

POLITECNICO DI TORINO

Department of Mechanical and Aerospace Engineering



**Politecnico
di Torino**

Master's Degree in Aerospace Engineering

Coupling OpenFOAM and OpenRadioss for Partitioned Fluid-Structure Interaction Frameworks

A Blast Wave - Wall Impact Simulation using the preCICE Library

Internal Supervisors

Prof. Andrea **Ferrero**

Prof. Dario **Pastrone**

Candidate

Elisa **Santoro**

Student ID: **326586**

External Supervisors

Univ.-Prof. Dr.-Ing. habil. Markus **Klein**

PD Dr.-Ing. habil. Josef **Hasslberger**

Dr.-Ing. Georgios **Michaloudis**

Thesis developed in collaboration with:

Bundeswehr University Munich



ACADEMIC YEAR 2025 - 2026

Abstract

The research presented in this thesis was conducted as a collaboration between the University of the Bundeswehr Munich (through its Institute of Applied Mathematics and Scientific Computing and Institute of Mechanics and Statics) and the University of Stuttgart (through its Institute for Parallel and Distributed Systems). The aim of this work is to configure and validate partitioned fluid-structure interaction (FSI) frameworks using OpenFOAM (CFD) and OpenRadioss (FEM) through the preCICE coupling library, for which an OpenRadioss adapter had undergone code-level verification but had never been validated against physically relevant test cases prior to this work. Despite extensive documentation of numerous partitioned FSI benchmarks configured through preCICE, no configuration of OpenRadioss as a structural participant had been previously documented in the literature.

In a peer-to-peer library approach, each participating solver is coupled through its own dedicated software interface known as the adapter, responsible for extracting coupling data from internal fields for transmission and imposing the exchanged quantities as boundary conditions and forcing terms upon reception. As the OpenRadioss adapter was still in an early stage of development at the start of this work, multiple partitioned FSI frameworks were configured to systematically identify defects in the adapter code and assess data transfer integrity. Faults identified throughout this process were directly reported to the OpenRadioss development team via the software's official GitHub repository, since this research did not concern the direct development of the adapter prototype itself, but rather the investigation of the causes underlying coupling failure. Upon defect resolution, the corrected coupling configurations were validated against physically relevant benchmark cases.

The coupled configurations established between OpenFOAM and OpenRadioss were assessed with particular attention devoted to the accuracy of the structural response. Two test cases were considered: the **Pulsating Flow over Flexible Flap**, for which benchmark reference data permitted quantitative assessment, and the **Blast Wave Impact against a Structural Wall**, selected to investigate previously uncharacterized structural responses and to deepen understanding of coupled interactions in undocumented configurations, given that fluid-structure interaction represents one of the most prevalent multiphysics phenomena in engineering applications. Across multiple flow regimes, the primary focus of the assessment was the spatial and temporal evolution of the structural participant. To ensure reliability of the numerical results, the validation campaign incorporated a study on the effect of element type selection for structural discretization (**HEX8** and **HEX20**) as well as a mesh refinement study on both participants, with the resulting findings subsequently applied to the configuration of the original blast loading simulation, for which no validation baseline exists.

The results confirmed the integrity and reliability of the coupling frameworks configured through preCICE in their entirety, thus establishing this work as the first documented and validated coupling integrating OpenRadioss as a structural participant and providing a complete numerical characterization of the coupled fluid-structural response in partitioned multiphysics applications.

Table of Contents

Abstract	I
List of Figures	IV
List of Tables	IX
1. Introduction	1
1.1 Thesis structure	2
2. Fluid Field Simulation: Physical and Numerical Principles	4
2.1 Overview of the OpenFOAM software	4
2.1.1 Solver architecture and open-source licensing	4
2.1.2 File structure of OpenFOAM cases	5
2.1.3 Flow configurations and solvers adopted in the present work	5
2.2 Mathematical formulation of fluid motion	7
2.2.1 The Navier-Stokes equations	7
2.2.2 Normal shock relations	11
2.3 Numerical solvers and algorithms	15
2.3.1 The pressure-based approach	15
2.3.2 Pressure-based solvers in OpenFOAM	16
2.3.3 The PIMPLE algorithm	17
3. Structural Response Simulation: Physical and Numerical Principles	19
3.1 Overview of the OpenRadioss software	19
3.1.1 Historical perspective	20
3.1.2 Software architecture and file structure of OpenRadioss cases	20
3.1.3 Use cases of OpenRadioss and structural configuration adopted in the present work	21
3.2 Material behavior and hyperelastic constitutive modeling	23
3.2.1 Kinematics of finite deformation	23
3.2.2 Elastic behavior and hyperelasticity	24
3.2.3 Modelling hyperelasticity	25
3.3 Finite element formulations and numerical integration	27
3.3.1 Hexahedral finite elements: HEX8 and HEX20	27
3.3.2 Gauss quadrature and locking phenomena	31
4. preCICE: an open-source library for partitioned multi-physics simulations	33
4.1 Fundamentals of numerical coupling	33
4.1.1 Equation coupling strategies	35
4.1.2 Data mapping at the coupling interface	37
4.1.3 Coupling configuration in preCICE	40
4.2 Solver integration with OpenRadioss: validation and fault identification in an early-stage adapter prototype	43
4.2.1 Partitioned Elastic Beam test case: restoration of displacement data transfer . . .	43
4.2.2 Perpendicular Flap test case: adaptive time step reduction to coupling time window size	47

5. Solver validation against a fluid-structure interaction benchmark: Pulsating Flow over Flexible Flap	53
5.1 The OpenFOAM participant: setup and configuration	53
5.1.1 Physical properties	53
5.1.2 Numerical implementation	55
5.2 The OpenRadioss participant: setup and configuration	57
5.2.1 Physical properties	57
5.2.2 Numerical implementation	58
5.3 preCICE: setup and configuration	59
5.4 Results and conclusions	61
5.4.1 Effect of element type: HEX8 and HEX20 discretization	61
5.4.2 Effect of mesh resolution: Coarse and Medium discretization	68
6. Numerical investigation of fluid-structure interactions under impulsive blast loading: Blast Wave Impact against a Structural Wall	74
6.1 The OpenFOAM participant: setup and configuration	74
6.1.1 Physical properties	74
6.1.2 Numerical implementation	76
6.1.3 Assessment of the uncoupled fluid domain	78
6.2 The OpenRadioss participant: setup and configuration	85
6.2.1 Physical properties	85
6.2.2 Numerical implementation	86
6.3 preCICE: setup and configuration	87
6.4 Results and conclusions	88
6.4.1 Preliminary assessment: thick-wall configuration at a low-intensity shock level	89
6.4.2 Effect of wall thickness: thin-wall configuration at a low-intensity shock level	95
6.4.3 Effect of blast wave intensity: thin-wall configuration at a high-intensity shock level	103
7. Conclusions and outlook	113
Bibliography	115
Acknowledgements	118

List of Figures

1.1	Logo of OpenFOAM.	1
1.2	OpenRadioss logo.	1
1.3	Logo of preCICE.	1
2.1	OpenFOAM: Case directory structure.	5
2.2	Pulsating Flow configuration: instantaneous velocity magnitude at $t = 2.5$ s. Post-processed in ParaView.	6
2.3	Blast Wave configuration: pressure field at $t = 0.02$ s. Post-processed in ParaView.	6
2.4	Conservation of mass for a material volume of fluid of mass m	7
2.5	Conservation of linear momentum for a material volume of fluid of mass m	8
2.6	Material volume of mass m moving with the particles it encloses.	9
2.7	Stationary normal shock wave.[8]	11
2.8	$M_2 = f(M_1)$ across a normal shock wave.	13
2.9	$\frac{\rho_2}{\rho_1} = f(M_1)$	15
2.10	$\frac{p_2}{p_1} = f(M_1)$	15
2.11	$\frac{T_2}{T_1} = f(M_1)$	15
2.12	Qualitative flowchart of the PIMPLE algorithm.[12] f represents the linear momentum.	17
2.13	Implementation-level flow chart of the PIMPLE algorithm in OpenFOAM.[13] <code>pimple.loop()</code> represents the SIMPLE cycle, while <code>pimple.correct()</code> the PISO loop.	18
3.1	OpenRadioss: Graphical User Interface. Each test case, commonly referred to as <i>job</i> , is configured by selecting the input Starter file, the number of threads (-nt) and the number of MPI domains (-np), and subsequently queued through the "Add Job" button.[17]	21
3.2	OpenRadioss: Case directory structure.[17]	21
3.3	OpenRadioss: Run Window.[17]	22
3.4	Structural configuration: perpendicular rectangular flap cantilevered at its base.[18]	22
3.5	Stress-strain curve σ - ϵ for ASTM A992 Steel. [21]	25
3.6	Stress-strain curve σ - ϵ for a generic hyperelastic material. [22]	25
3.7	Classification of three-dimensional finite elements.	28
3.8	Master cube of any HEX8 element.	28
3.9	Master cube of any HEX20 element.	30
4.1	Basic functional components of preCICE.[25]	33
4.2	Adapters are the interface layer between simulation codes and the preCICE library.	34
4.3	Sub-cycling mechanisms within a preCICE coupling time window.[26]	35
4.4	Participant execution sequencing in a preCICE coupled simulation: serial (staggered) and parallel (simultaneous) coupling schemes.	36
4.5	Serial explicit coupling scheme.[27]	37
4.6	Parallel explicit coupling scheme.[27]	37
4.7	Schematic comparison of consistent and conservative data mapping schemes at the coupling interface.[27]	38
4.8	Projection-based mapping methods: nearest-neighbor and nearest-projection.[25]	39

4.9	Radial basis function (RBF) mapping methods: Gaussian basis functions of support radius r , applied to a consistent mapping from an input mesh (orange) to an output sample location $\bar{x}_i^{S_2}$ (blue).[25]	40
4.10	Perpendicular Flap test case (discussed in Section 4.2.2): preCICE-config-visualizer .	40
4.11	Schematic representation of the beam loading condition.[32]	44
4.12	Analytical and numerical reaction force at $z = 0$ m. Post-processed in MATLAB .	44
4.13	Analytical and numerical reaction force at $z = 8$ m. Post-processed in MATLAB .	44
4.14	Monolithic beam fixed on both ends: displacement contour along the y -direction at $t = 3$ s. Post-processed in ParaView.	45
4.15	Partitioned Elastic Beam test case: faulted displacement field at $t = 0.01$ s under uni-directional coupling, showing absence of displacement transfer to beam-left . Post-processed in ParaView.	46
4.16	Partitioned beam fixed on both ends: displacement contour along the y -direction under uni-directional coupling, at $t = 3$ s. Post-processed in ParaView.	47
4.17	Unconstrained structural time stepping: displacement response of the Perpendicular Flap tip in the x -direction. Post-processed in MATLAB .	48
4.18	Fixed structural time stepping: displacement response of the Perpendicular Flap tip in the x -direction. Post-processed in MATLAB .	50
4.19	Perpendicular Flap test case: displacement field along the x -direction under bi-directional coupling, at $t = 0.75$ s. Post-processed in ParaView.	51
5.1	Parabolic inlet velocity profile at $t = 1$ s. Post-processed in ParaView.	54
5.2	Pulsating inlet velocity profile for $0 < t < 30$ s. Processed in MATLAB .	55
5.3	Pulsating Flow over Flexible Flap test case: preCICE-config-visualizer .	59
5.4	Flexible Flap watchpoint locations: Center-Front-Left-Edge (red) and Left-Back-Right-Edge (blue). Post-processed in ParaView.	60
5.5	Center-Front-Left-Edge watchpoint (HEX8 elements): displacement time history along the x -direction. Post-processed in MATLAB .	62
5.6	Left-Back-Right-Edge watchpoint (HEX8 elements): displacement time history along the x -direction. Post-processed in MATLAB .	63
5.7	Center-Front-Left-Edge watchpoint (HEX20 elements): displacement time history along the x -direction. Post-processed in MATLAB .	65
5.8	Left-Back-Right-Edge watchpoint (HEX20 elements): displacement time history along the x -direction. Post-processed in MATLAB .	65
5.9	Center-Front-Left-Edge watchpoint: displacement time history along the x -direction. Investigation on the effect of element formulation. Post-processed in MATLAB .	67
5.10	Left-Back-Right-Edge watchpoint: displacement time history along the x -direction. Investigation on the effect of element formulation. Post-processed in MATLAB .	67
5.11	Center-Front-Left-Edge watchpoint (Medium mesh): displacement time history along the x -direction. Post-processed in MATLAB .	68
5.12	Left-Back-Right-Edge watchpoint (Medium mesh): displacement time history along the x -direction. Post-processed in MATLAB .	69
5.13	Center-Front-Left-Edge watchpoint: displacement time history along the x -direction. Investigation on the effect of mesh refinement (HEX20 element formulation). Post-processed in MATLAB .	70
5.14	Left-Back-Right-Edge watchpoint: displacement time history along the x -direction. Investigation on the effect of mesh refinement (HEX20 element formulation). Post-processed in MATLAB .	70

5.15 Pulsating Flow over Flexible Flap: 3D simulation setup at $t = 2.5$ s. HEX20 solid element formulation, Medium mesh refinement. Post-processed in ParaView.	72
5.16 Pulsating Flow over Flexible Flap: 2D cross section at $z = 0$ m and $t = 2.5$ s. HEX20 solid element formulation, Medium mesh refinement. Post-processed in ParaView.	72
5.17 Pulsating Flow over Flexible Flap: 3D simulation setup at $t = 13$ s. HEX20 solid element formulation, Medium mesh refinement. Post-processed in ParaView.	73
5.18 Pulsating Flow over Flexible Flap: 2D cross section at $z = 0$ m and $t = 13$ s. HEX20 solid element formulation, Medium mesh refinement. Post-processed in ParaView.	73
6.1 Pressure distribution at $t = 0$ s, illustrating the Sod shock tube initial condition for the high-intensity shock level. Post-processed in ParaView.	75
6.2 Temperature distribution at $t = 0$ s, illustrating the Sod shock tube initial condition for the high-intensity shock level. Post-processed in ParaView.	76
6.3 Low-intensity shock level: pressure distribution at $t = 0.2$ s. Post-processed in ParaView.	79
6.4 Low-intensity shock level: pressure distribution at $t = 0.3$ s. Post-processed in ParaView.	79
6.5 Low-intensity shock level: temperature distribution at $t = 0.2$ s. Post-processed in ParaView.	79
6.6 Low-intensity shock level: temperature distribution at $t = 0.3$ s. Post-processed in ParaView.	80
6.7 Low-intensity shock level: magnitude of the velocity field at $t = 0.2$ s. Post-processed in ParaView.	80
6.8 Low-intensity shock level: magnitude of the velocity field at $t = 0.3$ s. Post-processed in ParaView.	80
6.9 Low-intensity shock level: flow velocity along the x -direction at $t = 0.2$ s. Post-processed in ParaView.	81
6.10 Low-intensity shock level: flow velocity along the x -direction at $t = 0.3$ s. Post-processed in ParaView.	81
6.11 High-intensity shock level: pressure distribution at $t = 0.2$ s. Post-processed in ParaView.	81
6.12 High-intensity shock level: pressure distribution at $t = 0.3$ s. Post-processed in ParaView.	82
6.13 High-intensity shock level: temperature distribution at $t = 0.2$ s. Post-processed in ParaView.	82
6.14 High-intensity shock level: temperature distribution at $t = 0.3$ s. Post-processed in ParaView.	82
6.15 High-intensity shock level: magnitude of the velocity field at $t = 0.2$ s. Post-processed in ParaView.	83
6.16 High-intensity shock level: magnitude of the velocity field at $t = 0.3$ s. Post-processed in ParaView.	83
6.17 High-intensity shock level: flow velocity along the x -direction at $t = 0.2$ s. Post-processed in ParaView.	83
6.18 High-intensity shock level: flow velocity along the x -direction at $t = 0.3$ s. Post-processed in ParaView.	84
6.19 Low-intensity shock level: oscillatory pressure profile along the x -direction, at $y = 1.75$ m and $t = 0.2$ s. Post-processed in ParaView.	84
6.20 High-intensity shock level: oscillatory pressure profile along the x -direction, at $y = 1.75$ m and $t = 0.2$ s. Post-processed in ParaView.	85
6.21 Blast Wave Impact against a Structural Wall: preCICE-config-visualizer.	87

6.22	Low-intensity shock level: density distribution at $t = 0.001$ s. Post-processed in ParaView.	89
6.23	Thick-wall coupled configuration at the low-intensity shock level: pressure distribution at $t = 0.2$ s. Post-processed in ParaView.	90
6.24	Thick-wall coupled configuration at the low-intensity shock level: pressure distribution at $t = 0.3$ s. Post-processed in ParaView.	90
6.25	Thick-wall coupled configuration at the low-intensity shock level: temperature distribution at $t = 0.2$ s. Post-processed in ParaView.	91
6.26	Thick-wall coupled configuration at the low-intensity shock level: temperature distribution at $t = 0.3$ s. Post-processed in ParaView.	91
6.27	Thick-wall coupled configuration at the low-intensity shock level: density distribution at $t = 0.2$ s. Post-processed in ParaView.	92
6.28	Thick-wall coupled configuration at the low-intensity shock level: density distribution at $t = 0.3$ s. Post-processed in ParaView.	92
6.29	Thick-wall coupled configuration at the low-intensity shock level: magnitude of the velocity field at $t = 0.2$ s. Post-processed in ParaView.	93
6.30	Thick-wall coupled configuration at the low-intensity shock level: magnitude of the velocity field at $t = 0.3$ s. Post-processed in ParaView.	93
6.31	Thick-wall coupled configuration at the low-intensity shock level: flow velocity along the x -direction at $t = 0.2$ s. Post-processed in ParaView.	94
6.32	Thick-wall coupled configuration at the low-intensity shock level: flow velocity along the x -direction at $t = 0.3$ s. Post-processed in ParaView.	94
6.33	Thick-wall coupled configuration at the low-intensity shock level: overlaid displacement time histories along the x -direction for comparison across all three watch-points. Post-processed in MATLAB.	95
6.34	Thin-wall coupled configuration at the low-intensity shock level: pressure distribution at $t = 0.2$ s. Post-processed in ParaView.	96
6.35	Thin-wall coupled configuration at the low-intensity shock level: pressure distribution at $t = 0.3$ s. Post-processed in ParaView.	96
6.36	Thin-wall coupled configuration at the low-intensity shock level: temperature distribution at $t = 0.2$ s. Post-processed in ParaView.	97
6.37	Thin-wall coupled configuration at the low-intensity shock level: temperature distribution at $t = 0.3$ s. Post-processed in ParaView.	97
6.38	Thin-wall coupled configuration at the low-intensity shock level: density distribution at $t = 0.2$ s. Post-processed in ParaView.	98
6.39	Thin-wall coupled configuration at the low-intensity shock level: density distribution at $t = 0.3$ s. Post-processed in ParaView.	98
6.40	Thin-wall coupled configuration at the low-intensity shock level: magnitude of the velocity field at $t = 0.2$ s. Post-processed in ParaView.	99
6.41	Thin-wall coupled configuration at the low-intensity shock level: magnitude of the velocity field at $t = 0.3$ s. Post-processed in ParaView.	99
6.42	Thin-wall coupled configuration at the low-intensity shock level: flow velocity along the x -direction at $t = 0.2$ s. Post-processed in ParaView.	100
6.43	Thin-wall coupled configuration at the low-intensity shock level: flow velocity along the x -direction at $t = 0.3$ s. Post-processed in ParaView.	100
6.44	Thin-wall coupled configuration at the low-intensity shock level: overlaid displacement time histories along the x -direction for comparison across all three watch-points. Post-processed in MATLAB.	101

6.45	Bottom-Left-Edge watchpoint: displacement time history along the x -direction. Investigation on the effect of wall thickness. Post-processed in MATLAB .	101
6.46	Mid-Left-Edge watchpoint: displacement time history along the x -direction. Investigation on the effect of wall thickness. Post-processed in MATLAB .	102
6.47	Tip-Left-Edge watchpoint: displacement time history along the x -direction. Investigation on the effect of wall thickness. Post-processed in MATLAB .	102
6.48	High-intensity shock level: magnitude of the velocity field at $t = 0.013$ s. Post-processed in ParaView.	104
6.49	Thin-wall coupled configuration at the high-intensity shock level: pressure distribution at $t = 0.2$ s. Post-processed in ParaView.	105
6.50	Thin-wall coupled configuration at the high-intensity shock level: pressure distribution at $t = 0.3$ s. Post-processed in ParaView.	105
6.51	Thin-wall coupled configuration at the high-intensity shock level: temperature distribution at $t = 0.2$ s. Post-processed in ParaView.	106
6.52	Thin-wall coupled configuration at the high-intensity shock level: temperature distribution at $t = 0.3$ s. Post-processed in ParaView.	106
6.53	Thin-wall coupled configuration at the high-intensity shock level: density distribution at $t = 0.2$ s. Post-processed in ParaView.	107
6.54	Thin-wall coupled configuration at the high-intensity shock level: density distribution at $t = 0.3$ s. Post-processed in ParaView.	107
6.55	Thin-wall coupled configuration at the high-intensity shock level: magnitude of the velocity field at $t = 0.2$ s. Post-processed in ParaView.	108
6.56	Thin-wall coupled configuration at the high-intensity shock level: magnitude of the velocity field at $t = 0.3$ s. Post-processed in ParaView.	108
6.57	Thin-wall coupled configuration at the high-intensity shock level: flow velocity along the x -direction at $t = 0.2$ s. Post-processed in ParaView.	109
6.58	Thin-wall coupled configuration at the high-intensity shock level: flow velocity along the x -direction at $t = 0.3$ s. Post-processed in ParaView.	109
6.59	Thin-wall coupled configuration at the high-intensity shock level: overlaid displacement time histories along the x -direction for comparison across all three watchpoints. Post-processed in MATLAB .	110
6.60	Bottom-Left-Edge watchpoint: displacement time history along the x -direction. Investigation on the effect of blast wave intensity. Post-processed in MATLAB .	110
6.61	Mid-Left-Edge watchpoint: displacement time history along the x -direction. Investigation on the effect of blast wave intensity. Post-processed in MATLAB .	111
6.62	Tip-Left-Edge watchpoint: displacement time history along the x -direction. Investigation on the effect of blast wave intensity. Post-processed in MATLAB .	111

List of Tables

4.1	Physical properties of the fluid domain.	48
4.2	Physical properties of the solid domain.	48
5.1	Spatial resolution of the coupled simulations. [40]	53
5.2	Physical properties of the fictitious Newtonian fluid.[39] [40]	54
5.3	Physical properties of the fictitious neo-Hookean material, as prescribed by Schott et al.[39] [40]	58
5.4	Center-Front-Left-Edge watchpoint: cross-solver displacement comparison for selected time instants. Investigation on the effect of element formulation.	63
5.5	Left-Back-Right-Edge watchpoint: cross-solver displacement comparison for selected time instants. Investigation on the effect of element formulation.	64
5.6	Center-Front-Left-Edge watchpoint: cross-solver displacement comparison for selected time instants. Investigation on the effect of element formulation (HEX8 and HEX20).	66
5.7	Left-Back-Right-Edge watchpoint: cross-solver displacement comparison for selected time instants. Investigation on the effect of element formulation (HEX8 and HEX20).	66
5.8	Center-Front-Left-Edge watchpoint: cross-solver displacement comparison for selected time instants. Investigation on the effect of mesh refinement (HEX20 element formulation).	68
5.9	Left-Back-Right-Edge watchpoint: cross-solver displacement comparison for selected time instants. Investigation on the effect of mesh refinement (HEX20 element formulation).	69
5.10	OpenFOAM participant: computational cost for the first 10 time steps of the coupled simulation. Investigation on the effect of mesh refinement.	71
5.11	OpenRadioss participant: total computational cost of the coupled simulation. Investigation on the effect of mesh refinement.	71
6.1	Blast wave intensity levels of the coupled simulations.	74
6.2	Physical properties of air under the perfect gas assumption.	75
6.3	Blast wave intensity levels of the coupled simulations.	76
6.4	Physical properties of the Mooney-Rivlin silicone rubber, as prescribed by Villani et al. [45] and Lu et al. [46]	85
6.5	Upstream properties of the low-intensity blast wave, derived from the upstream pressure p_{blast} using the normal shock relations.	89
6.6	Thick-wall coupled configuration at the low-intensity shock level: displacement comparison along the x -direction, for selected time instants, across all three watchpoints.	95
6.7	Thin-wall coupled configuration at the low-intensity shock level: displacement comparison along the x -direction, for selected time instants, across all three watchpoints.	103
6.8	Low-intensity shock level: displacement comparison, for selected time instants, across all three watchpoints locations. Investigation on the effect of wall thickness.	103
6.9	Upstream properties of the high-intensity blast wave, derived from the upstream pressure p_{blast} using the normal shock relations.	103
6.10	Thin-wall coupled configuration at the high-intensity shock level: displacement comparison along the x -direction, for selected time instants, across all three watchpoints.	112
6.11	Thin-Wall structural configuration: displacement comparison, for selected time instants, across all three watchpoints locations. Investigation on the effect of blast wave intensity.	112
6.12	Total computational cost for each configuration of the Blast Wave Impact against a Structural Wall test case.	112

Chapter 1

Introduction

The research presented in this work was conducted as a collaboration between the Polytechnic of Turin, the University of the Bundeswehr Munich (through its Institute of Applied Mathematics and Scientific Computing and Institute of Mechanics and Statics), and the University of Stuttgart (through its Institute for Parallel and Distributed Systems), investigating partitioned fluid-structure interaction (FSI) frameworks using OpenFOAM (CFD), OpenRadioss (FEM), and the preCICE coupling library.



Figure 1.1: Logo of OpenFOAM. Figure 1.2: OpenRadioss logo. Figure 1.3: Logo of preCICE.

Fluid-structure interaction belongs to a broader class of numerical methods known as **multiphysics simulations**; to contextualize the coupled behavior introduced above, a brief overview of this class of simulations is presented as follows. Multiphysics simulations are numerical configurations that combine into a single model several coupled domains, in order to capture how multiple physical phenomena (e.g. fluid dynamics, structural mechanics, heat transfer, electromagnetics) simultaneously interact to determine the response of a given system. Such a modeling approach accurately accounts for complex coupled effects across several industrial applications that would otherwise be intractable using single-physics frameworks, yielding a correspondingly higher predictive accuracy.

The multiphysics approach stands in contrast to single-physics simulations, which restrict the analysis to individual physical domains and consequently lead to interactions being addressed sequentially rather than concurrently. Although the latter approach provides detailed insights into individual phenomena by isolating one effect at a time, this same isolation also constitutes its inherent limitation, since it assumes that the physical world can be neatly subdivided into self-contained systems that fail to capture the full chain of cross-domain interactions.

Within the class of multiphysics simulations, the monolithic approach and the **partitioned** approach constitute the two opposing strategies available for implementing inherently coupled numerical frameworks.[1] In the monolithic approach, a single software constructs and solves one global system of equations for the complete coupled domain; in the partitioned approach, by contrast, the coupled problem is divided among multiple independent solvers, each governing a specific component of the physics involved, with inter-solver communication managed by a dedicated coupling tool. Building on the partitioned strategy, the present work configures fluid-structure interaction frameworks in which the fluid domain is solved with OpenFOAM and the structural domain with OpenRadioss, while preCICE serves as the coupling library that mediates their interaction to ensure solver synchronization and robust data exchange between **participants**. The main advantages of a partitioned approach lie in the ability to leverage pre-existing, field-specific solvers and the flexibility to combine components that are each optimized for their own subdomain.

As a coupling tool, preCICE adopts a **library** approach: each participating solver initiates the call to preCICE through a dedicated object called the adapter, as opposed to a framework approach, in which the coupling tool itself takes control and calls each solver directly. Since the coupled software require no modification at the source-code level under a library approach, but must solely expose the preCICE calls at the appropriate locations via an external interface (i.e. the adapter), this library-based strategy is markedly less invasive as it preserves solver independence and keeps control flow confined within each simulation code. Furthermore, rather than depending on a server-like coupling executable to coordinate software initialization, preCICE follows a **peer-to-peer** approach under which each simulation code is started individually.

In a peer-to-peer library approach, the adapter serves as the dedicated software interface which is responsible for extracting coupling data from internal fields for transmission and imposing the exchanged quantities as boundary conditions and forcing terms upon reception. At the beginning of this work, the OpenRadioss adapter prototype was still in an early stage of development, since it had exclusively undergone code-level verification and had never been validated against physically relevant numerical frameworks prior to this research. For this reason, despite the numerous partitioned FSI benchmarks

implemented through the preCICE library, no configuration of OpenRadioss as a structural participant had been previously documented in any application reported in literature. The aim of this work was therefore to address such a gap by configuring and validating two test cases both to reproduce a documented application and to investigate a previously unexplored numerical configuration.

Throughout the configuration of these partitioned FSI frameworks, the concurrent investigation systematically identified defects in the adapter code while assessing the integrity of data transfer mechanisms; faults identified over the course of this process were subsequently reported to the OpenRadioss development team via the software’s official GitHub repository, given that this research did not concern the direct development of the adapter prototype itself, but rather the diagnosis of the causes underlying coupling failure. Upon defect resolution, the corrected coupling configurations established between OpenFOAM and OpenRadioss were validated against physically relevant benchmark cases, with particular attention devoted to the accuracy of the structural response, as detailed in Chapters 5 and 6.

This research is structured around two test cases, each addressing a distinct aspect of the coupling methodology’s validation: the **Pulsating Flow over Flexible Flap**, selected to enable quantitative assessment against reference benchmark data, whereas the **Blast Wave Impact against a Structural Wall** was selected to establish the first characterization of a previously unexplored structural response in synchronized coupled interactions. The results confirmed the robustness and reliability of the coupling frameworks configured through preCICE, thus establishing this work as the first documented and validated integration of OpenRadioss as a structural participant within the preCICE ecosystem, delivering a detailed numerical characterization of coupled fluid–structure responses in partitioned multiphysics applications while providing further opportunities for future investigation and methodological development.

1.1 Thesis structure

The present work is organized into six chapters, which provide an overview of the numerical tools and solvers central to this research while detailing the implementation of two test case configurations along with the results obtained, spanning both validation against an established benchmark and characterization of a previously unexplored scenario. All simulation results reported in the following chapters are expressed in the MKS (meter-kilogram-second) unit system.

- Chapter 1 outlines the scientific motivations underlying this study and introduces the research objectives, the state of the art in multiphysics applications, and the software framework adopted in the present work.
- Chapter 2 provides the theoretical background on the fluid-side applications. The fundamental principles of fluid mechanics considered in this study are examined with particular attention to the governing equations of viscous flow and the normal shock relations, alongside a characterization of the OpenFOAM solver architecture and the numerical algorithms it relies upon.
- Chapter 3 establishes the theoretical framework underlying the structural-side applications. Material behavior and hyperelastic constitutive modeling are addressed in detail, together with an overview of finite element formulations and numerical integration as relevant to the applications considered in this work.
- Chapter 4 describes the fundamentals of numerical coupling underlying partitioned frameworks configured with preCICE. An overview of coupling strategies, data mapping at the shared interface, and configuration architecture governing software participation is provided. Furthermore, solver integration with OpenRadioss is thoroughly discussed with emphasis on numerical validation and fault identification across two preliminary test cases: the **Partitioned Elastic Beam** test case and the **Perpendicular Flap** simulation.
- Chapter 5 reproduces the **Pulsating Flow over Flexible Flap** test case to validate the OpenRadioss solver against a fluid–structure interaction benchmark in which OpenFOAM serves as the fluid participant. This test case follows the partitioned coupling framework implemented by Chourdakis et al. [40] and used as a benchmark for comparison, retaining the OpenFOAM domain and replacing the deal.II participant used for the **Flexible Flap** configuration with an equivalent OpenRadioss counterpart, and the results are further validated against the monolithic reference provided by Schott et al. [39]

- Chapter 6 introduces the **Blast Wave Impact against a Structural Wall** test case to characterize a previously unexplored coupled configuration simulating the cross-domain propagation of a blast wave and the structural deformation it induces, placing particular emphasis on the pressure field evolution and the associated watchpoint displacement response. Although the presented configuration requires further refinement considering anomalous trends and inconsistencies observed in the post-processed results, the associated findings establish a robust foundation for the continued development of partitioned fluid–structure interaction frameworks incorporating OpenRadioss as a structural solver within the preCICE coupling environment. Building on a promising preliminary outcome under the specific conditions considered, a dedicated test campaign employing a functionally equivalent structural participant would nonetheless clarify whether further refinement of the OpenRadioss configuration is warranted.
- Conclusions summarize the main findings of the thesis, highlighting the favorable outcomes of the structural software integration within the preCICE coupling ecosystem; an outlook is additionally offered on extended applications and refinements aimed at improving physical consistency and reducing predictive error in partitioned fluid-structure interaction cases configured using OpenRadioss.

Chapter 2

Fluid Field Simulation: Physical and Numerical Principles

The aim of this chapter is to present the notions that were used when implementing the fluid domain configurations defined across this thesis. A brief overview of the OpenFOAM software is provided, followed by a rundown of the physical and numerical properties encountered when characterizing the fluid fields that have been modeled throughout the project.

2.1 Overview of the OpenFOAM software

Computational Fluid Dynamics (CFD) is defined as the branch of fluid mechanics that employs numerical techniques to predict fluid behaviors across various engineering practices and applications¹.

To represent very complex phenomena with proper accuracy, adequate modeling is necessary: this is achieved both by properly defining the fluid domain properties (*pre-processing*) as well as throughout a coherent implementation of governing equations resolution procedures (*processing*). The obtained results are then visualized and analyzed during the *post-processing* stage.

OpenFOAM²(Open-source Field Operation And Manipulation) is a continuum mechanics simulation software, developed starting from the early 1990s at Imperial College London by researchers Henry Weller and Hrvoje Jasak.[2] This computational tool enables representation of a multitude of physical phenomena, encompassing incompressible, compressible and multiphase flows as well as turbulence, heat transfer, acoustics, chemical reactions and electromagnetism processes.

2.1.1 Solver architecture and open-source licensing

OpenFOAM does not provide a generic solver applicable to all cases; each flow solver is designed as a standalone tool capable of handling only a narrow set of problems, resulting in software applications that are inherently solver-dependent. Such architecture therefore relies on the user to choose the most appropriate tool to correctly depict the physical phenomenon of interest. Top-level solvers available within OpenFOAM closely mimic the capabilities of commercial CFD programs, whilst leaving room to the user for further vertical integration and customization [3]; this flexibility largely extends the software suitability to any system of time-dependent PDEs through direct manipulation of the source code.

The OpenFOAM code itself is written in C++, and it is publicly available on GitLab under the GNU General Public License v3.0³ guarantying the open-sourceness of the project while allowing private and commercial usage; it is actively maintained and developed by OpenCFD Ltd. Given the intellectual property rights pending on commercial software which are detained by the application creator or publisher, it is impossible for end users to fully access and modify said programs' numerical source code; moreover, usage of the provided software is contingent upon acceptance of the non-negotiable terms stated in the end-user license agreement (EULA). The open-source nature of OpenFOAM, conversely, allows end users to bypass these obstacles granting what Richard Stallman (founder of the Free Software Movement of 1983) stated as the four essential freedoms of a free software [4]:

1. Freedom to run a program, for any purpose;
2. Freedom to modify such program to suit one's needs: source code accessibility is a precondition to make this freedom effective in practice;
3. Freedom to redistribute copies to help one's neighbor;
4. Freedom to distribute modified versions of the program so that the whole community can benefit from one's improvements.

¹ScienceDirect: <https://sciencedirect.com/topics/materials-science/computational-fluid-dynamics>

²OpenFOAM: <https://www.openfoam.com/>

³GitLab - Official OpenFOAM Repository: <https://gitlab.com/openfoam/core/openfoam>

2.1.2 File structure of OpenFOAM cases

OpenFOAM is designed as a command-line-based toolbox for the Linux operating system, meaning that the software does not provide a native, official Graphical User Interface (GUI) and users must instead run their applications within the terminal environment; test cases are configured using plain-text configuration files referred to as **dictionaries**, which are included in each **case_directory** to specify simulation parameters, boundary conditions and solver settings. The basic directory structure of an OpenFOAM test case, shown in Figure 2.1, is described as follows.

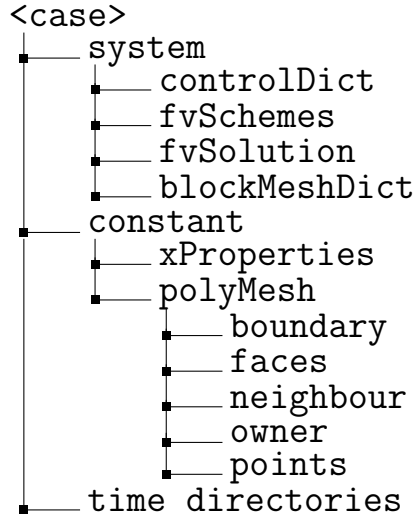


Figure 2.1: OpenFOAM: Case directory structure.

The **system** directory sets parameters associated with the solution procedure through the use of the following files: **controlDict** sets run control parameters such as start/end time, time step and parameters for data output, **fvSchemes** specifies the discretization schemes and **fvSolution** sets the equation solvers, tolerances and other algorithm controls.

The **constant** directory contains files specifying physical properties for the application concerned (e.g. *transportProperties*) as well as a full description of the case mesh within the **polyMesh** subdirectory.

Finally, the **'time' directories** contain individual files of data associated to specific physical fields; such data represents either initial values and boundary conditions specified by the user to define the problem (e.g. 0 directory), or results files written by OpenFOAM throughout the resolution procedure for post-processing purposes[5]. It's important to note that OpenFOAM fields must be initialized even when the solution does not strictly require so, as is the case for steady-state problems; the name of each output time directory is based on the simulated time at which the data is written, and they must be post-processed with external tools such as ParaView⁴, an open-source, multi-platform application designed for scientific data analysis and visualization purposes.

2.1.3 Flow configurations and solvers adopted in the present work

Two different fluid field configurations have been implemented and studied for the purpose of this research, as shown in Figure 2.2 and 2.3 :

1. the **Pulsating Flow over Flexible Flap** test case, in which a three-dimensional incompressible domain is characterized by an unsteady, non-uniform inlet velocity;
2. the **Blast Wave-Wall Impact** simulation, where a highly compressible phenomenon propagates across a pseudo two-dimensional supersonic field.

In light of the substantial differences between the two classes of problem, different solvers were chosen to implement the corresponding numerical configurations; namely, **pimpleFoam** was used to represent the incompressible domain whilst the supersonic field has been solved with **rhoPimpleFoam**. Both tools are transient, pressure-based solvers based on the PIMPLE algorithm: although the two

⁴ParaView: <https://www.paraview.org/>

support mesh motion and topology changes within the fluid domain, they differ significantly in their thermodynamic formulation. Whilst `pimpleFoam` doesn't solve the energy conservation equation and carries no dependence of density, `rhoPimpleFoam` couples momentum and pressure equations with an equation of state and the energy transport balance to resolve the full thermodynamic state of the field at each time step. The governing equations solved for each application, together with the algorithm configurations implemented, will be presented and discussed in the remainder of this chapter.

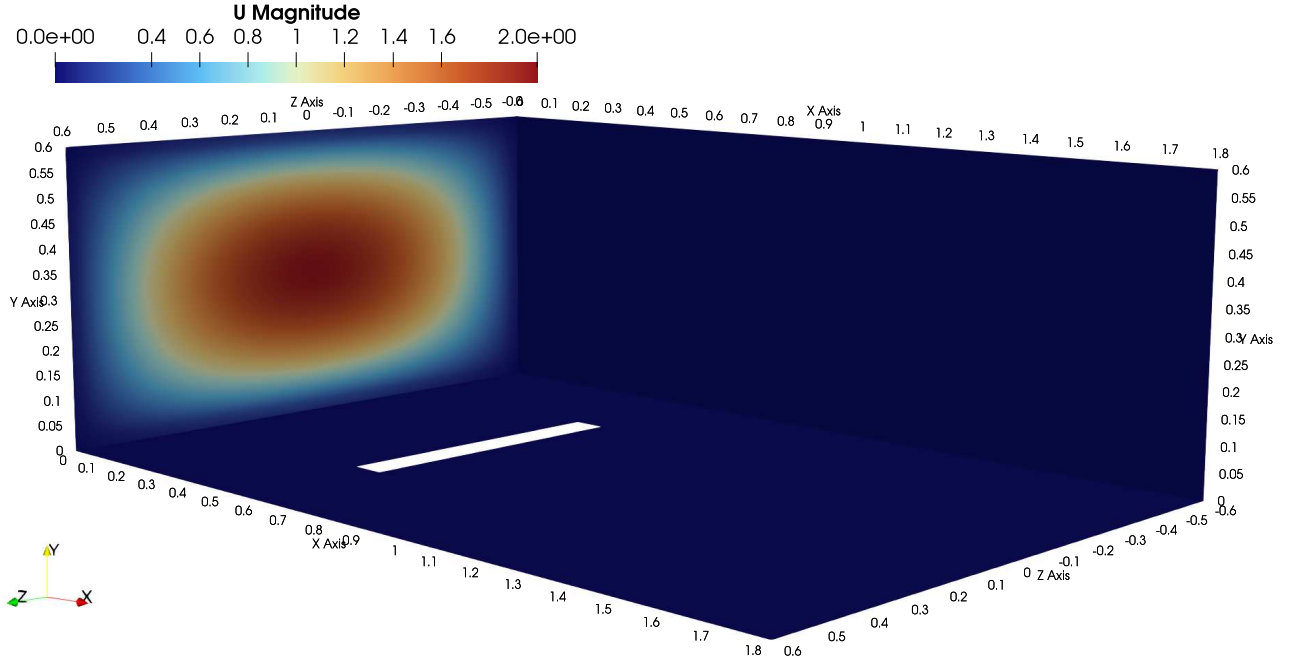


Figure 2.2: Pulsating Flow configuration: instantaneous velocity magnitude at $t = 2.5$ s. Post-processed in ParaView.

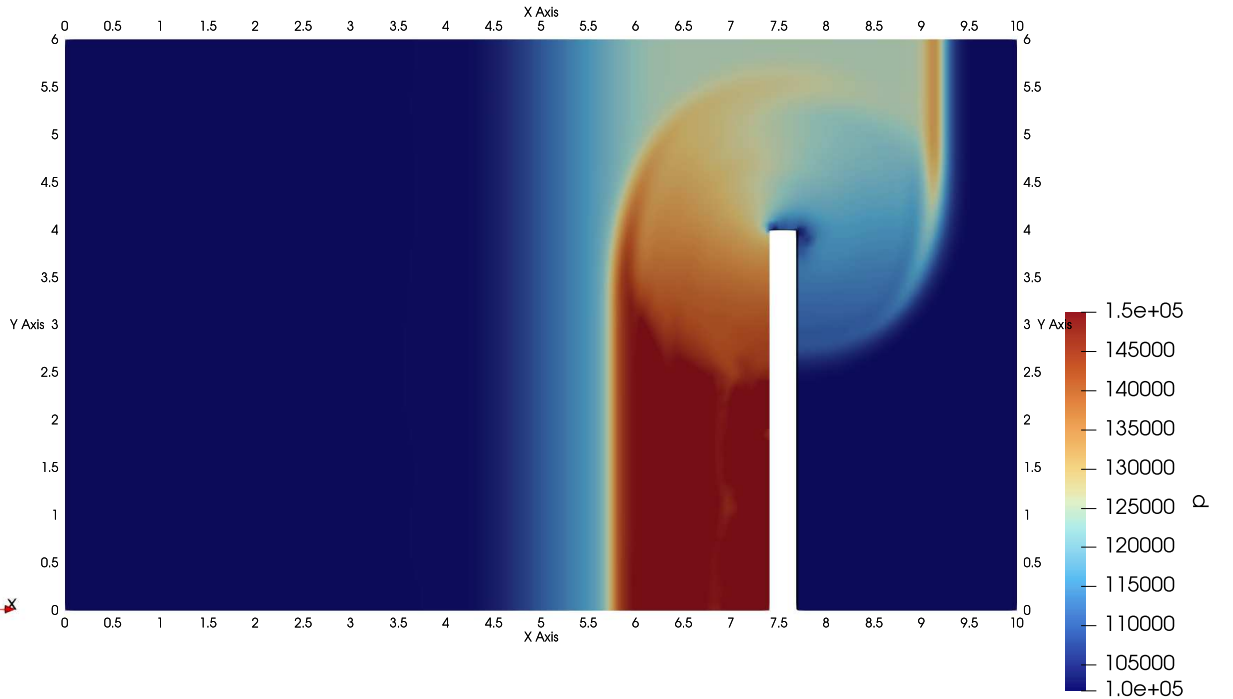


Figure 2.3: Blast Wave configuration: pressure field at $t = 0.02$ s. Post-processed in ParaView.

2.2 Mathematical formulation of fluid motion

2.2.1 The Navier-Stokes equations

Although fluids consist of molecules continuously in motion, most engineering applications deal with the average macroscopic effects generated by such particles. The **continuum hypothesis** allows to mathematically treat fluids as infinitely divisible without any change in character; material properties (such as viscosity, density or thermal conductivity) and variables (like pressure, temperature and velocity) can therefore be defined as the limit of the mean of such quantity across the molecular fluctuations.[6]

When the order of magnitude of the molecules mean free path is significantly smaller than the characteristic dimension of the problem of interest, continuum mechanics refers to the conceptual infinitesimal entities that constitute fluids as *fluid particles*, elementary volumes large enough to fit a statistically significant number of molecules within while being small enough to consider the corresponding mean parameters as constant. This simplification exempts the need to fully understand the internal micro- and nanostructure of fluid molecules to correctly predict their physical behaviors and corresponding mathematical descriptions, and flow fields therefore result characterized by properties that are continuous in space and time.

The governing equations of fluid motion are represented by a **system of highly non-linear, second order, partial differential equations set in four independent variables** (three spatial directions and time); by applying the principles of conservation of mass, momentum and total energy to a control volume, it's possible to solve for the scalar and vector fields that describe fluid particles behavior in unsteady three-dimensional viscous flows.

Conservation of mass (continuity equation)

The principle of conservation of mass states that, in the absence of mass sources and sinks, a region will conserve its mass on a local level.[7]

Considering the material volume of fluid shown in Figure 2.4 of mass m , density ρ and velocity $\mathbf{v}$, such kinematic condition can be written in an Eulerian coordinate system as:

$$\int_V \left(\frac{D\rho}{Dt} + \rho \nabla \cdot \mathbf{v} \right) dV = 0 \quad (2.1)$$

For the integral given in Eq.(2.1) to be true for any control volume V , the integrand should be equal to zero, giving the (non-conservative) differential form of the continuity equation as:

$$\frac{D\rho}{Dt} + \rho \nabla \cdot \mathbf{v} = 0 \quad (2.2)$$

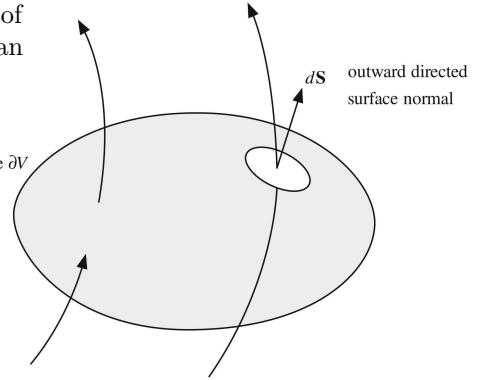


Figure 2.4: Conservation of mass for a material volume of fluid of mass m .

