at the last step: Δp as δq, E_u, E_v and E_p as field errors. At $\chi = 32$ the Δp and E_p points overlap. The dashed lines are the 10^{-4} and 10^{-2} thresholds of the tables.

OpenFOAM Cartesian solid mask format. The reconstruction maps the tensor network solution onto the staggered grid arrangement used by `maskedIcoFoam`, so that both velocity and pressure are expressed on the same grid, at the same field locations, and in the same file format as the reference solution. This ensures that the two solutions are directly comparable without any further interpolation or post-processing.

The visualisation of Figure 27 shows the absolute error $|\|\vec{\mathbf{u}}\|_{\text{Qrec}} - \|\vec{\mathbf{u}}\|_{\text{FV}}|$ at the final time $t_{\text{end}} = 2$ [s] on the 512×64 Cartesian solid mask grid. The reconstructed MPS solution $\|\vec{\mathbf{u}}\|_{\text{Qrec}}$ is obtained by mapping the MPS output at bond dimension $\chi = 64$ onto the solid mask grid, and $\|\vec{\mathbf{u}}\|_{\text{FV}}$ is the finite-volume solution from `maskedIcoFoam`.

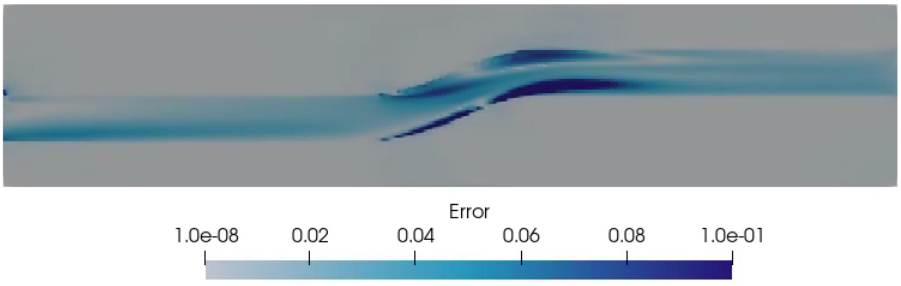


Figure 27: Absolute error $|\|\vec{\mathbf{u}}\|_{\text{Qrec}} - \|\vec{\mathbf{u}}\|_{\text{FV}}|$ at the final time $t_{\text{end}} = 2$ [s] on the 512×64 Cartesian solid mask grid. The reconstructed MPS solution $\|\vec{\mathbf{u}}\|_{\text{Qrec}}$ is obtained from the MPS output at bond dimension $\chi = 64$, and $\|\vec{\mathbf{u}}\|_{\text{FV}}$ is the finite-volume solution from `maskedIcoFoam`.

The three indirect-coupling cases presented in this section are accompanied by a step-by-step instruction set in the tutorial document [1], which describes how to prepare each case, generate the solid mask, assemble the QCFD inputs, and run the solver.

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

5.2 2D Problem - Matrix Coupling

This subsection presents the validation case developed to demonstrate the direct coupling strategy, in which the discretised system matrices arising from the finite difference formulation are converted into tensor network representations without an intermediate geometric remapping. The case uses the classical flow around a circular cylinder as the physical benchmark.

5.2.1 V2D-4a: Flow around cylinder

To illustrate the direct coupling framework, the incompressible Navier–Stokes equations are solved for steady flow around a stationary circular cylinder at $Re = 20$. The governing equations are

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \frac{1}{Re}\Delta \mathbf{u} + f \quad &\text{in } \Omega, \\ \nabla \cdot \mathbf{u} = 0 \quad &\text{in } \Omega, \end{aligned} \tag{14}$$

where $\mathbf{u}$ is the velocity field, p is the pressure, f is a body force term, and Ω denotes the computational domain.