Lastly, by breaking down the expression of the density substantial derivative in Eq.(2.2) into its local and convective terms, the conservation of mass in flux (conservative) form is obtained:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0 \quad (2.3)$$

It is acceptable to assume that, in absence of any significant change in total pressure or temperature within the flow field, the fluid can be considered incompressible, hence characterized by fluid particles which keep their original density as they move along the respective streamlines. Such condition can be mathematically expressed as $\frac{D\rho}{Dt} = 0$, turning Eq.(2.2) into the continuity equation for incompressible flows given by:

$$\nabla \cdot \mathbf{v} = 0 \quad (2.4)$$

which in integral form can be expressed as follows:

$$\int_S (\mathbf{v} \cdot \mathbf{n}) dS = 0 \quad (2.5)$$

Eq.(2.5) states that the net flow of mass across any control volume within incompressible flows is equal to zero.

Conservation of linear momentum

The principle of conservation of linear momentum states that, in the absence of any external force acting on a body, the body retains its total momentum which is defined as the product of its mass and velocity vector. Since momentum is a vector quantity, its components in any direction will also be conserved.[7]

As Newton's second law of motion states that the material volume of any substance can change its momentum only in presence of a net force acting on it (including both surface forces and body forces), by considering the material volume of fluid shown in Figure 2.5 of mass m , density ρ and velocity $\mathbf{v}$, Newton's law can be written in an Eulerian coordinate system in two different ways: the conservative (or flux) form, and the non-conservative form.

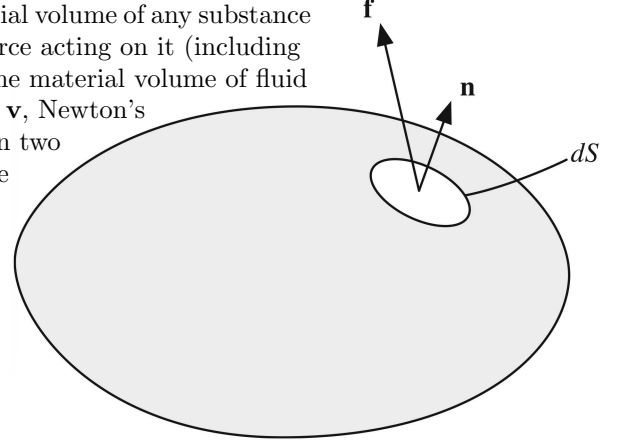


Figure 2.5: Conservation of linear momentum for a material volume of fluid of mass m .

Conservative form

By applying Reynolds transport theorem (as rigorously derived by Moukalled et al. in Section 3.3.2 of [7]) to the momentum balance equation, the conservative or flux version of the conservation principle is written in integral form as:

$$\int_V \left[\frac{\partial}{\partial t}(\rho \mathbf{v}) + \nabla \cdot (\rho \mathbf{v} \mathbf{v}) - \mathbf{f} \right] dV = 0 \quad (2.6)$$

where $\mathbf{f}$ is the external force per unit volume acting on the material volume. Such force is split into two terms, denoted by $\mathbf{f}_s$ and $\mathbf{f}_b$, which respectively represent surface and body forces so that $\mathbf{f} = \mathbf{f}_s + \mathbf{f}_b$. Once the terms $\mathbf{f}_s$ and $\mathbf{f}_b$ acting on the control volume are specified, the full form of the momentum equation is obtained; for a detailed characterization of these forces, the reader is referred to Sections 3.5.3 and 3.5.4 of Moukalled et al.[7]⁵.

The differential form of such equation is obtained by setting the integrand in Eq.(2.6) to zero for any volume V , leading to:

$$\frac{\partial}{\partial t}(\rho \mathbf{v}) + \nabla \cdot (\rho \mathbf{v} \mathbf{v}) = \mathbf{f} \quad (2.7)$$

Non-conservative form

By manipulating Eq.(2.6) and invoking the continuity constraint, the integral formulation of the momentum balance principle in its non-conservative form is given by:

$$\int_V \left[\frac{D}{Dt}(\rho \mathbf{v}) + (\rho \mathbf{v} \nabla \cdot \mathbf{v}) - \mathbf{f} \right] dV = 0 \quad (2.8)$$

For the integral given in Eq.(2.8) to be zero over any control volume V , the integrand must be zero, hence:

$$\frac{D}{Dt}(\rho \mathbf{v}) + (\rho \mathbf{v} \nabla \cdot \mathbf{v}) = \mathbf{f} \quad (2.9)$$

By expanding the substantial derivative of the momentum term and regrouping, the non-conservative form of the momentum equation in its differential form is obtained as:

$$\rho \frac{D\mathbf{v}}{Dt} + \underbrace{\mathbf{v} \left(\frac{D\rho}{Dt} + \rho \nabla \cdot \mathbf{v} \right)}_{\text{Continuity}} = \mathbf{f} \quad (2.10)$$

When the continuity constraint is applied to the equation above and the velocity substantial derivative is expanded, Eq.(2.10) reduces to:

$$\rho \left[\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla) \mathbf{v} \right] = \mathbf{f} \quad (2.11)$$

⁵For later reference: $\mathbf{f} = \mathbf{f}_s + \mathbf{f}_b = \nabla \cdot \mathbf{\Pi} + \mathbf{f}_b = -\nabla p + \nabla \cdot \boldsymbol{\tau} + \mathbf{f}_b$

By substituting the full expression⁵ of the external force per unit volume $\mathbf{f}$ into Eq.(2.12), the momentum equation becomes:

$$\rho \left[\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla) \mathbf{v} \right] = -\nabla p + \nabla \cdot \boldsymbol{\tau} + \mathbf{f}_b \quad (2.12)$$

For a complete expansion of the viscous stress tensor term, the reader is referred to Section 3.5.5 of Moukalled et al.[7], where the Newton-Stokes constitutive relation is substituted into $\boldsymbol{\tau}$ leading to the Navier-Stokes equation of linear momentum in its final explicit form.

Conservation of energy

The conservation of energy is governed by the first law of thermodynamics, which states that energy can be neither created nor destroyed during a process but can only change from one form (e.g. mechanical, kinetic, chemical) into another. Consequently, the sum of all forms of energy in an isolated system remains constant.[7]

Considering the material volume shown in Figure 2.6 of mass m , density ρ , and moving with a velocity $\mathbf{v}$, applying the first law of thermodynamics to such volume entails that the rate of change of its total energy (given by the sum of the internal and kinetic contributions) is equal to the rate of heat addition and work extraction through its boundaries. Mathematically, this is given by:

$$\left(\frac{dE}{dt} \right)_{MV} = \dot{Q} - \dot{W} \quad (2.13)$$

where the adopted sign convention is that work done by and heat added to the material volume are positive.

The net rate of heat transferred to the material element $\dot{Q}$ is given by $\dot{Q} = \dot{Q}_S + \dot{Q}_V$, where $\dot{Q}_S$ represents the rate transferred across the surface and $\dot{Q}_V$ the heat generated and destroyed within the material volume due to physical and chemical processes. The net rate of work done by the material volume $\dot{W}$ is made up, on the other hand, by the rate of work done by the surface forces $\dot{W}_S$ and the rate performed by body forces $\dot{W}_b$.

After some manipulations which allow to rewrite the first law of thermodynamics in terms of forces acting through a distance and apply the divergence theorem to the surface integral contributions, the mathematical description of energy conservation in terms of specific total energy (total energy per unit mass) is written as:

$$\int_V \left[\frac{\partial}{\partial t}(\rho e) + \nabla \cdot (\rho \mathbf{v} e) + \nabla \cdot \dot{q}_s + \nabla \cdot (p \mathbf{v}) - \nabla \cdot (\boldsymbol{\tau} \cdot \mathbf{v}) - \mathbf{f}_b \cdot \mathbf{v} - \dot{q}_V \right] dV = 0 \quad (2.14)$$

where $\dot{q}_V$ represents the rate of heat source or sink within the material volume per unit volume and $\dot{q}_s$ the rate of heat transfer per unit area across the surface area of the material element. For a complete formulation of these terms and their respective contributions, the reader is referred to Section 3.6.4 of Moukalled et al.[7].

For the volume integral in Eq.(2.14) to be true for any control volume V , the integrand must be zero resulting in the energy equation written in differential form as:

$$\frac{\partial}{\partial t}(\rho e) + \nabla \cdot (\rho \mathbf{v} e) = -\nabla \cdot \dot{q}_s - \nabla \cdot (p \mathbf{v}) + \nabla \cdot (\boldsymbol{\tau} \cdot \mathbf{v}) + \mathbf{f}_b \cdot \mathbf{v} + \dot{q}_V \quad (2.15)$$

The energy equation may also be written in terms of specific internal energy as follows.

Conservation of specific internal energy

The energy equation in terms of specific internal energy is defined by performing the dot product of the momentum equation with the velocity vector as follows:

$$\left[\frac{\partial}{\partial t}(\rho \mathbf{v}) + \nabla \cdot (\rho \mathbf{v} \mathbf{v}) \right] \cdot \mathbf{v} = \mathbf{f} \cdot \mathbf{v} \quad (2.16)$$

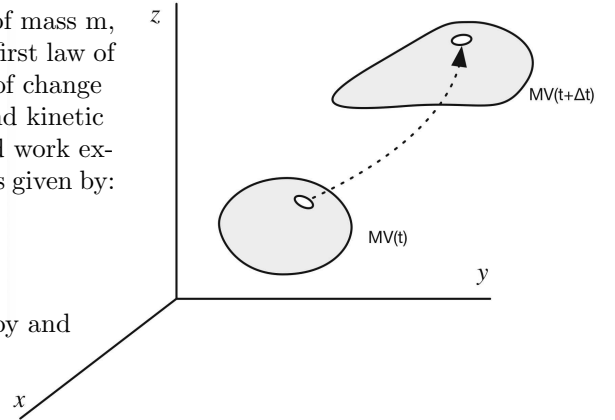


Figure 2.6: Material volume of mass m moving with the particles it encloses.

After some manipulations Eq.(2.16) becomes

$$\frac{\partial}{\partial t}(\rho \mathbf{v} \cdot \mathbf{v}) - \rho \mathbf{v} \cdot \frac{\partial \mathbf{v}}{\partial t} + \nabla \cdot [\rho(\mathbf{v} \cdot \mathbf{v})\mathbf{v}] - \rho \mathbf{v} \cdot [(\mathbf{v} \cdot \nabla)\mathbf{v}] = \mathbf{f} \cdot \mathbf{v} \quad (2.17)$$

Rearranging and collecting terms the following is obtained:

$$\frac{\partial}{\partial t}(\rho \mathbf{v} \cdot \mathbf{v}) + \nabla \cdot [\rho(\mathbf{v} \cdot \mathbf{v})\mathbf{v}] - \underbrace{\mathbf{v} \cdot \rho \left[\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla)\mathbf{v} \right]}_{= \mathbf{f} \text{ (From Eq.(2.12))}} = \mathbf{f} \cdot \mathbf{v} \quad (2.18)$$

By replacing $\mathbf{f}$ with its equivalent expression from Eq.(2.12), the balance equation for the flow kinetic energy is obtained as:

$$\frac{\partial}{\partial t} \left(\rho \frac{1}{2} \mathbf{v} \cdot \mathbf{v} \right) + \nabla \cdot \left[\rho \left(\frac{1}{2} \mathbf{v} \cdot \mathbf{v} \right) \mathbf{v} \right] = -\mathbf{v} \cdot \nabla p + \mathbf{v} \cdot (\nabla \cdot \boldsymbol{\tau}) + \mathbf{f}_b \cdot \mathbf{v} \quad (2.19)$$

By rewriting Eq.(2.19) in the following form:

$$\frac{\partial}{\partial t} \left(\rho \frac{1}{2} \mathbf{v} \cdot \mathbf{v} \right) + \nabla \cdot \left[\rho \left(\frac{1}{2} \mathbf{v} \cdot \mathbf{v} \right) \mathbf{v} \right] = -\nabla \cdot (p\mathbf{v}) + p\nabla \cdot \mathbf{v} + \nabla \cdot (\boldsymbol{\tau} \cdot \mathbf{v}) - (\boldsymbol{\tau} : \nabla \mathbf{v}) + \mathbf{f}_b \cdot \mathbf{v} \quad (2.20)$$

and subtracting Eq.(2.20) from Eq.(2.15), the energy equation in terms of specific internal energy is obtained as follows:

$$\frac{\partial}{\partial t}(\rho \hat{u}) + \nabla \cdot (\rho \mathbf{v} \hat{u}) = -\nabla \cdot \dot{q}_s - p\nabla \cdot \mathbf{v} + (\boldsymbol{\tau} : \nabla \mathbf{v}) + \dot{q}_V \quad (2.21)$$

where the sign $:$ is the double dot product between two tensors, defined as: $\boldsymbol{\tau} : \nabla \mathbf{v} = \sum_i \sum_j \tau_{ij} \frac{\partial v_i}{\partial x_j}$

The Navier-Stokes equations

By introducing the constitutive relations (as derived and formulated by Moukalled et al. in [7]) into the principles of conservation that were just presented, it is possible to obtain the Navier-Stokes equations system for an unsteady three-dimensional viscous flows as follows:

$$\begin{cases} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0 \\ \rho \left(\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla)\mathbf{v} \right) = -\nabla p + \nabla \cdot [\mu (\nabla \mathbf{v} + (\nabla \mathbf{v})^T) + \lambda (\nabla \cdot \mathbf{v})\mathbf{I}] + \mathbf{f}_b \\ \rho \left(\frac{\partial \hat{u}}{\partial t} + (\mathbf{v} \cdot \nabla)\hat{u} \right) = -p\nabla \cdot \mathbf{v} + [\mu (\nabla \mathbf{v} + (\nabla \mathbf{v})^T) + \lambda (\nabla \cdot \mathbf{v})\mathbf{I}] : \nabla \mathbf{v} + \nabla \cdot (k\nabla T) + \dot{q}_V \end{cases} \quad (2.22)$$

While this system represents the full mathematical formulation of fluid motion, its strongly non-linear nature due to the convective terms prevents the determination of a general closed-form analytical solution, making numerical approaches necessary for all but the simplest flow configurations. Physical assumptions about the fluid of interest can nevertheless lead to significant simplifications within the governing equations, with a notable example being the **incompressible flow assumption**: under this hypothesis, fluid particles within domains characterized by no significant variation in total pressure or temperature maintain their original density as they travel along their respective streamlines. Eq.(2.22) can be therefore rewritten as follows:

$$\begin{cases} \nabla \cdot \mathbf{v} = 0 \\ \rho \left(\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla)\mathbf{v} \right) = -\nabla p + \mu \nabla^2 \mathbf{v} + \mathbf{f}_b \\ \rho \left(\frac{\partial \hat{u}}{\partial t} + (\mathbf{v} \cdot \nabla)\hat{u} \right) = \left(\frac{\mu}{2} \|\nabla \mathbf{v} + (\nabla \mathbf{v})^T\|^2 \right)^6 + k\nabla^2 T + \dot{q}_V \end{cases} \quad (2.23)$$

⁶ $[\mu (\nabla \mathbf{v} + (\nabla \mathbf{v})^T)] : \nabla \mathbf{v} = \frac{\mu}{2} \|\nabla \mathbf{v} + (\nabla \mathbf{v})^T\|^2 \equiv \mu \Phi$

where $\|\cdot\|$ denotes the Frobenius norm and $\frac{1}{2}\|\nabla\mathbf{v} + (\nabla\mathbf{v})^T\|^2 = \Phi \geq 0$ is the viscous dissipation function, that is the rate at which kinetic energy is irreversibly converted into internal energy (heat) by viscous friction.

Since density is treated as constant by hypothesis within domains under the incompressible assumption, such variable carries no information about the local thermodynamic state of the system rendering the equation of state (which in general relates $\rho = f(p, T)$) superfluous to the description of the field dynamics. As the dynamic viscosity μ is likewise assumed constant and independent of temperature, the momentum equation forms a closed system with the continuity condition in the unknowns $(\mathbf{v}, p)$, which can be solved without any prior knowledge of the thermal field expression. Once the velocity field is fully solved for, it enters the energy equation as a known advection term giving rise to a one-way coupling between the thermal and the mechanical problem: while the flow field governs the evolution of temperature affecting its distribution, temperature exerts no feedback on the flow, allowing to conclude that under such physical assumptions the energy equation is decoupled from the other two and heat transfer is not always a primary concern for incompressible flows.

2.2.2 Normal shock relations

A shock wave is defined as a compression sound wave of finite strength that propagates through a fluid medium, inducing an abrupt, discontinuous change in the flow properties due to its extremely small thickness (typically on the order of micrometers); when such wave travels perpendicularly to the flow direction, it's referred to as a **normal shock wave**. The high gradients of velocity and temperature registered within the field due to the presence of such discontinuities lead to entropy production phenomena, making isentropic relations inapplicable across the shock; by manipulating the Navier-Stokes equations under the simplifying assumptions of one-dimensional, steady compressible flows, it is however possible to derive a set of equations relating the upstream and downstream properties of the flow across these waves: the normal shock relations⁷.

Considering the control volume shown in Figure 2.7 enclosing a normal shock wave within a fluid domain where 1 and 2 respectively denote the upstream and downstream flow states, the governing equations of fluid motion presented in Eq.(2.22) can be rewritten as follows:

$$\begin{cases} \rho_1 v_1 = \rho_2 v_2 \\ \rho_1 v_1^2 + p_1 = \rho_2 v_2^2 + p_2 \\ h_1 + \frac{1}{2}v_1^2 = h_2 + \frac{1}{2}v_2^2 \end{cases} \quad (2.24)$$

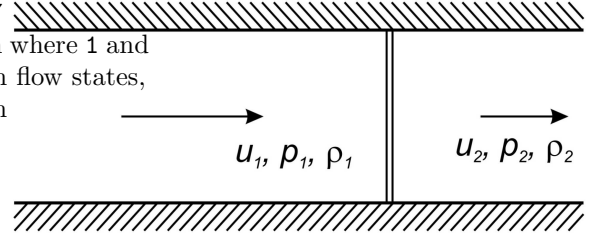


Figure 2.7: Stationary normal shock wave.[8]

under the simplifying hypothesis of one-dimensional, steady compressible flow.

Prandtl relation

Starting from the momentum equation: $\rho_1 v_1^2 + p_1 = \rho_2 v_2^2 + p_2$, dividing both sides by $\rho_1 v_1$ gives:

$$\frac{\rho_1 v_1^2}{\rho_1 v_1} + \frac{p_1}{\rho_1 v_1} = \frac{\rho_2 v_2^2 + p_2}{\rho_1 v_1} \implies v_1 + \frac{p_1}{\rho_1 v_1} = \frac{\rho_2 v_2^2 + p_2}{\rho_1 v_1} \quad (2.25)$$

By substituting $\rho_1 v_1 = \rho_2 v_2$ from the continuity equation to Eq.(2.25), the right-hand side becomes:

$$v_1 + \frac{p_1}{\rho_1 v_1} = \frac{\rho_2 v_2^2 + p_2}{\rho_2 v_2} \implies v_1 + \frac{p_1}{\rho_1 v_1} = \frac{\rho_2 v_2^2}{\rho_2 v_2} + \frac{p_2}{\rho_2 v_2} \implies v_1 + \frac{p_1}{\rho_1 v_1} = v_2 + \frac{p_2}{\rho_2 v_2} \quad (2.26)$$

Rearranging, Eq.(2.26) can be written as:

$$\frac{p_1}{\rho_1 v_1} - \frac{p_2}{\rho_2 v_2} = v_2 - v_1 \quad (2.27)$$

For a calorically perfect gas, where the speed of sound is defined as $a = \sqrt{\frac{\gamma p}{\rho}}$, Eq.(2.27) becomes:

$$\frac{a_1^2}{\gamma_1 v_1} - \frac{a_2^2}{\gamma_2 v_2} = v_2 - v_1 \quad (2.28)$$

⁷ScienceDirect: <https://www.sciencedirect.com/topics/engineering/normal-shock-wave>

Now shifting the attention to the energy equation, by expanding the definitions of enthalpy ($h = C_p T$) and specific heat at constant pressure ($C_p = \frac{\gamma R}{\gamma - 1}$) the balance reduces to:

$$C_p T_1 + \frac{1}{2} v_1^2 = C_p T_2 + \frac{1}{2} v_2^2 \implies \frac{\gamma R T_1}{\gamma - 1} + \frac{1}{2} v_1^2 = \frac{\gamma R T_2}{\gamma - 1} + \frac{1}{2} v_2^2 \quad (2.29)$$

By remembering the definition of the speed of sound a , Eq.(2.29) results:

$$\frac{a_1^2}{\gamma - 1} + \frac{1}{2} v_1^2 = \frac{a_2^2}{\gamma - 1} + \frac{1}{2} v_2^2 \quad (2.30)$$

which represents an alternative way of writing the energy balance equation between two generic states within the flow. By rearranging Eq.(2.30) to relate the flow velocity v and speed of sound a in any point of the field to the corresponding flow properties at sonic conditions (where $v = a = a^*$, indicated by the superscript $*$), the equilibrium reduces to:

$$\frac{a^2}{\gamma - 1} + \frac{1}{2} v^2 = \frac{a^{*2}}{(\gamma - 1)} + \frac{1}{2} a^{*2} \implies \frac{a^2}{\gamma - 1} + \frac{1}{2} v^2 = \frac{\gamma + 1}{2(\gamma - 1)} a^{*2} \quad (2.31)$$

$$\implies a^2 = -\frac{\gamma - 1}{2} v^2 + \frac{\gamma + 1}{2} a^{*2} \quad (2.32)$$

By evaluating Eq.(2.32) in the upstream and downstream flow states (1 and 2, respectively) of the control volume shown in Figure 2.7, the following relations can be derived:

$$\begin{cases} a_1^2 = \frac{\gamma + 1}{2} a^{*2} - \frac{\gamma - 1}{2} v_1^2 \\ a_2^2 = \frac{\gamma + 1}{2} a^{*2} - \frac{\gamma - 1}{2} v_2^2 \end{cases} \quad (2.33)$$

By replacing the values of a_1^2 and a_2^2 from the energy equation given by Eq.(2.33) in Eq.(2.27), it is possible to derive the relation between the upstream and downstream velocities across a normal shock wave by performing the analytical manipulations reported below.

$$\begin{aligned} & \frac{1}{\gamma v_1} \left(\frac{\gamma + 1}{2} a^{*2} - \frac{\gamma - 1}{2} v_1^2 \right) - \frac{1}{\gamma v_2} \left(\frac{\gamma + 1}{2} a^{*2} - \frac{\gamma - 1}{2} v_2^2 \right) = v_2 - v_1 \\ \implies & \left(\frac{\gamma + 1}{2\gamma} \right) a^{*2} \left(\frac{1}{v_1} - \frac{1}{v_2} \right) = \left(\frac{\gamma + 1}{2\gamma} \right) (v_2 - v_1) \implies a^{*2} \left(\frac{1}{v_1} - \frac{1}{v_2} \right) = (v_2 - v_1) \\ \implies & a^{*2} \left(\frac{v_2}{v_1 v_2} - \frac{v_1}{v_1 v_2} \right) = (v_2 - v_1) \implies \frac{1}{v_1 v_2} a^{*2} (v_2 - v_1) = (v_2 - v_1) \end{aligned} \quad (2.34)$$

The algebraic manipulations presented above culminate in the formulation of a cornerstone result in normal shock wave theory: the **Prandtl relation**.

$$a^{*2} = v_1 v_2 \quad (2.35)$$

As expressed by Eq.(2.35), only two possibilities are available when correlating the upstream and downstream velocities across a normal shock wave:

1. $v_1 > a^*$ and $v_2 < a^*$: such condition implies that the flow speed reduces through the shock wave;
2. $v_1 < a^*$ and $v_2 > a^*$: such condition implies that the flow speed across the shock wave increases, violating the 2nd law of thermodynamics and therefore being physically impossible.

Dividing Eq.(2.35) by a^{*2} on both sides gives:

$$1 = \frac{v_1}{a^*} \frac{v_2}{a^*} = M_1^* M_2^* \quad (2.36)$$

where $M^* = \frac{v}{a^*}$ is generally referred to as the speed ratio. The Prandtl relation can thus be reinterpreted as:

1. $M_1^* > 1$ and $M_2^* < 1$: such condition implies an upstream supersonic Mach number;
2. $M_1^* < 1$ and $M_2^* > 1$: such condition implies an upstream subsonic Mach number, violating the 2nd law of thermodynamics and therefore being physically impossible.

M^* can be related to the Mach number M at any point of the given fluid domain by dividing the energy equation as expressed in Eq.(2.31) by v^2 :

$$\frac{1}{v^2} \cdot \left(\frac{a^2}{\gamma - 1} + \frac{1}{2} v^2 \right) = \frac{1}{v^2} \cdot \left(\frac{\gamma + 1}{2(\gamma - 1)} a^{*2} \right) \implies \frac{1}{M^2(\gamma - 1)} + \frac{1}{2} = \frac{\gamma + 1}{2(\gamma - 1)} \frac{1}{M^{*2}} \quad (2.37)$$

resulting in the following:

$$M^{*2} = \frac{(\gamma + 1)M^2}{2 + (\gamma - 1)M^2} \quad (2.38)$$

By substituting this expression in Eq.(2.36) it is possible to express $M_2 = f(M_1)$, that is the dependency between the upstream (1) and downstream (2) Mach number registered across a normal shock wave.

$$\frac{(\gamma + 1)M_1^2}{2 + (\gamma - 1)M_1^2} \cdot \frac{(\gamma + 1)M_2^2}{2 + (\gamma - 1)M_2^2} = 1 \implies \frac{2}{M_2^2} = \frac{4\gamma M_1^2 - 2(\gamma - 1)}{2 + (\gamma - 1)M_1^2} \quad (2.39)$$

By rearranging for M_2^2 , Eq.(2.39) becomes:

$$M_2^2 = \frac{2 + (\gamma - 1)M_1^2}{2\gamma M_1^2 - (\gamma - 1)} \implies \begin{cases} M_2 \rightarrow 1 & \text{if } M_1 \rightarrow 1 \\ M_2 \rightarrow \sqrt{\frac{\gamma - 1}{2\gamma}} \approx 0.378 & \text{if } M_1 \rightarrow \infty \end{cases} \quad (2.40)$$

proving that for supersonic values of the upwind Mach number M_1 the downwind Mach number M_2 is always subsonic.

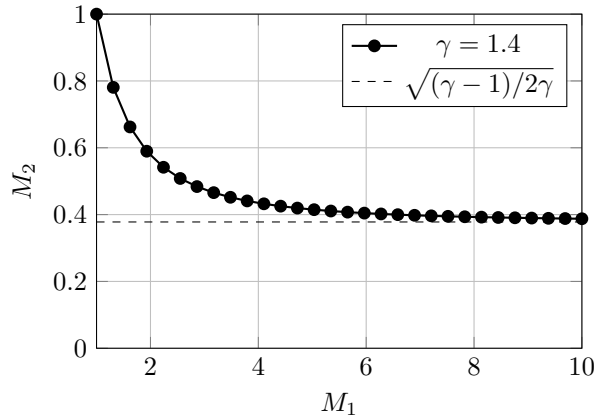


Figure 2.8: $M_2 = f(M_1)$ across a normal shock wave.

Shock jump conditions: density and velocity ratios

By manipulating the continuity equation (Eq.(2.24)) and invoking the Prandtl relation (Eq.(2.35)), the density ratio across a normal shock wave is given by:

$$\rho_1 v_1 = \rho_2 v_2 \implies \frac{\rho_2}{\rho_1} = \underbrace{\frac{v_1 \cdot v_1}{v_2 \cdot v_1}}_{\text{Eq.(2.35)}} = \frac{v_1^2}{a^{*2}} = M_1^{*2} \implies \frac{\rho_2}{\rho_1} = \frac{(\gamma + 1)M_1^2}{2 + (\gamma - 1)M_1^2} \quad (2.41)$$

From the continuity equation, the density ratio across a normal shock wave varies inversely with the velocity ratio; Eq.(2.41) therefore becomes:

$$\frac{\rho_2}{\rho_1} = \frac{v_1}{v_2} = \frac{(\gamma + 1)M_1^2}{2 + (\gamma - 1)M_1^2} \quad (2.42)$$

showing that the flow decelerates and the static density rises across the shock.

Shock jump conditions: the pressure ratio

By manipulating the momentum equation (Eq.(2.24)) and invoking the continuity condition, the pressure jump across a normal shock wave is given by:

$$p_2 - p_1 = \rho_1 v_1^2 - \rho_2 v_2^2 \xRightarrow{\rho_1 v_1 = \rho_2 v_2} p_2 - p_1 = \rho_1 v_1 \underbrace{(v_1 - v_2)}_{>0 \implies \Delta p > 0} = \rho_1 v_1^2 \left(1 - \frac{v_2}{v_1}\right) \quad (2.43)$$

Remembering that $a^2 = \gamma RT = \gamma \frac{p}{\rho}$, it's possible to replace $\rho = \gamma \frac{p}{a^2}$ in Eq.(2.43) as:

$$p_2 - p_1 = \frac{\gamma p_1}{a_1^2} v_1^2 \left(1 - \frac{v_2}{v_1}\right) = \gamma p_1 M_1^2 \left(1 - \frac{v_2}{v_1}\right) \quad (2.44)$$

As Eq.(2.42) expresses $\frac{v_1}{v_2} = f(M_1)$, expanding such expression in Eq.(2.44) gives:

$$p_2 - p_1 = \gamma p_1 M_1^2 \left(1 - \frac{2 + (\gamma - 1)M_1^2}{(\gamma + 1)M_1^2}\right) \implies \frac{p_2 - p_1}{p_1} = \gamma M_1^2 \left(1 - \frac{2 + (\gamma - 1)M_1^2}{(\gamma + 1)M_1^2}\right) \quad (2.45)$$

which simplifies to:

$$\frac{p_2}{p_1} = \frac{2\gamma M_1^2 - (\gamma - 1)}{(\gamma + 1)} \quad (2.46)$$

or, in relative terms to get an indication of the shock strength:

$$\frac{p_2 - p_1}{p_1} = \frac{\Delta p}{p_1} = \frac{2\gamma(M_1^2 - 1)}{(\gamma + 1)} \quad (2.47)$$

The illustration of $\frac{p_2}{p_1} = f(M_1)$ in Figure 2.10 shows that the pressure rises sharply across the normal shock wave, reaching a point of divergence as $M_1 \rightarrow \infty$.

Shock jump conditions: the temperature ratio

By taking the ratio of state equation ($p = \rho RT$) between the downstream (2) and upstream (1) conditions of a shock wave, one gets:

$$\frac{p_2}{p_1} = \frac{\rho_2}{\rho_1} \frac{T_2}{T_1} \quad (2.48)$$

where the pressure and density ratios are given by Eq.(2.46) and Eq.(2.42), respectively.

Replacing the aforementioned expressions in Eq.(2.48) and rearranging to isolate $\frac{T_2}{T_1}$ gives:

$$\frac{T_2}{T_1} = \left(\frac{a_2}{a_1}\right)^2 = 1 + \left[\frac{2(\gamma - 1)(\gamma M_1^2 + 1)(M_1^2 - 1)}{(\gamma + 1)^2 M_1^2} \right] \quad (2.49)$$

which shows that the static temperature rises too across the shock reaching a point of divergence as $M_1 \rightarrow \infty$ (though at a slower rate than pressure).

In conclusion, all the thermodynamics parameters defining the control volume shown in Figure 2.7 increase across the shock; Figure 2.9, 2.10 and 2.11 additionally show that as the shock rises in intensity the pressure and temperature fields reach a divergence condition while density attains an asymptotic value.

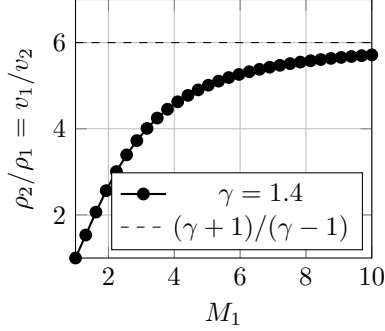


Figure 2.9: $\frac{\rho_2}{\rho_1} = f(M_1)$.

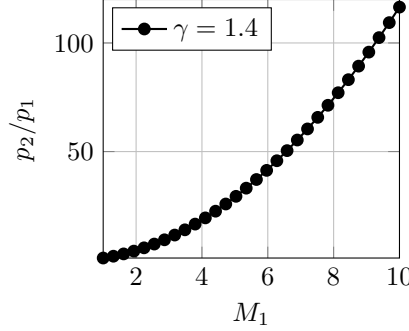


Figure 2.10: $\frac{p_2}{p_1} = f(M_1)$.

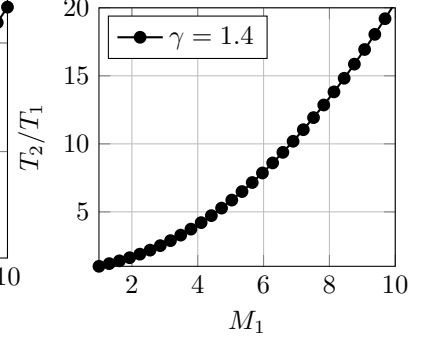


Figure 2.11: $\frac{T_2}{T_1} = f(M_1)$.

Thermodynamics ratios as a function of M_1 across a normal shock wave.

2.3 Numerical solvers and algorithms

2.3.1 The pressure-based approach

Numerical methods for the solution of fluid flows have traditionally been divided into two categories, based on the choice of primary unknown through which mass conservation is enforced when solving the Navier-Stokes equations: pressure-based solvers and density-based solvers.[9] [10] While both

formulations determine the velocity field expression from the linear momentum equations, a **pressure-based** approach treats pressure as the main thermodynamic variable to solve for within the flow and the continuity equation is repurposed as a kinematic constraint on the velocity field, so that a pressure correction equation (that enforces mass conservation) is derived. **Density-based** solvers, on the other hand, directly solve the continuity equation with density as the primary unknown, and pressure is subsequently recovered from the equation of state.

Historically, the preference for one method over the other was driven by the nature of the flow regime under investigation: whilst pressure-based solvers resulted quite efficient in resolving low-speed incompressible flows on structured staggered grids, density-based approaches were predominantly used to simulate transonic and supersonic fields that were typically encountered in aerodynamics applications for the aeronautical industry. Nevertheless, considerable efforts were devoted towards extending the applicability of each numerical method to flow regimes traditionally governed by the opposing approach, and to enable pressure-based solvers to correctly represent compressible subsonic, transonic and supersonic fields on unstructured collocated grids, a fundamental change in character for the pressure equation that is solved throughout the simulation was thereby implemented.

In the incompressible limit, the pressure equation is elliptic in nature as perturbations propagate instantaneously across the entire domain in all directions; compressible flows by contrast don't broadcast information globally but rather transport it directionally, due to the finite speed of disturbance propagation and the strictly downstream nature of information conveyance. For this reason, to correctly describe the mathematical behavior of such fields the pressure equation must rather present a hyperbolic character, and that is achieved through the introduction of a pseudo-advection term that increases in strength with the Mach number. Since density is no longer constant within compressible fields and pressure and density are coupled through the equation of state, its variations must be explicitly accounted for and they can be expressed in terms of pressure corrections as $\delta\rho = \frac{\partial\rho}{\partial p}\delta p$. By substituting this coupling into the continuity equation, the pseudo-advection term of pressure arises as the mathematical fingerprint for density-pressure coupling, and since its strength scales with M^2 it vanishes at low Mach numbers thus preserving the elliptic nature of the pressure equation for nearly incompressible flows. As the term increasingly dominates

within higher Mach numbers fields, however, the pressure equation character naturally transitions from elliptic to hyperbolic.

Such correction thereby relies on the dual role played by pressure within compressible fluid fields at different Mach numbers, which can be best summarized by considering the extreme cases presented below:

1. limit of incompressible flows ($M \leq 0.3$): pressure affects velocity through its gradient in the momentum equation, and it serves as the constraint through which mass conservation is enforced as at very low Mach numbers density is weakly related to both pressure and velocity. Such loose coupling consequently reduces the continuity equation to a kinematic constraint on the velocity field, as the near-independence of density from velocity variations renders it no longer meaningful as the equation for the evolution of density.
2. limit of hypersonic flows ($M \geq 5$): the continuity equation governs the evolution of density, which is strongly dependent on pressure through the equation of state. Pressure thereby affects the continuity and momentum equations simultaneously by acting on velocity through its gradient in the former and on density through the equation of state in the latter.

Advantages of pressure-based methods over density-based solvers

Just as pressure-based solvers required a shift in character (from elliptic to hyperbolic) for the pressure correction equation to extend their applicability to compressible flow regimes, density-based methods had to restore to *artificial* compressibility techniques (which lower the speed of sound to a closer value to the local flow velocity) in order to operate in the incompressible flows limit while avoiding performance degradation from arising.

As density-based solvers propagate information at the speed of sound, when this parameter is vastly greater than the actual flow velocity the system matrices that formulate the numerical problem tend to become very stiff, thus causing convergence degradation to occur; preconditioning techniques are therefore necessary to artificially reduce the speed of sound, essentially reformulating the density equation into some form of pressure equation.

Since this technique effectively borrows the mathematical structure of pressure-based solvers in applications where density-based formulations fall short and are inherently ill-conditioned, an important advantage presented by pressure-based methods lies in their ability to guarantee robustness, promote convergence and stabilize computations without restoring to artificial treatments but rather pseudo-advection terms that encode the density-pressure coupling.

2.3.2 Pressure-based solvers in OpenFOAM

To address pressure-momentum coupling in the Navier-Stokes equations OpenFOAM provides several algorithms, the most widely used being SIMPLE, PISO, and PIMPLE. For the purpose of this thesis, both numerical configurations discussed in Section 2.1.3 have been implemented choosing solvers based on the PIMPLE algorithm (`pimpleFoam` and `rhoPimpleFoam`), thus the remainder of this chapter will provide an overview of this numerical method and a brief description of the algorithm flowchart.[11]

In its incompressible form, the pressure-momentum coupling problem arises because the momentum equation (Eq.(2.23)) contains four unknowns (pressure p and the three components of velocity $\mathbf{v}$) with only three scalar equations to represent the equilibrium (one for each spatial direction in $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$), so a fourth equation is needed to describe the physical problem in order to close the system. Since the continuity equation governs the evolution of density and contains no pressure term, Eq.(2.23) must be algebraically manipulated to derive the pressure-dependent relation known as the Poisson equation, given by:

$$\nabla^2 p = -\rho \nabla \cdot (\mathbf{u} \cdot \nabla \mathbf{u}) \quad (2.50)$$

By combining the momentum equation in its vector form with Eq.(2.50), the system is fully defined by four equations in four unknowns and it's ready to be solved.

The strategy behind the numerical algorithm is to find a pressure and velocity field that satisfy the mass conservation equation. To that end, the system is typically treated sequentially: each variable is computed independently while the others are held constant, with velocity being determined first and pressure subsequently.

As the PIMPLE algorithm is used to simulate transient problems by combining the PISO and SIMPLE strategies, a brief outline of each method is presented below.

- **SIMPLE**: Semi-Implicit-Method-Of-Pressure-Linked-Equations. OpenFOAM usually implements such algorithm for steady-state analysis.
- **PISO**: Pressure-Implicit-of-Split-Operations. OpenFOAM relies on this method for transient calculations where the time step is limited by the Courant number condition ($Co < 1$).
- **PIMPLE**: Merged PISO–SIMPLE algorithm. Solvers relying on such algorithm combine both the SIMPLE and PISO resolution procedures, thus enabling the use of larger time steps ($Co \gg 1$).

2.3.3 The PIMPLE algorithm

The main advantage of the PIMPLE algorithm is that time step adopted in the simulation can be increased drastically, as the implicit nature of the numerical method allows Courant numbers significantly greater than one ($Co \gg 1$) to be used. The PIMPLE algorithm also results more stable than the standalone SIMPLE approach, though the increased robustness comes at the price of higher computational cost.

The principle of the algorithm, illustrated by the flow charts in Figure 2.12 and 2.13, is as follows.

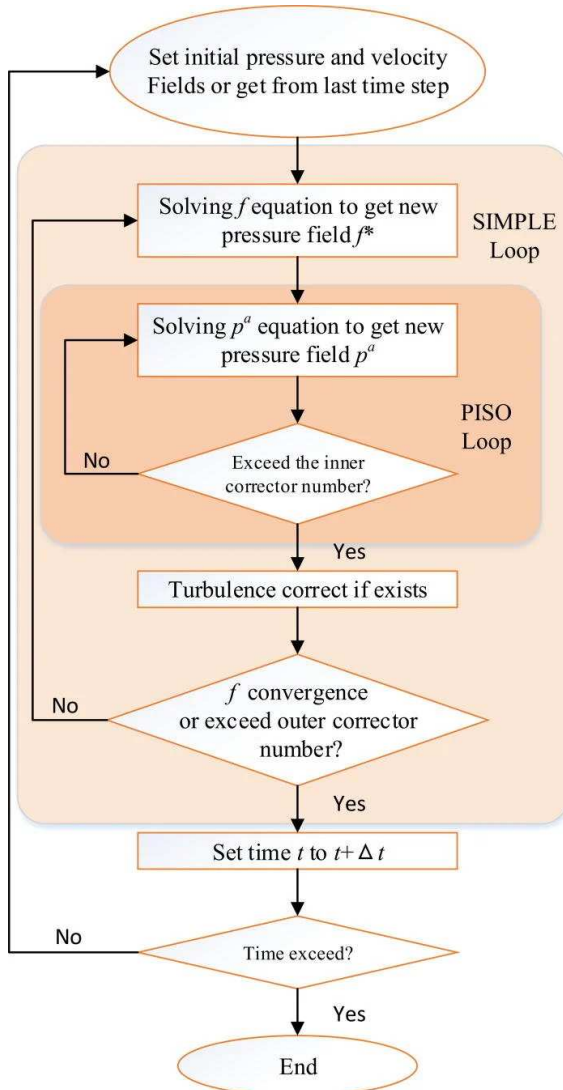


Figure 2.12: Qualitative flowchart of the PIMPLE algorithm.[12] f represents the linear momentum.

Within a single time step the PIMPLE algorithm is governed by an outer loop, referred to as the SIMPLE loop, and an inner cycle, called the PISO loop. The aim of the outer loop is to search for a steady-state solution of the flow field, essentially adopting the convergence strategy of SIMPLE-based solvers. To this end, the SIMPLE loop iterates internally until the flow field variables (e.g. velocity, pressure) satisfy the equilibrium condition, and an under-relaxation strategy is adopted to stabilize this iterative process preventing it from overshooting. In other words, rather than applying the full solution computed at each iteration to advance the field expression, only a fraction of it is used to slow down the flow update and damp numerical oscillations. As all explicit terms in the governing equations are computed with values from previous iterations, the outer corrector loops ensure that they are consistent with the current solution and have reached full convergence.

Nested inside each SIMPLE loop iteration, the PISO cycle is run to iteratively couple the pressure and velocity fields until both the momentum and continuity equations are simultaneously satisfied. The number of PISO corrector iterations is governed by two parameters found in the source code: `nCorrPiso` and `corrPiso`. The former sets the number of corrector iterations performed per inner loop, and is defined as a user input in the `fvSolution` dictionary to allow for direct control of the pressure-velocity coupling accuracy at the expense of additional computational cost. The actual number of corrector iterations performed is then tracked by `corrPiso`, a counter that resets at the end of every PISO cycle before the algorithm advances to the next SIMPLE iteration.

The PIMPLE algorithm at each given time step exits the SIMPLE loop once the governing equations residuals drop below a user-defined threshold, meaning that the solution has satisfied a tolerance criterion deemed acceptable against the `residualControl` parameter configured in the `fvSolution`

dictionary; at that point the converged state is treated as the solution for the respective time instant, and the process is repeated with the next one until the simulation reaches its prescribed end time.

While the pressure-velocity coupling problem is typical of incompressible flows, pressure-based solvers can be extended to compressible regimes where density variations must also be accounted for, as discussed in Section 2.3.1. Since such configurations require the density field to be calculated at each time step, solvers implementing the PIMPLE algorithm such as `rhoPimpleFoam` leverage the pressure-density coupling given by the equation of state to obtain it; the pressure solution from each PIMPLE iteration is thus passed to the thermodynamic library so that it is used to determine the density field accordingly.

From a quantitative analysis of the OpenFOAM source code performed by Holzmann [11], it's possible to conclude that the `pimpleControl` object (which is declared in the `pimpleControl.H` header file) stores the following information:

- **nCorrectors**: number of PISO correction iterations performed;
- **nOuterCorrectors**: amount of SIMPLE iterations configured;
- whether turbulence properties are updated for each outer iteration or only at the last one, if applicable;
- convergence criterion information to stop the solver for steady-state simulations, if applicable;
- convergence criterion information to stop the steady-state calculation within each transient simulation time step, if applicable.

The settings which control the PIMPLE algorithm configuration are defined in the test case `fvSolution` dictionary; users can set the method to operate only one SIMPLE loop by setting `nOuterCorrectors = 1`, essentially restoring the solver to its PISO mode. Of the two source code parameters which control the inner cycle (`nCorrPiso` and `corrPiso`), only `nCorrPiso` is explicitly defined in the `fvSolution` dictionary through the `nCorrectors` command.

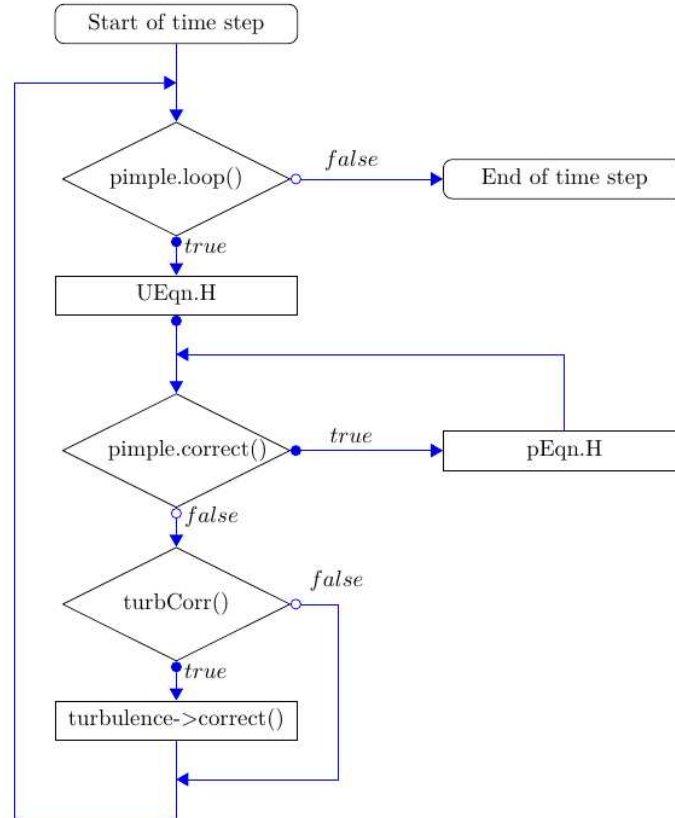


Figure 2.13: Implementation-level flow chart of the PIMPLE algorithm in OpenFOAM.[13] `pimple.loop()` represents the SIMPLE cycle, while `pimple.correct()` the PISO loop.

Chapter 3

Structural Response Simulation: Physical and Numerical Principles

The aim of this chapter is to present the notions that were used when implementing the structural configurations defined across this thesis. A brief overview of the OpenRadioss software is provided, followed by a rundown of the physical and numerical properties encountered when characterizing the structural participants that have been modeled throughout the project.