The tensor network solution is compared against a classical finite-difference (FD) reference computed on the same problem. The comparison is performed on a body-fitted grid of $N = 256 \times 512$ cells, where $n_\eta = 8$ is the number of grid points in the radial direction η (normal to the cylinder surface) and $n_\xi = 9$ is the number of grid points in the azimuthal direction ξ (along the cylinder circumference). The radial direction resolves the boundary layer around the cylinder, while the azimuthal direction follows the curvature of the body.

Fractional-Step Time Integration The simulation is based on a fractional-step algorithm [15], analogous to classical projection methods [9]. The scheme advances the solution in time through a sequence of sub-steps that separate the treatment of the convective, diffusive, and pressure terms, and it preserves the mathematical structure of the incompressible Navier–Stokes equations. The key operations — vector addition, scalar multiplication, elementwise multiplication for nonlinear terms, and MPO–MPS contraction for applying differential operators — are all performed in the tensor train format. After each operation, a truncation step is applied to control the growth of bond dimensions.

The simulation starts from a uniform initial condition

$$\mathbf{u}_{\text{init}} = (U_\infty, 0)^\top, \tag{15}$$

in which the streamwise velocity is set to the free-stream value U_∞ and the transverse velocity is zero. The time step Δt is chosen so that the convective Courant number remains below unity, ensuring the stability of the convective term. The solution is then advanced in pseudo time until a steady state is reached, at which point the fields are extracted for comparison with the reference solution.

Reconstruction of the Tensor Network Solution After the tensor network calculation has converged, the solution is recovered in dense form by contracting the TT-cores along their bond indices. Because the tensor network simulation is performed on the same grid as the finite-difference reference, no interpolation or remapping is required: the reconstructed fields are expressed directly at the same grid points, in the same layout, and can be compared against the reference without further post-processing.

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

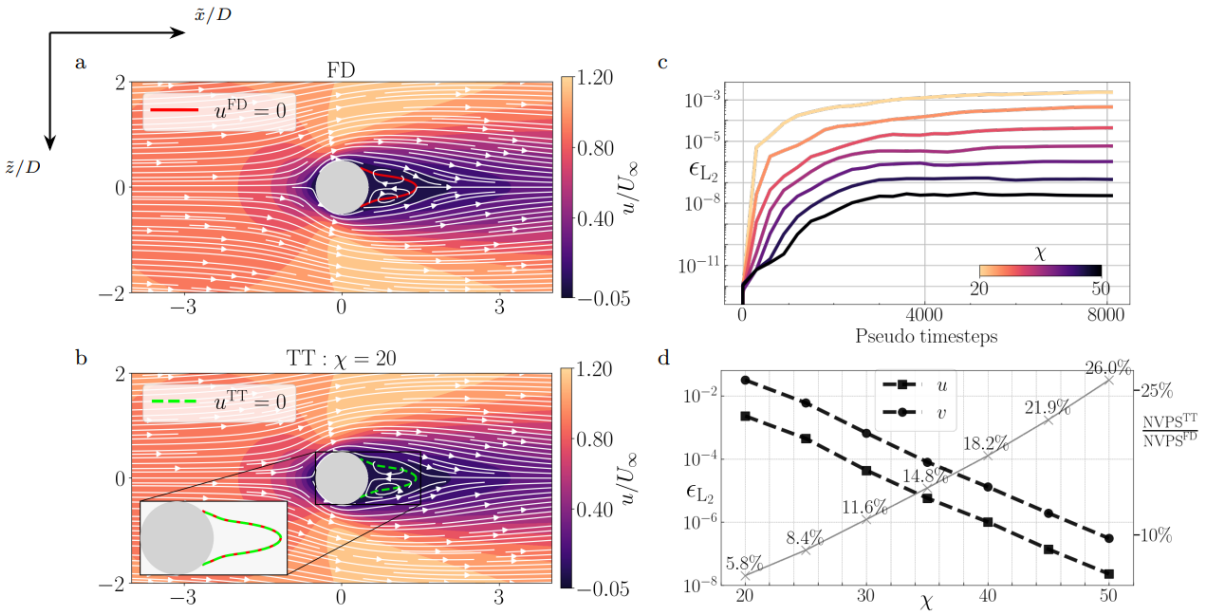


Figure 28: Streamlines and horizontal velocity (u) contours for the steady-state flow at $Re = 20$, predicted by (a) the classical FD method and (b) the TT method with $\chi = 20$. The $u = 0$ isoline is highlighted with an inset in panel (b) showing a close-up comparison. The predictive difference ϵ_{L2} between the TT and the classical FD solutions is shown for the velocity magnitude (c) and the individual Cartesian velocity components u, v (d). Panel (c) shows the temporal evolution for different bond dimensions χ, while panel (d) depicts the steady-state results supplemented by the NVPS fraction for each χ to indicate the corresponding compression rate.

Figure 28 presents the comparison in four panels. Panels (a) and (b) show the streamlines and the horizontal velocity (u) contours of the steady-state solution obtained with the classical FD method and with the tensor train (TT) method at bond dimension $\chi = 20$, respectively. The $u = 0$ isoline is highlighted, and an inset in panel (b) provides a close-up comparison of the two isolines. The two solutions are visually indistinguishable at this scale, confirming that the tensor network representation reproduces the steady-state flow field around the cylinder. Panels (c) and (d) quantify the predictive difference between the TT and FD solutions through the relative error ϵ_{L2}. Panel (c) shows the temporal evolution of this error for several bond dimensions χ, illustrating how the accuracy of the tensor network solution improves as the bond dimension is increased. Panel (d) reports the steady-state values of ϵ_{L2} for the individual Cartesian velocity components u and v. It also shows the NVPS fraction for each χ, that is computed as

$$\text{NVPS} = 2\sum_{k=1}^{n} \alpha_{k-1}\alpha_k, \tag{16}$$

where α_k denotes the bond index between the k-th and $(k+1)$-th TT-core and $n = \log(N)$ is the number of TT-cores. The NVPS for a TT-vector therefore grows only as $O(n\chi^2)$ [13]. A small NVPS fraction corresponds to a large compression rate, and the plot illustrates the trade-off between compression and accuracy as the bond dimension is varied. Taken together, the panels demonstrate that the direct coupling strategy is able to reproduce the classical reference solution with a substantial compression of the degrees of freedom.

5.2.2 V2D-4b: Verification against OpenFOAM

The comparison presented above relies on a classical finite-difference (FD) reference solution. This reference is produced by the in-house FD solver of the UHH framework, which has been verified against OpenFOAM and against reference values from the literature for the flow around a circular cylinder. The verification covers the drag coefficient C_D over the Reynolds number range $Re \in [1, 200]$, spanning the Stokes and subcritical regimes. The comparison is summarised in Figure 29 and shows good agreement across the range, with a small constant offset that is attributed to differences in discretisation, grid resolution, and boundary treatment. These results confirm that the FD solver provides a reliable basis for benchmarking the tensor network solver, and the interested reader is referred to Appendix H of

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

Ref. [9] for the full verification study. A direct reconstruction of the tensor network output onto an OpenFOAM grid is not performed in this work; instead, the TT solution is compared against the FD solver, which has itself been verified against OpenFOAM and literature values.