3.1 Overview of the OpenRadioss software

The Finite Element Method, or Analysis (FEM or FEA), is a numerical technique used to find approximate solutions to complex physical problems by subdividing a computational domain into a set of interconnected, simple geometric parts (e.g. triangular or prism-shaped) referred to as finite elements¹. Based on the partial differential equations which govern complex real-world phenomena (typically related to material responses under external stress), shape functions over each element are thus developed and combined to find an approximate solution which minimizes the potential energy of the assembled discretized system². This approach is widely used in structural design to analyze and predict stresses and deformation within articulate, real-world engineering systems, following a structured multi-stage modeling procedure consisting of a *pre-processing*, *processing*, and *post-processing* phase as outlined in Chapter 2.

OpenRadioss³ is the free, open-source version of the long-standing industry-proven FEA dynamic simulation code known as Altair Radioss⁴. Developed starting from the late 1980s by the CAE software company Mecalog, the commercial version of Radioss has been integrated into HyperWorks in 2006 following Altair's acquisition of Mecalog and has since been employed by several industrial, research and development, and educational organizations to solve complex linear and non-linear engineering problems. Ever since its first introduction, the explicit dynamics simulation code has been used to represent and simulate highly dynamic events such as shocks and impacts, automotive crashes and safety assessments, bird strikes and emergency landing procedures and consumer electronics drop testing.

In September 2022 the Altair company introduced OpenRadioss as a free, open-source FEA alternative which relies on robust numerical methods based on explicit time integration schemes and uses the same input format and source code as the Radioss commercial software⁵. The aim of this initiative has been to embrace the four essential freedoms of a free software in order to bring together a worldwide community of experts and users, so that the latest challenges brought on by rapidly evolving technologies could be addressed more cohesively. By empowering end-users to directly contribute to the development of tools that commercial alternatives have yet to implement, OpenRadioss leverages the first-hand expertise of those who rely on the software daily as they are those best positioned to drive targeted improvements and bridge the gap between state-of-the-art research and industry needs. Such a bottom-up approach accelerates the global pace of innovation and allows multidisciplinary challenges across a wide range of industries to be tackled more effectively. At the same time, individual contributors benefit from working within a professionally maintained and version-controlled code base, removing a significant barrier for domain experts who, despite lacking a dedicated software development background, can meaningfully participate in the code's technical evolution.

Finally, as OpenRadioss can also read and run models written with Ansys LS-DYNA⁶ syntax (essentially enabling its community to contribute and harmonize model building), the software is positioned to become the most accessible and powerful open-source software for accurate simulation of complex multi-physics dynamic events.

¹ScienceDirect: <https://www.sciencedirect.com/topics/chemistry/finite-element-modeling>

²ScienceDirect: <https://www.sciencedirect.com/topics/mathematics/finite-element-method>

³OpenRadioss: <https://openradioss.org/>

⁴Altair Radioss: <https://altair.com/radioss>

⁵Altair Radioss FEA solver now available as open-source: <https://altair.com/newsroom/news-releases>

⁶LS-DYNA is an explicit dynamics commercial software distributed by the Ansys company.

3.1.1 Historical perspective

Among the publicly available, explicit dynamics simulation tools capable of calculating the structural response of highly non-linear dynamic events, OpenRadioss stands out as one of the most accessible options due to its open-source nature. Historically [14], the simulation code traces back to the DYNA3D program that was developed by Dr. John O. Hallquist at Lawrence Livermore National Laboratory in 1976. Given the complex physics underlying explosion simulations, which represented the original intended application of DYNA3D, a full three-dimensional finite element code was developed to accurately capture non-linear dynamics through explicit time integration. As the program expanded into general structural mechanics simulations, its growing versatility quickly established DYNA3D as a dominant computational code leading to the unrestricted public release of its source code in 1978.

Following the DYNA3D source code release, in the late 1970s a French company called ESI Group⁷ started developing PAM-CRASH, a software able to perform numerous impact event simulations ranging from large scale (e.g. a military fighter plane crashing into a nuclear power plant [15]) to automotive crash scenarios [16]. As the PAM-CRASH program proved highly capable within automotive industry applications, the Radioss software was in turn developed by the Mecalog company drawing on the foundations established by PAM-CRASH. After Altair acquired Mecalog in 2006 and integrated Radioss into its HyperWorks suite, development continued within Altair's commercial ecosystem until it was announced in September 2022 that the Radioss program had been released as an open-source technology under the name OpenRadioss, retaining nearly all features of its commercial counterpart.

One notable exception is the encryption routine support, available in Radioss but absent from OpenRadioss. This feature is accessed via the `/ENCRYPT` input keyword and allows users to run simulations that integrate proprietary third-party models (such as commercial crash test dummies) without exposing their underlying geometry, material properties, or contact definitions. The `/ENCRYPT` keyword thus allows the affected model to be placed as a black box inside the numerical framework while keeping the protected internal response opaque to the user, which may be a requirement in certain industrial workflows.

Another feature exclusive to the commercial version of Radioss is the finite volume method implemented for airbag response modeling through the `/MONVOL/FVMBAG1` input keyword. As this method is physically more accurate at representing inflation of monitored volumes with respect to the uniform pressure method (which assumes homogeneous pressure distribution inside the airbag), the absence of this property card in the open-source version of the program likely reflects proprietary algorithmic implementations subject to commercial licensing constraints.

3.1.2 Software architecture and file structure of OpenRadioss cases

The OpenRadioss code itself is primarily written in **Fortran**, with some additional programming in **C++** and **C** to encode some components within the build system. The software is publicly available on GitHub under the GNU Affero General Public License v3.0⁸, and it is actively maintained and developed by Altair Engineering.

Although OpenRadioss was originally designed as a command-line-based toolbox for Windows and Linux, community members have developed a **Python**-based Graphical User Interface for job submission and execution management, bundled with installation executable packages since November 2024[17]. The scope of this tool is to simplify the user experience in launching and queuing simulations without directly invoking the solver from the terminal, though model configuration and results visualization usually rely on third-party programs. In particular, Altair HyperMesh is the best suited pre-processing software given its native support of Radioss as a solver environment, offering a dedicated interface for mesh generation, boundary condition assignment, and solver parameter definition within the starter and engine files; post-processing, on the other hand, is usually carried out in ParaView. The GUI window, shown in Figure 3.1, is launched within a system by executing the `OpenRadioss_gui.bat` file (Windows operating system) or the `OpenRadioss_gui.bash` file (Linux operating system) found inside the OpenRadioss installation repository.

⁷ESI Group: <https://www.siemens.com/en-us/ecosystem/esi-group/>

⁸GitHub - Official OpenRadioss Repository: <https://github.com/OpenRadioss/OpenRadioss.git>

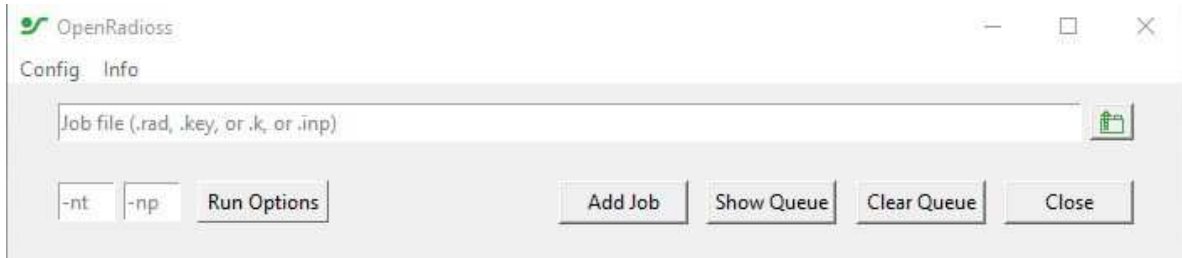


Figure 3.1: OpenRadioss: Graphical User Interface. Each test case, commonly referred to as *job*, is configured by selecting the input **Starter** file, the number of threads (-nt) and the number of MPI domains (-np), and subsequently queued through the "Add Job" button.[17]

Jobs within OpenRadioss are configured using plain-text configuration files referred to as the **Starter** and **Engine** files, which are included in each **case_directory** to specify boundary conditions, simulation parameters and solver settings. The basic directory structure of an OpenRadioss test case, shown in Figure 3.2, is described as follows.

```
<case>
├── jobFile_0000.rad
└── jobFile_0001.rad
```

Figure 3.2: OpenRadioss: Case directory structure.[17]

The OpenRadioss **Starter** file, identified by the **_0000.rad** suffix, defines the initial state of the model prior to the simulation run. It is structured around keyword cards in the **/KEYWORD** format, through which the complete model definition is specified, including geometry, mesh, material properties, part and property assignments, boundary conditions, contact interfaces, and initial conditions. The keyword cards are generally processed by the solver in the order of their identifiers, regardless of their position in the input deck.

The **Engine** file, carrying the **_0001.rad** suffix, governs the dynamic execution of the simulation and is read by the solver once the **Starter** has been successfully processed. It defines the computational and output control parameters, including time step settings, run termination criteria, and output file generation for results visualization. Like the **Starter**, the **Engine** is also structured through **/KEYWORD** format cards.

The OpenRadioss output log for both the **Starter** and **Engine** files is presented in the **Run Window**, shown in Figure 3.3; the tab opens within the OpenRadioss GUI after a job is submitted, allowing the user to monitor and control the numerical execution.

It is also worth noting that OpenRadioss supports input files written with Ansys LS-DYNA⁶ syntax to broaden community contributions and improve cross-solver model portability. LS-DYNA is an explicit dynamics commercial software developed by Dr. John O. Hallquist after his departure from the Lawrence Livermore National Laboratory; it was first released by the Livermore Software Technology Company in 1988, and its source code has never been made public even after Ansys' acquisition of LSTC in 2019. As both OpenRadioss and LS-DYNA are based on an identical base code represented by DYNA3D, the two solvers are largely interchangeable with OpenRadioss often being labeled as a free, comparably capable alternative to LS-DYNA[14].

3.1.3 Use cases of OpenRadioss and structural configuration adopted in the present work

OpenRadioss has been applied across a wide range of highly dynamic simulation scenarios demonstrating its versatility across a broad spectrum of engineering applications[14]. Besides the numerous car crash examples published extensively as benchmarks, the literature documents several impact simulations for the aviation industry (e.g. bird strike, drone strike), validation of protective structure designs against standard ISO test protocols, ballistic impacts and explosive burning investigations as well as fluid-structure interaction simulations demonstrating the feasibility of arbitrary Lagrangian-Eulerian (ALE) frameworks.

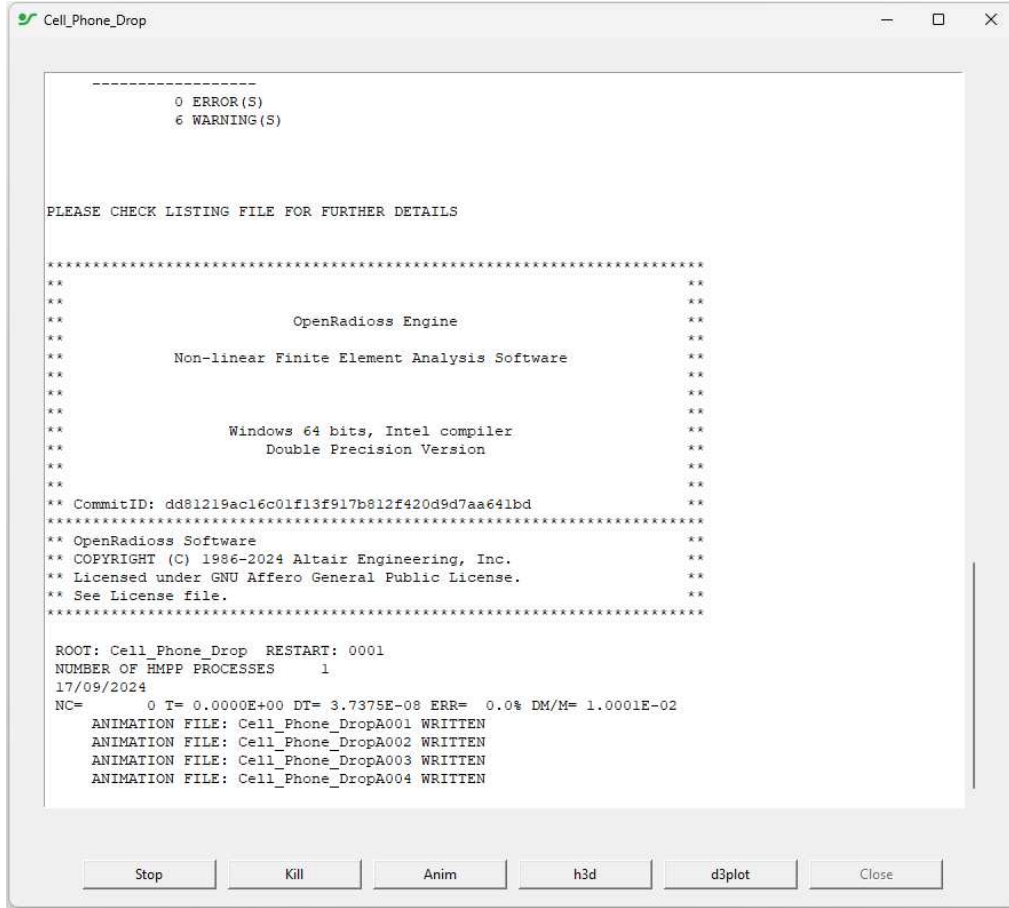


Figure 3.3: OpenRadioss: Run Window.[17]

As the fluid domain was resolved separately through OpenFOAM as discussed in Chapter 2, the present work exclusively employed OpenRadioss as the structural solver within partitioned fluid-structure interaction scenarios; the recurring structural configuration adopted throughout this research is illustrated in Figure 3.4.

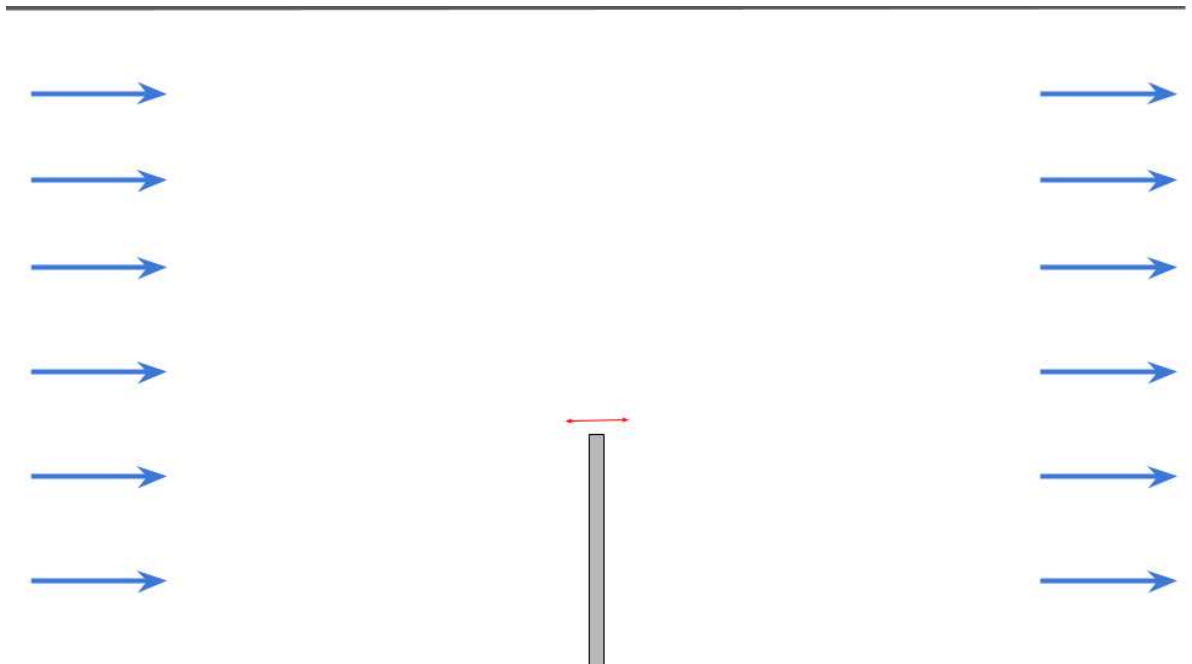


Figure 3.4: Structural configuration: perpendicular rectangular flap cantilevered at its base.[18]

While the *Pulsating Flow over Flexible Flap* and the *Blast Wave-Wall Impact* simulations present substantial differences in the fluid domain configurations, both cases share, in principle, the same structural configuration: a three-dimensional rectangular flap clamped at its base as the sole constraint imposed. No internally generated stresses are applied to the structure, as it is subject exclusively to fluid-induced loading that is transmitted through the lateral walls (serving as the physical and numerical fluid-structure interface). The material properties assigned to each case, together with the domain discretization choices made for each configuration, will be presented and discussed in the remainder of this chapter.

3.2 Material behavior and hyperelastic constitutive modeling

Selecting the appropriate material in the design of structural components is one of the most critical aspects of continuum mechanics. As this choice directly influences the structural response and governs performance in terms of integrity, load-bearing capacity, stiffness and dynamic behavior, accurate constitutive modeling represents a fundamental task in material characterization so that structural engineers are provided the most adequate tools to make informed selections based on specific project requirements.

Materials encountered in engineering applications can be broadly classified into four macro-families: metals and alloys, ceramics and glasses, polymers and elastomers, and composites, which optimize mechanical properties through constituents combination from different material families. Among these classes, the most commonly adopted for fluid-structure interactions are metals and alloys, which provide high stiffness and strength for load-bearing applications, as well as polymers and elastomers, particularly suited for applications involving large reversible deformations and flexibility.

To correctly predict material response and characterize their mechanical behavior, all material classes can be associated with a range of constitutive models which define the constitutive equations governing the structural internal equilibrium. Constitutive equations are mathematical relations correlating the stresses arising from material deformation to the corresponding strains [19], and depending on the material class under consideration they may be strain-rate-dependent (e.g. viscous fluid models) or time-independent (e.g. the small strain linear elasticity equations involving the Young's modulus E and the Poisson ratio ν).

3.2.1 Kinematics of finite deformation

Kinematics provides the geometric foundation for finite deformation mechanics, describing the motion and deformation of a body independently of the forces causing it.[19] The deformation gradient $\mathbf{F}$, defined as a function of the displacement field gradient $\mathbf{F} = \mathbf{I} + \nabla \mathbf{d}$, is the central kinematic quantity in finite deformation mechanics as it encodes the complete local deformation information of the body at any given point. To ensure that the deformation field is physically admissible, thus precluding self-penetration and volumetric collapse, the Jacobian $J = \det(\mathbf{F})$, which measures the local volumetric change, must satisfy $J > 0$.

From a mathematical point of view, the tensor $\mathbf{F}$ can be decomposed into a stretch and a rotation component yielding the unique multiplicative decomposition known as the polar decomposition:

$$\mathbf{F} = \mathbf{R}\mathbf{U} \quad (3.1)$$

where $\mathbf{R}$ represents the rigid rotation tensor ($\mathbf{R}^T \mathbf{R} = \mathbf{I}$, $\det \mathbf{R} = 1$) and $\mathbf{U}$ is a symmetric, positive-definite pure stretch tensor ($\mathbf{U} = \mathbf{U}^T > 0$).

From the polar decomposition it is also possible to derive a rotation-free strain measure called the Right Cauchy-Green deformation tensor $\mathbf{C}$, which is defined as:

$$\mathbf{C} = \mathbf{F}^T \mathbf{F} = \mathbf{U}^2 \quad (3.2)$$

and inherits the symmetry and positive-definiteness of $\mathbf{U}$.

To capture large rotations and stretches through a non-linear dependence on the displacement gradient, the Green-Lagrange strain tensor $\mathbf{E}$ is introduced from $\mathbf{C}$ as:

$$\mathbf{E} = \frac{1}{2} (\mathbf{F}^T \mathbf{F} - \mathbf{I}) = \frac{1}{2} (\nabla \mathbf{d} + \nabla \mathbf{d}^T + \nabla \mathbf{d}^T \nabla \mathbf{d}) \quad (3.3)$$

which simplifies in the linearized limit of small displacement gradients ($\|\nabla \mathbf{d}\| \ll 1$) as its higher-order term $\nabla \mathbf{d}^T \nabla \mathbf{d}$ becomes negligible. Linearly elastic materials therefore admit the reduced form of the strain tensor $\mathbf{E}$, given by the infinitesimal Cauchy strain tensor $\boldsymbol{\varepsilon}$ which is denoted as:

$$\boldsymbol{\varepsilon} = \frac{1}{2} (\nabla \mathbf{d} + \nabla \mathbf{d}^T), \quad \varepsilon_{ij} = \frac{1}{2} (d_{i,j} + d_{j,i}) \quad (3.4)$$

These kinematic quantities provide the foundation of hyperelastic constitutive formulations, which relate material deformations to internal energy through the definition of a strain energy density function $W(\mathbf{F})$. The role of these tensors as kinematic arguments of the function $W(\mathbf{F})$ is addressed in the following subsection.

3.2.2 Elastic behavior and hyperelasticity

The most widely adopted constitutive framework in structural mechanics is the classical linear elasticity model, founded on the small-strain assumption and the linear form of Hooke's law as the fundamental relation between the stress tensor $\boldsymbol{\sigma}$ and the strain tensor $\boldsymbol{\varepsilon}$ through the fourth-order elasticity tensor $\mathbb{C}$. Of the material families commonly encountered in fluid-structure interaction applications, metals and alloys are those that follow the classical linearly elastic behavior. Classical linear elasticity is a fundamental theory in continuum mechanics describing the mechanical response of solid bodies under two key assumptions: deformations remain infinitesimally small and the relationship between stresses and strains is linear. Deformation is quantified through the symmetric, second-order tensor $\boldsymbol{\varepsilon}$ which measures how a body locally stretches and distorts, in the linearized limit of small displacement gradients ($\|\nabla \mathbf{d}\| \ll 1$), with respect to its reference configuration.

The constitutive behavior, understood as the material-specific link between deformation and internal forces, is then governed by the generalized form of Hooke's law, which postulates a linear proportionality between the stress tensor $\boldsymbol{\sigma}$ and $\boldsymbol{\varepsilon}$ through the fourth-order elasticity tensor $\mathbb{C}$ as $\sigma_{ij} = \mathbb{C}_{ijkl} \varepsilon_{kl}$. For isotropic materials, which are characterized by identical mechanical properties in all directions, the full elasticity tensor reduces to two independent scalar parameters such that the constitutive law takes the explicit form:

$$\sigma_{ij} = \frac{E}{1 + \nu} \left(\varepsilon_{ij} + \frac{\nu}{1 - 2\nu} \varepsilon_{kk} \delta_{ij} \right) \quad (3.5)$$

with the Young's modulus E governing stiffness in uniaxial loading and the Poisson's ratio ν quantifying the lateral contraction accompanying axial deformation.

Though this theoretical framework proves particularly well-suited at describing the behavior of metals under typical service conditions, elastomers and rubber-like materials exhibit a fundamentally different response to applied stress. The mechanical behavior of metals within the elastic regime is reversible and path-independent, allowing the body to fully recover its original undeformed configuration upon load removal with no residual strain; beyond the yield stress σ_Y , however, metals depart from their linear elastic response and exhibit an elasto-plastic behavior. The yield stress σ_Y thus represents a critical threshold on the experimental stress-strain curve $\boldsymbol{\sigma}$ - $\boldsymbol{\varepsilon}$ (Figure 3.5), beyond which metals progressively accumulate permanent and irrecoverable deformations. Elastomers and rubber-like materials, on the other hand, can sustain deformations far exceeding the small-strain assumption while fully recovering their original configuration upon load removal. Materials capable of remaining fully elastic under any loading condition without exhibiting plastic dissipation are classified as hyperelastic.

Hyperelasticity is a non-linear elastic constitutive framework commonly used to represent the response of elastomers, rubbers, and other soft flexible materials undergoing large-strain deformations[20]. As traditional linear elastic models fail to capture the non-linear stress-strain relationship of the hyperelastic regime, characterized by a **S-shaped** curve and a path-independent response, the constitutive model of hyperelastic materials is based on the fundamental principle that stress equals the energetic cost of incremental deformation; to mathematically formulate that, a scalar potential per unit volume is introduced as the strain energy density function $W(\mathbf{F})$, also referred to as **SEDF** or stored energy function.

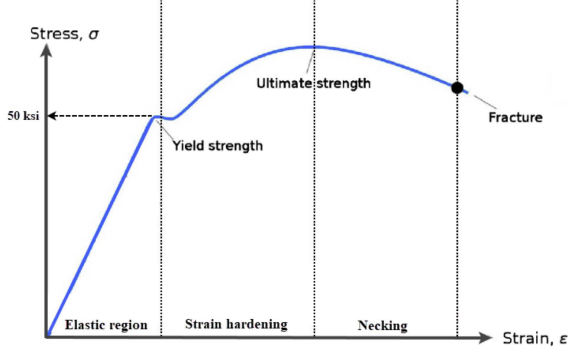


Figure 3.5: Stress-strain curve σ - ϵ for ASTM A992 Steel. [21]

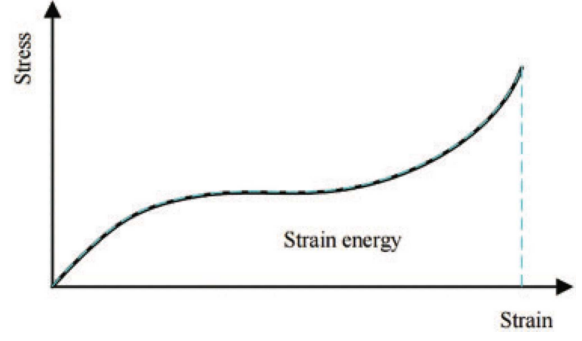


Figure 3.6: Stress-strain curve σ - ϵ for a generic hyperelastic material. [22]

By differentiating this parameter with respect to the right Cauchy–Green tensor $\mathbf{C}$, the stress tensor $\boldsymbol{\sigma}$ at any point inside the body is fully determined as follows:

$$\boldsymbol{\sigma} = \frac{2}{J} \mathbf{F} \frac{\partial W}{\partial \mathbf{C}} \mathbf{F}^T \quad (3.6)$$

guaranteeing that stresses are solely determined by the current state of deformation, that the elastic response is independent of strain history and rate, and that thermodynamic consistency is inherently ensured as no energy is created or dissipated over a closed deformation cycle. The existence of a strain energy density function $W(\mathbf{F})$ has a rigorous thermodynamic justification: when an elastic material (which exhibits no plastic dissipation) undergoes an isentropic process (hence adiabatic and reversible), its strain energy density function $W(\mathbf{F})$ assumes the physical meaning of the free energy stored reversibly per unit volume, coinciding with the Helmholtz free energy density Ψ of the material.

3.2.3 Modelling hyperelasticity

As the hyperelastic stress tensor $\boldsymbol{\sigma}$ is directly tied to the strain energy density function $W(\mathbf{F})$ as defined in Eq.(3.6), selecting an appropriate formulation for $W(\mathbf{F})$ and accurately determining the material parameters governing the constitutive response is the central challenge presented by hyperelastic modeling. Based on several experimental studies documented in the literature, a hyperelastic model is well-posed if it presents the following qualities[20]:

- it exactly reproduces the full **S-shaped** stress-strain curve;
- it remains accurate under all deformation modes (e.g. an exact model for uniaxial tension must also accurately describe simple shear or equibiaxial tension);
- it involves a small number of calibration parameters, thereby reducing the number of experimental tests needed to characterize the material.

To formulate a constitutive model that is frame-independent (i.e. independent of the choice of basis), the stored energy function for isotropic hyperelastic materials is expressed as a function of the principal invariants I_1 , I_2 , I_3 of the Right Cauchy–Green tensor $\mathbf{C}$, rather than of $\mathbf{F}$ directly. The three invariants can in turn be expressed in terms of the principal stretches λ_1 , λ_2 and λ_3 , defined as the eigenvalues of $\mathbf{U}$ (or equivalently the square roots of the eigenvalues of $\mathbf{C}$) and representing the ratio of deformed to reference length along each principal direction ($\lambda_i = \frac{l_i}{l_{0i}}$):

$$I_1 = \lambda_1^2 + \lambda_2^2 + \lambda_3^2, \quad I_2 = \lambda_1^2 \lambda_2^2 + \lambda_2^2 \lambda_3^2 + \lambda_3^2 \lambda_1^2, \quad I_3 = \lambda_1^2 \cdot \lambda_2^2 \cdot \lambda_3^2 = J^2 \quad (3.7)$$

In Eq.(3.7), I_1 is the sum of squared stretches, I_2 measures the sum of squared area changes and $J = \sqrt{I_3}$ quantifies the volumetric change, as it represents the ratio of deformed to reference volume ($J = \frac{V}{V_0} = \frac{l_1 \cdot l_2 \cdot l_3}{l_{01} \cdot l_{02} \cdot l_{03}} = \lambda_1 \cdot \lambda_2 \cdot \lambda_3$).

For materials exhibiting a bulk modulus many orders of magnitude larger than their shear modulus ($\frac{K}{\mu} \rightarrow \infty$), incompressible hyperelasticity can be introduced as a valid idealization characterized by the following kinematic constraint:

$$J = \det(\mathbf{F}) = 1 \quad \Leftrightarrow \quad I_3 = 1 \quad (3.8)$$

since incompressible materials never undergo volumetric changes.

Although by definition incompressible materials exhibit a theoretical Poisson's ratio of $\nu = 0.5$, numerically implementing this condition exactly would result in an infinite bulk modulus ($K = \mu \cdot \frac{2(1+\nu)}{3(1-2\nu)} \rightarrow \infty$) causing the speed of sound to diverge and the solid element time step to approach zero. To avoid this singularity, it is common practice in finite element codes to set $\nu = 0.495$ while modeling incompressible materials so that a proper balance between accuracy and computational efficiency is achieved.

The invariant representation discussed in Eq.(3.7) drastically reduces the space of admissible SEDFs, thereby providing the kinematic foundation upon which the earliest isotropic hyperelastic models, known as the Neo-Hookean and Mooney-Rivlin solids, were developed. Though numerous more sophisticated formulations have been proposed since and finite element simulation codes support a broad range of hyperelastic constitutive models, the discussion in this thesis is restricted to the implementation of the Neo-Hookean and Mooney-Rivlin material models in OpenRadioss as the two structural configurations adopted in the present work rely on these models exclusively.

The material card /MAT/LAW42 implemented in OpenRadioss defines the strain energy density function $W(\mathbf{F})$ as:[23]

$$W(\lambda_1, \lambda_2, \lambda_3) = \sum_{p=1}^5 \frac{\mu_p}{\alpha_p} (\bar{\lambda}_1^{\alpha_p} + \bar{\lambda}_2^{\alpha_p} + \bar{\lambda}_3^{\alpha_p} - 3) + \frac{K}{2}(J - 1)^2 \quad (3.9)$$

where $\bar{\lambda}_i = J^{-\frac{1}{3}}\lambda_i$ is the deviatoric stretch and α_p and μ_p represent the material constant pairs governing the constitutive response. While the numerical implementation of this material law supports up to five coefficient pairs to define the constitutive behavior, the number adopted in practice is governed by the complexity of the material response and the desired curve-fitting accuracy.

In addition to that, the initial shear modulus μ and bulk modulus K are given by:

$$\mu = \frac{\sum_{p=1}^5 \mu_p \cdot \alpha_p}{2}, \quad K = \mu \cdot \frac{2(1+\nu)}{3(1-2\nu)} \quad (3.10)$$

where the product of the material constant pairs must always be $\mu_p \cdot \alpha_p > 0$ for material stability (i.e. $\mu = \frac{\sum_{p=1}^5 \mu_p \cdot \alpha_p}{2} > 0$) and the Poisson's ratio ν is used exclusively to compute the bulk modulus.

Neo-Hookean model

The simplest constitutive model of hyperelastic materials is the Neo-Hookean model, first proposed in 1948 by R. Rivlin and implemented in OpenRadioss using the following strain energy density function:

$$W = C_{10}(I_1 - 3) \quad (3.11)$$

where I_1 is the first invariant of the right Cauchy-Green tensor and C_{10} is the only experimental material constant.

This representation corresponds to a specialization of the general SEDF of Eq. 3.9, derived by setting:

$$\mu_1 = 2C_{10}, \quad \alpha_1 = 2, \quad \mu_2 = \alpha_2 = 0 \quad (3.12)$$

so that the only input parameters required by the software are μ_1 and α_1 . Given its simplicity, the Neo-Hookean material model is typically only accurate for strains less than 20%. [23]

Mooney-Rivlin model

A slightly more complex constitutive model of hyperelastic materials offered by OpenRadioss is the Mooney-Rivlin model, first proposed in 1940 by M. Mooney and expressed in terms of invariants by R. Rivlin in 1948. It represents the strain energy density function as:

$$W = C_{10}(I_1 - 3) + C_{01}(I_2 - 3) \quad (3.13)$$

where I_1 and I_2 are the first and second invariants of the right Cauchy-Green tensor and C_{10} and C_{01} represent the material experimental constants.

This representation corresponds to a specialization of the general SEDF of Eq. 3.9, derived by setting:

$$\mu_1 = 2C_{10}, \quad \mu_2 = -2C_{01}, \quad \alpha_1 = 2, \quad \alpha_2 = -2 \quad (3.14)$$

so the input parameters required by the software are μ_1 , α_1 , μ_2 and α_2 , which are typically provided by the material suppliers or retrieved through experimental testing. Owing to its greater complexity, the Mooney-Rivlin material law is typically accurate for strains up to 100%.^[23]

3.3 Finite element formulations and numerical integration

Once a structural component is fully characterized by its geometry, material properties, loading conditions, and boundary constraints, the resulting model is numerically tested and optimized through a finite element analysis, which approximates the solution of the governing equations by discretizing the continuous domain into small, simply shaped elements called finite elements. Such an approach aims at approximating the displacement field within each element by using shape functions, which are mathematical relations representing how a field variable varies within an element whilst ensuring a partition of unity and compact support⁹; after each element stiffness matrix $\mathbf{k}_e$ is derived, it is then possible to assemble the global stiffness matrix $\mathbf{K}$ (which relates nodal displacements $\mathbf{d}$ in global coordinates (x, y, z) to total forces $\mathbf{F}$ acting on the structure^[24]) by summing all elemental contributions over the discretized domain. The quality of the structural solution is critically influenced by two key factors: a proper mesh density, which governs solution accuracy and computational cost, and the choice of element type, which determines interpolation accuracy and sensitivity to numerical artifacts such as locking.

As both cases discussed in the present work involve three-dimensional structural components, the following discussion is restricted to 3D finite elements and hexahedral solids in particular, presenting their integration formulations alongside a critical assessment of the advantages and limitations of each element type and numerical approach.

3.3.1 Hexahedral finite elements: HEX8 and HEX20

Three-dimensional finite elements are typically classified into three families: tetrahedral elements, hexahedral elements, and triangular prism elements, as illustrated in Figure 3.7.

Tetrahedral and hexahedral elements are obtained by adding a new dimension to TRI3 (3-noded linear triangle) and QUAD4 (4-noded quadrilateral plate) discretization cells, respectively, thus representing a direct extension of two-dimensional elements through the addition of a spatial coordinate and displacement component; triangular prism elements, on the other hand, represent a hybrid cell type that finds no counterpart in two-dimensional numerical applications. As the present work exclusively relies on hexahedral elements, a further distinction is drawn between HEX8 (tri-linear hexahedra) and HEX20 (quadratic hexahedra) discretization elements based on the number of nodes characterizing each cell.

Both element types are isoparametric, meaning that the geometry and displacement field are interpolated using the same shape functions; since physical coordinates (x, y, z) of each element vary across the mesh, these functions are defined in a reference coordinate system (ξ, η, ζ) on a fixed master element instead so that every physical cell within the discretized domain, regardless of how geometrically distorted it may be, is always mapped to the same master cube through a coordinate transformation.

Within this framework, tri-linear hexahedra, also known as the eight-node brick element, feature one node at each corner of their cuboidal structure; to preserve the dimensionless setting defined by the reference coordinate system (ξ, η, ζ) , any HEX8 element of arbitrary shape is mapped to the eight reference corner nodes located at $(\xi, \eta, \zeta) = (\pm 1, \pm 1, \pm 1)$, as shown in Figure 3.8.

⁹ScienceDirect: <https://www.sciencedirect.com/topics/engineering/shape-function>

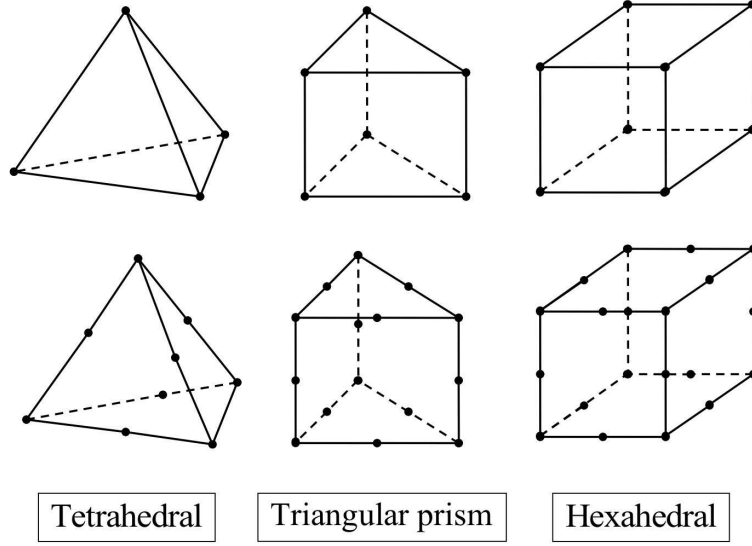


Figure 3.7: Classification of three-dimensional finite elements.

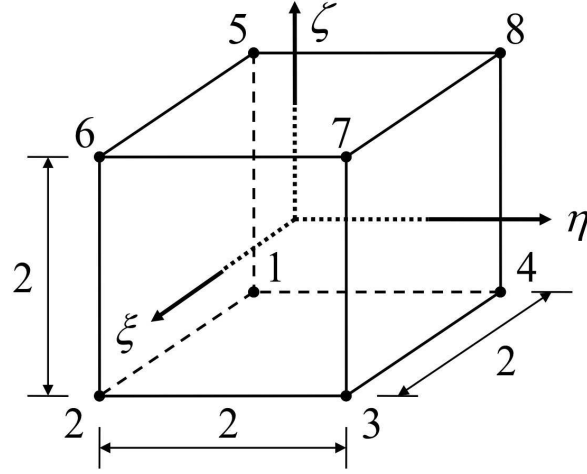


Figure 3.8: Master cube of any HEX8 element.

For an eight-node brick element with three spatial coordinates, the shape function of a generic node i located at (ξ_i, η_i, ζ_i) is given by:[24]

$$N_i(\xi, \eta, \zeta) = \frac{1}{8}(1 + \xi_i\xi)(1 + \eta_i\eta)(1 + \zeta_i\zeta) \quad (3.15)$$

where (ξ_i, η_i, ζ_i) and (x_i, y_i, z_i) represent the nodal coordinates in the reference and physical coordinate systems, respectively. As this representation is linear along each spatial direction, the resulting coordinate transformation is referred to as trilinear interpolation.

The geometry and displacement field of any eight-node brick element are thus given by:

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \sum_{i=1}^8 N_i(\xi, \eta, \zeta) \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \sum_{i=1}^8 N_i(\xi, \eta, \zeta) \mathbf{X}_i \quad (3.16)$$

and

$$\mathbf{d} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \sum_{i=1}^8 N_i(\xi, \eta, \zeta) \begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = \sum_{i=1}^8 N_i(\xi, \eta, \zeta) \mathbf{d}_i \quad (3.17)$$

The Jacobian matrix $\mathbf{J}$ of the coordinate mapping, which relates derivatives in the reference coordinate system (ξ, η, ζ) to those in the physical space (x, y, z) , is defined as:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix} = \sum_{i=1}^8 \begin{bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \\ \frac{\partial N_i}{\partial \zeta} \end{bmatrix} \begin{bmatrix} x_i & y_i & z_i \end{bmatrix} \quad (3.18)$$

where the derivatives of the shape functions N_i with respect to ξ, η , and ζ are given by:

$$\begin{aligned} \frac{\partial N_i}{\partial \xi} &= \frac{1}{8} \xi_i (1 + \eta_i \eta) (1 + \zeta_i \zeta) \\ \frac{\partial N_i}{\partial \eta} &= \frac{1}{8} \eta_i (1 + \xi_i \xi) (1 + \zeta_i \zeta) \\ \frac{\partial N_i}{\partial \zeta} &= \frac{1}{8} \zeta_i (1 + \xi_i \xi) (1 + \eta_i \eta) \end{aligned} \quad (3.19)$$

The strain-displacement relationship is therefore expressed as:

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{xz} \end{bmatrix} = \begin{bmatrix} \partial u / \partial x \\ \partial v / \partial y \\ \partial w / \partial z \\ \partial u / \partial y + \partial v / \partial x \\ \partial v / \partial z + \partial w / \partial y \\ \partial u / \partial z + \partial w / \partial x \end{bmatrix} = \sum_{i=1}^8 \begin{bmatrix} \partial N_i / \partial x & 0 & 0 \\ 0 & \partial N_i / \partial y & 0 \\ 0 & 0 & \partial N_i / \partial z \\ \partial N_i / \partial y & \partial N_i / \partial x & 0 \\ 0 & \partial N_i / \partial z & \partial N_i / \partial y \\ \partial N_i / \partial z & 0 & \partial N_i / \partial x \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = \sum_{i=1}^8 \mathbf{B}_i \mathbf{d}_i \quad (3.20)$$

where $\mathbf{d}_i$ is the nodal displacement vector for each element (of size $n_{\text{dof}} \times 1$) whilst $\mathbf{B}$ represents the strain-displacement matrix (of size $6 \times n_{\text{dof}}$), which varies spatially within the element as a function of (ξ, η, ζ) .

To compute the derivatives of the shape functions $N_i = f(\xi, \eta, \zeta)$ with respect to the physical coordinates (x, y, z) and assemble the strain-displacement matrix $\mathbf{B}$, the following derivative transformation is applied:

$$\begin{bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \\ \frac{\partial N_i}{\partial \zeta} \end{bmatrix} \quad (3.21)$$

Once the matrix $\mathbf{B}$ is fully established, the stiffness matrix $\mathbf{K}$ of the discretized domain is expressed through the principle of virtual work as:

$$\mathbf{K} = \int_V \mathbf{B}^T \mathbf{C} \mathbf{B} dV = \int_{-1}^1 \int_{-1}^1 \int_{-1}^1 \mathbf{B}^T(\xi, \eta, \zeta) \mathbf{C} \mathbf{B}(\xi, \eta, \zeta) \det(\mathbf{J}) d\xi d\eta d\zeta \quad (3.22)$$

where $\mathbf{C}$ represents the structural elasticity tensor, which remains constant throughout each element for a homogeneous material, and the volume element transformation is applied as $dV = \det(\mathbf{J}) d\xi d\eta d\zeta$.

As both $\mathbf{B}$ and $\det(\mathbf{J})$ are polynomial functions of (ξ, η, ζ) , this integral cannot be evaluated analytically for a general HEX8 element but is instead approximated numerically through a weighted sum computed at specific integration points (ξ_i, η_i, ζ_i) known as Gauss points:

$$\mathbf{K} \approx \sum_{i=1}^{n_{\text{gp}}} w_i \mathbf{B}^T(\xi_i, \eta_i, \zeta_i) \mathbf{C} \mathbf{B}(\xi_i, \eta_i, \zeta_i) \det(\mathbf{J}_i) \quad (3.23)$$

where n_{gp} is the number of integration points and w_i represents the Gauss weights.

As an extension of the HEX8 formulation, HEX20 discretization elements introduce twelve additional midside nodes[24], each placed at the midpoint of one edge of the master cube, alongside the eight existing corner nodes illustrated in Figure 3.8. The resulting twenty-node brick element configuration is shown in Figure 3.9.

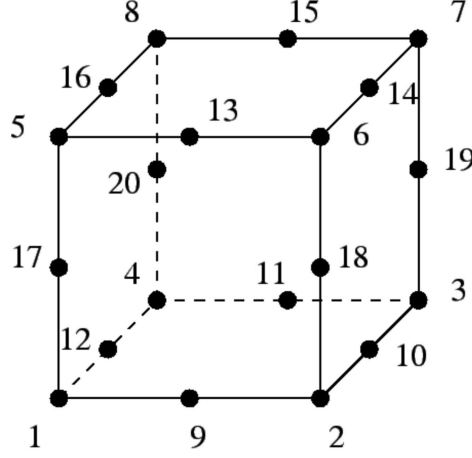


Figure 3.9: Master cube of any HEX20 element.

Within the previously introduced isoparametric framework, any HEX20 element of arbitrary shape maps its twenty nodes to two geometrically distinct families: eight corner nodes, located at $(\xi, \eta, \zeta) = (\pm 1, \pm 1, \pm 1)$, and twelve midside nodes, where exactly one of the three coordinates (ξ, η, ζ) is always set to 0 and the remaining two are ± 1 .

The shape function of each node i placed at (ξ_i, η_i, ζ_i) therefore depends on its nodal family, leading to the following expressions:

- 8 corner nodes:

$$N_i(\xi, \eta, \zeta) = \frac{1}{8}(1 + \xi_i\xi)(1 + \eta_i\eta)(1 + \zeta_i\zeta)(\xi_i\xi + \eta_i\eta + \zeta_i\zeta - 2) \quad (3.24)$$

extending the HEX8 corner node shape function defined in Eq. 3.15 by the additional factor $(\xi_i\xi + \eta_i\eta + \zeta_i\zeta - 2)$ to ensure it vanishes at all midside node locations.

- 12 midside nodes:

- 4 nodes located on a ξ -direction edge $(\xi_i, \eta_i, \zeta_i) = (0, \pm 1, \pm 1)$:

$$N_i(\xi, \eta, \zeta) = \frac{1}{4}(1 - \xi^2)(1 + \eta_i\eta)(1 + \zeta_i\zeta) \quad (3.25)$$

- 4 nodes located on a η -direction edge $(\xi_i, \eta_i, \zeta_i) = (\pm 1, 0, \pm 1)$:

$$N_i(\xi, \eta, \zeta) = \frac{1}{4}(1 + \xi_i\xi)(1 - \eta^2)(1 + \zeta_i\zeta) \quad (3.26)$$

- 4 nodes located on a ζ -direction edge $(\xi_i, \eta_i, \zeta_i) = (\pm 1, \pm 1, 0)$:

$$N_i(\xi, \eta, \zeta) = \frac{1}{4}(1 + \xi_i\xi)(1 + \eta_i\eta)(1 - \zeta^2) \quad (3.27)$$

where the squared term $(1 - \square^2)$ always corresponds to the coordinate direction along the edge of interest, following a consistent cyclic permutation pattern across the three spatial directions.

The key distinction between HEX8 and HEX20 elements lies in the order of their shape functions. As twenty-node brick elements incorporate quadratic terms such as ξ^2, η^2, ζ^2 , the displacement field varies quadratically within each cell rather than linearly, thus enabling the accurate representation of bending and curved geometries while HEX8 elements are limited to linear displacement fields. This interpolation of degree 2 along each coordinate direction is known as serendipity interpolation.