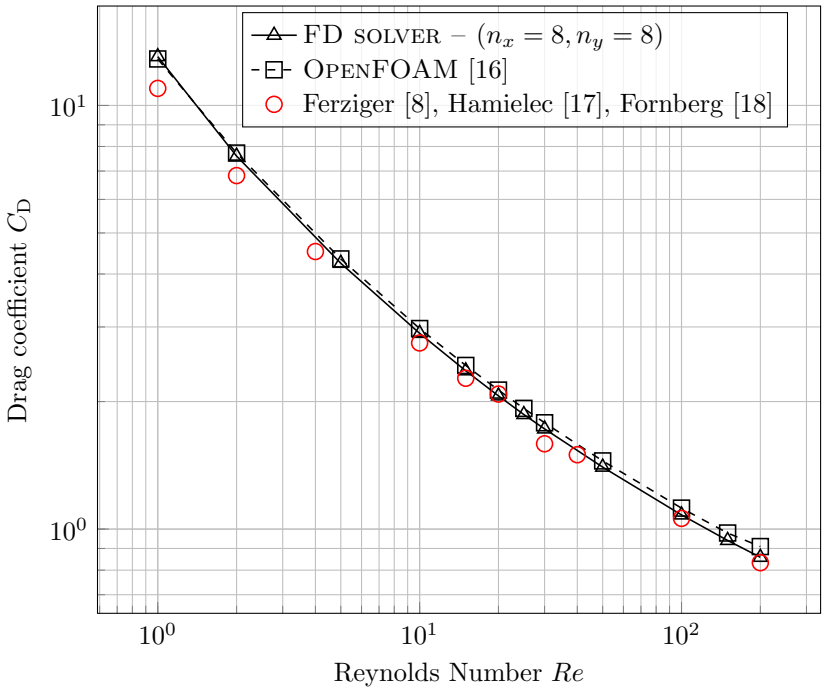


Figure 29: Drag coefficient as a function of the Reynolds number Re for the flow around a cylinder from Stokes to subcritical regimes. The benchmark study compares results from literature [15, 18, 17], OpenFOAM [16] and the developed FD solver.

6 Conclusions

This deliverable has presented the OpenFOAM-QCFD software interface and library, together with the validation cases that demonstrate its capabilities. Two coupling strategies have been implemented and tested. The indirect strategy, developed by ENGYS, maps OpenFOAM geometries onto a Cartesian solid mask and encodes it as a matrix product state for the QCFD solver, and has been validated on three two-dimensional cases: heat conduction in a square with an internal cylinder, laminar flow around a cylinder at $\mathrm{Re} = 20$, and laminar flow in an S-Bend channel representative of BEV cooling geometries. The direct strategy, developed by TUHH, converts discretised system matrices directly into tensor network representations, and has been validated against a finite-difference reference for steady flow around a cylinder at $\mathrm{Re} = 20$, with that reference itself verified against OpenFOAM and the literature. In every case, the interface was demonstrated to reproduce the classical reference solution, and the error was quantified as a function of the bond dimension χ. The mapping strategy was shown to be accurate for velocity and temperature fields, with the pressure field requiring larger bond dimensions to reach the same accuracy. The direct strategy was shown to achieve substantial compression of the degrees of freedom, with the tensor network solution matching the finite-difference reference while retaining only a small fraction of the variables. Across all cases, the fundamental capability of the interface has been demonstrated. Nothing in the pipeline remains unproven: every stage, from geometry extraction through to the comparison of the reconstructed solution against the reference, has been exercised and validated.

Several directions remain open for future work. The accuracy of the wall representation in the indirect coupling requires improvement, since the immersed boundary treatment of solid walls is the dominant source of error in the cases presented. Turbulence modelling has not been addressed and would need to be added for industrial-scale flows. Higher grid resolutions and larger bond dimensions would allow the accuracy of both strategies to be pushed further, at a cost that scales with the problem size. For the direct strategy in particular, a fundamental limitation must be acknowledged: the classical system

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

matrix must still be assembled before it can be converted into a tensor network, and this assembly accounts for roughly half of the total solution cost. Even if the tensor network solve itself were free, the overall procedure would still cost at least half of the classical solution, which means that the direct matrix-to-tensor conversion is not competitive unless the matrix assembly step can be fundamentally restructured. The indirect strategy does not suffer from this limitation, because it does not require the classical matrix to be formed. Looking further ahead, the interface provides the basis for a longer-term commercial approach. In the current workflow, OpenFOAM serves as the entry point for geometry and configuration data, which is valuable for communication and for distributing the method across the CFD community, but the OpenFOAM step is not required in a production setting. A commercial product could take the geometry directly, encode it as a solid mask, and set up and solve the problem entirely within the tensor network framework, with post-processing performed on the tensor network output. The OpenFOAM interface would remain useful as a bridge for existing users, but the long-term path is a direct geometry-to-solution pipeline. Provided that tensor network solvers can be made scalable to industrial problem sizes, and that quantum hardware continues to mature, this approach has the potential to become a transformative product.