Following the same isoparametric mapping adopted for **HEX8** elements (Eq. 3.16 and Eq. 3.17), geometry and displacement within each **HEX20** cell are interpolated using twenty shape functions defined above, with the summation now extending over all twenty nodes as $\mathbf{x} = \sum_{i=1}^{20} N_i(\xi, \eta, \zeta) \mathbf{x}_i$ and $\mathbf{d} = \sum_{i=1}^{20} N_i(\xi, \eta, \zeta) \mathbf{d}_i$. The Jacobian matrix $\mathbf{J}$ and the strain-displacement matrix $\mathbf{B}$ also remain conceptually identical to those presented in Eq. 3.18 and Eq. 3.20, following the same assembly procedure introduced for **HEX8** elements and only differing in size and content due to the increased number of nodes.

Beyond the geometric mapping, the linear derivatives in (ξ, η, ζ) yielded by quadratic shape functions N_i support a linear variation of the strain field within each **HEX20** element, whereas the linear shape functions of **HEX8** cells yield constant derivatives resulting in a piecewise constant strain field. Owing to the linear strain field, **HEX20** elements can capture bending stress gradients within a single element while **HEX8** elements can only approximate this behavior through significant mesh refinement.

As the definition of the stiffness matrix $\mathbf{K}$ is a direct application of the principle of virtual work, which is independent of element type choice, its integral form remains identical to Eq. 3.22, with the only difference between the two formulations being the polynomial degree of the integrand $\mathbf{B}^T \mathbf{C} \mathbf{B}$ in each direction, increasing from 2 for **HEX8** elements to 4 for **HEX20** cells. As the integral cannot be solved analytically, it is approximated numerically instead through the weighted sum of Eq. 3.23.

3.3.2 Gauss quadrature and locking phenomena

The weighted sum shown in Eq. 3.23 is an application of the Gauss quadrature rule, which approximates the stiffness matrix by evaluating the integrand at specific sampling points (also known as Gauss points) assigning them weights in order to minimize the integration error; the accuracy of the numerical integration directly depends on the number of Gauss points selected, though at the cost of increased computational expense. A one-dimensional Gauss rule states that n sampling points integrate polynomials exactly up to a degree of $2n - 1$, where points ξ_i and weights w_i are determined analytically and tabulated in the literature to minimize the number of evaluations required. The extension of this quadrature rule to the three-dimensional master cube is achieved through a tensor product generating a $p \times q \times r$ scheme, which applies a p -point rule in ξ , a q -point rule in η , and an r -point rule in ζ , yielding a total of $p \cdot q \cdot r$ Gauss points with three-dimensional weights w_i, w_j, w_k . The resulting Gauss points are distributed symmetrically within the integration domain $[-1, 1]^3$, with symmetric points sharing equal weights.

For **HEX8** elements, full integration uses a $2 \times 2 \times 2$ scheme (for a total of 8 Gauss points) to integrate exactly the polynomial content of $\mathbf{B}^T \mathbf{C} \mathbf{B}$ for regular undistorted elements[24], which is of degree 2 in each direction due to the linear polynomial entries of the strain-displacement matrix $\mathbf{B}$. The sampling points are located at all combinations of $(\xi, \eta, \zeta) = \left(\pm \frac{1}{\sqrt{3}}, \pm \frac{1}{\sqrt{3}}, \pm \frac{1}{\sqrt{3}}\right)$, and all weights are set to $w_i = 1$.

In distorted elements, however, the non-constant Jacobian entries raise the polynomial degree of the integrand beyond what the $2 \times 2 \times 2$ scheme can integrate exactly, introducing an integration error. While this effect is typically negligible for mildly distorted meshes, a numerical phenomenon known as locking arises when the polynomial space of the element fails to represent the deformation field correctly, thus resulting in an artificially over-stiff structural response which restricts the deformation of elements under the applied loading. This phenomenon manifests in two distinct forms: volumetric locking and shear locking.

- Volumetric locking occurs in structures made of nearly incompressible materials when the shape functions used in finite element analysis cannot properly approximate the incompressibility condition throughout each element, which constrains the displacement field to satisfy near-zero volumetric strains ($\nabla \cdot \mathbf{d} \approx 0$) at each Gauss point. Mathematically, fully integrated **HEX8** elements exhibit an artificially over-stiff response because the 8 incompressibility constraints over-constrain the $n_{\text{dof}} = 24$ degrees of freedom of each cell.
- Shear locking arises when trilinear **HEX8** shape functions are forced to approximate out-of-plane bending deformations; since the linear polynomial space of **HEX8** elements cannot exactly represent the quadratic displacement field of pure bending solutions, the incorrect representation of shear strains generates spurious strains γ_{xy} and γ_{xz} that render the element artificially over-stiff.

To guarantee numerical convergence without resorting to significant mesh refinement, a reduced integration scheme addresses locking by relying on a single Gauss point, located at the centroid $(\xi, \eta, \zeta) = (0, 0, 0)$ of the master cube with corresponding weight $w = 8$, thus imposing a single volumetric strain constraint on the discretization element; furthermore, the central sampling point relieves shear locking by avoiding the face regions where spurious shear strains arise. This solution, however, introduces hourglass instabilities that require hourglass control¹⁰ in order to suppress the resulting spurious deformation modes; for this reason, the approach was not adopted in the present work.

To avoid shear locking and hourglass instabilities altogether, **HEX20** elements were adopted throughout this thesis as their quadratic shape functions represent bending displacement fields exactly. Since the integrand function $\mathbf{B}^T \mathbf{C} \mathbf{B}$ is of degree 4 in each direction due to the quadratic polynomial entries of the strain-displacement matrix $\mathbf{B}$, full integration for **HEX20** elements requires a $3 \times 3 \times 3$ Gauss quadrature scheme[24], with points located at all combinations of $\xi, \eta, \zeta \in \left\{ -\sqrt{\frac{3}{5}}, 0, +\sqrt{\frac{3}{5}} \right\}$ with corresponding weights $\left\{ \frac{5}{9}, \frac{8}{9}, \frac{5}{9} \right\}$. While second-order cells eliminate the shear locking exhibited by fully integrated **HEX8** elements, this is achieved at the cost of significantly higher computational expense as the number of integration points increases from eight to twenty-seven and the element stiffness matrix $\mathbf{k}_e$ grows from 24×24 to 60×60 . Since the volumetric over-constraint is independent of polynomial order but depends solely on its ratio to degrees of freedom, fully integrated **HEX20** elements remain susceptible to volumetric locking. To address this shortcoming, OpenRadioss implements a variable integration scheme for **HEX20** elements in which the number of Gauss points is adapted based on the local deformation state of each element, thereby reducing the number of volumetric constraints while preserving the ability to represent bending deformation modes exactly.

¹⁰ScienceDirect: <https://www.sciencedirect.com/topics/engineering/hourglass-control>

Chapter 4

preCICE: an open-source library for partitioned multi-physics simulations

4.1 Fundamentals of numerical coupling

preCICE (Precise Code Interaction Coupling Environment)¹ is a free, open-source coupling library developed from the early 2010s at the University of Stuttgart and the Technical University of Munich. Through black-box coupling of separate numerical solvers, the software enables partitioned, multi-physics simulations by allowing coupled simulation codes (i.e. *participants*) to exchange physical quantities at their shared boundaries (i.e. *interfaces*).^[25] The preCICE library is written in C++, and it is publicly available on GitHub under the GNU Lesser General Public License v3.0², which guarantees the open-source nature of the library without imposing licensing restrictions when linked against proprietary solver codes; it is actively maintained and developed by its founding universities and sustained by direct contributions of an active, research-driven community.

Importantly, preCICE has no notion of physics; to achieve simultaneous, cross-domain interactions, its core components only handle the technical and numerical aspects of coupling, as illustrated in Figure 4.1.

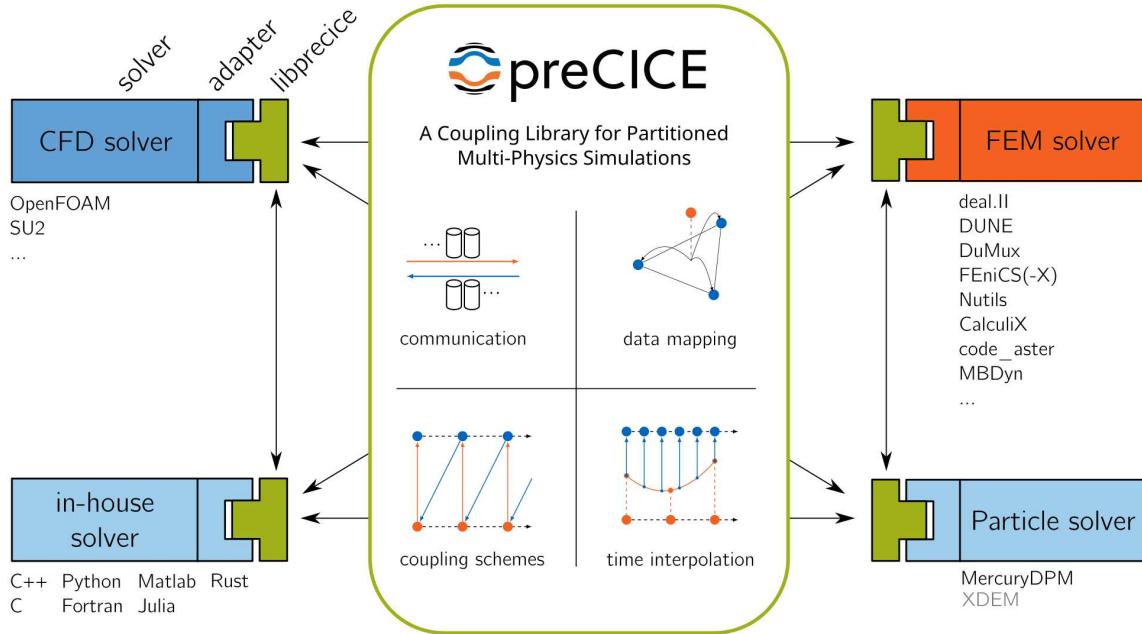


Figure 4.1: Basic functional components of preCICE.^[25]

The main implementation aspects of the coupling protocol for data interpolation and synchronization can be outlined as follows:^[25]

- All coupling numerics and communication are implemented in a peer-to-peer library approach. This means that, while coupled participants are separate executables running as independent processes with their own MPI communicators, preCICE is not a standalone server process but rather a coupling middleware (i.e. a set of functions that each solver calls from within its own execution).
- The library is used via a high-level application programming interface (API) in a minimally invasive way (from the coupled solvers' perspective). As a result, minimal restructuring is needed

¹preCICE: <https://precice.org/>

²GitHub – Official preCICE Repository: <https://github.com/precice/precice>

on the participant source codes (i.e. the solvers' internal physics is left untouched) and instead preCICE defines the common coupling protocol through a few standardized function calls (e.g. initialize, read data, write data, advance) at well-defined points in the solver's execution.

- Communication is established in an asynchronous, fully parallelized way, targeting only the specific ranks of each participant that require coupling data. By keeping the cost of data exchange proportional to the size of the coupling interface rather than the full simulation, communication scales with the overall computation preventing it from becoming a bottleneck.

To establish numerical coupling between participants, each simulation code requires a purpose-built adapter (i.e. a glue code) that mediates data exchange between the solver and the preCICE library, as illustrated by Figure 4.2. Depending on the participant, an adapter can be:

- a module or a class of the participant's code: this solution enables calls to preCICE from within the solver by directly embedding the adapter into the source code;
- a standalone executable: this solution is generally adopted when the participant's source code is inaccessible;
- a sophisticated script that orchestrates the solver-to-preCICE coordination externally: this solution undermines the library approach, that is designed to operate with calls from within a running solver instead.

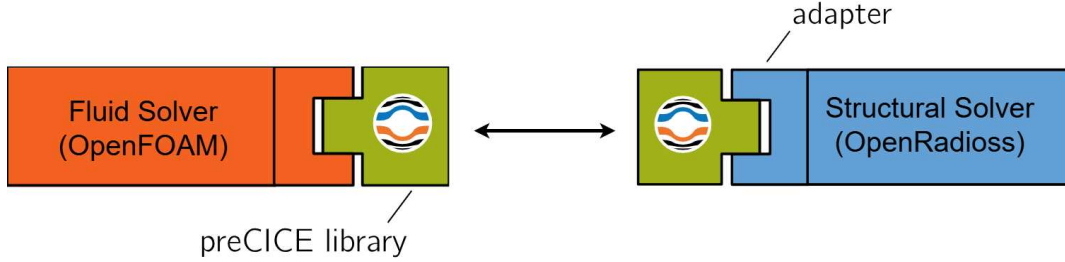


Figure 4.2: Adapters are the interface layer between simulation codes and the preCICE library.

The adapter contains all API calls to the preCICE library, thus guaranteeing access to the exchanged data and all control operations (e.g. variable reading and writing on specified boundaries, time step control for simulation synchronization) while remaining abstracted from point-to-point communication, data mapping and coupling algorithms. More specifically, an adapter bears two key responsibilities: providing nodal coupling values to preCICE by extracting them from each solver's internal fields, and translating those sent from preCICE into boundary conditions or forcing terms on the receiving solver.

Data transfer between coupled solvers takes place at every *coupling time window*, which represents a synchronization checkpoint between participants. The coupling time window t^n must not be confused with the solver time step τ^k , which instead represents the discrete temporal increment used to advance the solver's internal state. Since the two do not need to coincide within a partitioned simulation (the coupling time window can exceed both solvers' time steps), preCICE handles any resulting temporal mismatches between t^n and τ^k through temporal interpolation, as illustrated in Figure 4.3.

When a mismatch between t^n and τ^k occurs, the simulation codes must advance through multiple time steps before reaching the coupling checkpoint, undergoing a process known as sub-cycling. Since preCICE samples only the instantaneous state of the exchanged data at the end of each coupling time window, sub-cycling applied to fluid-structure interactions can generate two classes of problem. First, numerical instabilities may result from energy inconsistencies introduced by sampling rate and conserved quantities (e.g., forces, momentum) instantaneously rather than integrating them over the full coupling window. Second, when the structural adapter repeatedly reads the total structural displacement over multiple fluid sub-steps, such displacement amplification imposed on the fluid mesh may cause premature solver crashes due to geometry corruption. Since temporal mismatches can additionally drive solver desynchronization, compounding the risk of premature crashes, all simulations presented in this work therefore enforce a uniform time stepping strategy in which τ^k is set equal to t^n across all participating solvers.

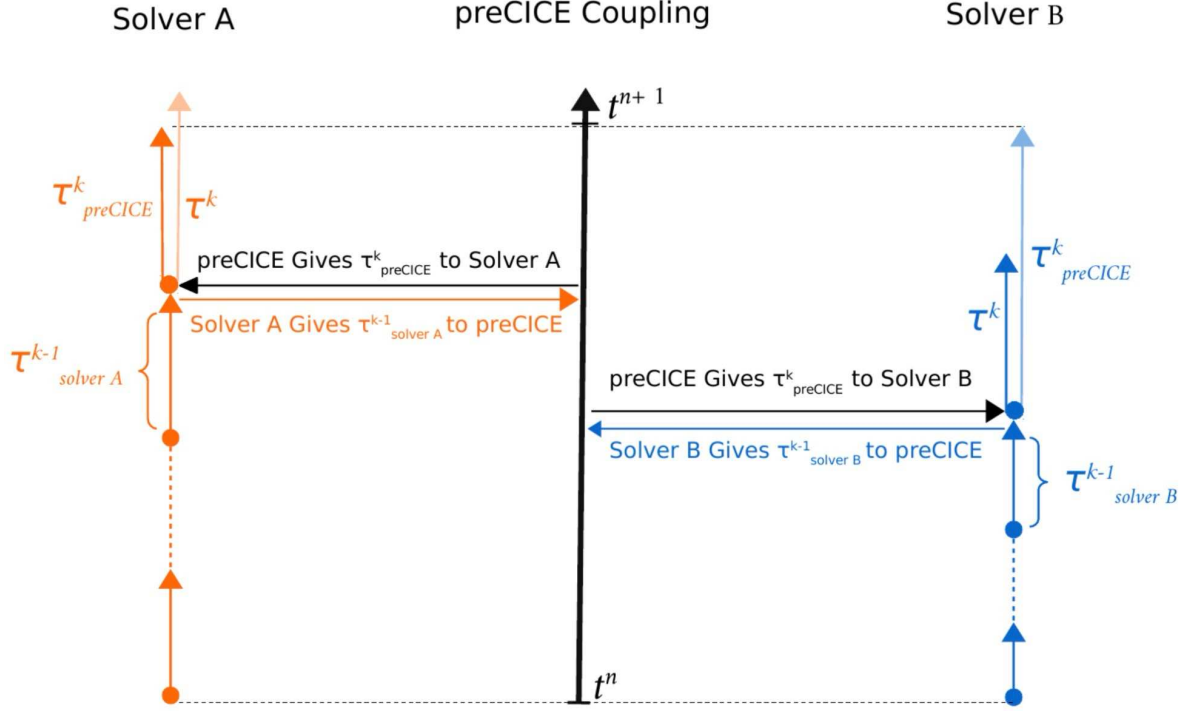


Figure 4.3: Sub-cycling mechanisms within a preCICE coupling time window.[26]

Beyond the three-dimensional coupling between a single pair of solvers (OpenFOAM and OpenRadioss) discussed in the present work, preCICE natively supports a broad range of coupling configurations (e.g. surface/volumetric, explicit/implicit, serial/parallel) across arbitrarily many participating solvers. Several mapping methods are provided to accommodate data exchange between participants with non-matching or non-conforming meshes, and the available coupling schemes orchestrate the coupling flow (i.e. data routing between participants, synchronization of time step execution). Given their central role in the configuration of the simulations presented in the following chapters, coupling schemes and data mapping are each discussed in detail in the subsections below.

4.1.1 Equation coupling strategies

Coupling schemes represent the core of preCICE’s architecture, since they govern data routing between participants and synchronization procedures. preCICE supports the following coupling options, all independently configurable at runtime:[25]

- uni-directional / bi-directional;
- explicit / implicit;
- parallel / serial.

Since fluid-structure interactions inherently involve data dependencies in both directions (as opposed to one-way coupled systems), all applications considered in the present work have been configured with bi-directional coupling.

To guarantee physical accuracy and numerical stability of the coupled fields, the values computed on both sides of the interface must be mutually consistent. For a canonical fluid-structure interaction scenario in which two solvers S_1 and S_2 represent a fluid and a structural participant, respectively, physical variables are mapped between vector spaces X_1 and X_2 . More specifically, S_1 computes forces acting on the deformable interface, and S_2 uses these forces to compute the resulting displacement and velocity of the shared boundary.[27] These mappings can be expressed as:

$$S_1: X_1 \rightarrow X_2 \quad \text{and} \quad S_2: X_2 \rightarrow X_1 \quad (4.1)$$

Given that the output of each solver serves as input for the other, the resulting fixed-point problem can be addressed through two distinct coupling approaches, namely explicit and implicit coupling. Explicit

schemes execute each participant once per time step without verifying the quality of the exchanged data, while implicit formulations perform multiple iterations within a single time step until the coupled values satisfy a convergence criterion.

Furthermore, coupling schemes can be classified according to the sequencing of participant execution, as illustrated in Figure 4.4: serial coupling entails staggered solver execution, while parallel coupling prescribes simultaneous execution. The inter-field coupling parallelism established by preCICE should not be confused with parallelism internal to solver execution; in this regard, all solvers employed in the present work were configured to run a sequential simulation internally.

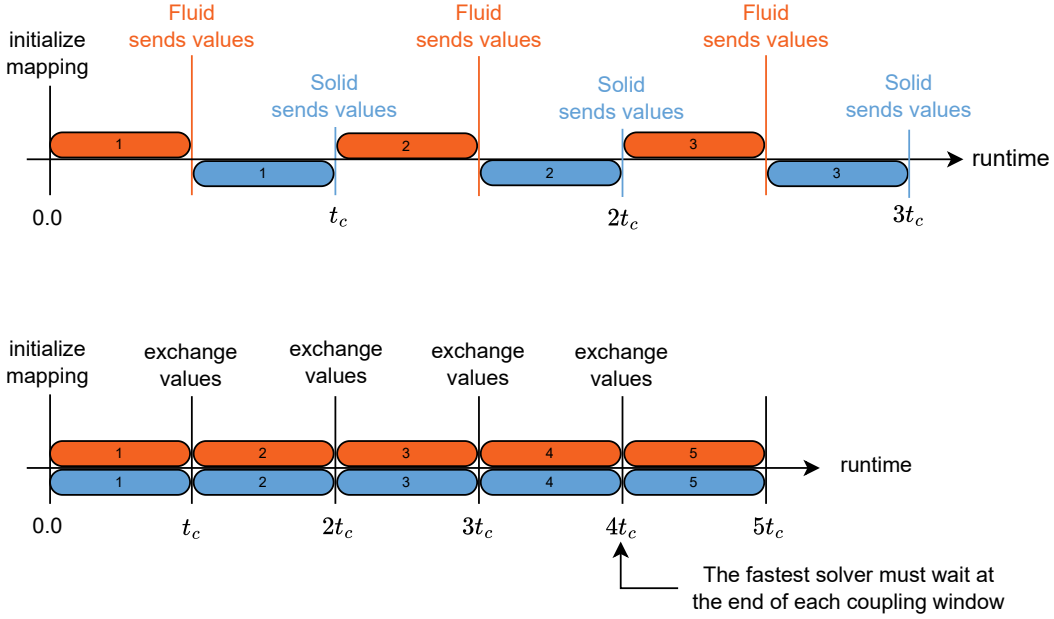


Figure 4.4: Participant execution sequencing in a preCICE coupled simulation: serial (staggered) and parallel (simultaneous) coupling schemes.

In terms of how output values are updated at each iteration, serial coupling always feeds the most recent state variables to the receiving solver, while parallel coupling supplies both solvers with output values from the previous iteration. As a result, serial coupling scenarios require solvers to alternately wait for each other, while parallel coupling eliminates such imposition.

Turning to the mathematical formulation of explicit coupling, the leading solver S_1 in a serial execution computes output values $x_2^{(n+1)}$ from boundary values $x_1^{(n)}$ evaluated at the (preceding) n^{th} time step:[27]

$$x_2^{(n+1)} = S_1^{(n)}(x_1^{(n)}) \quad (4.2)$$

Solver S_2 , in turn, awaits the output of S_1 ; upon receiving the updated $x_2^{(n+1)}$, it computes the corresponding $x_1^{(n+1)}$ from boundary values evaluated at the (current) $n^{th} + 1$ time step:

$$x_1^{(n+1)} = S_2^{(n)}(x_2^{(n+1)}) \quad (4.3)$$

To minimize cross-solver waiting time and improve computational efficiency, parallel coupling is typically preferred as it enables solvers to exchange data only upon mutual completion. Mathematically, both solvers operate with input values from the same time step n as neither waits for the other's updated result; the key distinction from the serial case lies in the argument of S_2 which now operates on $x_2^{(n)}$ rather than the updated $x_2^{(n+1)}$.

$$x_2^{(n+1)} = S_1^{(n)}(x_1^{(n)}) \quad (4.4)$$

$$x_1^{(n+1)} = S_2^{(n)}(x_2^{(n)}) \quad (4.5)$$

The two explicit coupling schemes are depicted in Figures 4.5 and 4.6; the illustrated procedure is repeated at every time step until the simulation runtime is exhausted.

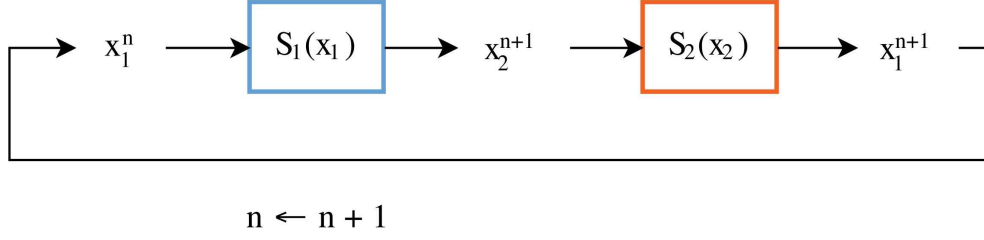


Figure 4.5: Serial explicit coupling scheme.[27]

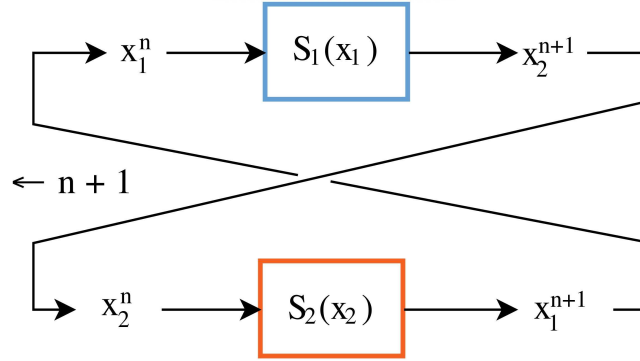


Figure 4.6: Parallel explicit coupling scheme.[27]

Explicit coupling schemes are prone to numerical instability, as no convergence check is enforced on the data exchanged at the shared interface. To overcome this limitation, implicit coupling schemes perform multiple coupling iterations within a single time window, advancing to the next only once the coupled values satisfy a convergence criterion. To support implicit coupling, preCICE requires solvers to revert to a previous physical state upon convergence failure within a given time window; to this end, each solver's complete internal state (e.g. all field and mesh information) is stored in checkpoints, which solvers write and read at every time step to enable full state restoration.[25] In the event of insufficient convergence, preCICE signals the adapter and instructs it to reload the solver state from the latest checkpoint, which corresponds to the last successfully completed coupling window, to repeat the coupling iteration with updated boundary conditions. Given that the implicit coupling loop is managed by the adapter, implicit coupling can only be configured if checkpoint operations are natively supported at the participant source code level; as OpenRadioss lacks native checkpoint support, implicit coupling could not be configured throughout this work and its mathematical formulation is therefore omitted from the present discussion.

4.1.2 Data mapping at the coupling interface

In a partitioned simulation, participants exchange data across coupling meshes that may be non-matching owing to differences in the spatial discretization of the shared interface. As such discrepancies may introduce interpolation errors that could compromise mass and energy conservation, accurate transfer of physical variables is ensured through data mappings.[25]

To formalize, consider a data mapping from the coupling mesh of participant S_1 to that of participant S_2 . At initialization, the only information required is the vertex coordinates of both meshes, defined as:

$$\mathcal{M}_{S_1} = \{\vec{x}_1^{S_1}, \dots, \vec{x}_{n_1}^{S_1}\} \quad \text{with} \quad \vec{x}_i^{S_1} \in \mathbb{R}^d, \quad \mathcal{M}_{S_2} = \{\vec{x}_1^{S_2}, \dots, \vec{x}_{n_2}^{S_2}\} \quad \text{with} \quad \vec{x}_i^{S_2} \in \mathbb{R}^d. \quad (4.6)$$

The purpose of data mapping is to compute $\vec{v}^{S_2} = (v_1^{S_2}, v_2^{S_2}, \dots, v_{n_2}^{S_2})^T$ from $\vec{v}^{S_1} = (v_1^{S_1}, v_2^{S_1}, \dots, v_{n_1}^{S_1})^T$, transferring field values from the vertices of $\mathcal{M}_{S_1}$ to those of $\mathcal{M}_{S_2}$.

All methods available in preCICE are provided in both a consistent and a conservative variant, as illustrated in Figure 4.7. Consistent mapping reproduces field values of $\mathcal{M}_{S_1}$ at the corresponding vertices of $\mathcal{M}_{S_1}$, while conservative mapping preserves the integral of the mapped quantity across the interface; the former is therefore suitable for physical variables such as displacements, velocities, pressure, and stress, while the latter is appropriate for cumulative variables (e.g. forces) where global conservation must be maintained.

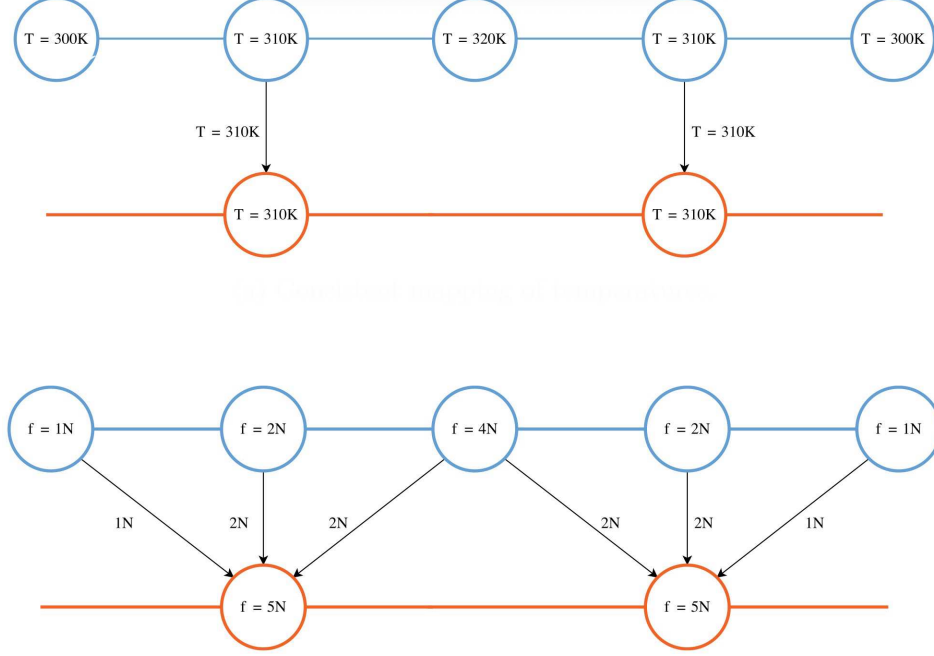


Figure 4.7: Schematic comparison of consistent and conservative data mapping schemes at the coupling interface.[27]

The mapping operation is formalized as a linear transformation:

$$\mathbf{M} \vec{v}^{S_1} = \vec{v}^{S_2} \quad (4.7)$$

with the mapping matrix $\mathbf{M} \in \mathbb{R}^{n_2 \times n_1}$. In a consistent mapping, each row of the matrix $\mathbf{M}$ must sum to one, whereas in a conservative method the entries of each column of M must sum to unity (i.e. the conservation requirement imposes $\sum_i v_i^{S_1} = \sum_i v_i^{S_2}$). It follows that conservative mapping methods are obtained by transposing the corresponding consistent mapping matrix.

Beyond the consistent-conservative distinction, data mapping in preCICE can be further classified into two categories according to their spatial interpolation strategy: projection-based methods and radial basis function (RBF) methods. Projection-based mappings rely on spatial proximity searches across the mesh entities of both participants, and are further distinguished into nearest-neighbor and nearest-projection methods.

Nearest-neighbor mapping associates each vertex $\vec{x}_i^{S_2}$ of the output mesh $\mathcal{M}_{S_2}$ with the spatially nearest vertex $\vec{x}_{j(i)}^{S_1}$ on the input mesh $\mathcal{M}_{S_1}$, defined as:

$$v_i^{S_2} = v_{j(i)}^{S_1} \quad (4.8)$$

Nearest-projection mapping, by contrast, exploits connectivity information across multiple vertices on the input mesh $\mathcal{M}_{S_1}$ to interpolate field values to the vertices of $\mathcal{M}_{S_2}$. More specifically, each output vertex $\vec{x}_j^{S_2}$ is mapped by projecting onto the entities of $\mathcal{M}_{S_1}$, interpolating the field value at the projection point $p(\vec{x}_j^{S_2})$, and assigning the result to $\vec{x}_j^{S_2}$. Since this method was not employed in the present work, its mathematical formulation is omitted.

In contrast to projection-based methods, RBF mappings operate on scattered data to construct global interpolation functions, generally offering greater accuracy at the expense of increased computational effort; these functions are expressed as linear combinations of radially symmetric basis functions centered at the vertices $\vec{x}_i^{S_1}$ of $\mathcal{M}_{S_1}$ and evaluated at the vertices $\vec{x}_j^{S_2}$ of the output mesh $\mathcal{M}_{S_2}$.

To ensure exact interpolation of constant and linear functions, a first-order polynomial term $q(\vec{x})$ is added to the interpolant $s : \mathbb{R}^d \rightarrow \mathbb{R}$ as:

$$s(\vec{x}) = \sum_{i=1}^{n_1} \lambda_i \cdot \phi \left(\left\| \vec{x} - \vec{x}_i^{S_1} \right\|_2 \right) + q(\vec{x}), \quad (4.9)$$

where ϕ denotes the radial basis function, the polynomial term takes the form $q(\vec{x}) = \beta_0 + \beta_1 x_1 + \dots + \beta_d x_d$, and the interpolation weights $\lambda_i \in \mathbb{R}$, $i = 1, \dots, n_{S_1}$ are determined by enforcing the following interpolation conditions:

$$s(\vec{x}_i^{S_1}) = v_i^{S_1} \quad \forall i = 1, \dots, n_1 \quad (4.10)$$

Since the polynomial term renders the system underdetermined, the following conditions are imposed to enforce closure and ensure a unique solution:

$$\sum_{i=1}^{n_1} \lambda_i \vec{x}_i^{S_1} = \vec{0} \quad \text{and} \quad \sum_{i=1}^{n_1} \lambda_i = 0. \quad (4.11)$$

In matrix notation, this yields a linear system of dimensions $n_{S_1} \times n_{S_1}$ solved within preCICE to determine the interpolation weights λ_i .

$$\begin{pmatrix} C & Q \\ Q^T & \mathbf{0} \end{pmatrix} \begin{pmatrix} \vec{\lambda} \\ \vec{\beta} \end{pmatrix} = \begin{pmatrix} \vec{v}^{S_1} \\ \vec{0} \end{pmatrix}, \quad (4.12)$$

where $\vec{\lambda} = (\lambda_1, \lambda_2, \dots, \lambda_{n_1})^T$, $\vec{\beta} = (\beta_0, \beta_1, \dots, \beta_d)^T$, $C = \left(\phi \left(\left\| \vec{x}_i^{S_1} - \vec{x}_j^{S_1} \right\|_2 \right) \right)_{i,j=1,\dots,n_1}$ and $Q = \left(1, x_{1,i}^{S_1}, \dots, x_{d,i}^{S_1} \right)_{i=1,\dots,n_1}$. The system is sparse or dense depending on the choice of basis function.

For the sampling, the mapping reads:

$$\vec{v}^{S_2} = (\tilde{C} \quad \tilde{Q}) \begin{pmatrix} C & Q \\ Q^T & \mathbf{0} \end{pmatrix}^{-1} \begin{pmatrix} \vec{v}^{S_1} \\ \vec{0} \end{pmatrix}, \quad (4.13)$$

so that the interpolant is evaluated at the vertices of $\mathcal{M}_{S_2}$ via a matrix-vector product of dimensions $n_{S_2} \times n_{S_1}$.

Since radial basis functions depend only on the Euclidean distance between points $\|\vec{x} - \vec{x}_i\|_2$ (i.e. they are radially symmetric in all spatial dimensions), distances between the coupling meshes $\mathcal{M}_{S_1}$ and $\mathcal{M}_{S_2}$ do not need to be explicitly addressed unlike the nearest-projection method. The accuracy of the RBF mapping does, however, degrade with increasing gaps or overlaps between the two meshes.

A comparative overview of the geometric principles underlying each mapping strategy is provided in Figures 4.8 and 4.9.

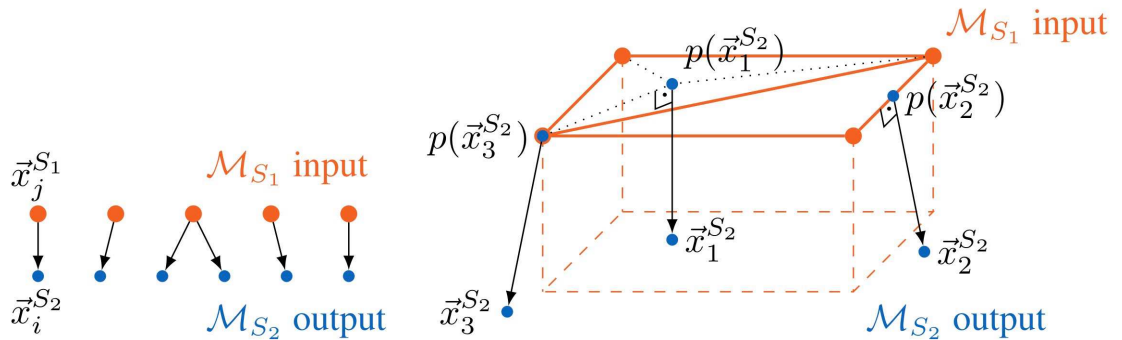


Figure 4.8: Projection-based mapping methods: nearest-neighbor and nearest-projection.[25]

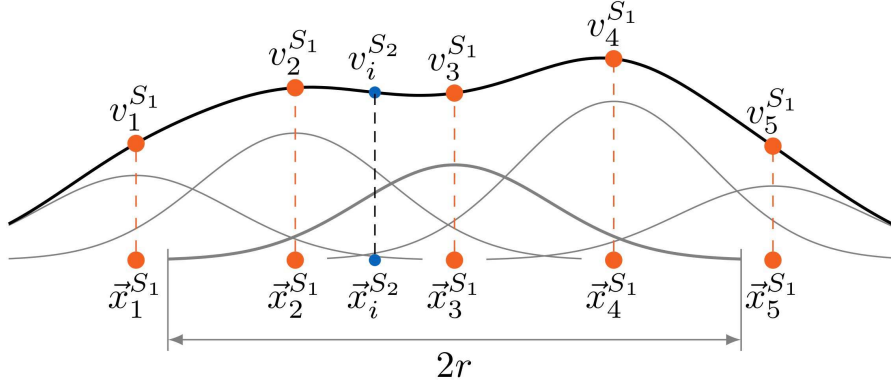


Figure 4.9: Radial basis function (RBF) mapping methods: Gaussian basis functions of support radius r , applied to a consistent mapping from an input mesh (orange) to an output sample location $\vec{x}_i^{S_2}$ (blue).[25]

4.1.3 Coupling configuration in preCICE

In a partitioned simulation, all participants exchange data at their shared interface while running as independent, separate processes. To orchestrate their simultaneous, cross-domain interactions, preCICE relies on a central **XML configuration file** that acts as the canonical blueprint for the entire coupled simulation. In addition, each solver requires an adapter configuration file to invoke preCICE and initialize the coupling.

The **preCICE configuration file** is an XML file shared among all participants, defining their roles, communication channels, mesh mapping and coupling interfaces, data exchange, coupling scheme, and time-stepping.[28] To better understand, handle and debug a given configuration file, the **preCICE-config-visualizer** tool natively interprets and translates its abstract, text-based structure into a clear, interactive visual graph.[29] Figure 4.10 shows an example generated from a case studied in the present work.

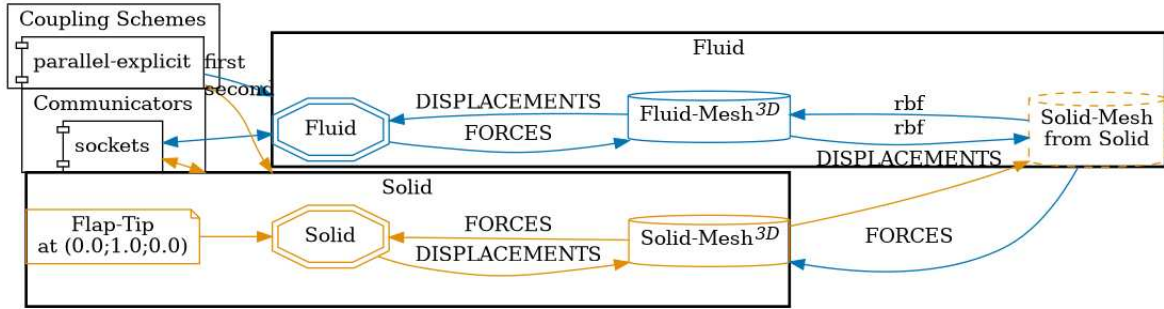


Figure 4.10: Perpendicular Flap test case (discussed in Section 4.2.2): preCICE-config-visualizer.

Dissecting the information contained in the **config-visualizer** output shown in Figure 4.10, the following can be identified:

- Participants and roles: The three-dimensional simulation involves a **Fluid** and a **Solid** participant, owning a **Fluid-Mesh** and a **Solid-Mesh**, respectively. The exchanged quantities are forces and displacements, transferred at the interface as follows:
 - Fluid: write Forces, read Displacements;
 - Solid: write Displacements, read Forces.
- Coupling scheme: the coupling is parallel-explicit, with **Fluid** as the first participant and **Solid** as the second.
- Mesh mapping: data between non-conforming meshes is projected onto the interface via RBF interpolation.

```

1 <data:vector name="FORCES" />
2 <data:vector name="DISPLACEMENTS" />
3
4 <mesh name="Fluid-Mesh" dimensions="3">
5   <use-data name="FORCES" />
6   <use-data name="DISPLACEMENTS" />
7 </mesh>
8
9 <mesh name="Solid-Mesh" dimensions="3">
10   <use-data name="DISPLACEMENTS" />
11   <use-data name="FORCES" />
12 </mesh>
13
14 <participant name="Fluid">
15   <provide-mesh name="Fluid-Mesh" />
16   <receive-mesh name="Solid-Mesh" from="Solid" />
17   <write-data name="FORCES" mesh="Fluid-Mesh" />
18   <read-data name="DISPLACEMENTS" mesh="Fluid-Mesh" />
19   <mapping:rbf direction="write" from="Fluid-Mesh" to="Solid-Mesh"
20     constraint="conservative">
21     <basis-function:compact-polynomial-c6 support-radius="0.05" />
22   </mapping:rbf>
23   <mapping:rbf direction="read" from="Solid-Mesh" to="Fluid-Mesh" constraint
24     ="consistent">
25     <basis-function:compact-polynomial-c6 support-radius="0.05" />
26   </mapping:rbf>
27   <export:vtu directory="precice-exports" every-n-time-windows="500" />
28 </participant>
29
30 <participant name="Solid">
31   <provide-mesh name="Solid-Mesh" />
32   <write-data name="DISPLACEMENTS" mesh="Solid-Mesh" />
33   <read-data name="FORCES" mesh="Solid-Mesh" />
34   <export:vtu directory="precice-exports" every-n-time-windows="500" />
35   <watch-point mesh="Solid-Mesh" name="Flap-Tip" coordinate="0.0;1.0;0.0" />
36 </participant>

```

Listing 4.1: Excerpt of the preCICE configuration file encoding the coupling described above.

The configuration file further encodes time-stepping parameters, which are not reflected in the `preCICE-config-visualizer`, as follows.

```

1 <coupling-scheme:parallel-explicit>
2   <time-window-size value="1e-4" />
3   <max-time value="5.0" />
4   <participants first="Fluid" second="Solid" />
5   <exchange data="FORCES" mesh="Solid-Mesh" from="Fluid" to="Solid" />
6   <exchange data="DISPLACEMENTS" mesh="Solid-Mesh" from="Solid" to="Fluid" />
7 </coupling-scheme:parallel-explicit>

```

Listing 4.2: Excerpt of the preCICE configuration file specifying the coupling scheme, time-stepping settings, and data routing.

Adapter Configuration Files: OpenFOAM and OpenRadioss

Alongside the preCICE configuration file, each solver requires its own adapter configuration file to invoke preCICE and initialize the coupling. The OpenFOAM adapter configuration is handled through a dictionary file called `preciceDict`, located in the `system` directory and structured as follows.[30]

```
1   FoamFile
2   {
3       version      2.0;
4       format       ascii;
5       class        dictionary;
6       location     "system";
7       object       preciceDict;
8   }
9
10  preciceConfig "../precice-config.xml";
11
12  participant Fluid;
13
14  modules (FSI);
15
16  interfaces
17  {
18      Interface
19      {
20          mesh          Fluid-Mesh;
21          patches       (flap);
22          locations     faceCenters;
23
24          readData
25          (
26              DISPLACEMENTS
27          );
28
29          writeData
30          (
31              FORCES
32          );
33      };
34  }
35
36  FSI
37  {
38      rho rho [1 -3 0 0 0 0 0] 1;
39  }
```

Listing 4.3: OpenFOAM adapter configuration: `system/preciceDict` input file.

This file contains the following key entries:

- **participant**: specifies the solver's identity within the coupled simulation. It must match the participant name defined in the preCICE configuration file, whose path is specified in quotation marks via `preciceConfig`.
- **modules**: the FSI option enables the exchange of common data fields relevant to the simulation (e.g. Forces, Displacements, Stress). The fluid density field used for force computation is further specified via the `rho` entry in the FSI block, located at the bottom of the file.
- **interfaces**: specifies the boundary patches participating in the coupled simulation, assigned to the coupling interfaces in the `system/blockMeshDict` input file. The `mesh` entry must match the one specified in the preCICE configuration file, while `locations` defines the interface mesh sampling points with respect to the mesh cells (e.g. face centers, face nodes).
- **readData** and **writeData**: define the OpenFOAM-side data routing at the coupling interface. These must match the corresponding entries specified in the preCICE configuration file.

In the case of OpenRadioss, the adapter is configured through a **.cpl** file located directly in the solver’s working directory and structured as follows.

```

1  /PRECICE/PARTICIPANT_NAME/Solid
2  /PRECICE/CONFIG_FILE/./precice-config.xml
3  /PRECICE/MESH_NAME/Solid-Mesh
4  /PRECICE/READ/FORCES
5  /PRECICE/WRITE/DISPLACEMENTS
6  /PRECICE/GRNOD/2

```

Listing 4.4: OpenRadioss adapter configuration: **.cpl** input file.

This file contains the following key entries:

- **PARTICIPANT_NAME**: specifies the solver’s identity within the coupled simulation. It must match the participant name defined in the preCICE configuration file, with its path provided via the **CONFIG_FILE** entry.
- **MESH_NAME**: identifies the coupling mesh and must match the corresponding entry in the preCICE configuration file.
- **READ** and **WRITE**: define the OpenRadioss-side data routing at the coupling interface. These must match the corresponding entries specified in the preCICE configuration file.
- **GRNOD**: specifies the group of nodes located on the boundary and participating in the coupled simulation. The **/GRNOD** card with the matching tag ID is assigned to the coupling interfaces in the **_0000.rad** starter file.

4.2 Solver integration with OpenRadioss: validation and fault identification in an early-stage adapter prototype

The aim of this research is to configure partitioned fluid-structure interaction frameworks using preCICE, coupling OpenFOAM with the structural solver OpenRadioss for which no prior preCICE support existed. As a result, the OpenRadioss adapter required systematic validation, and the simulations carried out proved effective in exposing previously undetected faults as the initially provided adapter had undergone code-level verification exclusively, without validation against physically relevant test cases. It should be noted that this work did not aim to directly resolve code-level faults, as bug resolution was addressed through issue reporting to the preCICE and OpenRadioss development teams via the official GitHub repository; the objective was rather to validate coupling and, in the event that was unsuccessful, identify the underlying causes of coupling failure.

Two main issues were identified and resolved over the course of this research: the restoration of displacement data transfer between coupled participants, and the adjustment of the structural solver’s time step, regulated by OpenRadioss’s default use of adaptive time stepping, whenever it exceeded the coupling time window size enforced by preCICE.

4.2.1 Partitioned Elastic Beam test case: restoration of displacement data transfer

The first issue encountered was the failure to establish displacement data transfer between OpenRadioss participants, as evidenced by a benchmark structure-structure interaction test case detailed in the following subsection. Given the early stage of maturity of the originally provided OpenRadioss adapter, a preliminary coupling between two solver instances was configured as a validation step to assess the integrity of inter-participant data transfer; the test case was designed to replicate the **Partitioned Beam** tutorial publicly available on the preCICE website[31], with minor modifications to the geometry and loading conditions applied to the structure.

As an initial validation attempt, the model was tested through a standalone OpenRadioss simulation. The test featured a monolithic beam of cross-sectional dimensions $0.5 \times 0.5 \text{ m}^2$ and length $l = 8 \text{ m}$, fixed

at both ends and loaded on its upper surface over $6 < z < 8$ m with a distributed load $q = 914285 \frac{\text{N}}{\text{m}}$ along the negative y -direction for $1 < t < 3$ s. Under these conditions, the loading yields analytical reaction forces of approximately 100 kN and 1730 kN at $z = 0$ m and $z = 8$ m, respectively, in accordance with standard structural mechanics formulations.[32]

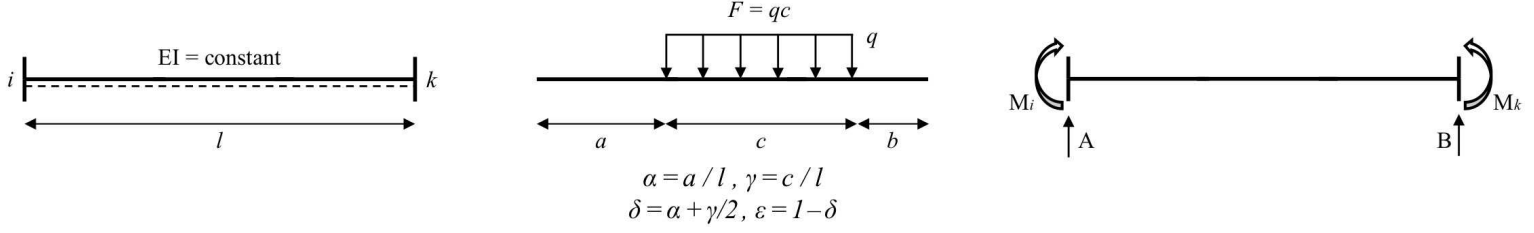


Figure 4.11: Schematic representation of the beam loading condition.[32]

The analytical solution yields $A = 100$ kN at $z = 0$ m and $B = 1730$ kN at $z = 8$ m, given by

$$A = \left[\epsilon + (\epsilon - \delta) \left(\delta \epsilon - \frac{\gamma^2}{4} \right) \right] F, \quad B = F - A \quad (4.14)$$

where $a = 6$ m, $c = 2$ m, $b = 0$ m, $F = q \cdot c = 914285 \cdot 2 = 1828570$ N = 1828.57 kN.

Numerical validation was performed via a MATLAB script, with results shown in Figures 4.12 and 4.13.

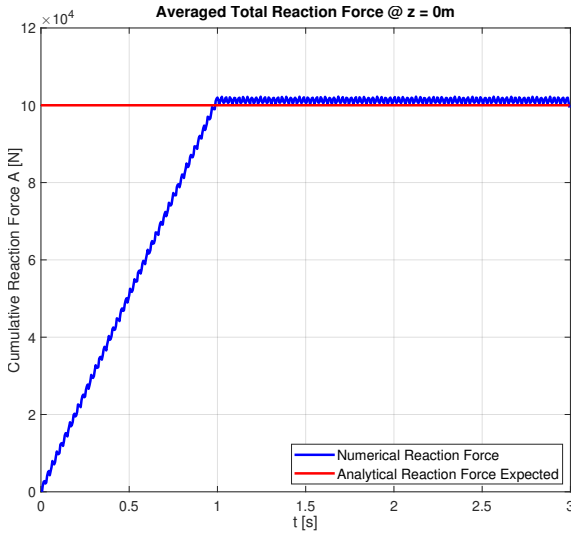


Figure 4.12: Analytical and numerical reaction force at $z = 0$ m. Post-processed in MATLAB.

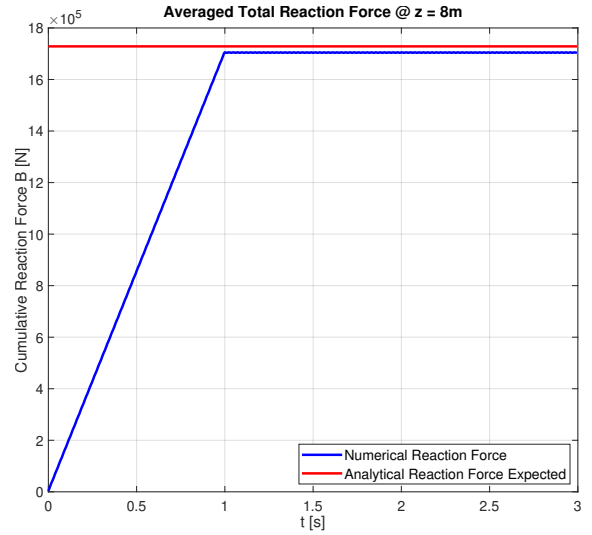


Figure 4.13: Analytical and numerical reaction force at $z = 8$ m. Post-processed in MATLAB.

The corresponding displacement distribution is presented in Figure 4.14, where the maximum displacement recorded along the negative y -direction is $\Delta y \approx 1 \times 10^{-3}$ m at $z \approx 5$ m.

Having established the physical accuracy of the monolithic case, the same configuration was reproduced as a bi-directional partitioned simulation with identical geometry and loading conditions in which the first participant corresponded to the loaded segment (**beam-right**) and the second to the unloaded portion (**beam-left**), to assess the integrity of data transfer between participants through the adapter prototype; the configuration prescribed that **beam-right** transferred displacements and received forces, while **beam-left** transferred forces and received displacements.

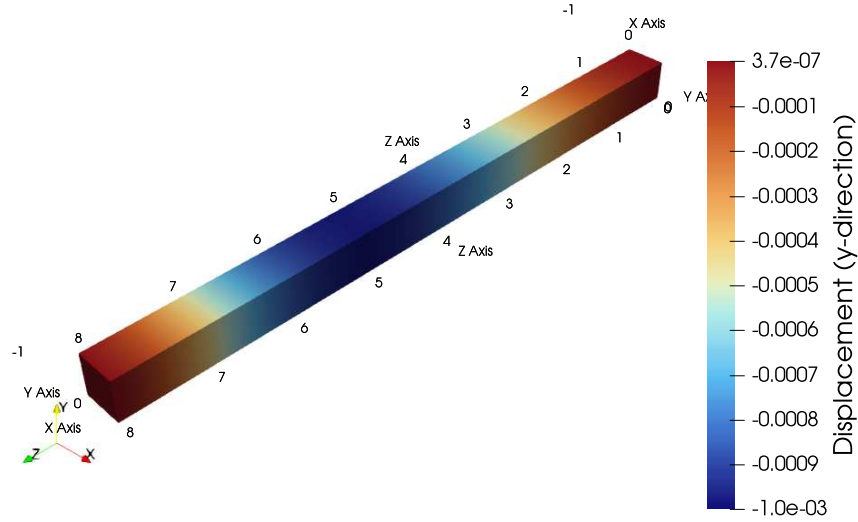


Figure 4.14: Monolithic beam fixed on both ends: displacement contour along the y -direction at $t = 3$ s. Post-processed in ParaView.

The simulation, however, failed at an early stage, terminating after approximately ten coupling time windows of duration 10^{-4} s each, with the following error originating in `beam-left`:

```

1 MESSAGE ID :      259
2 ** WARNING ZERO OR NEGATIVE VOLUME: 3D-ELEMENT ID: 5988
3 ELEMENT IS SWITCHED TO SMALL STRAIN OPTION
4 ---[precice] ERROR: advance() cannot be called with a time step size of 0.
5 terminate called after throwing an instance of 'precice::Error'
6 what(): advance() cannot be called with a time step size of 0.
7 User or system abort detected (CTRL-C) !
8 ERROR TERMINATION

```

Listing 4.5: Error output generated upon premature termination of the partitioned simulation.

The zero-volume warning issued for `beam-left` indicates that interface elements underwent inadmissibly large deformations, causing the solver to switch to the small strain option and consequently supply a zero time step size to preCICE, which triggered the `advance()` error.

```

1 <participant name="s1">
2   <provide-mesh name="OpenRadioSS-Mesh1" />
3   <receive-mesh name="OpenRadioSS-Mesh2" from="s2" />
4   <write-data name="FORCES" mesh="OpenRadioSS-Mesh1" />
5   <read-data name="DISPLACEMENTS" mesh="OpenRadioSS-Mesh1" />
6   <mapping:nearest-neighbor
7     direction="write"
8     from="OpenRadioSS-Mesh1"
9     to="OpenRadioSS-Mesh2"
10    constraint="conservative" />
11   <mapping:nearest-neighbor
12     direction="read"
13     from="OpenRadioSS-Mesh2"
14     to="OpenRadioSS-Mesh1"
15     constraint="consistent" />
16   <export:vtu directory="precice-exports"/>
17 </participant>
18 <participant name="s2">
19   <export:vtu directory="precice-exports"/>
20   <provide-mesh name="OpenRadioSS-Mesh2" />
21   <write-data name="DISPLACEMENTS" mesh="OpenRadioSS-Mesh2" />
22   <read-data name="FORCES" mesh="OpenRadioSS-Mesh2" />
23 </participant>

```

Listing 4.6: Excerpt of the preCICE configuration file showing the export files implementation for both participants.

As an initial diagnostic step, the coupling configuration was modified to prescribe uni-directional displacement transfer only, temporarily excluding force data, and the simulation was re-run. The export files, visualized as spheres at the coupling interface nodes in Figure 4.15, confirmed that displacement data was not transferred from **beam-right** to preCICE and consequently not received by **beam-left**, which remained undeformed in contrast to the loaded segment that underwent the expected deformation. The fault was thereby localized to the adapter prototype, which failed to correctly handle displacement data in both the write and read directions.

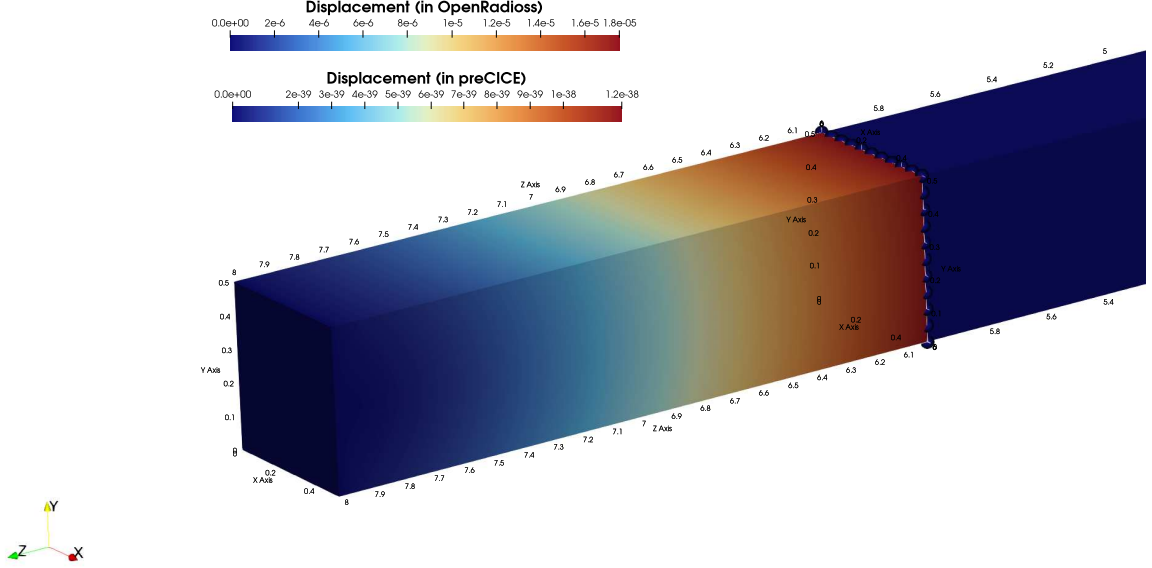


Figure 4.15: Partitioned Elastic Beam test case: faulted displacement field at $t = 0.01$ s under uni-directional coupling, showing absence of displacement transfer to **beam-left**. Post-processed in ParaView.

Following identification of the fault, an issue was opened in the OpenRadioss GitHub repository and subsequently resolved by the OpenRadioss development team.[33] The fault originated in a hardcoded data type mapping located in the adapter **Fortran** interface, which rendered the adapter **.cpl** configuration file ineffective at runtime by disregarding any data types specified in the XML **configuration**. Since **read** and **write** operations were resolved against a fixed set of predefined combinations instead, when the **coupling_displacements** option was configured through the solver **.cpl** file, the OpenRadioss adapter incorrectly passed **coupling_positions** to the read function of the **C++** adapter instead causing the adapter to check for exchange of position data; when no such configuration was found in the implemented coupling, the **read** operation was consequently skipped entirely. For a detailed account of the debugging process, the reader is referred to[33] and[34], as the aim of this research is to validate and assess the adapter prototype rather than to contribute directly to its development.

Following the patch, a uni-directional partitioned coupling prescribing displacement transfer only was configured to confirm that the fault had been resolved; as Figure 4.16 illustrates, the simulation completed successfully, yielding a displacement distribution consistent with that obtained in the monolithic case.

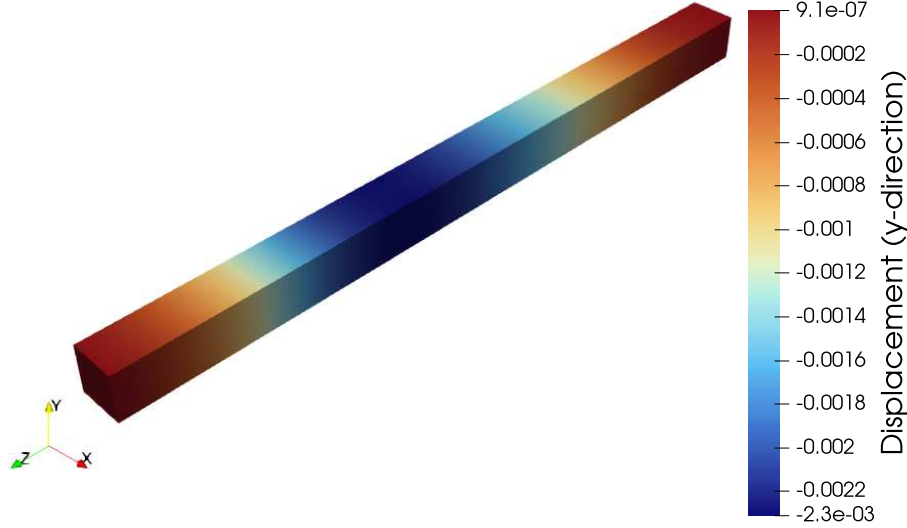


Figure 4.16: Partitioned beam fixed on both ends: displacement contour along the y -direction under uni-directional coupling, at $t = 3$ s. Post-processed in ParaView.

Although the uni-directional configuration does not fully replicate the displacement values observed in the monolithic case, as the unloaded participant transmitted no forces in response rendering the two cases physically non-equivalent, the qualitative agreement between the displacement fields, as evidenced by the displacement contour, confirmed that data was correctly transferred between participants and that the fault had been resolved.

Upon reinstatement of the original bi-directional coupling configuration, the test case proved unsuitable for validation purposes owing to numerical instabilities that arose throughout the simulation runs given the explicit nature of *both* solvers involved; no exact replication of the monolithic case was therefore achieved. The validation of bi-directional partitioned coupling was therefore redirected toward a fluid-structure interaction configuration with OpenFOAM serving as the second participant, where the implicit time integration provided by the different solver and the elimination of adaptive time stepping (inherent to explicit dynamics solvers by default, yet a known source of instability in partitioned coupling) yielded a considerably more stable simulation. Nevertheless, this test case enabled the identification and resolution of a significant fault that would otherwise have compromised the success of fluid-structure interaction configurations as well.

4.2.2 Perpendicular Flap test case: adaptive time step reduction to coupling time window size

As the bi-directional coupling between two structural participants was ultimately unsuccessful, owing to numerical instabilities introduced by the explicit dynamics formulation of both participants, data transfer integrity through the OpenRadioss adapter was assessed via a second preliminary coupling incorporating a participant of a different nature: a fluid domain modelled in OpenFOAM. The test case was designed to replicate the **Perpendicular Flap** tutorial publicly available on the preCICE website[18], adjusted to a three-dimensional geometry (instead of the pseudo-2D domain featured in the reference) in order to accommodate the absence of two-dimensional coupling support in the OpenRadioss adapter prototype. The **config-visualizer** of the coupled configuration is shown in Figure 4.10.

The test case featured the $6 \times 4 \times 1$ m OpenFOAM domain provided by the tutorial, prescribing a constant inflow profile on the left boundary, an outflow condition on the right boundary and a no-slip condition on the top and bottom walls of the channel and flap surface,[18] and the $0.1 \times 1 \times 1$ m OpenRadioss participant representing a solid, elastic flap fixed at its base placed perpendicularly to the flow. An incompressible, fictitious fluid crossed the channel at a velocity of $v = 10$ m/s in the x -direction and impacted the flap, composed of an equally fictitious material and located at the center of the domain ($x = 0$), causing the structure to oscillate in response to the fluid pressure acting on its surface. The physical properties of both domains are summarized in Table 4.1 and 4.2, and the setup is illustrated schematically in Figure 3.4.

Density ρ_F	Kinematic Viscosity ν_F
$\rho_F = 1.0 \text{ kg/m}^3$	$\nu_F = 1.0 \text{ m}^2/\text{s}$

Table 4.1: Physical properties of the fluid domain.

Density ρ_S	Young's Modulus E	Poisson's Ratio ν_S
$\rho_S = 3.0 \cdot 10^3 \text{ kg/m}^3$	$E = 4.0 \cdot 10^6 \text{ kg/ms}^2$	$\nu_S = 0.3$

Table 4.2: Physical properties of the solid domain.

To extend the simulation to a fully three-dimensional configuration, the unit-depth dimension of both domains along the z -direction was discretized with 5 cells. The simulation was run for $t = 5 \text{ s}$. In order to enable comparison against a validation baseline, established by the mutual agreement of reference logs produced by alternative structural solvers under identical fluid coupling conditions, the temporal evolution of the flap tip was tracked through a watchpoint placed at $(x, y, z) = (0, 1, 0) \text{ m}$ to monitor the exchanged coupling quantities at that location over time.

In a preliminary coupling assessment, the test case was configured with no explicit control applied to the structural participant time step, although adaptive time stepping and consequent sub-cycling are known to introduce instabilities and lead to participant desynchronization in fluid-structure interactions. The fluid solver, on the other hand, operated at a fixed time step of $\Delta t_F = 1 \times 10^{-4} \text{ s}$, set equal to the preCICE coupling time window size. The displacement response along the x -direction obtained under this configuration is shown in Figure 4.17; the reference output logs produced by alternative structural solvers are publicly available at [35].

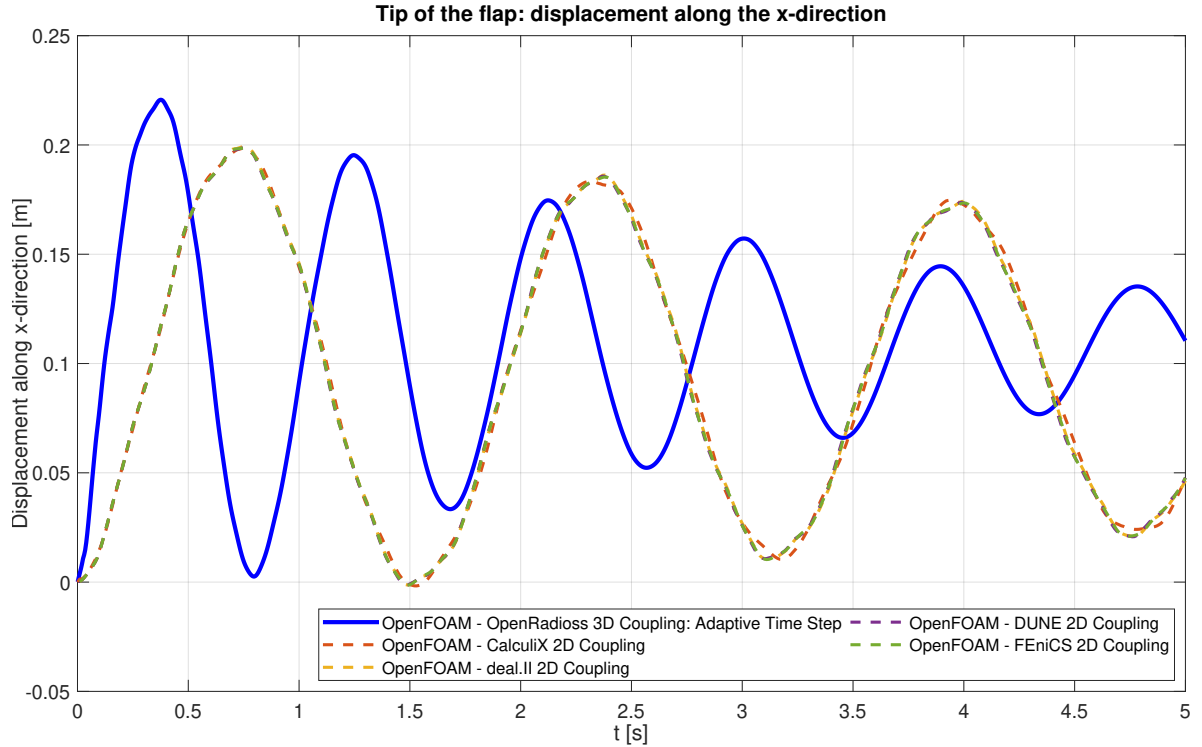


Figure 4.17: Unconstrained structural time stepping: displacement response of the **Perpendicular Flap** tip in the x -direction. Post-processed in MATLAB.

These results indicate that configurations featuring an unconstrained structural time step fail to reproduce the physical behavior of the system, as evidenced by the oscillation frequency mismatch, and yield a physically inconsistent structural response as the OpenRadioss flap undergoes six oscillation cycles compared to the three recorded for all other structural participants sharing the same physical and geometrical properties.

Upon inspection of the OpenRadioss simulation logs, the **_0001.out** output file revealed that the structural solver running on an adaptive time step operated with a $\text{solverDt} \approx 6.5 \times 10^{-4}$ s, as shown in Listing 4.7. Given that the solverDt exceeded the coupling time window size (set to $\text{preciceDt} = 1 \times 10^{-4}$ s), this violation exposed a defect in the adapter's time step enforcement logic, under which solverDt should have been capped at preciceDt .

CYCLE	TIME	TIME-STEP	ELEMENT	ERROR	I-ENERGY	K-ENERGY	T
K-ENERGY R	EXT-WORK	MAS.ERR	TOTAL MASS	MASS ADDED			
0	0.000	0.6736E-03	SOLID	196	0.0%	0.000	0.000
	0.000	0.000	0.000	0.000	300.0	0.000	
ANIMATION FILE: solid-openradiossA001 WRITTEN							
100	0.1000E-01	0.6703E-03	SOLID	109	0.0%	4.392	10.82
	0.000	0.000	0.000	0.000	300.0	0.000	
200	0.2000E-01	0.6703E-03	SOLID	136	0.0%	4.586	12.36
	0.000	0.000	0.000	0.000	300.0	0.000	
300	0.3000E-01	0.6700E-03	SOLID	31	0.0%	4.499	15.46
	0.000	0.000	0.000	0.000	300.0	0.000	
400	0.4000E-01	0.6701E-03	SOLID	136	0.0%	4.828	17.05
	0.000	0.000	0.000	0.000	300.0	0.000	
500	0.5000E-01	0.6702E-03	SOLID	136	0.0%	4.840	20.16
	0.000	0.000	0.000	0.000	300.0	0.000	
ANIMATION FILE: solid-openradiossA002 WRITTEN							
600	0.6000E-01	0.6694E-03	SOLID	136	0.0%	4.770	23.73
	0.000	0.000	0.000	0.000	300.0	0.000	
700	0.7000E-01	0.6680E-03	SOLID	136	0.0%	5.122	25.75
	0.000	0.000	0.000	0.000	300.0	0.000	
800	0.8000E-01	0.6687E-03	SOLID	136	0.0%	5.320	28.12
	0.000	0.000	0.000	0.000	300.0	0.000	
900	0.9000E-01	0.6694E-03	SOLID	138	0.0%	5.715	29.54
	0.000	0.000	0.000	0.000	300.0	0.000	
1000	0.1000	0.6692E-03	SOLID	138	0.0%	6.463	29.64
	0.000	0.000	0.000	0.000	300.0	0.000	

Listing 4.7: Excerpt from the **_0001.out** output file: first 1000 iterations of the OpenRadioss simulation. The adaptive time step solverDt ($\approx 6.5 \times 10^{-4}$ s) exceeds the coupling time window size preciceDt .

```

1 <coupling-scheme:serial-explicit>
2   <time-window-size value="1e-4" />
3   <max-time value="5.0" />
4   <participants first="Fluid" second="Solid" />
5   <exchange data="FORCES" mesh="Solid-Mesh" from="Fluid" to="Solid" />
6   <exchange data="DISPLACEMENTS" mesh="Solid-Mesh" from="Solid" to="Fluid"
7   />
8 </coupling-scheme:serial-explicit>

```

Listing 4.8: Excerpt from the preCICE XML configuration, showing that the coupling time window size was set to 1×10^{-4} s.

As a diagnostic step, the OpenRadioss configuration was modified to constrain the adaptive time step to a fixed value and the simulation was re-run. To this end, the lines shown in Listing 4.9 were added to the **_0001.rad** input file, where the **/DT** engine keyword defines a scale factor for the time step (set to its default value of 0.9) and the minimum time step for all elements, while **/DTIX** defines the initial and maximum time step allowed within the simulation:

```

1 /DT
2 0.9 0.0001
3 #
4 /DTIX
5 0.0001 0.0001

```

Listing 4.9: Excerpt from the **_0001.rad** input file, showing the adaptive time step range restriction to a fixed value.

Since OpenRadioss provides no direct mechanism to disable adaptive time stepping within the participant, a fixed time step can be prescribed only by imposing identical upper and lower bounds on the adaptive time step range.

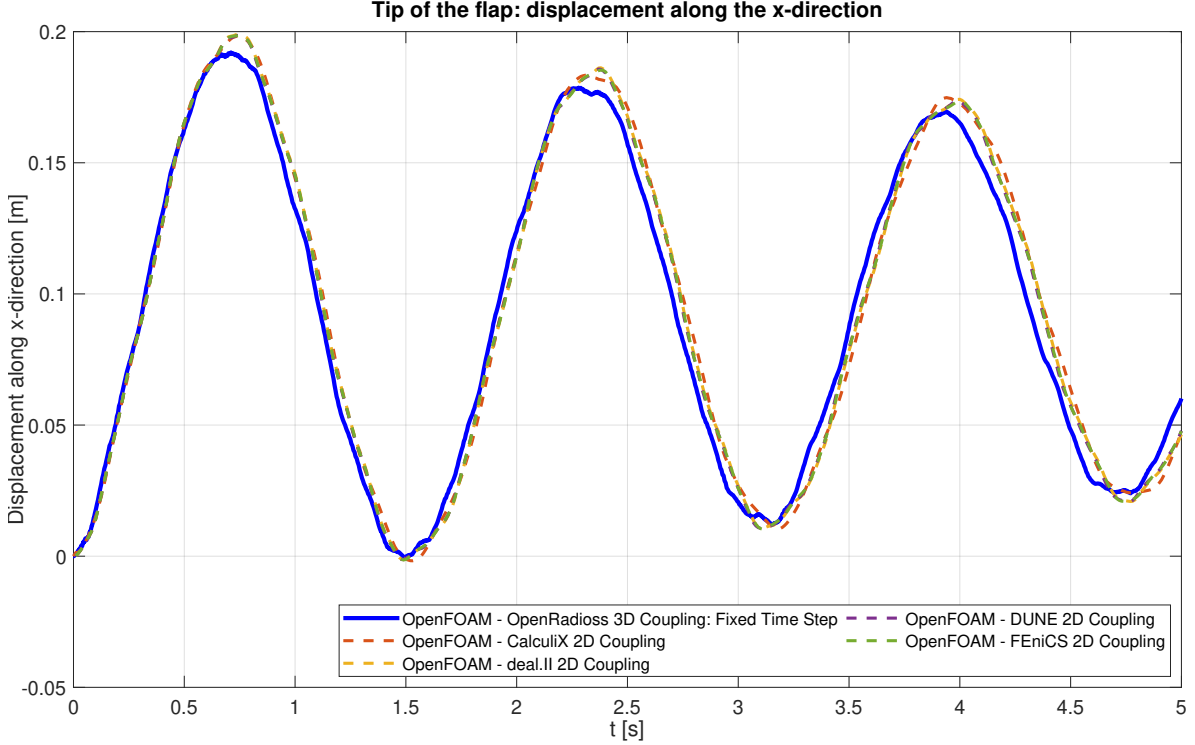


Figure 4.18: Fixed structural time stepping: displacement response of the **Perpendicular Flap** tip in the x -direction. Post-processed in MATLAB.

Under the constrained configuration, the post-processing plot shown in Figure 4.18 is considerably more consistent with the reference output logs produced by alternative structural solvers, with the minor discrepancies arising from the dimensional mismatch between the three-dimensional OpenRadioss configuration and the pseudo-two-dimensional reference cases. These findings confirmed that the physically inconsistent response documented above originated from the adaptive time step configuration and the faulted time step enforcement logic at the adapter level; following identification of the defect, an additional issue was opened in the OpenRadioss GitHub repository and subsequently resolved by the OpenRadioss development team.[36]

As noted above, the fault originated in the adapter's failure to verify whether the solver-side time step exceeded the coupling window boundary. As a result, the structural solver overshoot coupling synchronization points by advancing in steps significantly larger than permitted, and OpenRadioss covered approximately six preCICE windows within a single iteration ($\text{solverDt} / \text{preciceDt} = 6 \times 10^{-4} / 1 \times 10^{-4} \approx 6$). From an implementation standpoint, the time step handling logic coded within the adapter's C++ interface was affected by a defect in which the enforcement pattern $\text{dt} = \min(\text{preciceDt}, \text{solverDt})$ proved ineffective at runtime due to its misplacement within an unrelated conditional branch, despite already being present in the adapter source. More specifically, the enforcement block was contingent on the presence of contact interfaces in the model, rendering the restriction unreachable in configurations where no contact interfaces were defined. In conclusion, the physically inconsistent structural response observed in Figure 4.17 was attributable to the fact that, at the coupling level, the overshooting was interpreted as a sequence of completed coupling windows, although the structural solver's internal clock had advanced beyond the prescribed coupling window boundaries. Consequently, the time instants at which data exchange occurred represented incorrectly resolved time levels. For a detailed account of the debugging process, the reader is referred to[36], [37] and[38], as the aim of this research is to validate and assess the adapter prototype rather than to contribute directly to its development.

Following the patch, the **Perpendicular Flap** simulation was repeated to verify that the fault had been resolved, as confirmed by the output logs shown in Listing 4.10.

1	CYCLE	TIME	TIME-STEP	ELEMENT	ERROR	I-ENERGY	K-ENERGY	T
2		K-ENERGY R	EXT-WORK	MAS.ERR	TOTAL	MASS	MASS ADDED	
2	0	0.000	0.1000E-04	FIXED	0	0.0%	0.000	0.000
			0.000	0.000	0.000	300.0	0.000	
3	ANIMATION FILE: solid-openradiossA001 WRITTEN							
4	100	0.1000E-02	0.1000E-04	FIXED	0	0.0%	0.4997	0.4918
		0.000	0.000	0.000	0	0.0%	0.5166	0.4258
5	200	0.2000E-02	0.1000E-04	FIXED	0	0.0%	0.4223	0.4829
		0.000	0.000	0.000	0	0.0%	0.4575	0.4408
6	300	0.3000E-02	0.1000E-04	FIXED	0	0.0%	0.4631	0.4189
		0.000	0.000	0.000	0	0.0%	0.4236	0.4727
7	400	0.4000E-02	0.1000E-04	FIXED	0	0.0%	0.3974	0.4985
		0.000	0.000	0.000	0	0.0%		
8	500	0.5000E-02	0.1000E-04	FIXED	0	0.0%		
		0.000	0.000	0.000	0	0.0%		
9	600	0.6000E-02	0.1000E-04	FIXED	0	0.0%		
		0.000	0.000	0.000	0	0.0%		
10	700	0.7000E-02	0.1000E-04	FIXED	0	0.0%		
		0.000	0.000	0.000	0	0.0%		

Listing 4.10: Excerpt from the `_0001.out` output file: first 700 iterations of the corrected OpenRadioss simulation. The solver time step `solverDt` appears to be consistent with the preCICE coupling time window `preciceDt`, meaning that the fault shown in Listing 4.7 has been resolved.

Given that sub-cycling arising from adaptive time stepping is known to introduce instabilities in partitioned fluid-structure interaction frameworks, which are independent of the solver-side time step adjustment and are therefore not directly caused by bugs in the adapter source, even following confirmation that the aforementioned defect had been resolved all subsequent fluid-structure interaction cases discussed in the present work were maintained with a forced fixed time step, as detailed in Listing 4.9. This decision was motivated by the fact that, although the dimensional mismatch between the OpenRadioss configuration and the reference domain introduced minor discrepancies, the coupling established by the preCICE tutorial was replicated successfully, with a watchpoint evolution consistent with that recorded for the alternative structural solvers.

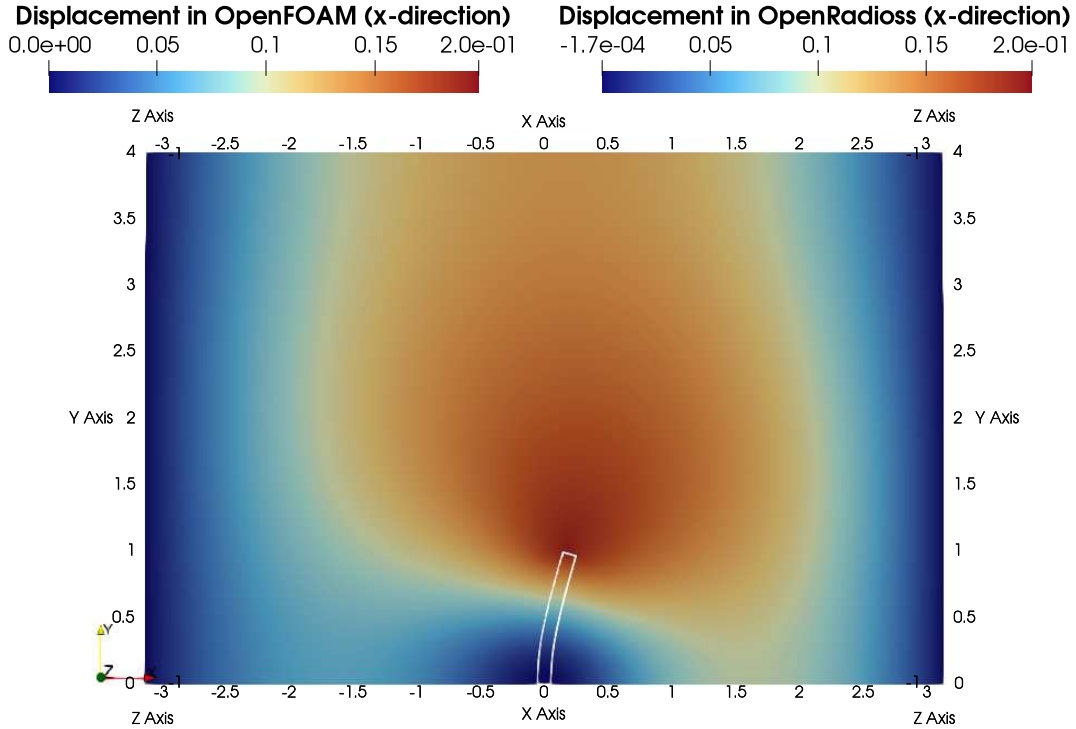


Figure 4.19: Perpendicular Flap test case: displacement field along the x -direction under bi-directional coupling, at $t = 0.75$ s. Post-processed in ParaView.

The successful reproduction of the **Perpendicular Flap** tutorial, with a fixed time step enforced in the OpenRadioss participant, confirmed that displacement and force data were correctly transferred between coupled participants and that the previously identified defects had been resolved. Building on this outcome it was possible to proceed to more complex fluid-structure interaction cases, both to further validate partitioned frameworks against established reference data in geometrically consistent configurations, as well as to investigate previously undocumented coupled behavior by implementing original setups without an existing validation baseline.

Chapter 5

Solver validation against a fluid-structure interaction benchmark: Pulsating Flow over Flexible Flap

The aim of this chapter is to present and discuss a benchmark case originally introduced as a monolithic reference by Schott et al. [39] and first validated as a partitioned framework using preCICE to couple an OpenFOAM fluid solver with a deal.II structural participant by Chourdakis et al. [40]. Consistent with the validation objective of this thesis, the present implementation replaces the structural solver of the reference study with an equivalent OpenRadioss participant. The case consists of a three-dimensional computational domain comprising an hyperelastic flap clamped at its base and immersed in a laminar channel flow; the simulation runtime is set to $t = 30$ s. For complete access to the simulation setup, including the OpenFOAM, OpenRadioss, and preCICE configurations, the reader is referred to [41].

The fluid domain features an unsteady, non-uniform inlet velocity profile; it is simulated with the `pimpleFoam` solver, whose numerical settings are described in detail in Section 5.1.2. The structural participant, in turn, represents a flap composed of a compressible non-linear neo-Hookean material, whose constitutive behavior is modeled in OpenRadioss via the `/MAT/LAW42` material card.

The test case has been simulated on two different mesh refinement levels, referred to as Coarse and Medium cases [40], whose cell counts are listed in Table 5.1. Additionally, a mesh grading was applied to the fluid mesh to achieve finer resolution in the proximity of the flexible flap.

Mesh Refinement	Coarse	Medium
# Fluid Cells	13 710	105 960
# Solid Cells	48	384

Table 5.1: Spatial resolution of the coupled simulations. [40]

Validation was performed by comparing the tip displacement of the flap along the x -direction, for both the Coarse and Medium OpenRadioss meshes, at the centerline ($x = 0.5$ m, $y = 0.35$ m, $z = 0.0$ m) and at the left corner ($x = 0.57$ m, $y = 0.35$ m, $z = -0.3$ m) against both the corresponding deal.II watchpoint time histories [40] and the $\mathcal{A}2$ -fine results of the monolithic reference [39].

The subsequent sections provide a detailed description of the implementation of each participant and the coupling configuration, as well as a thorough discussion of the results obtained and the conclusions drawn therefrom.

5.1 The OpenFOAM participant: setup and configuration

5.1.1 Physical properties

The fluid participant consists of an OpenFOAM domain modeling a three-dimensional laminar channel flow, driven by an unsteady, non-uniform inlet velocity profile; the dimensions of the fluid domain are prescribed as $\Omega_F = [0, 1.8] \times [0, 0.6] \times [-0.6, 0.6]$ m. The configuration follows the reference implementation provided by Chourdakis et al. [40]¹ to ensure reproducibility and direct comparability with the benchmark results.

¹Pre-configured OpenFOAM domain available at: <https://journal.openfoam.com/index.php/ofj/article/view/88>

The inlet boundary condition prescribes a pulsating parabolic velocity profile following the definition of Schott et al. [39]:

$$\mathbf{u} = (u, 0, 0) \cdot g(t), \quad (5.1)$$

where

$$u = u_{\max} \frac{2500}{81} y(y - 0.6)(z + 0.6)(z - 0.6), \quad u_{\max} = 2 \text{ m/s}, \quad (5.2)$$

and $g(t)$ is a temporal modulation function defined as:

$$g(t) = \begin{cases} \frac{1}{2} (1 - \cos(\pi t)) & t \in [0, 10.0] \text{ s} \\ 0 & t \in (10.0, 30.0] \text{ s} \end{cases} \quad (5.3)$$

Figure 5.1 illustrates the non-uniform inlet velocity profile at $t = 1$ s, alongside the temporal evolution of $g(t)$ which is depicted in Figure 5.2.

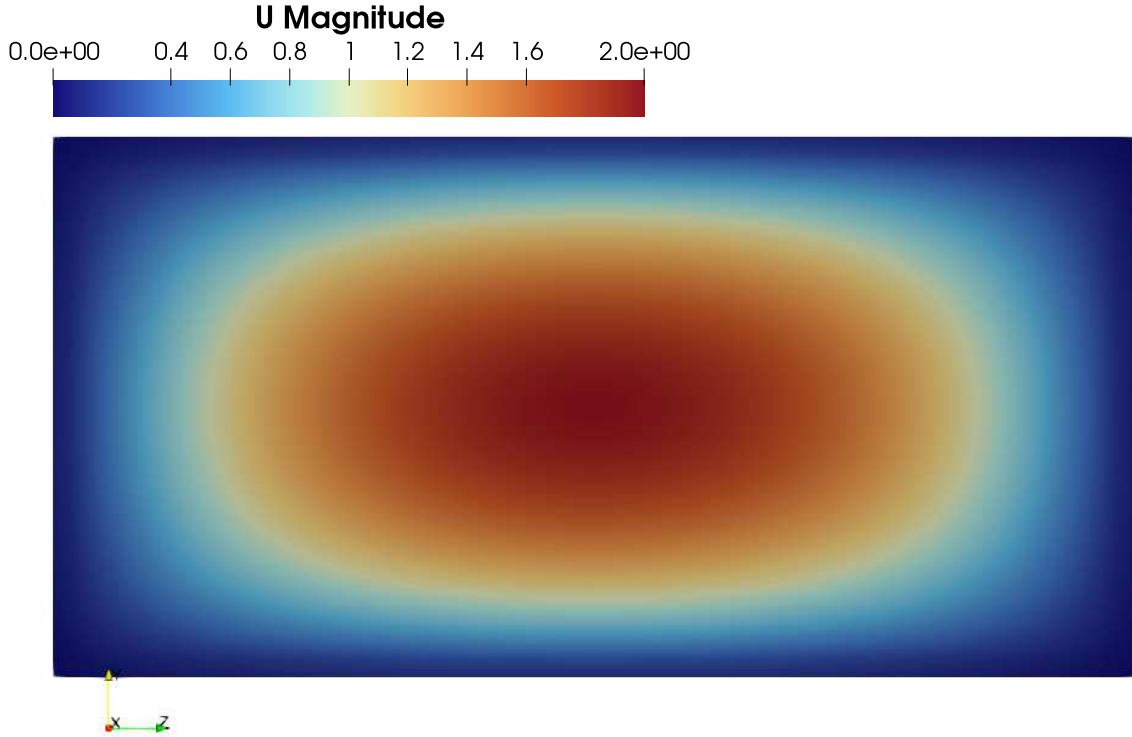


Figure 5.1: Parabolic inlet velocity profile at $t = 1$ s. Post-processed in ParaView.

Additionally, the domain features a fixed pressure condition at the outlet and a no-slip condition on all lateral walls, and a `movingWallVelocity` condition is prescribed at the shared boundary (i.e. the flap interface) to accommodate the mesh deformation imposed by the structural participant.

The OpenFOAM domain contains an incompressible, laminar Newtonian fluid whose physical properties were artificially modified from the channel flow configuration proposed by Gerstenberger et al.[42] to enhance dynamic interactions and coupling effects between the pulsating flow and the flexible structure. The Newtonian transport model characterizes the fictitious fluid by a constant kinematic viscosity ν as listed in Table 5.2.

Kinematic Viscosity ν [m ² /s]
0.01

Table 5.2: Physical properties of the fictitious Newtonian fluid.[39] [40]

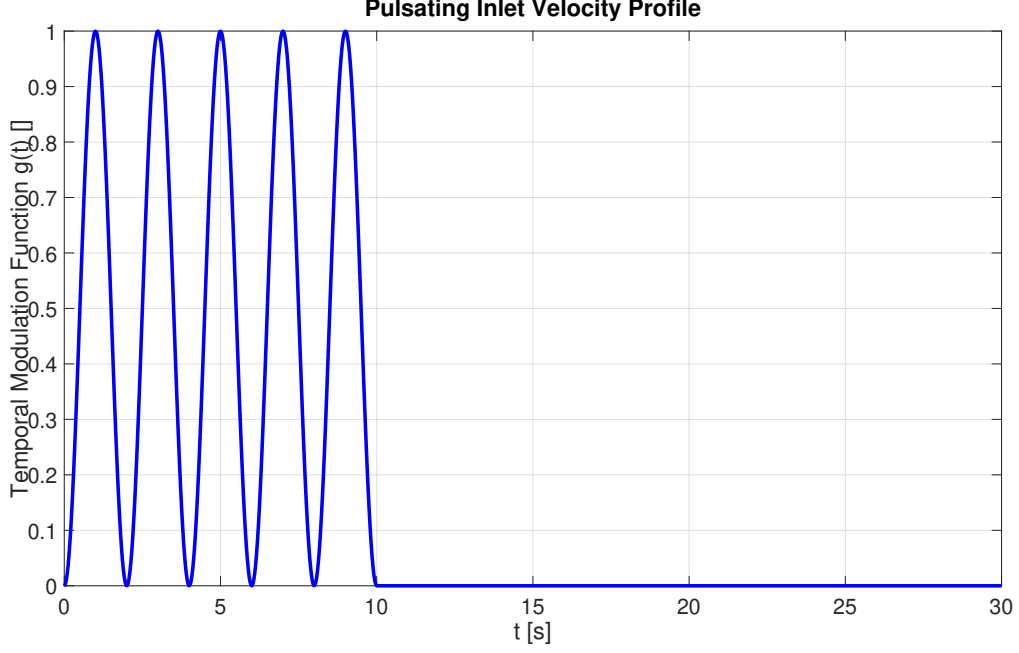


Figure 5.2: Pulsating inlet velocity profile for $0 < t < 30$ s. Processed in MATLAB.

Finally, to accommodate the mesh deformation induced by the moving flap, the dynamic mesh motion is governed by a Laplacian diffusion equation ensuring smooth propagation of boundary displacements throughout the fluid domain. The mesh diffusivity is prescribed as a quadratic inverse function of the distance from the fluid-structure interface (Listing 5.2), allowing for a smooth variation of diffusivity gradients in regions of large displacement whereas regions remote from the flap retain steeper gradients as they undergo minimal deformation. As a result, mesh quality is preserved during large structural displacements since the diffusivity distribution confines significant deformation to the cells immediately surrounding the flap interface.

5.1.2 Numerical implementation

The numerical implementation of the OpenFOAM participant employs the `pimpleFoam` solver, which is designed for transient simulations of incompressible flows supporting both laminar and turbulent regimes under Newtonian and non-Newtonian fluid models. As a pressure-based solver, `pimpleFoam` relies on the PIMPLE algorithm to resolve pressure-momentum coupling, as described in Section 2.3.3. Additionally, its native support for dynamic mesh morphing extends its applicability to moving boundary problems, making `pimpleFoam` the appropriate solver for the present numerical configuration.

The boundary and initial conditions of the fluid domain are specified within the `0` time directory, which contains, among other input files, the mandatory pressure (`0/p`) and velocity (`0/U`) fields required by the pressure-based solver. The pressure field prescribes a `zeroGradient` condition at all boundary faces except the outlet, where a fixed pressure condition is imposed; the pulsating parabolic inlet defined in Eq. 5.1, by contrast, is prescribed within the `0/U` file. The profile, which depends on both space and time, is implemented through a `codeFixedValue` condition, a fixed-value boundary condition that supports user-defined coded expressions. The corresponding configuration imposed at the inlet boundary is shown in Listing 5.1.