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085

References

[1] QCFD Project. OpenFOAM-QCFD Interface: Tutorials for the Indirect Coupling Strategy. Project tutorial document (PDF), 2026. Available through the QCFD community repository at the University of Hamburg, `https://www.fdr.uni-hamburg.de/communities/qcfd/`.

[2] Jeffrey P Slotnick, Abdollah Khodadoust, Juan Alonso, David Darmofal, William Gropp, Elizabeth Lurie, and Dimitri J Mavriplis. Cfd vision 2030 study: a path to revolutionary computational aerosciences. Technical report, NASA, 2014.

[3] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, 2018.

[4] Amara Katabarwa, Katerina Gratsea, Athena Caesura, and Peter D. Johnson. Early fault-tolerant quantum computing. *PRX Quantum*, 5:020101, 2024.

[5] Peyman Givi, Andrew J. Daley, Dimitri Mavriplis, and Mujeeb Malik. Quantum speedup for aeroscience and engineering. *AIAA Journal*, 58(8):3715–3727, 2020.

[6] Quantum Computational Fluid Dynamics Consortium. Quantum Computational Fluid Dynamics – Core Benchmark CFD Set – Deliverable D1.1. Technical report, TUHH, UHH, ENGYS, FZJ, planqc, ENGYS, TUM, TUC, 2024.

[7] Charles S Peskin. Flow patterns around heart valves: A numerical method. *Journal of Computational Physics*, 10(2):252–271, 1972.

[8] Joel H. Ferziger, Milovan Perić, and Robert L. Street. *Computational Methods for Fluid Dynamics*. Springer International Publishing, 2020.

[9] Nis-Luca Van Hülst, Pia Siegl, Paul Over, Sergio Bengoechea, Tomohiro Hashizume, Mario Guillaume Cecile, Thomas Rung, and Dieter Jaksch. Quantum-inspired tensor-network fractional-step method for incompressible flow in curvilinear coordinates. *Computer Physics Communications*, page 110169, 2026.

[10] Alexandre Joel Chorin. Numerical solution of the Navier-Stokes equations. *Mathematics of Computation*, 22(104):745–762, 1968.

[11] Ulrich Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 326(1):96–192, 2011. January 2011 Special Issue.

[12] Román Orús. Tensor networks for complex quantum systems. *Nature Reviews Physics*, 1(9):538–550, 2019.

[13] Martin Kiffner and Dieter Jaksch. Tensor network reduced order models for wall-bounded flows. *Physical Review Fluids*, 8(12):124101, 2023.

[14] Quantum Computational Fluid Dynamics Consortium. Quantum Computational Fluid Dynamics – Core Benchmark CFD Set – Deliverable D2.1. Technical report, TUHH, ENG, UHH, 2026.

[15] Joel H Ferziger, Milovan Perić, and Robert L Street. *Computational methods for fluid dynamics*, volume 3. Springer, 2002.

[16] Engys ltd, helyx, 2026.

[17] AE Hamielec and JD Raal. Numerical studies of viscous flow around circular cylinders. *The physics of fluids*, 12(1):11–17, 1969.

[18] Bengt Fornberg. Steady viscous flow past a circular cylinder up to reynolds number 600. *Journal of Computational Physics*, 61(2):297–320, 1985.

This project receives funding from the European Union's Horizon 2020 HORIZON Research and Innovation Actions Programme under Grant Agreement #101080085