The incompressible nature of the flow considered requires that the volumetric flux through the cell faces also be initialized in the `0` time directory, through the `0/phi` input file. This field is imposed at all boundaries, since no flux information can be derived from the density field in the incompressible formulation, and its value is computed from the velocity field at each time step rather than prescribed independently. Finally, to enable dynamic mesh morphing and couple the fluid domain to structural deformation, a `0/pointDisplacement` field is initialized to track the displacement vector $\mathbf{d}$ of each flap interface node from its reference position and propagate it through the mesh. The field assigns zero displacement to all rigid boundaries, whereas the `flap` patch, initialized to zero, is updated at each coupling iteration. To further preserve mesh integrity during flap deformation, a `slip` condition is imposed on the upper wall of the channel to accommodate tangential motion along that boundary.

```

1 boundaryField
2 {
3     inlet
4     {
5         type            codedFixedValue;
6         value            uniform (0 0 0);
7         name            pulsatingParabolicInlet;
8
9         code
10        #{
11            const vectorField& Cf = patch().Cf();
12            const vectorField& n  = patch().nf();
13            vectorField& field    = *this;
14
15            const scalar pi = constant::mathematical::pi;
16            const scalar t  = this->db().time().value();
17
18            forAll(Cf, faceI)
19            {
20                const scalar y = Cf[faceI].y();
21                const scalar z = Cf[faceI].z();
22
23                const scalar poly =
24                    -2.0 * (2500.0/81.0) * y * (y - 0.6) * (z + 0.6) * (z - 0.6);
25
26                vector para = poly * n[faceI];
27
28                if (t < 10.0)
29                {
30                    scalar ramp = 0.5 * (1.0 - cos(pi * t));
31                    field[faceI] = ramp * para;
32                }
33                else
34                {
35                    field[faceI] = vector::zero;
36                }
37            }
38        #};
39    }
40 }

```

Listing 5.1: Pulsating parabolic inlet profile, prescribed via `codeFixedValue` in `0/U`.

The fluid physical parameters and flow regime definition are specified in the `constant` directory through the `transportProperties` and `turbulenceProperties` files, respectively, where the conditions presented in Section 5.1.1 are initialized. Dynamic mesh morphing is configured in the `constant/dynamicMeshDict` dictionary to accommodate the mesh deformation induced by flap displacement, and the motion is governed by a Laplacian diffusion equation with diffusivity inversely proportional to the distance from the fluid-structure interface. The mesh motion solver operates directly on the displacement field, with the prescribed displacement enforced as a Dirichlet boundary condition at each coupling time step. An absolute tolerance of 10^{-6} is set for `cellDisplacement`, and the diffusivity coefficient (`displacementLaplacianCoeffs`) is defined as a quadratic inverse function of the distance from the flexible flap. The corresponding implementation is shown in Listing 5.2.

The `system` directory completes the solver implementation by hosting the numerical configuration, algorithm settings and tolerances, and runtime control parameters. The `system/fvSchemes` file specifies the discretization schemes adopted for the simulation. Time derivatives are discretized with a first-order implicit Euler scheme, while gradient and divergence terms are discretized through Gauss linear interpolation of face values, with a cell-limited Gauss linear scheme for the velocity gradient and a limited linear scheme for the convective term to maintain boundedness and suppress numerical oscillations. To preserve accuracy on non-orthogonal meshes, Laplacian terms and surface-normal gradients are linearly interpolated with an explicit non-orthogonal correction.

```

1 dynamicFvMesh dynamicMotionSolverFvMesh;
2
3 motionSolverLibs ("libfvMotionSolvers.so");
4
5 solver          displacementLaplacian;
6
7 displacementLaplacianCoeffs {
8
9     diffusivity quadratic inverseDistance (flap);
10
11 }

```

Listing 5.2: `constant/dynamicMeshDict`: configuration of the Laplacian diffusion equation governing the mesh motion.

Following the configuration proposed by Chourdakis et al. [40], the PIMPLE loop implemented in the `system/fvSolution` file is configured to perform three PIMPLE pressure corrections (`nCorrectors`), five SIMPLE corrections (`nOuterCorrectors`) and three non-orthogonal corrector loop iterations (`nNonOrthogonalCorrectors`) to account for large mesh distortions.

Runtime control is specified in the `system/controlDict` dictionary, which sets the simulation duration to $t = 30$ s, a time step of $\Delta t_F = 0.001$ s, and a write interval of 0.5 s. In addition, the dictionary configures the calls to the preCICE library and registers the preCICE adapter as an OpenFOAM function object as follows:

```

1 functions
2 {
3     preCICE_Adapter
4     {
5         type preciceAdapterFunctionObject;
6         libs ("libpreciceAdapterFunctionObject.so");
7     }
8 }

```

Listing 5.3: `system/controlDict`: configuration of the preCICE adapter as an OpenFOAM function object.

Finally, the coupling parameters of the fluid participant are specified in the `system/preciceDict` dictionary, following the configuration presented in Listing 4.3 and discussed in Section 4.1.3.

5.2 The OpenRadioss participant: setup and configuration

5.2.1 Physical properties

The structural participant consists of an OpenRadioss domain modeling a three-dimensional flexible flap clamped at its base and composed of a neo-Hookean material; the dimensions of the structural domain are prescribed as $\Omega_S = [0.5, 0.57] \times [0, 0.35] \times [-0.3, 0.3]$ m. This configuration aims to recreate the reference deal.II participant provided by Chourdakis et al.[40], since implementing an equivalent participant in OpenRadioss would serve as the first documented attempt to integrate the previously unsupported structural solver within the preCICE framework. A successful implementation would therefore validate the established coupling through reproducibility and direct comparability with the benchmark results.

The OpenRadioss domain is constituted by a compressible, non-linear neo-Hookean material, whose physical properties were also artificially modified from the structural configuration proposed by Gerstenberger et al.[42] to enhance dynamic interactions and coupling effects between the pulsating flow and the flexible structure. The fictitious structure is characterized by the neo-Hookean material parameters listed in Table 5.3.

Initial Density ρ [kg/m ³]	Poisson's Ratio ν []	Modulus of Elasticity E [Pa]
250	0.40	500

Table 5.3: Physical properties of the fictitious neo-Hookean material, as prescribed by Schott et al.[39] [40]

The clamp at the base of the flexible flap constitutes the only physical boundary condition imposed on the structural participant. In addition, the discretization nodes located on the fluid–structure interface are grouped through a `/GRNOD` card in the `_0000.rad` starter file and provided as input to the OpenRadioss `.cpl` configuration file.

5.2.2 Numerical implementation

The constitutive behavior of the flexible flap is implemented in OpenRadioss through the `/MAT/LAW42` material card, based on the neo-Hookean formulation parameters proposed by Schott et al. [39]. Since the `/MAT/LAW42` card requires the material's initial density ρ , Poisson's ratio ν and shear modulus μ as input, the standard elasticity parameters listed in Table 5.3 are converted into the card's expected inputs through the following calculations.

Whereas the material's initial density $\rho = 250 \frac{\text{kg}}{\text{m}^3}$ and Poisson's ratio $\nu = 0.4$ are specified directly in the `/MAT/LAW42` property card, the initial shear modulus μ is derived from the Young's modulus E and ν as:

$$\mu = \frac{E}{2(1 + \nu)} = \frac{500}{2(1 + 0.4)} = \frac{500}{2.8} = 178.5714 \text{ Pa} \quad (5.4)$$

The OpenRadioss implementation of μ follows the formulation established in Eq. (3.10), recalled here for clarity alongside the input parameters μ_p and α_p prescribed by OpenRadioss for the neo-Hookean constitutive model[23] (Eq. (3.12)):

$$\mu = \frac{\sum_{p=1}^{10} \mu_p \cdot \alpha_p}{2} \quad (5.5)$$

$$\alpha_1 = 2, \quad \mu_2 = 0, \quad \alpha_2 = 0 \quad (5.6)$$

The only remaining unknown in the summation defined by Eq.(5.5) is therefore μ_1 , which following the above reasoning evaluates to:

$$\mu = 178.5714 \text{ Pa} = \frac{\mu_1 \cdot \alpha_1}{2} = \frac{\mu_1 \cdot 2}{2} \longrightarrow \mu_1 = 178.5714 \text{ Pa} \quad (5.7)$$

The resulting material card specified for the present configuration is shown in Listing 5.4.

```

1  #--1---|---2---|---3---|---4---|---5---|---6---|---7---|---8---|---9---|--10---|
2  #-  2. MATERIAL DEFINITION:
3  #--1---|---2---|---3---|---4---|---5---|---6---|---7---|---8---|---9---|--10---|
4  /MAT/LAW42/1
5  flapMaterial
6  #      RHO_I
7      250
8  #      Nu      sigma_cut      funIDbulk      Fscale_bulk      M      I_form
9      0.4      0      0      0      0      1
10 #      Mu_1      Mu_2      Mu_3      Mu_4      Mu_5
11      178.5714      0
12 #      Mu_6      Mu_7      Mu_8      Mu_9      Mu_10
13
14 #      alpha_1      alpha_2      alpha_3      alpha_4      alpha_5
15      2      0
16 #      alpha_6      alpha_7      alpha_8      alpha_9      alpha_10
17
18 #--1---|---2---|---3---|---4---|---5---|---6---|---7---|---8---|---9---|--10---|

```

Listing 5.4: `_0000.rad` starter file: `/MAT/LAW42` material card specifying the neo-Hookean constitutive parameters.

Turning to the numerical discretization of the structural participant, two distinct combinations of element type and integration scheme were explored to investigate which best reproduced the dynamic response of the reference deal.II participant. As a first attempt, the flexible flap was discretized using HEX8 elements with a fully integrated formulation as introduced in Section 3.3; since the comparison presented in Section 5.4.1 revealed poor agreement with the reference benchmark, likely due to volumetric locking in the discretization elements, these results demonstrated that the formulation was inadequate for replicating the reference behavior and validating the established coupling. By adopting HEX20 elements instead, thereby increasing the element order and enabling an accurate representation of bending and curved geometries while retaining the same number of discretization cells, combined with a variable number of integration points to mitigate locking phenomena, a substantial improvement in the results was achieved, as shown in Figures 5.7 and 5.8. The results and conclusions regarding the distinct structural responses obtained with the two element types are discussed in Section 5.4.1.

The runtime control parameters of the solver configuration are defined in the `_0001.rad` engine file; the simulation duration is consistently set to $t = 30$ s, with a fixed time step of $\Delta t_S = 0.001$ s enforced through the procedure discussed in Listing 4.9, and a write interval of 0.5 s. Finally, the coupling parameters of the structural participant are specified in the `.cpl` configuration file following the setup presented in Listing 4.4, as discussed in Section 4.1.3. The `/GRNOD/2` card defines the group of discretization nodes located on the shared boundary, whose tag ID = 2 is prescribed in the `_0000.rad` starter file to identify the coupling interface.

5.3 preCICE: setup and configuration

The preCICE configuration of the Pulsating Flow over Flexible Flap coupled simulation is established in accordance with the benchmark specification, as illustrated in Figure 5.3.

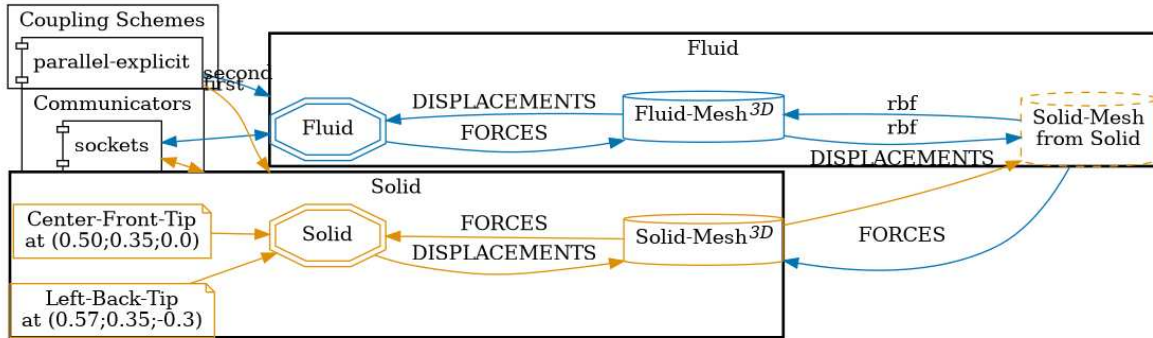


Figure 5.3: Pulsating Flow over Flexible Flap test case: preCICE-config-visualizer.

The configuration reveals that the participant definitions of the Fluid and Solid domains remain identical to those established in Section 4.1.3 for the **Perpendicular Flap** test case, with forces and displacements exchanged at the coupling interface as follows:

- Fluid: write Forces, read Displacements;
- Solid: write Displacements, read Forces.

The parallel-explicit coupling scheme is retained, with Fluid and Solid assigned as first and second participant, respectively. Due to the considerable extent of the interface meshes, data exchanged across non-conforming grids is projected onto the shared boundary via a compactly supported radial-basis-function data mapping to ensure sparse mapping matrices; each RBF mapping is configured to cover 5 to 10 vertices in each radial direction through an appropriately chosen support radius. The corresponding implementation is reported in Listing 5.5.

Two watchpoints are introduced to monitor the flap tip displacement throughout the simulation, constituting the primary difference with respect to the **Perpendicular Flap** configuration in which the linear elastic response was adequately captured by a single monitoring point located at the flap tip center. To more accurately characterize the hyperelastic structural response of the present configuration, the watchpoints configured on the flexible flap are placed at the center of the flap tip on the left edge of the interface, corresponding to $(x, y, z) = (0.5, 0.35, 0.0)$ m, and at the back of the flap tip on the right

edge of the interface, located at $(x, y, z) = (0.57, 0.35, -0.3)$ m. For reference, the two watchpoints are depicted as a red and a blue dot, respectively, in Figure 5.4.

```

1  <participant name="Fluid">
2    <provide-mesh name="Fluid-Mesh" />
3    <receive-mesh name="Solid-Mesh" from="Solid" />
4    <write-data name="FORCES" mesh="Fluid-Mesh" />
5    <read-data name="DISPLACEMENTS" mesh="Fluid-Mesh" />
6    <mapping:rbf direction="write" from="Fluid-Mesh" to="Solid-Mesh"
7      constraint="conservative">
8      <basis-function:compact-polynomial-c6 support-radius="0.065" />
9    </mapping:rbf>
10
11    <mapping:rbf direction="read" from="Solid-Mesh" to="Fluid-Mesh" constraint="consistent">
12      <basis-function:compact-polynomial-c6 support-radius="0.065" />
13    </mapping:rbf>
14    <export:vtu directory="precice-exports" every-n-time-windows="500" />
15  </participant>

```

Listing 5.5: Excerpt from the preCICE XML configuration: RBF mapping configuration for the Medium mesh refinement.

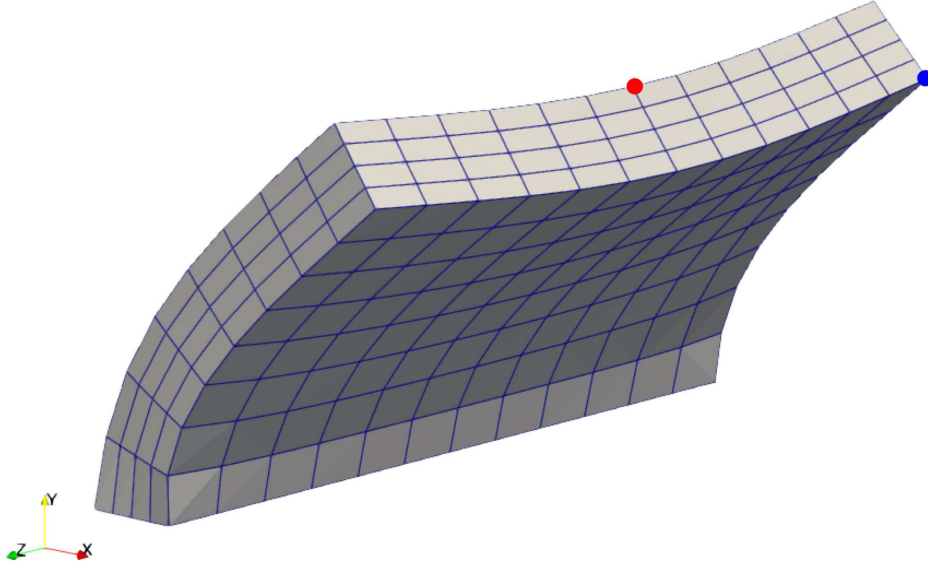


Figure 5.4: Flexible Flap watchpoint locations: Center-Front-Left-Edge (red) and Left-Back-Right-Edge (blue). Post-processed in ParaView.

The corresponding watchpoint implementation is reported in Listing 5.6.

```

1  <participant name="Solid">
2    <provide-mesh name="Solid-Mesh" />
3    <write-data name="DISPLACEMENTS" mesh="Solid-Mesh" />
4    <read-data name="FORCES" mesh="Solid-Mesh" />
5    <export:vtu directory="precice-exports" every-n-time-windows="500" />
6    <watch-point mesh="Solid-Mesh" name="Center-Front-Tip" coordinate="0.50;0.35;0.0" />
7    <watch-point mesh="Solid-Mesh" name="Left-Back-Tip" coordinate="0.57;0.35;-0.3" />
8  </participant>

```

Listing 5.6: Excerpt from the preCICE XML configuration: watchpoint implementation on the OpenRadioss structural participant.

5.4 Results and conclusions

The OpenRadioss participant’s progressive improvement in predictive accuracy, achieved through iterative refinement of the structural discretization, is presented alongside two independent reference solutions serving as benchmarks for the present validation study: the deal.II participant results by Chourdakis et al. [40], reported at a comparable mesh refinement level, and the monolithic results of Schott et al. [39], presented at the finest resolution available (*A2-fine*) of the corresponding unfitted, stationary mesh. The primary metric for this validation study is the displacement time history at the flap tip, recorded along the x -direction at two locations as shown in Figure 5.4: the **Center-Front-Left Edge** and the **Left-Back-Right-Edge**.

A preliminary investigation was conducted on the Coarse mesh to isolate the effect of finite element type on the accuracy of the structural response, and an initial numerical discretization employing fully integrated HEX8 elements was configured. The resulting poor agreement observed with the reference solutions prompted a reassessment of the element type, and a HEX20 configuration was subsequently adopted on the same Coarse mesh to assess whether higher-order elements could improve the accuracy of the solution. Motivated by the encouraging improvement provided by HEX20 elements, the mesh resolution of both participants was increased next to determine if finer meshes further improved the accuracy of the structural response. Since the overall results closely reproduce the reference benchmarks, and the post-processing curves display consistent convergence upon mesh refinement, these findings indicate that the OpenRadioss participant captures the behavior of its deal.II counterpart and the validation study is therefore deemed successful.

5.4.1 Effect of element type: HEX8 and HEX20 discretization

Following the configuration of both the OpenFOAM domain and the **Flexible Flap** participant within OpenRadioss by incorporating the aforementioned physical parameters and numerical settings, the investigation turned to assessing the influence of the discretization formulation on the structural solution to identify which element type yielded the closest agreement with the reference benchmarks.

As a preliminary attempt, the **Flexible Flap** participant was discretized with fully integrated HEX8 elements as introduced in Section 3.3. Since the initial tests were conducted on the Coarse mesh (whose element count is reported in Table 5.1), the structural participant comprised $2 \times 4 \times 6$ first-order elements across the three spatial directions, and the element formulation was specified through the /PROP/SOLID property card implemented in the `_0000.rad` starter file. The corresponding implementation is reported in Listing 5.7, where the fully integrated 8-node solid element formulation, characterized by a total of $2 \times 2 \times 2$ Gauss integration points, is denoted by the flag $I_{\text{solid}} = 18$.

```

1  #--1---|---2---|---3---|---4---|---5---|---6---|---7---|---8---|---9---|--10---|
2  #-  3. PROPERTIES DEFINITION
3  #--1---|---2---|---3---|---4---|---5---|---6---|---7---|---8---|---9---|--10---|
4  /PROP/SOLID/1
5  flapProperty
6  #  Isolid  Ismstr      Iale      Icpres      Itetra10  Inpts  Itetra4  Iframe      Dn
7      18          0          0          0          0          0          0          0          0.0
8  #          qa          qb          h          Lambda      Mu
9          0.0          0.0          0.0          0.0          0.0
10 #      deltaT_min      Vdef_min      Vdef_max      ASP_max      COL_min
11          0.0          0.0          0.0          0.0          0.0
12 #--1---|---2---|---3---|---4---|---5---|---6---|---7---|---8---|---9---|--10---|

```

Listing 5.7: `_0000.rad` starter file: /PROP/SOLID property card specifying the HEX8 solid element formulation.

Once the domain discretization was defined and the boundary conditions had been prescribed on the corresponding structural nodes, the partitioned simulation was executed and completed successfully, demonstrating that the coupling framework operated as intended without introducing numerical instabilities. After post-processing the numerical results, however, the watchpoint time histories revealed that the structural response along the x -direction produced by the first-order HEX8 discretization was physically inconsistent with the reference benchmark data, as shown in Figures 5.5 and 5.6.

Center-Front-Left-Edge watchpoint (HEX8 elements)

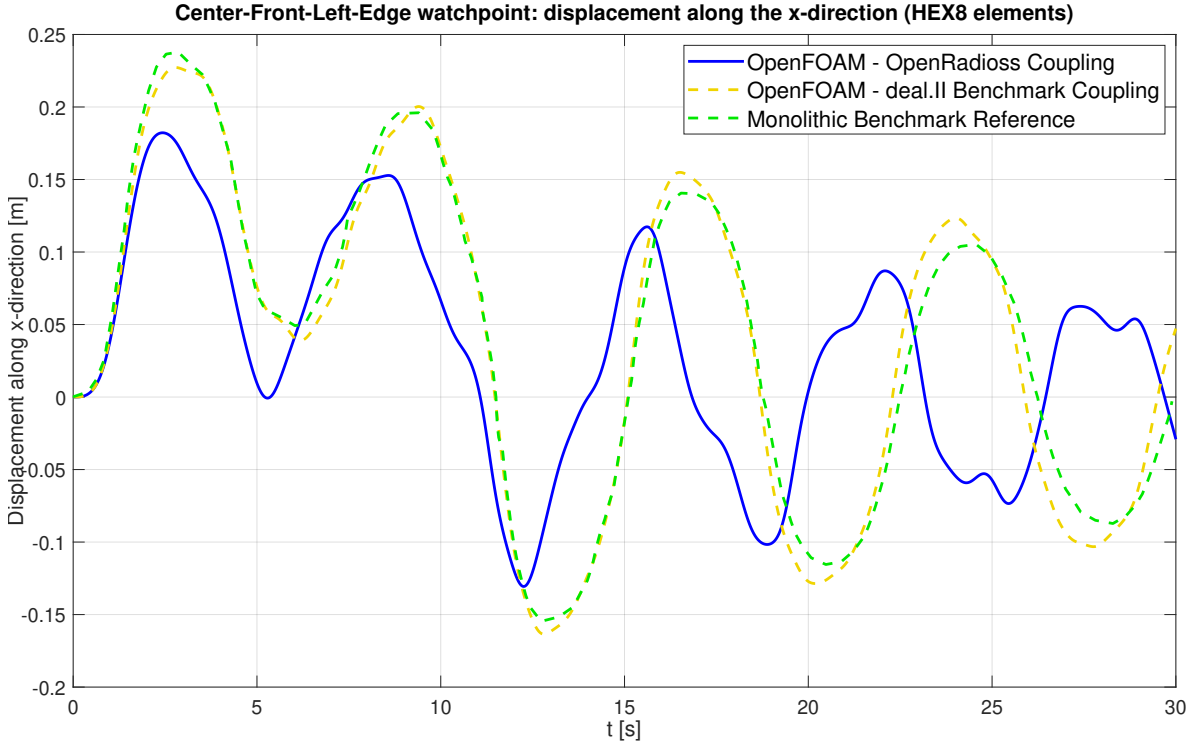


Figure 5.5: **Center-Front-Left-Edge** watchpoint (HEX8 elements): displacement time history along the x -direction. Post-processed in MATLAB.

Although the coupling is successfully established, with inter-solver communication taking place as expected and synchronization not being compromised as a result of the prescribed fixed time step, the poor agreement shown in Figure 5.5 and observed from $t \approx 1.5$ s onward suggests that first-order solid elements fail to reproduce the structural response along the x -direction accurately. At $t = 2.5$ s (i.e. the instant of the first displacement maximum), the displacement predicted by the OpenRadioss participant is significantly lower than the values reported in the reference benchmarks, and that is presumably attributable to the structural over-stiffness induced by volumetric locking. This consequently manifests across the entire structural response, which presents a broadly comparable temporal evolution with respect to the reference periodicity until $t \approx 19$ s, beyond which the cumulative error becomes excessive and the periodic character of the benchmark response is entirely lost.

Left-Back-Right-Edge watchpoint (HEX8 elements)

A similar conclusion is drawn from an analysis of the **Left-Back-Right-Edge** watchpoint time history, shown in Figure 5.6, where the response accuracy along the x -direction is retained until $t \approx 1.7$ s and any comparable temporal evolution is lost beyond $t \approx 12$ s.

To enable a quantitative assessment of the accuracy achieved by the OpenRadioss structural participant, selected displacement data along the x -direction associated with the curves shown in Figures 5.5 and 5.6 are reported below in Table 5.4 and 5.5.

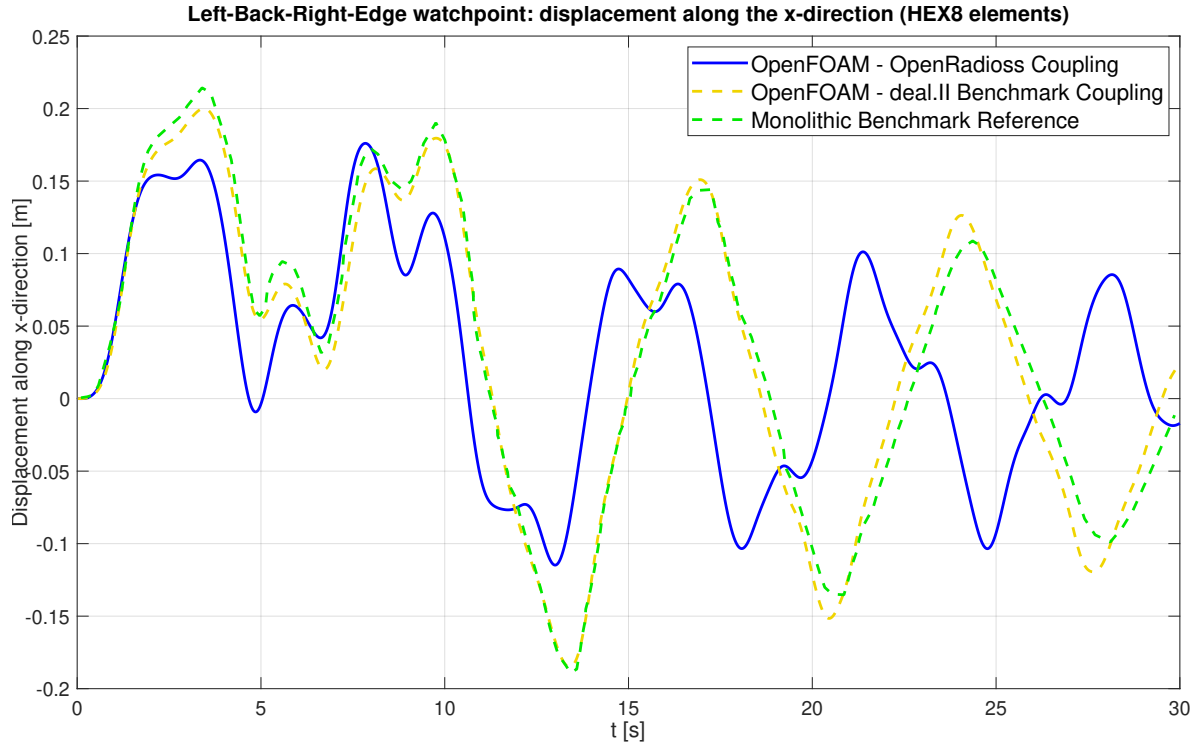


Figure 5.6: Left-Back-Right-Edge watchpoint (HEX8 elements): displacement time history along the x -direction. Post-processed in MATLAB.

	Center-Front-Left-Edge: displacement along x -direction [m]		
Time [s]	OpenRadioss (HEX8 solid elements)	deal.II benchmark	Monolithic benchmark
1.5	0.1123	0.1247	0.1395
3.0	0.1667	0.2258	0.2335
10.0	0.0665	0.1720	0.1623
13.0	-0.0698	-0.1615	-0.1532
16.5	0.0348	0.1549	0.1406
20.0	0.0038	-0.1272	-0.1108
24.0	-0.0543	0.1234	0.1023
28.0	0.0536	-0.1004	-0.0861

Table 5.4: Center-Front-Left-Edge watchpoint: cross-solver displacement comparison for selected time instants. Investigation on the effect of element formulation.

	Left-Back-Right-Edge: displacement along x -direction [m]		
Time [s]	OpenRadioss (HEX8 solid elements)	deal.II benchmark	Monolithic benchmark
1.5	0.1218	0.1202	0.1291
3.0	0.1571	0.1901	0.2022
5.0	-0.0037	0.0546	0.0571
9.0	0.0861	0.1391	0.1499
10.0	0.1109	0.1747	0.1784
14.0	0.0154	-0.1237	-0.1229
20.0	-0.0427	-0.1213	-0.1030
24.0	-0.0368	0.1260	0.1040
28.0	0.0835	-0.1081	-0.0989

Table 5.5: **Left-Back-Right-Edge** watchpoint: cross-solver displacement comparison for selected time instants. Investigation on the effect of element formulation.

By replacing the first-order **HEX8** formulation with **HEX20** solid elements, a substantial improvement was achieved in the accuracy of the structural response. Maintaining the $2 \times 4 \times 6$ element discretization, the structural participant adopted a second-order formulation instead, corresponding to the fully integrated quadratic 20-node solid element characterized by a variable number of integration points to prevent volumetric locking. The corresponding implementation, specified through the `/PROP/SOLID` property card in the `_0000.rad` starter file, is updated by setting $I_{\text{solid}} = 16$, as shown in Listing 5.8.

```

1  #--1---|---2---|---3---|---4---|---5---|---6---|---7---|---8---|---9---|---10---|
2  #-  3. PROPERTIES DEFINITION
3  #--1---|---2---|---3---|---4---|---5---|---6---|---7---|---8---|---9---|---10---|
4  /PROP/SOLID/1
5  flapProperty
6  #  Isolid  Ismstr    Iale    Icpres    Itetra10    Inpts  Itetra4    Iframe    Dn
7      16      0      0      0      0      0      0      0      0.0
8  #          qa          qb          h          Lambda          Mu
9          0.0          0.0          0.0          0.0          0.0
10 #      deltaT_min      Vdef_min      Vdef_max      ASP_max      COL_min
11          0.0          0.0          0.0          0.0          0.0
12 #--1---|---2---|---3---|---4---|---5---|---6---|---7---|---8---|---9---|---10---|

```

Listing 5.8: `_0000.rad` starter file: `/PROP/SOLID` property card specifying the **HEX20** solid element formulation.

Upon updating the element formulation to **HEX20** without modifying any other configuration parameter, the simulation was re-executed and the resulting structural response demonstrated considerable improvement in both physical consistency and temporal evolution accuracy. The post-processed watchpoint time histories are shown in Figures 5.7 and 5.8.

Center-Front-Left-Edge and Left-Back-Right-Edge watchpoints (HEX20 elements)

Inspection of the **Center-Front-Left-Edge** and **Left-Back-Right-Edge** watchpoint time histories, shown in Figures 5.7 and 5.8, reveals that the response accuracy along the x -direction is considerably more consistent with the reference benchmarks, and that both the structural behavior and its periodic character are accurately reproduced. The discrepancies observed with respect to the monolithic reference, particularly beyond $t \approx 24$ s, are attributable to the poor mesh resolution of the OpenRadioss participant, a limitation that will be addressed in the following section. These results nonetheless establish **HEX20** elements as the appropriate choice of element formulation, constituting an encouraging preliminary validation with respect to the quality and integrity of the exchanged data.

To enable direct comparison with the **HEX8** results, Tables 5.6 and 5.7 present the equivalent **HEX20** displacement data at the same time instants as Tables 5.4 and 5.5; a combined displacement plot for each watchpoint is also provided for completeness in Figures 5.9 and 5.10.

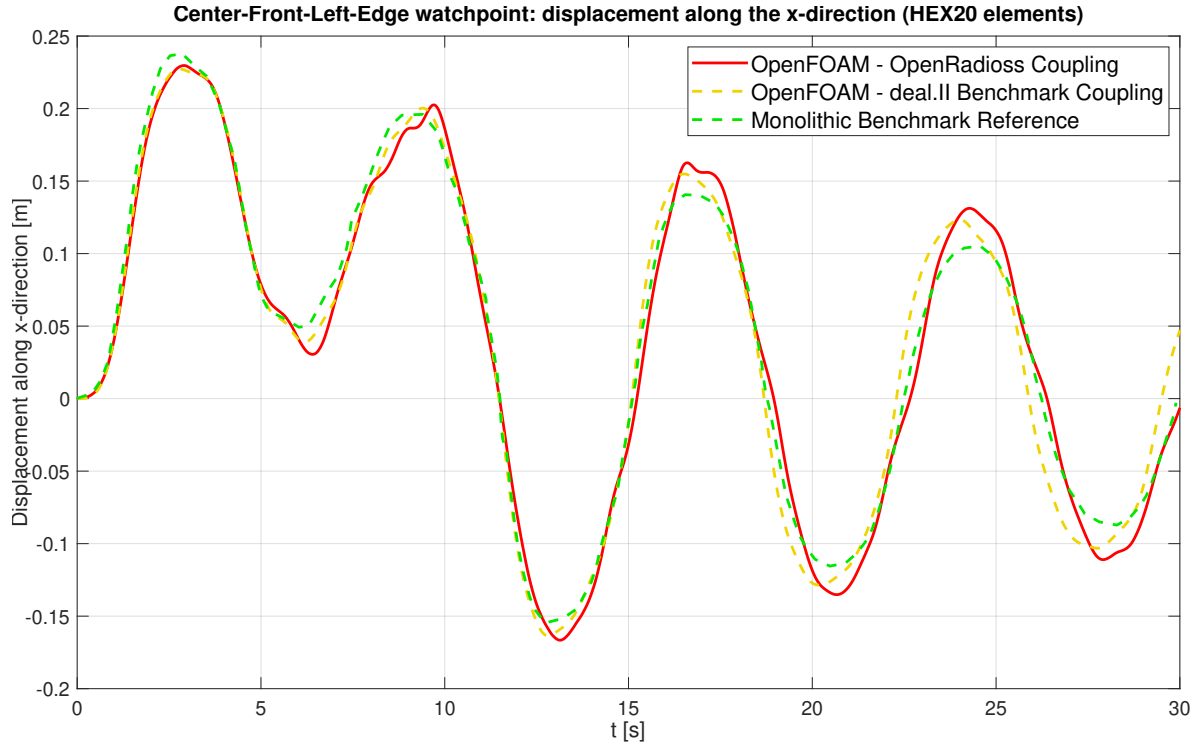


Figure 5.7: Center-Front-Left-Edge watchpoint (HEX20 elements): displacement time history along the x -direction. Post-processed in MATLAB.

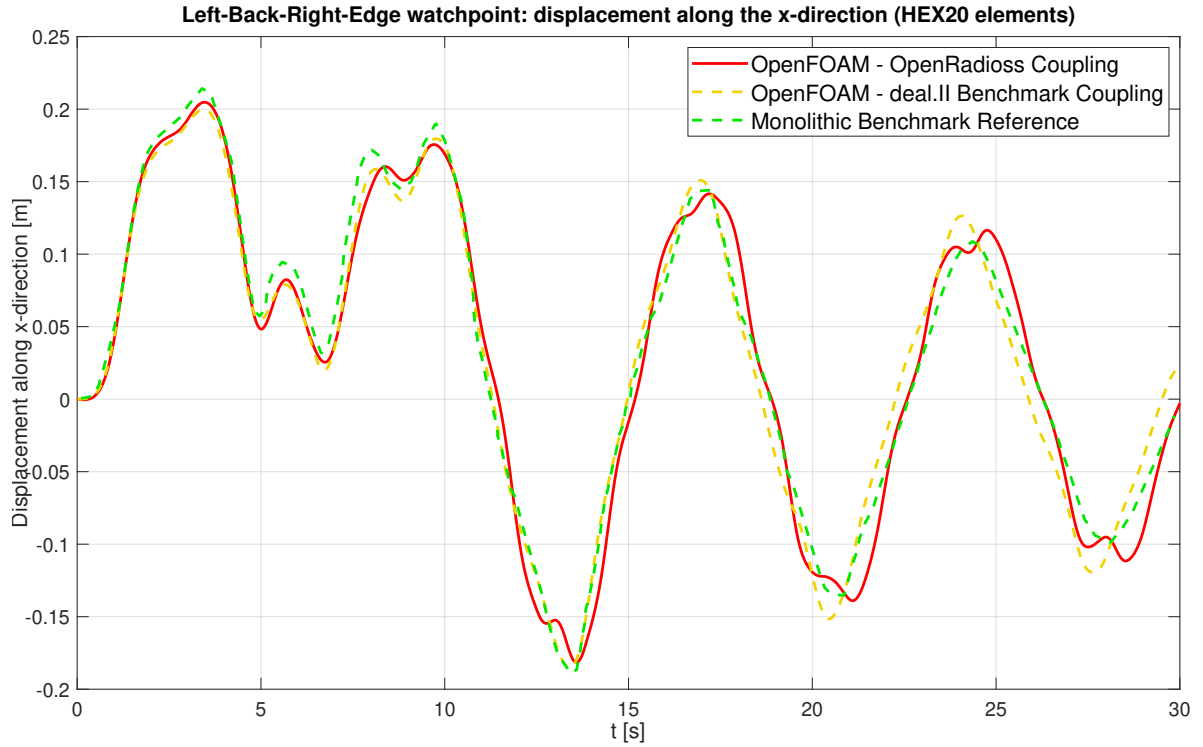


Figure 5.8: Left-Back-Right-Edge watchpoint (HEX20 elements): displacement time history along the x -direction. Post-processed in MATLAB.

Having established HEX20 elements as the appropriate formulation to replicate the mechanical behavior of the deal.II participant within OpenRadioss, the mesh resolution constitutes the next parameter of investigation due to the residual discrepancies observed upon post-processing of the Coarse mesh numerical results. To this end, Section 5.4.2 investigates whether further mesh refinement yields improvements in the accuracy and fidelity of the structural response.

	Center-Front-Left-Edge: displacement along x -direction [m]			
Time [s]	OpenRadioss (HEX8 elements)	OpenRadioss (HEX20 elements)	deal.II benchmark	Monolithic benchmark
1.5	0.1123	0.1199	0.1247	0.1395
3.0	0.1667	0.2290	0.2258	0.2335
10.0	0.0665	0.1861	0.1720	0.1623
13.0	-0.0698	-0.1649	-0.1615	-0.1532
16.5	0.0348	0.1610	0.1549	0.1406
20.0	0.0038	-0.1185	-0.1272	-0.1108
24.0	-0.0543	0.1259	0.1234	0.1023
28.0	0.0536	-0.1105	-0.1004	-0.0861

Table 5.6: **Center-Front-Left-Edge** watchpoint: cross-solver displacement comparison for selected time instants. Investigation on the effect of element formulation (HEX8 and HEX20).

	Left-Back-Right-Edge: displacement along x -direction [m]			
Time [s]	OpenRadioss (HEX8 elements)	OpenRadioss (HEX20 elements)	deal.II benchmark	Monolithic benchmark
1.5	0.1218	0.1222	0.1202	0.1291
3.0	0.1571	0.1926	0.1901	0.2022
5.0	-0.0037	0.0483	0.0546	0.0571
9.0	0.0861	0.1519	0.1391	0.1499
10.0	0.1109	0.1690	0.1747	0.1784
14.0	0.0154	-0.1543	-0.1237	-0.1229
20.0	-0.0427	-0.1192	-0.1213	-0.1030
24.0	-0.0368	0.1044	0.1260	0.1040
28.0	0.0835	-0.0951	-0.1081	-0.0989

Table 5.7: **Left-Back-Right-Edge** watchpoint: cross-solver displacement comparison for selected time instants. Investigation on the effect of element formulation (HEX8 and HEX20).

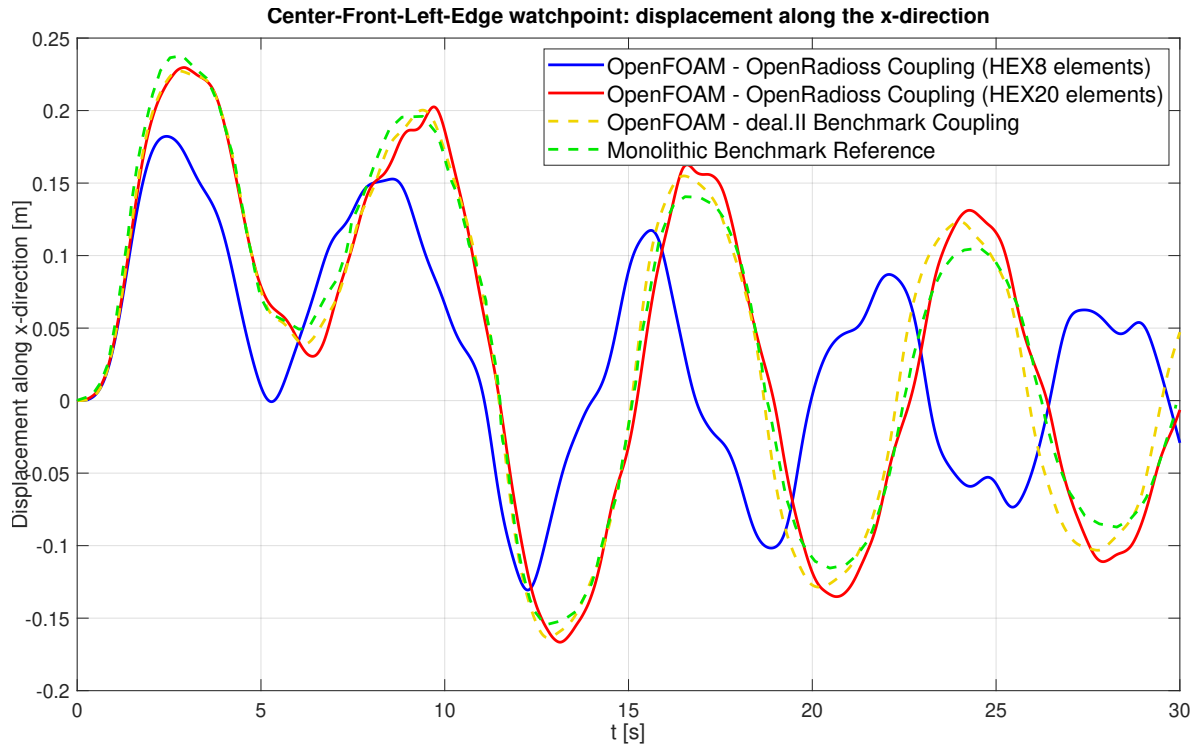


Figure 5.9: Center-Front-Left-Edge watchpoint: displacement time history along the x -direction. Investigation on the effect of element formulation. Post-processed in MATLAB.

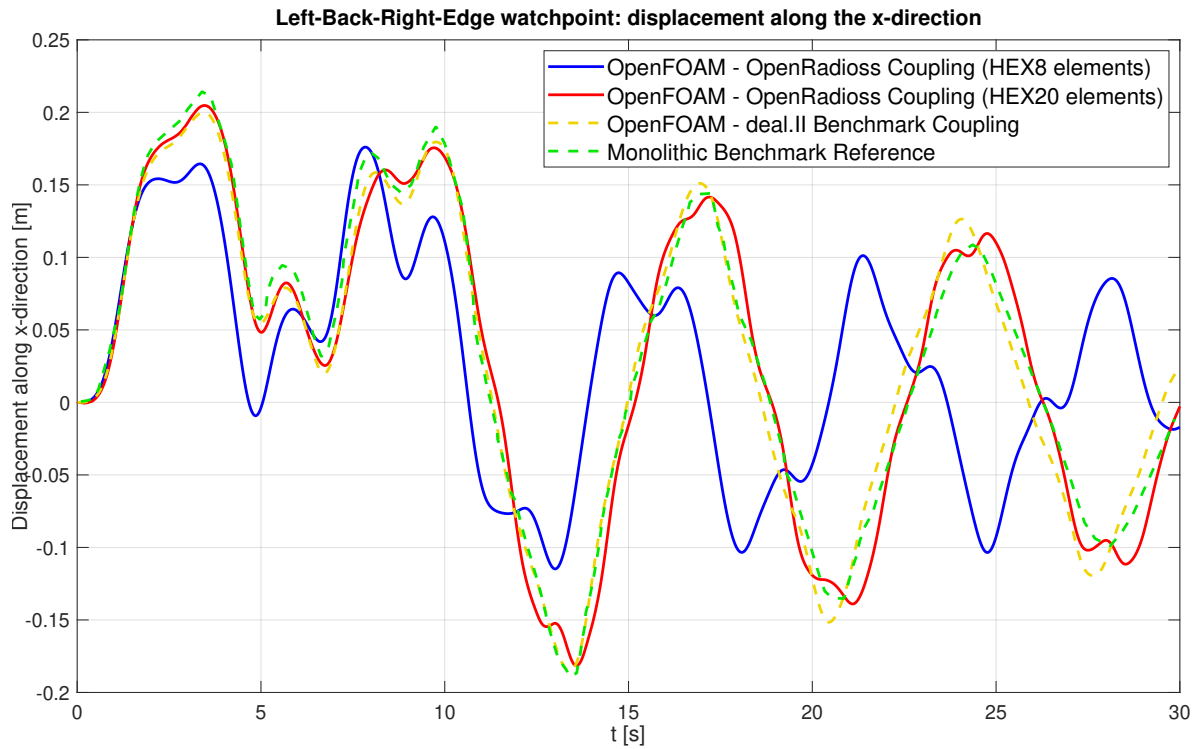


Figure 5.10: Left-Back-Right-Edge watchpoint: displacement time history along the x -direction. Investigation on the effect of element formulation. Post-processed in MATLAB.

5.4.2 Effect of mesh resolution: Coarse and Medium discretization

Building on the validated HEX20 element formulation, to assess the effect of mesh refinement on solution accuracy both the fluid and structural domains were subsequently re-meshed at Medium resolution and the coupled simulation was repeated; element counts for both discretizations are reported in Table 5.1. As evidenced by the **Center-Front-Left-Edge** and **Left-Back-Right-Edge** watchpoint results shown in Figures 5.11 and 5.12, refinement of both participants yielded a substantial improvement in the accuracy of the structural response.

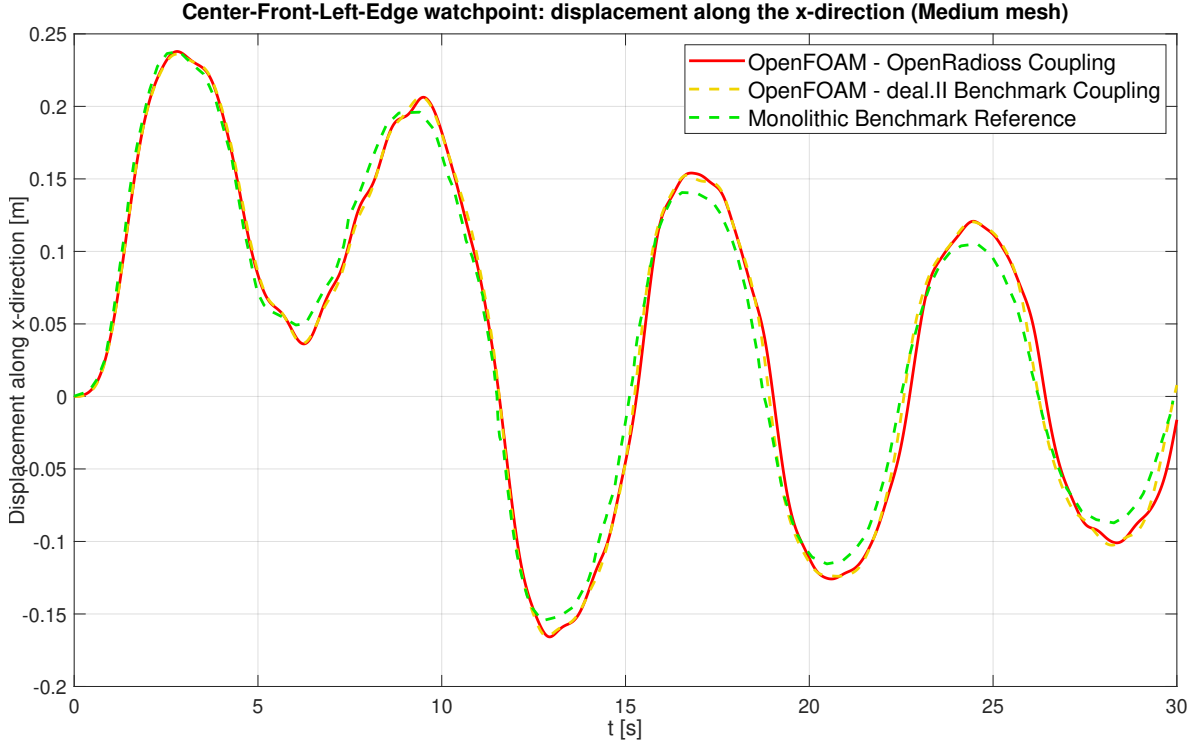


Figure 5.11: **Center-Front-Left-Edge** watchpoint (Medium mesh): displacement time history along the x -direction. Post-processed in MATLAB.

Furthermore, Tables 5.8 and 5.9 enable a quantitative assessment of the mesh resolution effect between the HEX20 Coarse and Medium mesh configurations, and a plot overlaying both displacement responses for each watchpoint is provided in Figures 5.13 and 5.14 for direct visual comparison.

	Center-Front-Left-Edge: displacement along x -direction [m]			
Time [s]	OpenRadioss (Coarse Mesh)	OpenRadioss (Medium Mesh)	deal.II benchmark	Monolithic benchmark
1.5	0.1199	0.1257	0.1247	0.1395
3.0	0.2290	0.2355	0.2258	0.2335
10.0	0.1861	0.1822	0.1720	0.1623
13.0	-0.1649	-0.1653	-0.1615	-0.1532
16.5	0.1610	0.1488	0.1549	0.1406
20.0	-0.1185	-0.1116	-0.1272	-0.1108
24.0	0.1259	0.1093	0.1234	0.1023
28.0	-0.1105	-0.0958	-0.1004	-0.0861

Table 5.8: **Center-Front-Left-Edge** watchpoint: cross-solver displacement comparison for selected time instants. Investigation on the effect of mesh refinement (HEX20 element formulation).

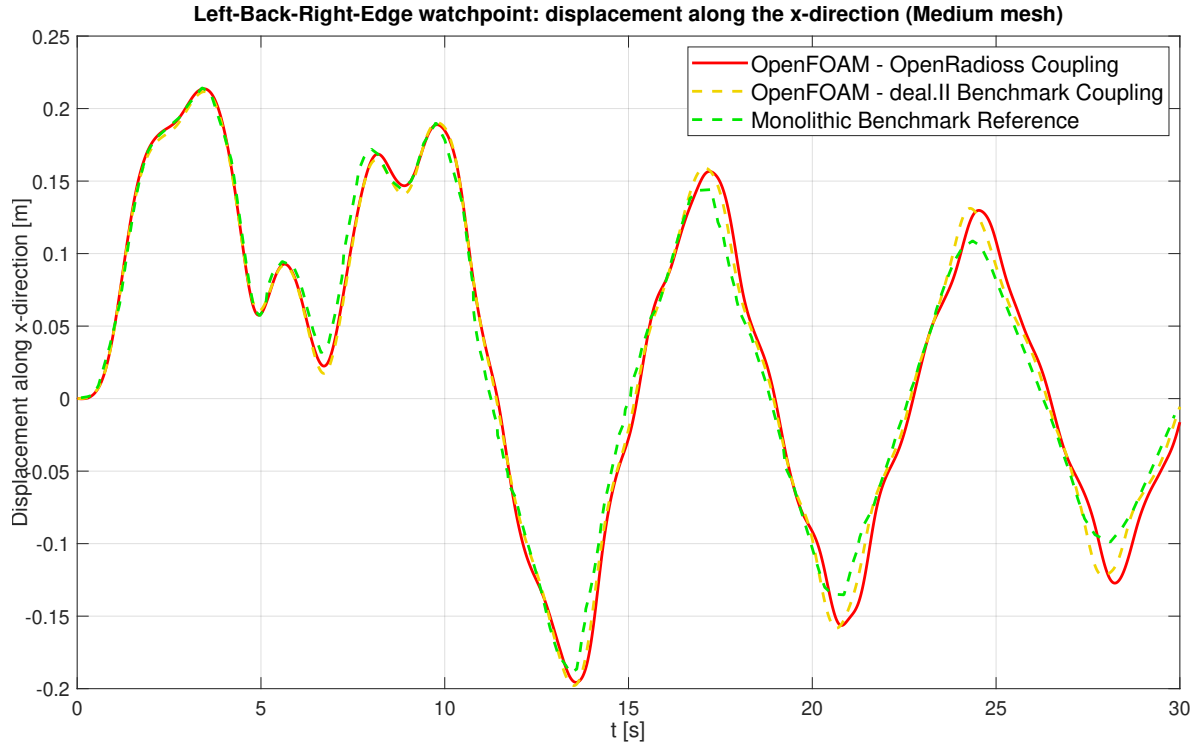


Figure 5.12: Left-Back-Right-Edge watchpoint (Medium mesh): displacement time history along the x -direction. Post-processed in MATLAB.

	Left-Back-Right-Edge: displacement along x -direction [m]			
Time [s]	OpenRadioss (Coarse Mesh)	OpenRadioss (Medium Mesh)	deal.II benchmark	Monolithic benchmark
1.5	0.1222	0.1296	0.1202	0.1291
3.0	0.1926	0.2024	0.1901	0.2022
5.0	0.0483	0.0585	0.0546	0.0571
9.0	0.1519	0.1478	0.1391	0.1499
10.0	0.1690	0.1848	0.1747	0.1784
14.0	-0.1543	-0.1648	-0.1237	-0.1229
20.0	-0.1192	-0.0916	-0.1213	-0.1030
24.0	0.1044	0.0965	0.1260	0.1040
28.0	-0.0951	-0.1185	-0.1081	-0.0989

Table 5.9: Left-Back-Right-Edge watchpoint: cross-solver displacement comparison for selected time instants. Investigation on the effect of mesh refinement (HEX20 element formulation).

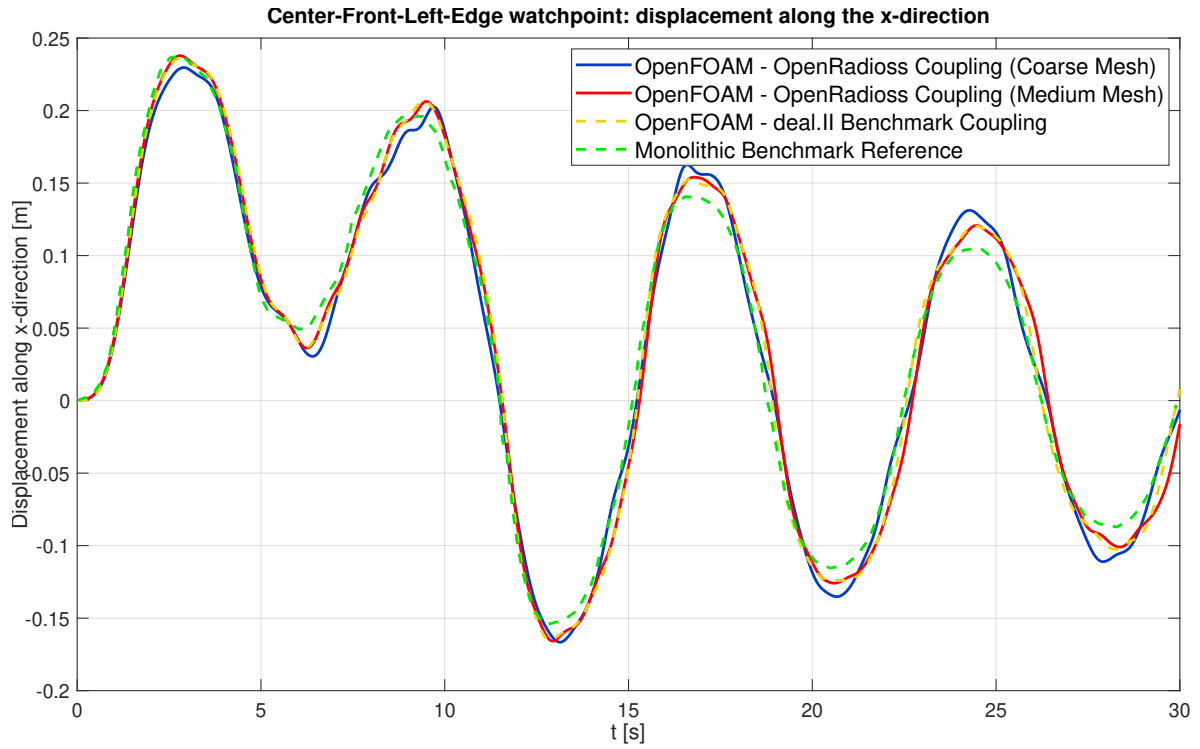


Figure 5.13: Center-Front-Left-Edge watchpoint: displacement time history along the x -direction. Investigation on the effect of mesh refinement (HEX20 element formulation). Post-processed in MATLAB.

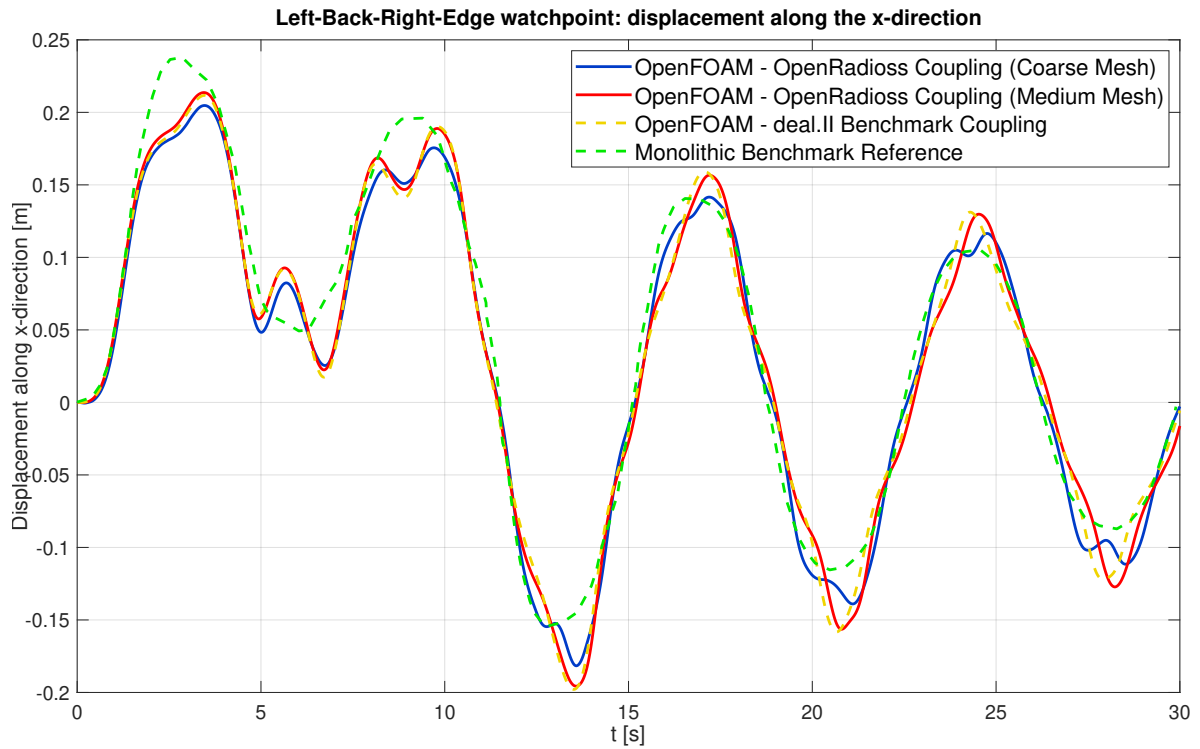


Figure 5.14: Left-Back-Right-Edge watchpoint: displacement time history along the x -direction. Investigation on the effect of mesh refinement (HEX20 element formulation). Post-processed in MATLAB.

The **runtime** overview provided in Tables 5.10 and 5.11 shows that the improvement in accuracy, however, came at the expense of a substantially higher computational cost of the Medium mesh configuration, with an increase of approximately 1180% in computational expense relative to the Coarse mesh execution.

	OpenFOAM ClockTime [s]		OpenFOAM ExecutionTime [s]	
Simulation Time [s]	Coarse Mesh	Medium Mesh	Coarse Mesh	Medium Mesh
0.001	8	16	0.62	6.44
0.002	9	20	0.94	11.24
0.003	9	25	1.30	15.98
0.004	9	30	1.62	20.62
0.005	10	35	1.96	25.35
0.006	10	39	2.30	30.04
0.007	11	44	2.67	34.95
0.008	11	49	2.99	39.73
0.009	11	54	3.35	44.34
0.010	12	58	3.69	49.07

Table 5.10: OpenFOAM participant: computational cost for the first 10 time steps of the coupled simulation. Investigation on the effect of mesh refinement.

	Coarse Mesh	Medium Mesh
Elapsed Time [s]	13 083.65	167 597.85
Elapsed Time [hh:mm:ss]	3:38:03	46:33:17
Computational cost increase [%]		1180 %

Table 5.11: OpenRadioss participant: total computational cost of the coupled simulation. Investigation on the effect of mesh refinement.

Nevertheless, the combination of the **HEX20** solid element formulation and Medium mesh refinement enabled validation of the established coupling with minimal residual error, as the reference structural response presented by the deal.II participant was successfully reproduced by OpenRadioss. To illustrate the interaction between the fluid and structural domains, a selection of the resulting datasets post-processed in ParaView is provided in Figures 5.15, 5.16, 5.17 and 5.18 to depict the best-performing configuration.

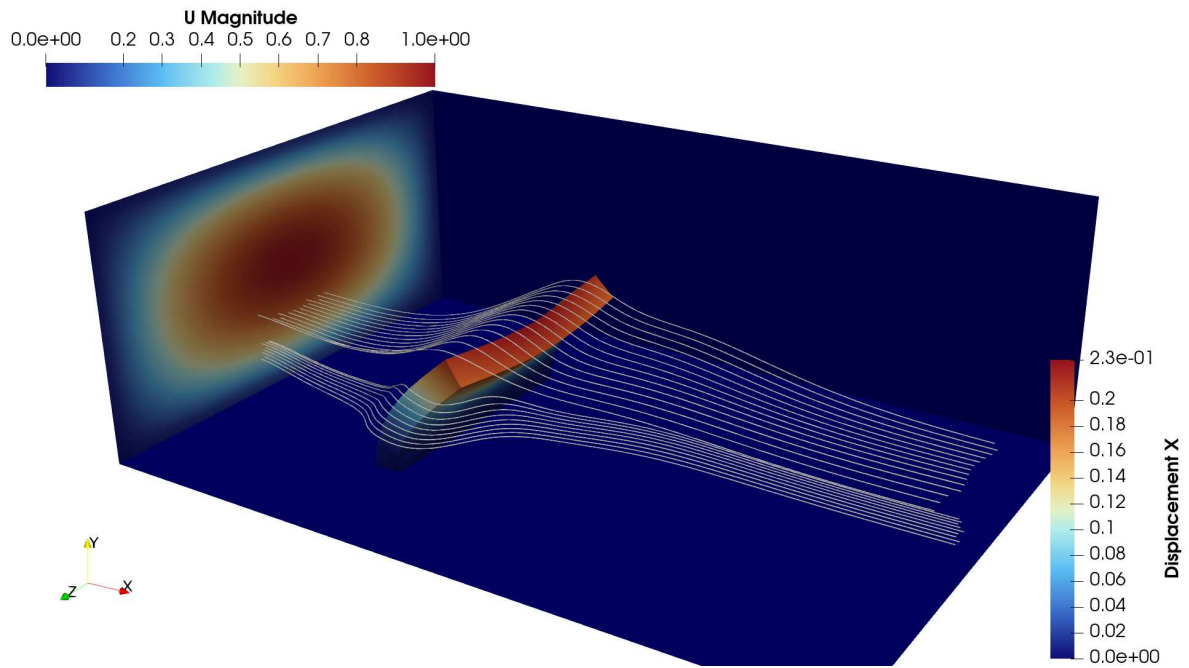


Figure 5.15: Pulsating Flow over Flexible Flap: 3D simulation setup at $t = 2.5$ s. HEX20 solid element formulation, Medium mesh refinement. Post-processed in ParaView.

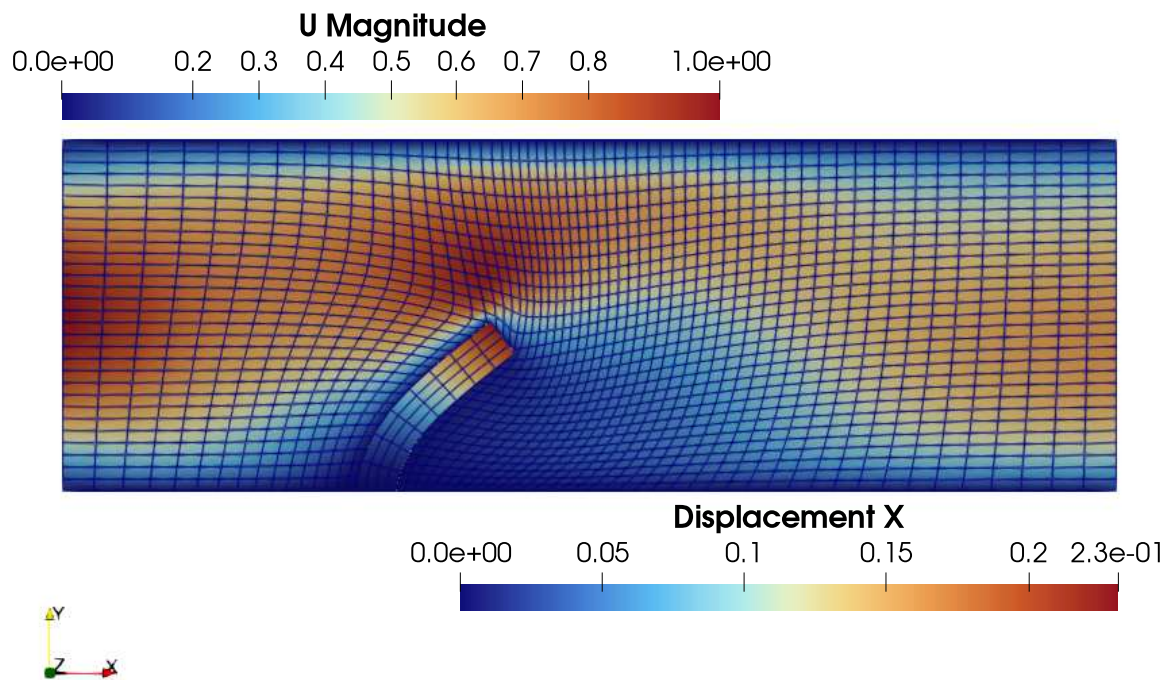


Figure 5.16: Pulsating Flow over Flexible Flap: 2D cross section at $z = 0$ m and $t = 2.5$ s. HEX20 solid element formulation, Medium mesh refinement. Post-processed in ParaView.

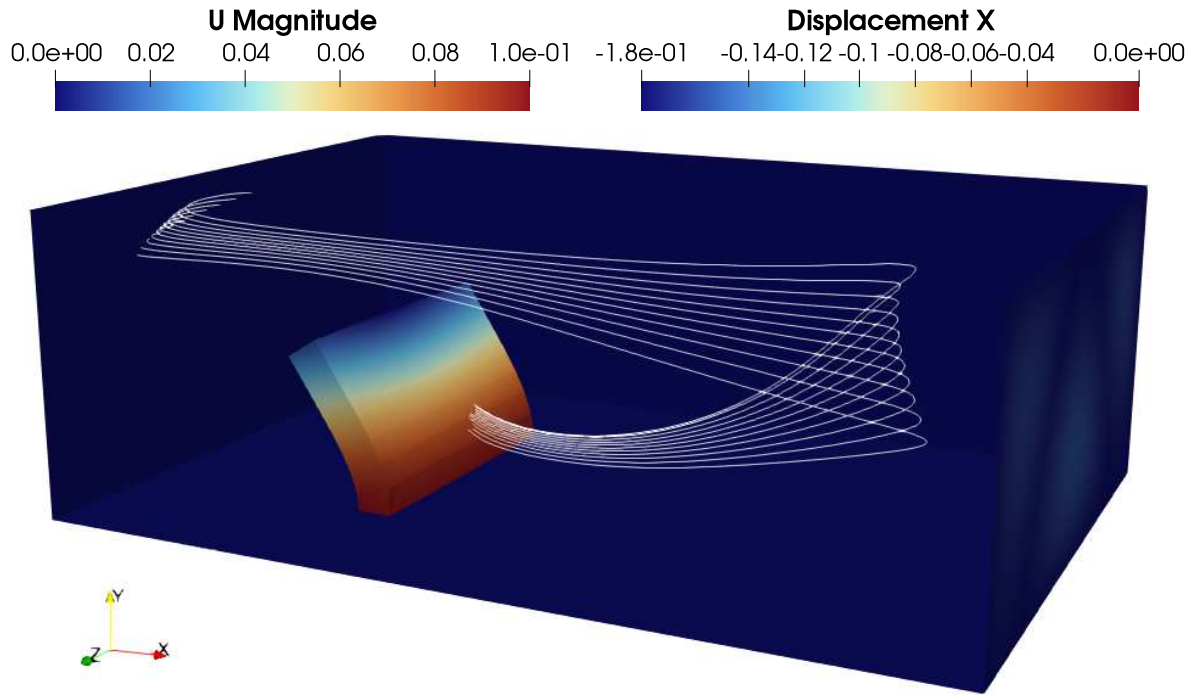


Figure 5.17: Pulsating Flow over Flexible Flap: 3D simulation setup at $t = 13$ s. HEX20 solid element formulation, Medium mesh refinement. Post-processed in ParaView.

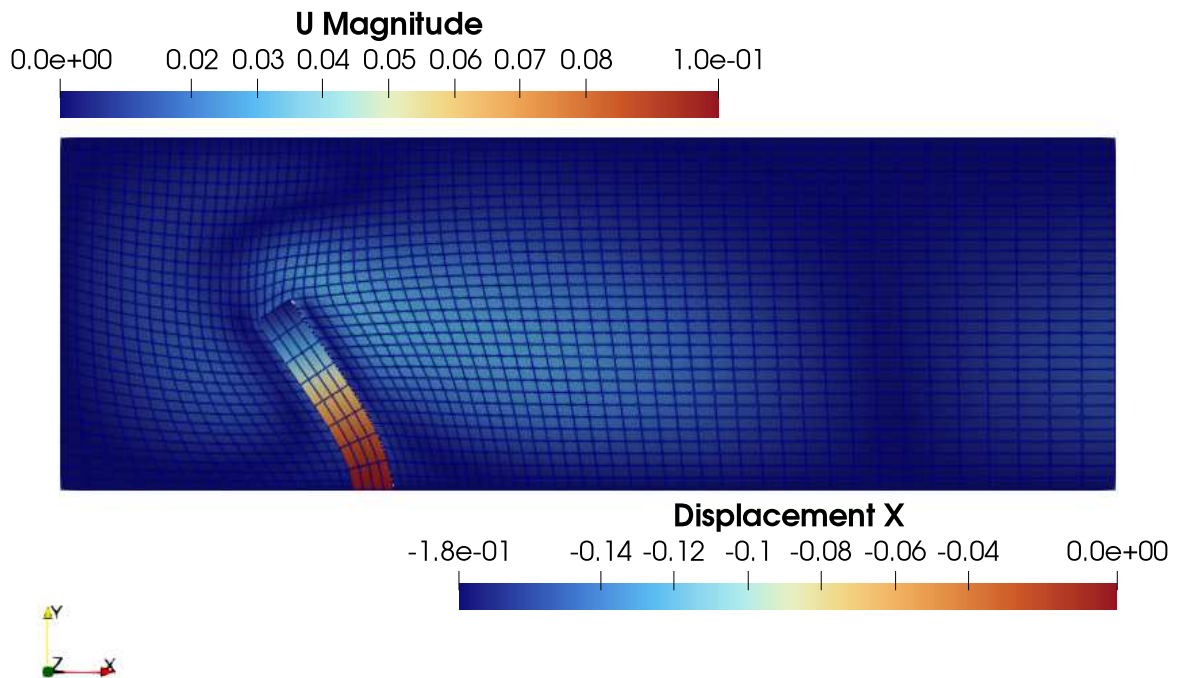


Figure 5.18: Pulsating Flow over Flexible Flap: 2D cross section at $z = 0$ m and $t = 13$ s. HEX20 solid element formulation, Medium mesh refinement. Post-processed in ParaView.

Chapter 6

Numerical investigation of fluid-structure interactions under impulsive blast loading: Blast Wave Impact against a Structural Wall

The aim of this chapter is to investigate a novel fluid-structure interaction setup for which no validation baseline currently exists, to characterize the undocumented coupled behavior of a blast wave as it propagates through the fluid domain and impacts a structural wall. The case consists of a pseudo two-dimensional computational domain, discretized with three cells along the depth (z) direction in both subdomains to accommodate the absence of two-dimensional coupling support in the current version of the OpenRadioss adapter. A hyperelastic structural wall, clamped at its base, is immersed in a laminar flow field through which the shock front of a blast wave propagates before impinging upon the wall. The simulation runtime is set to $t = 0.03$ s. Full details of the simulation setup, including the OpenFOAM, OpenRadioss, and preCICE configurations, are provided in [43] and [44].

Within the fluid domain, a shock wave propagates through the flow field generating sharp pressure discontinuities; it is simulated using the `rhoPimpleFoam` solver, whose numerical settings are described in detail in Section 6.1.2. The structural participant, in turn, consists of a 0.1 m thin flap composed of an incompressible non-linear Mooney-Rivlin material, following the formulation of Villani et al. [45] and Lu et al. [46] with its constitutive behavior modeled in OpenRadioss via the `/MAT/LAW42` material card.

The test case has been simulated at two shock wave intensity levels, referred to as **low-intensity** and **high-intensity**, whose initial conditions are listed in Table 6.1. Given the absence of a validation baseline, a conservative preliminary configuration was adopted for the **low-intensity** case, featuring a thicker structural wall, so the effect of wall thickness on the coupled system response is investigated first in Section 6.4.2, prior to the assessment of shock wave intensity effects discussed in Section ??.

Shock Intensity Level	Blast Wave Intensity [Pa]
Low	150 000
High	300 000

Table 6.1: Blast wave intensity levels of the coupled simulations.

The coupled response was characterized by monitoring the flap displacement along the x -direction at three sampling locations on the flap height: the **Bottom-Left-Edge** ($x = 7.4$ m, $y = 0.6$ m, $z = 0.2$ m), the **Mid-Left-Edge** ($x = 7.4$ m, $y = 2.0$ m, $z = 0.2$ m), and the **Tip-Left-Edge** ($x = 7.4$ m, $y = 4.0$ m, $z = 0.2$ m) watchpoints defined on the OpenRadioss structural mesh.

The subsequent sections provide a detailed description of the implementation of each participant and the coupling configuration, as well as a thorough discussion of the results obtained and the conclusions drawn therefrom.

6.1 The OpenFOAM participant: setup and configuration

6.1.1 Physical properties

The fluid participant consists of a pseudo two-dimensional OpenFOAM domain, with dimensions $\Omega_F = [0, 10] \times [0, 6] \times [0, 0.2]$ m and comprising $200 \times 120 \times 3$ discretization elements across the three spatial directions, through which a blast wave propagates and generates sharp pressure discontinuities.

The pressure field is initialized as a Sod shock tube, in which the left subdomain ($\Omega_{\text{High_Pressure}} = [0, 2] \times [0, 6] \times [0, 0.2]$ m) is set to the overpressure defined in Table 6.1 for the prescribed shock intensity level, while the right subdomain ($\Omega_{\text{Low_Pressure}} = (2, 10] \times [0, 6] \times [0, 0.2]$ m) is set to atmospheric pressure

($p_{\text{atm}} = 101\,325\text{ Pa}$). The resulting pressure distribution at $t = 0\text{ s}$ is illustrated in Figure 6.1.

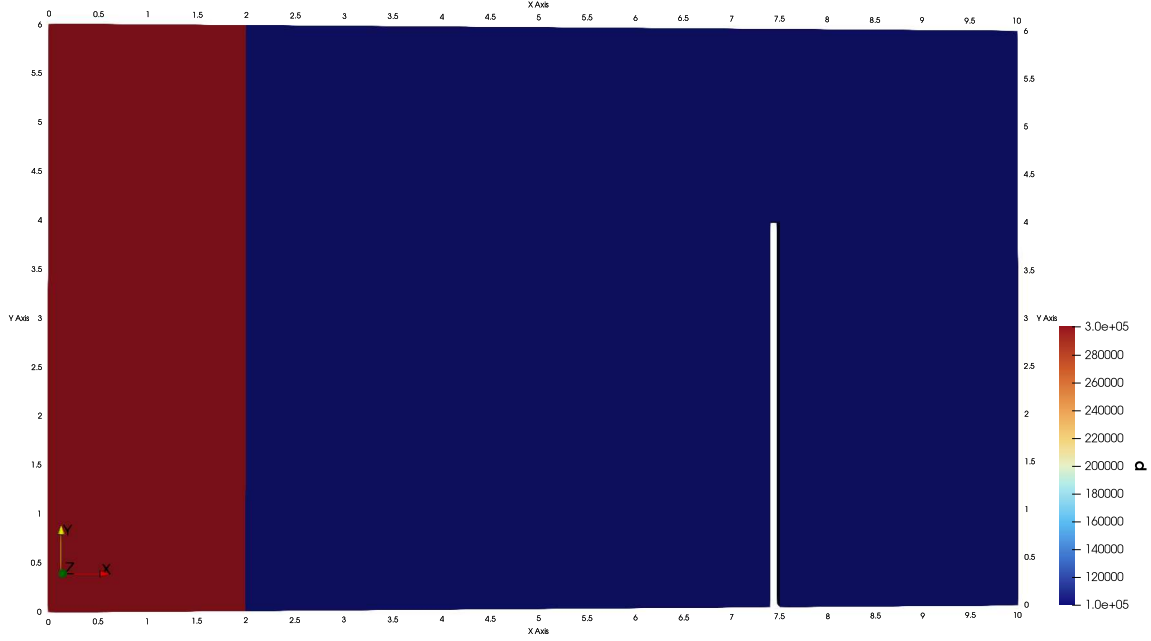


Figure 6.1: Pressure distribution at $t = 0\text{ s}$, illustrating the Sod shock tube initial condition for the **high-intensity** shock level. Post-processed in ParaView.

The fluid is modeled as a compressible, laminar Newtonian gas following the perfect gas assumption, with thermophysical properties corresponding to those of air under standard atmospheric conditions; the adopted values are listed in Table 6.2. Consistently with the atmospheric reference state, the $\Omega_{\text{Low_Pressure}}$ subdomain is initialized with a uniform temperature of $T = 300\text{ K}$.

ThermoType Model	hePsiThermo
Molar Mass M [g/mol]	28.9
Heat Capacity C_v [J/(kg · K)]	712
Reference Enthalpy H_f [J/kg]	0
Dynamic Viscosity μ [Pa · s]	1.8×10^{-5}
Prandtl Number Pr []	0.7

Table 6.2: Physical properties of air under the perfect gas assumption.

To ensure thermodynamic consistency across the high-pressure subdomain, the temperature field within $\Omega_{\text{High_Pressure}}$ is also initialized to the post-shock temperature derived from the normal shock relations introduced in Eq.(2.46) and Eq.(2.49). Specifically, the temperature ratio across the field discontinuity is computed as a function of the shock wave Mach number, which is in turn determined from the pressure ratio set by the shock intensity level. For completeness, Eq.(2.46) and Eq.(2.49) are recalled below for reference, while further details of the temperature initialization are provided in Section 6.1.2.

$$\frac{p_{\text{atm}}}{p_{\text{blast}}} = \frac{2\gamma M_1^2 - (\gamma - 1)}{(\gamma + 1)} \rightarrow M_1 = \sqrt{\frac{1}{2\gamma} \left[\frac{p_{\text{atm}}}{p_{\text{blast}}} (\gamma + 1) + (\gamma - 1) \right]} \quad (6.1)$$

$$\frac{T_{\text{atm}}}{T_{\text{blast}}} = 1 + \left[\frac{2(\gamma - 1)(\gamma M_1^2 + 1)(M_1^2 - 1)}{(\gamma + 1)^2 M_1^2} \right] \rightarrow T_{\text{blast}} = T_{\text{atm}} \cdot \left\{ 1 + \left[\frac{2(\gamma - 1)(\gamma M_1^2 + 1)(M_1^2 - 1)}{(\gamma + 1)^2 M_1^2} \right] \right\}^{-1} \quad (6.2)$$

Shock Intensity Level	Blast Wave Mach Number M_1 []	Blast Wave Upstream Temperature T_{blast} [K]
Low	1.19	336
High	1.64	425

Table 6.3: Blast wave intensity levels of the coupled simulations.

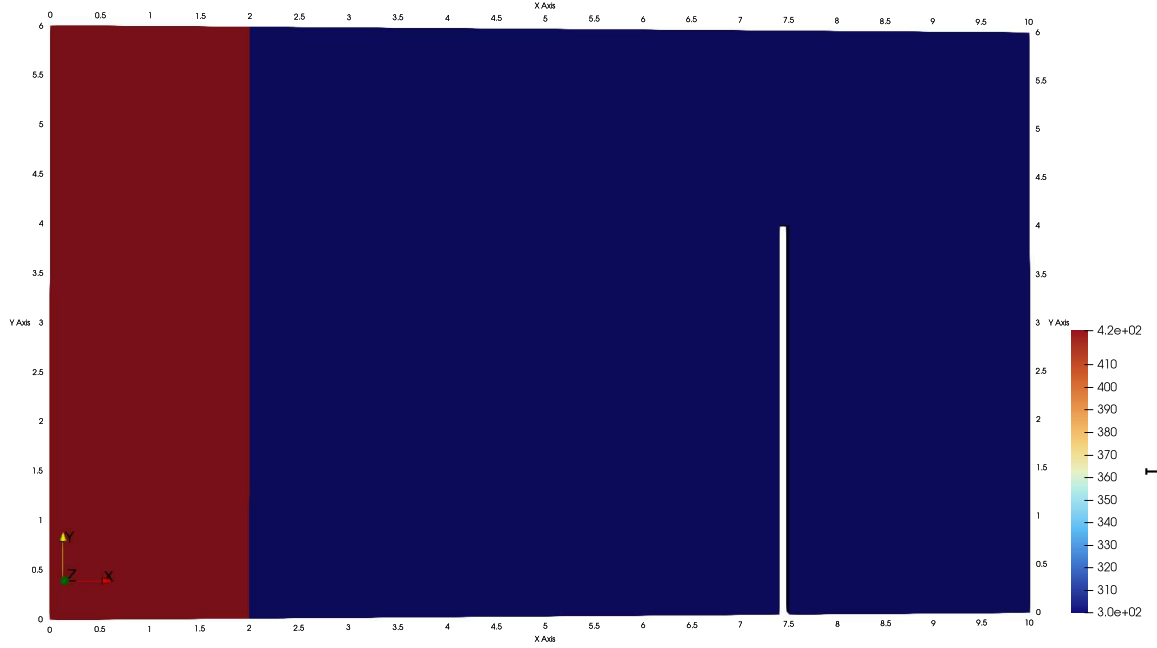


Figure 6.2: Temperature distribution at $t = 0$ s, illustrating the Sod shock tube initial condition for the **high-intensity** shock level. Post-processed in ParaView.

Finally, to accommodate the mesh deformation induced by the structural participant, a `movingWallVelocity` condition is prescribed at the shared boundary (i.e. the wall interface) and the dynamic mesh motion is configured following the same implementation discussed in Listing 5.2.

6.1.2 Numerical implementation

The numerical implementation of the OpenFOAM participant employs the `rhoPimpleFoam` solver, which is designed for transient simulations of compressible flows across both laminar and turbulent regimes undergoing temperature and density variations. As a pressure-based solver, `rhoPimpleFoam` relies on the PIMPLE algorithm to resolve pressure-momentum coupling, as described in Section 2.3.3, while compressibility effects are accounted for through the solution of the energy equation. Crucially for the present configuration, `rhoPimpleFoam` natively supports dynamic mesh morphing, extending its applicability to moving boundary problems. Its default central differencing schemes, however, lack the upwind bias required both to stabilize the sharp pressure gradients across the shock front and to resolve the shock thickness at the cell scale; as a result, the insufficient numerical dissipation fails to suppress spurious oscillations, as evidenced by the minor numerical instabilities reported in Section 6.1.3. Nevertheless, since dedicated shock-capturing solvers such as the density-based `rhoCentralFoam` do not support dynamic mesh morphing, `rhoPimpleFoam` remains the most appropriate compromise for the present configuration despite its shock-capturing limitations.

Consistently with the configuration outlined in Chapter 5, the boundary and initial conditions of the fluid domain are specified within the 0 time directory, which contains, among other input files, the mandatory pressure (0/p) and velocity (0/U) fields required by the pressure-based solver. The pressure field is assigned a `zeroGradient` condition at all boundary faces, while a no-slip condition is applied to the velocity field at all solid boundaries with the exception of the fluid-structure interface, which is assigned a `movingWallVelocity` condition to account for the mesh deformation induced

by the structural participant. Given the compressible nature of the flow under investigation, the temperature field must also be initialized to complete the thermodynamic configuration. This is accomplished through the `O/T` input file, in which the fluid domain is assigned a uniform temperature of $T = 300$ K. Finally, to enable dynamic mesh morphing in response to the structural deformation, a `O/pointDisplacement` field is initialized to track the displacement vector $\mathbf{d}$ of each interface node consistently with the approach adopted for the **Pulsating Flow** domain in Section 5.1.2.

The fluid physical parameters and flow regime definition are specified in the `constant` directory through the `thermophysicalProperties` and `turbulenceProperties` files, respectively, following the conditions outlined in Section 6.1.1. Within this framework, the `hePsiThermo` thermoType model governs the thermodynamic state of the domain through the compressibility field ψ , employed as the primary thermodynamic variable, from which density is recovered as $\rho = \psi p$. Air is modeled under the perfect gas assumption as a compressible, non-reacting pure mixture, as reported in Listing 6.1. The `thermo` entry is set to `eConst`, which assumes a constant specific heat at constant volume C_v , and the energy equation is solved accordingly in terms of sensible internal energy as specified by the `sensibleInternalEnergy` entry. The mixture thermophysical properties, selected consistently with the standard atmospheric conditions established in Section 6.1.1, are those reported in Table 6.2.

```

1 thermoType
2 {
3     type            hePsiThermo;
4     mixture         pureMixture;
5     transport       const;
6     thermo          eConst;
7     equationOfState perfectGas;
8     specie          specie;
9     energy          sensibleInternalEnergy;
10 }
```

Listing 6.1: `constant/thermophysicalProperties`: configuration of the perfect gas transport model for air.

Finally, dynamic mesh morphing is configured in the `constant/dynamicMeshDict` dictionary following the same implementation described for the **Pulsating Flow over Flexible Flap** case, as detailed in Section 5.1.2 through Listing 5.2.

The `system` directory completes the case configuration by hosting the discretization schemes, algorithm settings and tolerances, runtime control parameters, and the $\Omega_{\text{High Pressure}}$ subdomain initialization which constitutes the sole addition with respect to the **Pulsating Flow** case. The `system/fvSchemes` file specifies the discretization schemes for the simulation. Time derivatives are discretized with a second-order implicit Crank-Nicolson scheme; gradients employ a multi-directional cell-limited formulation (`cellMDLimited`) to constrain gradient magnitudes and reduce spurious oscillations, while divergence terms are discretized through a cell-limited Gauss linear scheme to maintain boundedness. Finally, Laplacian terms and surface-normal gradients are linearly interpolated with an explicit non-orthogonal correction to preserve accuracy on non-orthogonal meshes. Following the implementation described in the previous chapter, the PIMPLE loop configured in the `system/fvSolution` file is set to execute three PIMPLE pressure corrections (`nCorrectors`), five SIMPLE corrections (`nOuterCorrectors`) and three non-orthogonal corrector iterations (`nNonOrthogonalCorrectors`) to account for large mesh distortions. Runtime control is specified in the `system/controlDict` dictionary, which sets the simulation duration to $t = 0.03$ s = $3 \cdot 10^{-2}$ s, a time step of $\Delta t_F = 0.00001$ s = $1 \cdot 10^{-5}$ s, and a write interval of 0.001 s = $1 \cdot 10^{-3}$ s. In addition, the dictionary configures the calls to the preCICE library and registers the preCICE adapter as an OpenFOAM function object as reported in Listing 5.3.

The $\Omega_{\text{High Pressure}}$ subdomain is initialized through the `system/setFieldsDict` dictionary, which enables the prescription of non-uniform initial conditions within specified geometric regions by defining default field values across the entire domain and selectively overwriting them within the targeted areas. In the present case, a `boxToCell` condition prescribes an overpressure and an elevated temperature within the region defined by $(x, y, z) \in [0, 2] \times [0, 6] \times [0, 0.2]$. The corresponding implementation is reported in Listings 6.2 and 6.3.

```

1 defaultFieldValues
2 (
3     volScalarFieldValue p 101325
4     volScalarFieldValue T 300
5 );
6
7 regions
8 (
9     boxToCell
10    {
11        min (0 0 0);
12
13        max (2 6 0.2);
14
15        fieldValues
16        (
17            volScalarFieldValue p 150000
18            volScalarFieldValue T 336
19        );
20    }
21 );

```

Listing 6.2: `system/setFieldsDict`: configuration of the non-uniform initial conditions within the $\Omega_{\text{High_Pressure}}$ subdomain for the low-intensity shock level.

```

1 defaultFieldValues
2 (
3     volScalarFieldValue p 101325
4     volScalarFieldValue T 300
5 );
6
7 regions
8 (
9     boxToCell
10    {
11        min (0 0 0);
12
13        max (2 6 0.2);
14
15        fieldValues
16        (
17            volScalarFieldValue p 300000
18            volScalarFieldValue T 425
19        );
20    }
21 );

```

Listing 6.3: `system/setFieldsDict`: configuration of the non-uniform initial conditions within the $\Omega_{\text{High_Pressure}}$ subdomain for the high-intensity shock level.

Finally, the coupling parameters of the fluid participant are specified in the `system/preciceDict` dictionary, following the configuration presented in Listing 4.3 and discussed in Section 4.1.3.

6.1.3 Assessment of the uncoupled fluid domain

Since no established validation baseline exists for the present fluid domain configuration, all simulations whose coupled results are discussed in Section 6.4 were first run as standalone OpenFOAM cases by suppressing all calls to the preCICE library; these decoupled executions served to assess the quality of the implementation and provide a comparative reference for the fluid behavior. A selection of the resulting datasets, post-processed in ParaView, illustrates the corresponding flow field as shown in Figures 6.1 through 6.18.

Low-intensity shock level

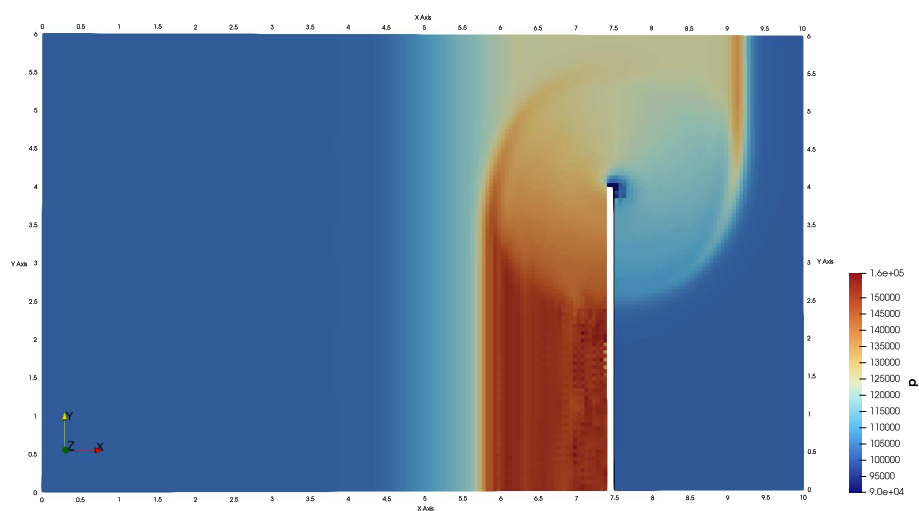


Figure 6.3: Low-intensity shock level: pressure distribution at $t = 0.2$ s. Post-processed in ParaView.

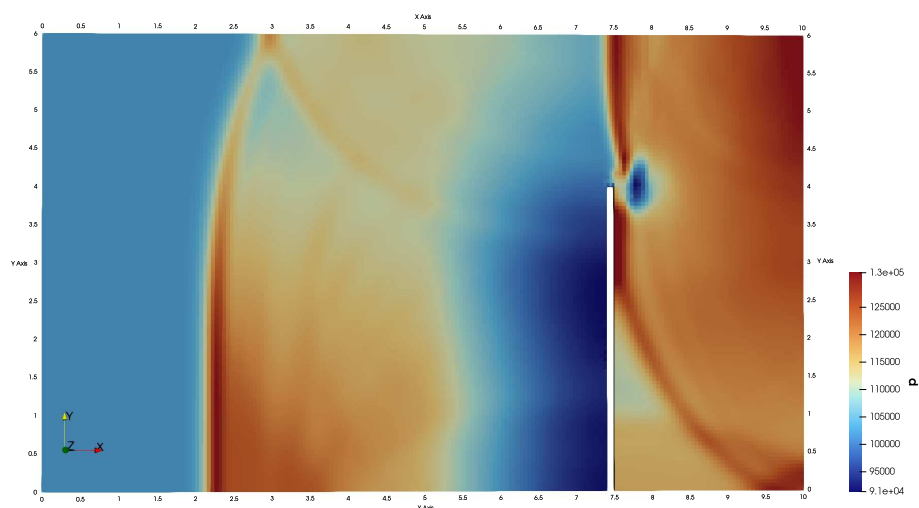


Figure 6.4: Low-intensity shock level: pressure distribution at $t = 0.3$ s. Post-processed in ParaView.

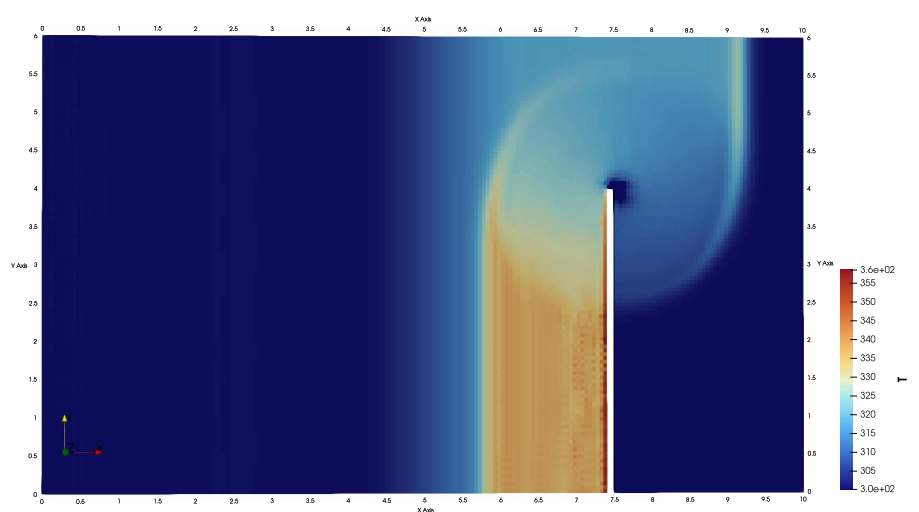


Figure 6.5: Low-intensity shock level: temperature distribution at $t = 0.2$ s. Post-processed in ParaView.

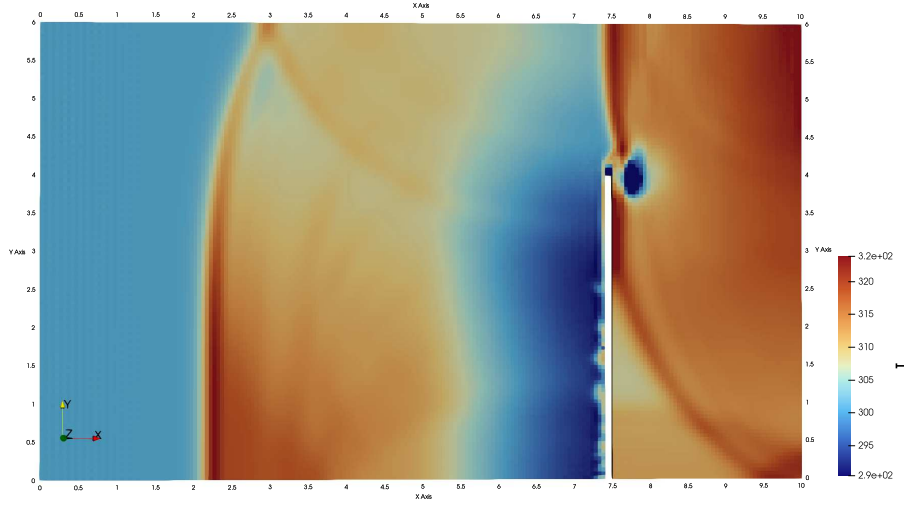


Figure 6.6: Low-intensity shock level: temperature distribution at $t = 0.3$ s. Post-processed in ParaView.

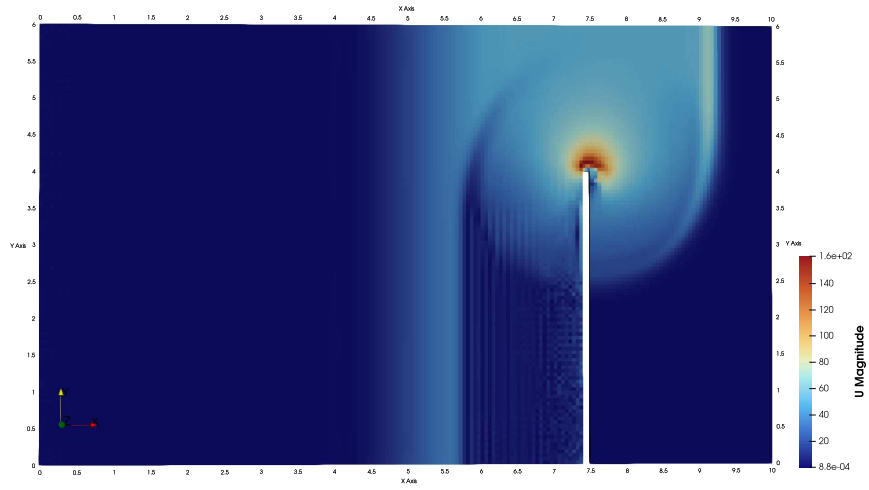


Figure 6.7: Low-intensity shock level: magnitude of the velocity field at $t = 0.2$ s. Post-processed in ParaView.

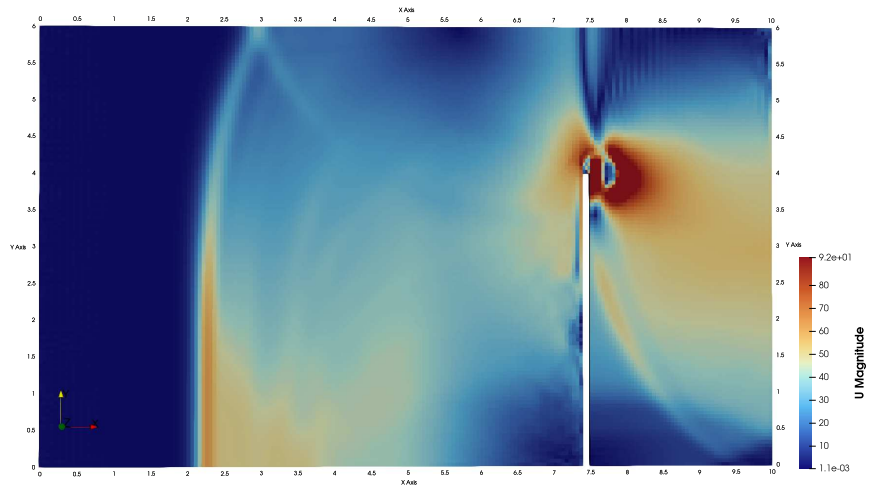


Figure 6.8: Low-intensity shock level: magnitude of the velocity field at $t = 0.3$ s. Post-processed in ParaView.

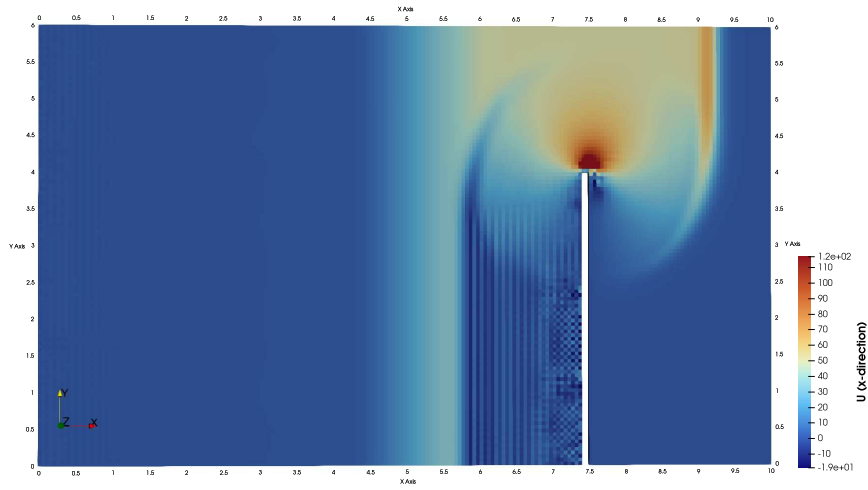


Figure 6.9: Low-intensity shock level: flow velocity along the x -direction at $t = 0.2$ s. Post-processed in ParaView.

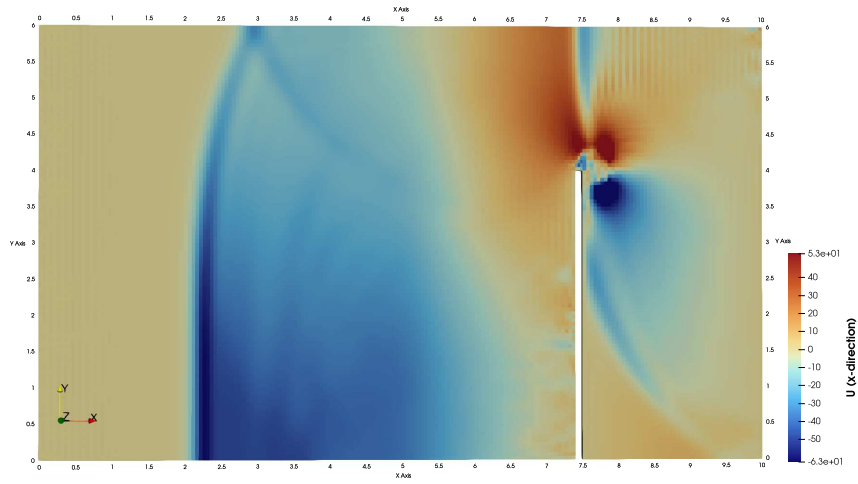


Figure 6.10: Low-intensity shock level: flow velocity along the x -direction at $t = 0.3$ s. Post-processed in ParaView.

High-intensity shock level

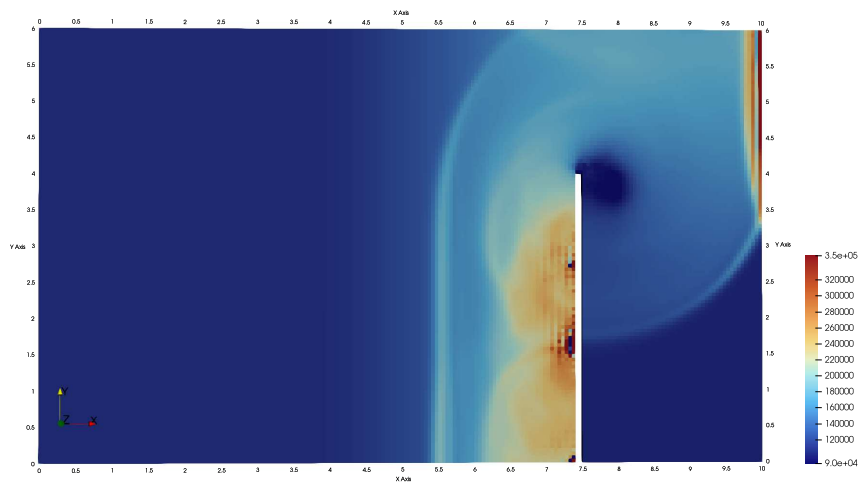


Figure 6.11: High-intensity shock level: pressure distribution at $t = 0.2$ s. Post-processed in ParaView.

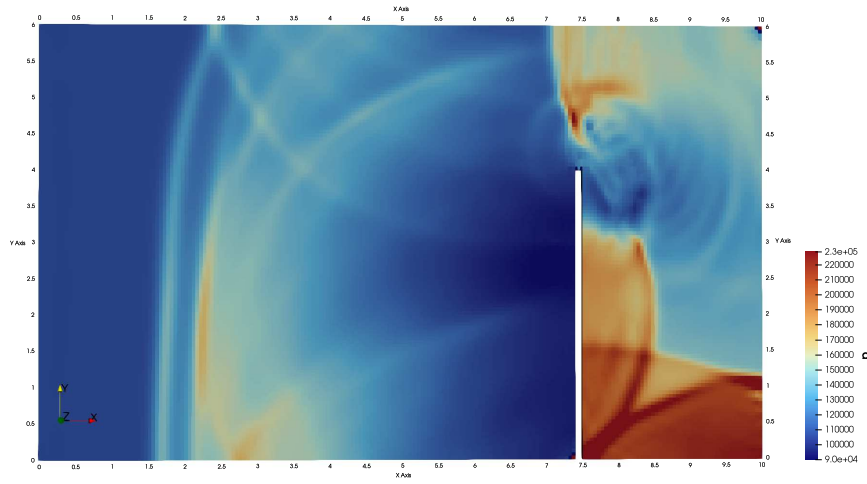


Figure 6.12: High-intensity shock level: pressure distribution at $t = 0.3$ s. Post-processed in ParaView.

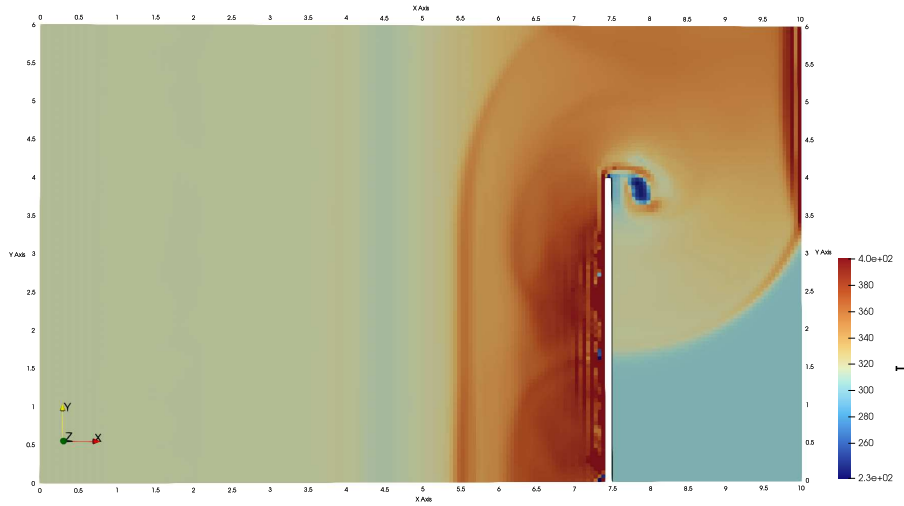


Figure 6.13: High-intensity shock level: temperature distribution at $t = 0.2$ s. Post-processed in ParaView.

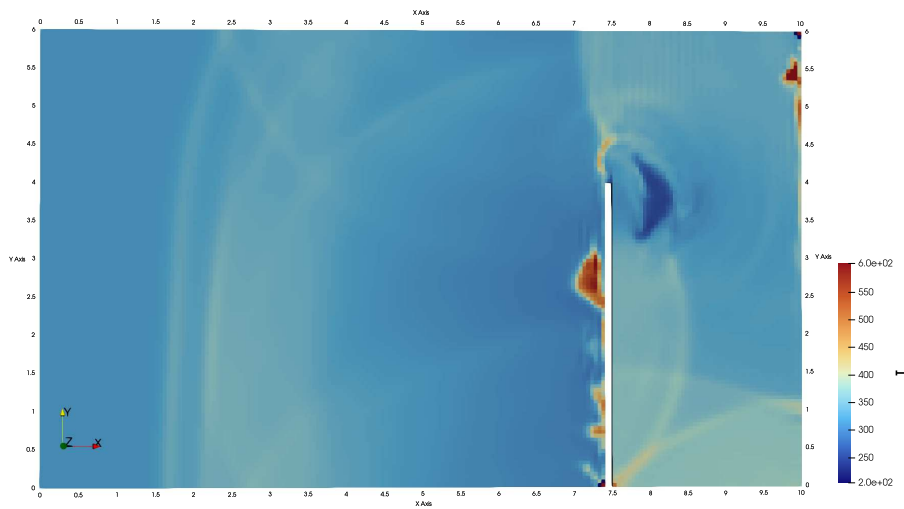


Figure 6.14: High-intensity shock level: temperature distribution at $t = 0.3$ s. Post-processed in ParaView.

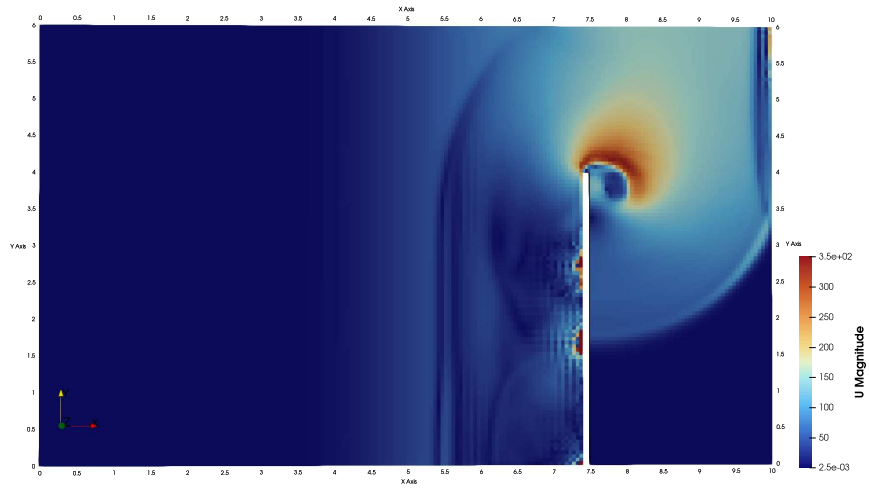


Figure 6.15: High-intensity shock level: magnitude of the velocity field at $t = 0.2$ s. Post-processed in ParaView.

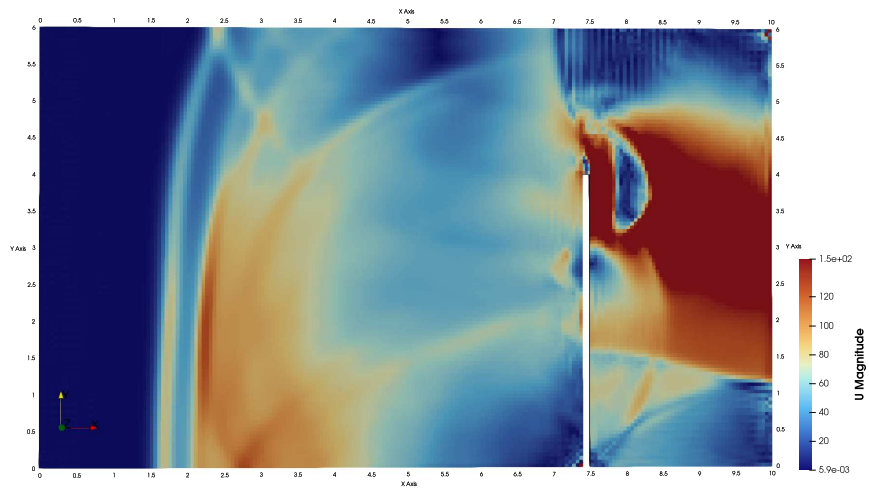


Figure 6.16: High-intensity shock level: magnitude of the velocity field at $t = 0.3$ s. Post-processed in ParaView.

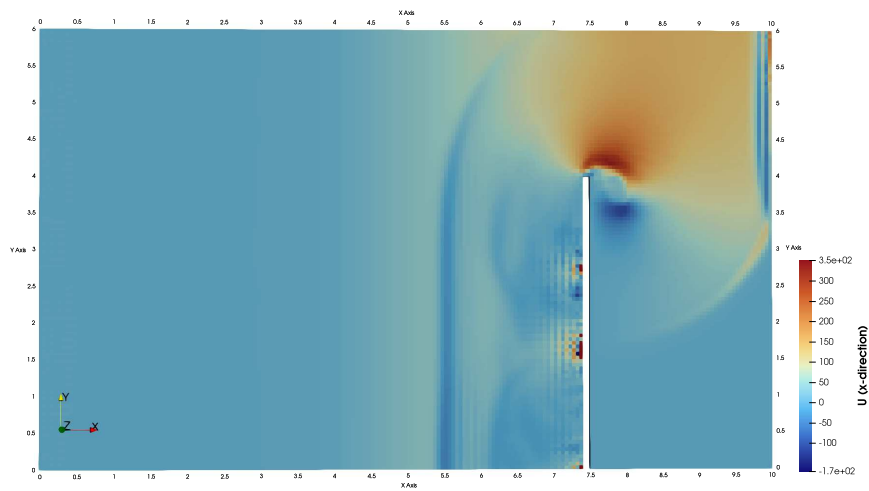


Figure 6.17: High-intensity shock level: flow velocity along the x -direction at $t = 0.2$ s. Post-processed in ParaView.

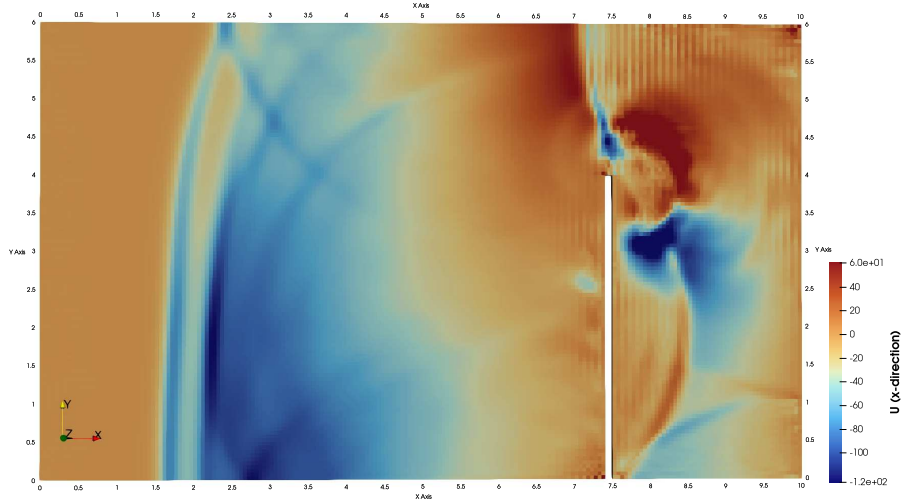


Figure 6.18: High-intensity shock level: flow velocity along the x -direction at $t = 0.3$ s. Post-processed in ParaView.

The post-processed results reveal localized oscillations in the near-wall region immediately following the impact event, attributable to the aforementioned shock-capturing limitations of the pressure-based solver formulation. To avoid interpolation-induced smoothing of the field values, all post-processed fields presented above are visualized as cell data rather than point data.

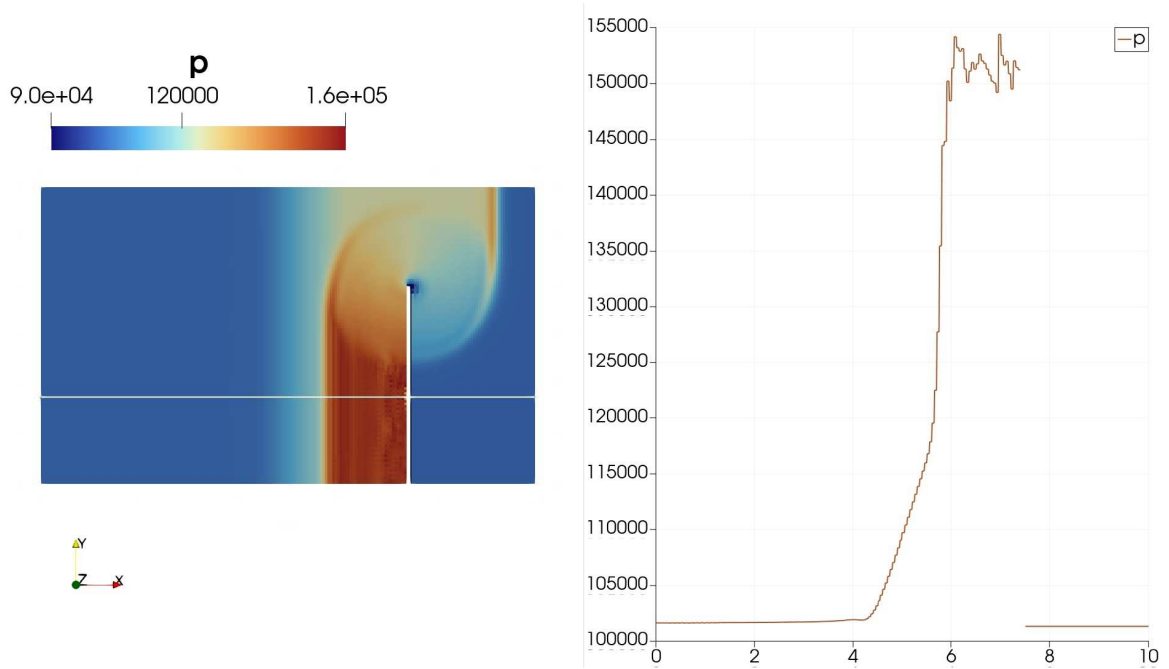


Figure 6.19: Low-intensity shock level: oscillatory pressure profile along the x -direction, at $y = 1.75$ m and $t = 0.2$ s. Post-processed in ParaView.

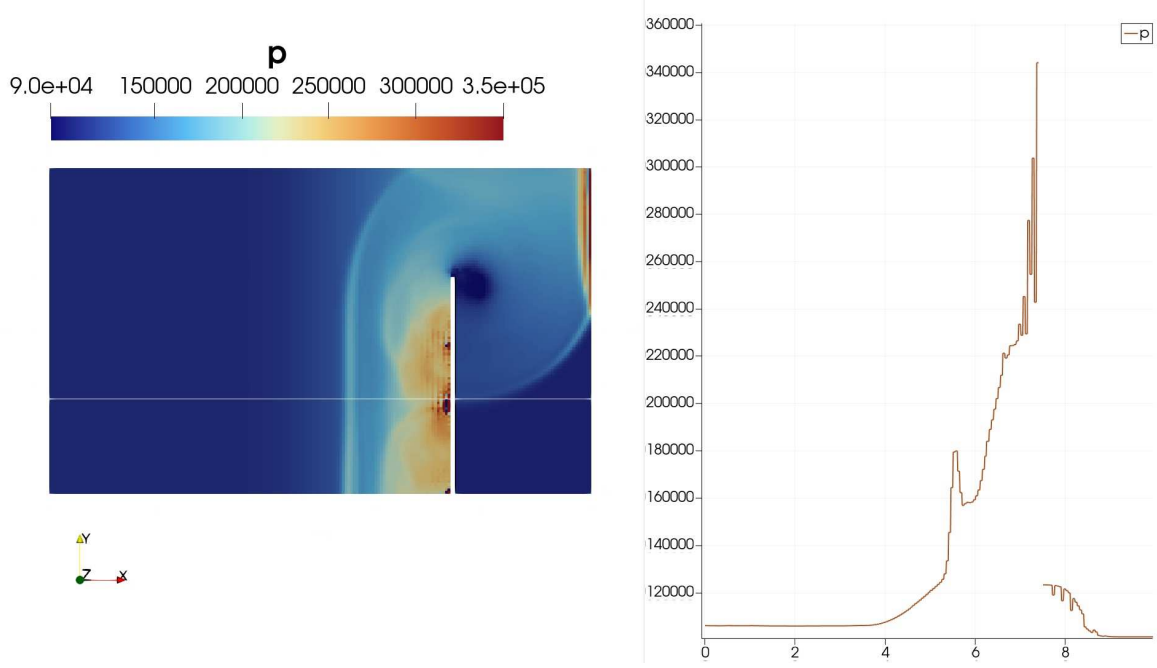


Figure 6.20: High-intensity shock level: oscillatory pressure profile along the x -direction, at $y = 1.75$ m and $t = 0.2$ s. Post-processed in ParaView.

6.2 The OpenRadioss participant: setup and configuration

6.2.1 Physical properties

The structural participant comprises an OpenRadioss domain that models a pseudo two-dimensional wall, which is clamped at its base and composed of a polydimethylsiloxane (PDMS) silicone rubber treated as a Mooney-Rivlin hyperelastic solid; the dimensions of the structural domain are defined as $\Omega_S = [7.4, 7.5] \times [0, 4] \times [0, 0.2]$ m, though a conservative preliminary configuration featured a thicker wall sized $\Omega_S^{\text{thick}} = [7.4, 7.7] \times [0, 4] \times [0, 0.2]$ m. The present implementation represents the first independently developed integration of OpenRadioss as a structural participant within the preCICE framework, with the aim of investigating the coupled system response in a previously undocumented numerical configuration.

The OpenRadioss domain consists of an incompressible, non-linear Mooney-Rivlin material, whose physical properties and experimental constants are drawn from the M20 end-linked PDMS rubber blend formulation reported by Villani et al. [45] and Lu et al. [46] The Mooney-Rivlin material parameters characterizing the constitutive response are listed in Table 6.4.

Initial Density ρ [kg/m ³]	Poisson's Ratio ν []	C_{10} [kPa]	C_{01} [kPa]	Shear Modulus μ [kPa]
1020	0.49	1808	1781	7178

Table 6.4: Physical properties of the Mooney-Rivlin silicone rubber, as prescribed by Villani et al. [45] and Lu et al. [46]

From these parameters, the Young's modulus E is derived as:

$$E = 2 \cdot \mu \cdot (1 + \nu) = 2 \cdot 7178 \cdot (1 + 0.49) = 21\,390 \text{ kPa} = 21.39 \text{ MPa} \quad (6.3)$$

Although blast impact applications typically rely on elastic materials with stiffness on the order of 10^2 GPa (e.g. aluminum, steel), the present configuration departs from this convention by adopting a hyperelastic material owing to its lower longitudinal wave speed c . The associated reduction in c yields a larger critical time step, defined as $\Delta T_{\text{crit}} = \frac{l_c}{c}$ (where l_c denotes the characteristic mesh

size), limiting the associated computational cost while retaining a physically representative material model. A quantitative derivation of the critical time step for the present application is provided in Section 6.2.2.

The clamp at the base of the structural wall is the only physical boundary condition applied to the participant. In addition, the discretization nodes located on the fluid–structure interface are collected through a /GRNOD card in the `_0000.rad` starter file and provided as input to the OpenRadioss `.cpl` configuration file.

6.2.2 Numerical implementation

The structural domain for the present configuration is discretized with $3 \times 80 \times 3$ elements across the three spatial directions. The constitutive behavior of the structural wall is implemented in OpenRadioss through the /MAT/LAW42 material card, based on the Mooney-Rivlin formulation parameters reported by Villani et al. [45] and Lu et al. [46] for the M20 end-linked polydimethylsiloxane (PDMS) rubber blend.

The constitutive parameters listed in Table 6.4 can be specified directly as input fields in the /MAT/LAW42 property card, where the initial shear modulus μ is defined according to Eq. (3.10). The input parameters μ_p and α_p prescribed by OpenRadioss for the Mooney-Rivlin constitutive formulation are recalled from Eq.(3.14) as follows[23]:

$$\mu_1 = 2 \cdot C_{10}, \quad \mu_2 = -2 \cdot C_{01}, \quad \alpha_1 = 2, \quad \alpha_2 = -2 \quad (6.4)$$

where C_{10} and C_{01} are the experimental material constants reported by Villani et al. [45]

By substituting the experimental constants C_{10} and C_{01} into Eq. 6.4, the resulting coefficients μ_1 and μ_2 evaluate to:

$$\mu_1 = 2 \cdot C_{10} = 2 \cdot 1808 \text{ kPa} = 3616 \text{ kPa}, \quad \mu_2 = -2 \cdot C_{01} = -2 \cdot 1781 \text{ kPa} = -3562 \text{ kPa} \quad (6.5)$$

The resulting /MAT/LAW42 material card, specified for the present configuration, is shown in Listing 6.4.

```

1  #--1---|---2---|---3---|---4---|---5---|---6---|---7---|---8---|---9---|---10---|
2  #-  2. MATERIAL DEFINITION:
3  #--1---|---2---|---3---|---4---|---5---|---6---|---7---|---8---|---9---|---10---|
4  /MAT/LAW42/1
5  flapMaterial
6  #          RHO_I
7          250
8  #          Nu          sigma_cut          funIDbulk          Fscale_bulk          M          I_form
9          0.4              0              0              0              0              1
10 #          Mu_1          Mu_2          Mu_3          Mu_4          Mu_5
11          3616000.0        -3562000.0
12 #          Mu_6          Mu_7          Mu_8          Mu_9          Mu_10
13
14 #          alpha_1          alpha_2          alpha_3          alpha_4          alpha_5
15          2              -2
16 #          alpha_6          alpha_7          alpha_8          alpha_9          alpha_10
17
18 #--1---|---2---|---3---|---4---|---5---|---6---|---7---|---8---|---9---|---10---|

```

Listing 6.4: `_0000.rad` starter file: /MAT/LAW42 material card specifying the Mooney-Rivlin constitutive parameters.

The numerical discretization of the structural participant employs **HEX20** solid elements, with the same integration scheme presented in Listing 5.8, to accurately represent bending and curved geometries through higher-order elements without excessively increasing the element count. The runtime control parameters of the solver configuration are defined in the `_0001.rad` engine file; the simulation duration is consistently set to $t = 0.03 \text{ s} = 3 \cdot 10^{-2} \text{ s}$, with a fixed time step of $\Delta t_S = 0.00001 \text{ s} = 1 \cdot 10^{-5} \text{ s}$ enforced through the procedure discussed in Listing 4.9, and a write interval of $0.001 \text{ s} = 1 \cdot 10^{-3} \text{ s}$.

The enforced fixed time step Δt_S follows from the critical time step evaluation of the present Ω_S domain, whose value ΔT_{crit} drives the selection of the time discretization governing the coupled simulation.

For the Ω_S domain, the characteristic mesh sizes l_c along the three spatial directions are:

- $l_{c,x} = \frac{0.1}{3} = 3.\bar{3} \cdot 10^{-2} \text{ m}$;
- $l_{c,y} = \frac{4}{80} = 5 \cdot 10^{-2} \text{ m}$;
- $l_{c,z} = \frac{0.2}{3} = 6.\bar{6} \cdot 10^{-2} \text{ m}$;

The speed of sound for the present pseudo two-dimensional structure is computed as:

$$c = \sqrt{\frac{E}{\rho}} = \sqrt{\frac{21.39 \cdot 10^6 \text{ Pa}}{1020 \text{ kg/m}^3}} = 144.81 \text{ m/s} \quad (6.6)$$

and the corresponding critical time step therefore evaluates to:

$$\Delta T_{\text{crit}} = \frac{l_c}{c} = \frac{6.\bar{6} \times 10^{-3} \text{ m}}{144.81 \text{ m/s}} = 4.6 \times 10^{-5} \text{ s}. \quad (6.7)$$

To avoid solution divergence and simulation failure caused by the violation of such critical time step condition, a conservative fixed time step of $\Delta t_S = 1 \cdot 10^{-5} \text{ s}$ is therefore enforced throughout the coupled simulation.

Finally, the coupling parameters of the structural participant are specified in the **.cpl** configuration file following the setup presented in Listing 4.4, as discussed in Section 4.1.3. The **/GRNOD/2** card defines the group of discretization nodes located on the shared boundary, whose tag ID = 2 is prescribed in the **_0000.rad** starter file to identify the coupling interface.

6.3 preCICE: setup and configuration

The preCICE configuration of the **Blast Wave Impact against a Structural Wall** mirrors that of the **Pulsating Flow over Flexible Flap** simulation in nearly all respects, as illustrated in Figure 6.21.

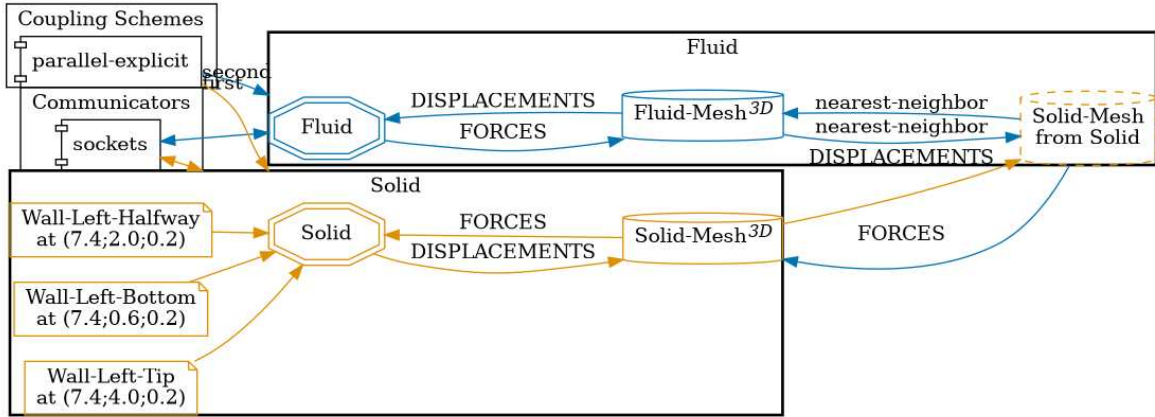


Figure 6.21: Blast Wave Impact against a Structural Wall: preCICE-config-visualizer.

For a detailed overview of Figure 6.21, the reader is referred to Section 5.3; the primary difference from Figure 5.3 lies in the data mapping method configured at the shared boundary. Owing to the novelty of the present coupled configuration, the interface meshes of both participants are constructed as conforming grids to eliminate mapping artifacts and inconsistencies that could otherwise arise from a poorly defined data mapping, as the resulting inaccuracies would be indistinguishable from those inherent to the coupled setup. Accordingly, to preclude an ill-defined support radius that could destabilize the radial-basis-function mapping and compromise coupling stability, the mapping method is set to **nearest-neighbor** so that each interface node exchanges data directly with its counterpart on the conforming grid. The corresponding implementation is detailed in Listing 6.5.

```

1  <participant name="Fluid">
2    <provide-mesh name="Fluid-Mesh" />
3    <receive-mesh name="Solid-Mesh" from="Solid" />
4    <write-data name="FORCES" mesh="Fluid-Mesh" />
5    <read-data name="DISPLACEMENTS" mesh="Fluid-Mesh" />
6    <mapping:nearest-neighbor direction="write" from="Fluid-Mesh" to="Solid-Mesh"
7      constraint="conservative"/>
8    <mapping:nearest-neighbor direction="read" from="Solid-Mesh" to="Fluid-Mesh"
9      constraint="consistent"/>
10   <export:vtu directory="precice-exports" every-n-time-windows="500" />
11 </participant>

```

Listing 6.5: Excerpt from the preCICE XML configuration: nearest-neighbor mapping configuration.

To monitor the displacement of the wall along the x -direction, three watchpoints are distributed along the height on its left face to characterize the hyperelastic structural response across its full extent. The watchpoints are located at $(x, y, z) = (7.4, 0.6, 0.2)$ m in the lower portion of the wall, $(x, y, z) = (7.4, 2.0, 0.2)$ m at mid-height, and $(x, y, z) = (7.4, 4.0, 0.2)$ m at the top. The corresponding watchpoint implementation is reported in Listing 6.6.

```

1  <participant name="Solid">
2    <provide-mesh name="Solid-Mesh" />
3    <write-data name="DISPLACEMENTS" mesh="Solid-Mesh" />
4    <read-data name="FORCES" mesh="Solid-Mesh" />
5    <export:vtu directory="precice-exports" every-n-time-windows="500" />
6    <watch-point mesh="Solid-Mesh" name="Wall-Left-Tip" coordinate="7.4;4.0;0.2" />
7    <watch-point mesh="Solid-Mesh" name="Wall-Left-Halfway" coordinate="7.4;2.0;0.2" />
8    <watch-point mesh="Solid-Mesh" name="Wall-Left-Bottom" coordinate="7.4;0.6;0.2" />
9  </participant>

```

Listing 6.6: Excerpt from the preCICE XML configuration: watchpoint implementation on the OpenRadioss structural participant.

6.4 Results and conclusions

The constitutive response of a hyperelastic structural wall subjected to blast wave loading is investigated across three configurations of progressively increasing severity: a **thick-wall** baseline at a **low-intensity** shock, a **thin-wall** configuration at the same blast intensity to isolate the effect of wall thickness under conditions of material non-linearity, and a **thin-wall** configuration at a **high-intensity** shock to assess the system response under extreme loading conditions. Across all three configurations, the structural response is quantified through the displacement time histories recorded at the **Bottom-Left-Edge**, **Mid-Left-Edge**, and **Tip-Left-Edge** watchpoints, complemented by the pressure field evolution within the fluid domain to track the propagation of the shock front and its interaction with the structural wall.

The investigation opens with the **thick-wall** configuration subjected to a **low-intensity** shock, serving as a conservative baseline to establish whether the coupled framework correctly captures the structural response under impulsive loading or whether the simulation breaks down due to physical material failure. The **thin-wall** configuration is subsequently considered at the same shock intensity level, with the fluid-side boundary conditions left unchanged to isolate the effect of wall thickness on the coupled response. Finally, the shock intensity is increased while retaining the **thin-wall** geometry to assess how the coupled solution evolves under more demanding loading conditions. In each configuration, the displacement time histories recorded at the watchpoints confirm the correct data exchange throughout the fluid-structure coupling, while the pressure field evolution is examined to provide a first characterization of blast wave propagation and its interaction with the structural wall within this previously uncharacterized setting. Detailed results for each configuration are presented in the following sections.

6.4.1 Preliminary assessment: thick-wall configuration at a low-intensity shock level

The **thick-wall** configuration at a **low-intensity** shock level defines the initial setup of the coupled domain: a 0.3 m thick wall, characterized by the material model detailed in Listing 6.4, and a 150 000 Pa blast wave, implemented as in Listing 6.2. The normal shock relations reported in Eq.(6.1), Eq.(6.2) and Eq.(2.41), combined with the ideal-gas definition of the speed of sound ($a = \sqrt{\gamma RT}$) and Eq.(2.36), yield the shock wave properties listed in Table 6.5; those are computed as a function of the upstream pressure p_{blast} , prescribed via `system/setFieldsDict` as the overpressure set by the `volScalarFieldValue p` entry.

Low-intensity blast wave: upstream properties	
Mach Number []	1.19
Temperature [K]	336
Density [kg/m ³]	1.62
Shock Propagation Velocity [m/s]	436.70
Instant of Shock Incidence [s]	0.012

Table 6.5: Upstream properties of the **low-intensity** blast wave, derived from the upstream pressure p_{blast} using the normal shock relations.

These values result from a theoretical back-calculation that recovers the upstream shock properties from the known downstream state and the prescribed upstream pressure, thus reversing the standard shock-tube analysis sequence in which downstream conditions are computed from a fully known upstream state. Furthermore, the normal shock relations assume an idealized discontinuity rather than the finite-thickness shock structure resolved within the simulation, and the solver's known limitations in representing sharp discontinuities introduce additional deviation from the theoretical values listed above. The resulting estimates therefore serve as a reference for comparison rather than as a precise prediction of the simulated shock wave behavior.

This interpretation is supported by the ParaView post-processing results, which confirms the shock impact occurs at $t_{\text{simulation}} \approx 0.015$ s (in reasonable agreement with the theoretical travel time, from which the predicted shock Mach number and propagation velocity follow as consistent with the simulated values), while the density field dataset at $t = 0.001$ s, shown in Figure 6.22, yields $\rho \approx 1.6$ kg/m³, in agreement with the theoretical estimate.

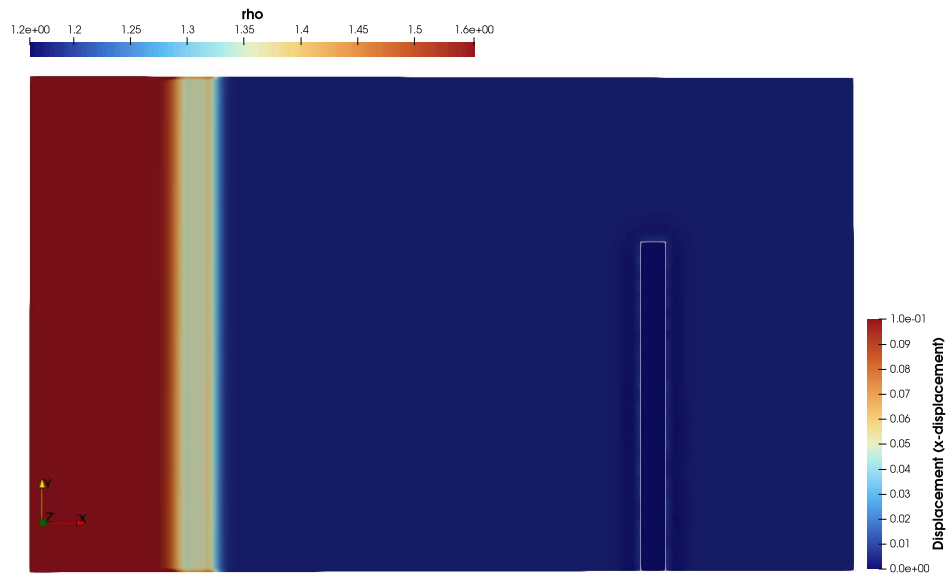


Figure 6.22: Low-intensity shock level: density distribution at $t = 0.001$ s. Post-processed in ParaView.

As a first characterization of the previously unexplored numerical configuration, selected field snapshots at $t = 0.02$ s and $t = 0.03$ s are provided below (Figures 6.23 through 6.32) to illustrate the pressure, temperature, velocity (magnitude and x -direction components) and density fields enabling the shock front propagation to be traced and offering insight into the evolution of the coupled fluid-structure response.

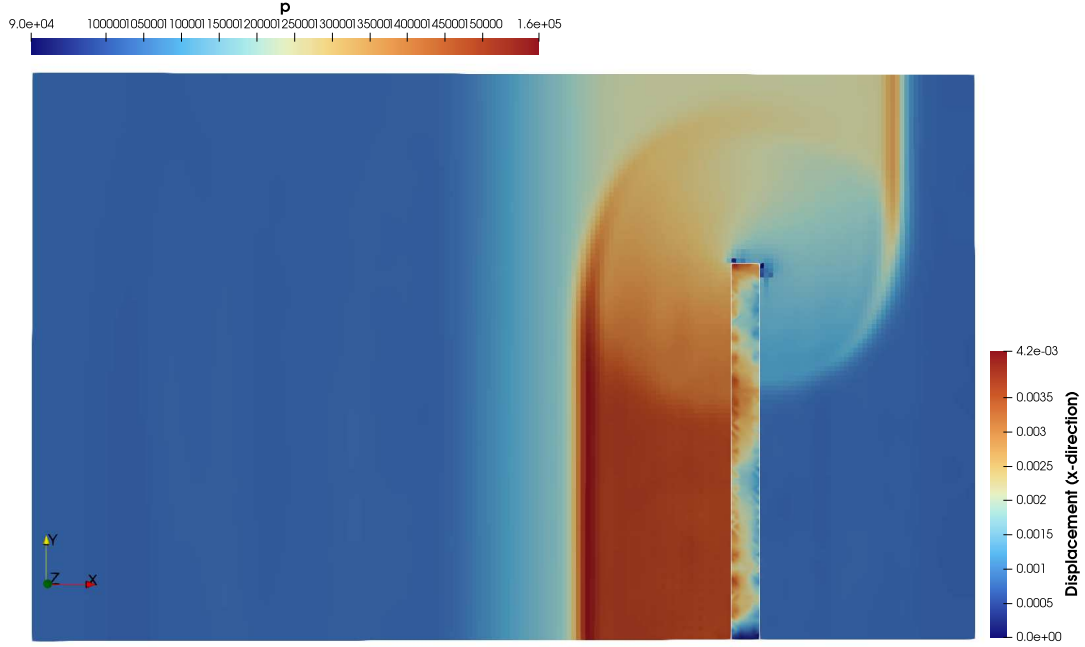


Figure 6.23: Thick-wall coupled configuration at the low-intensity shock level: pressure distribution at $t = 0.2$ s. Post-processed in ParaView.

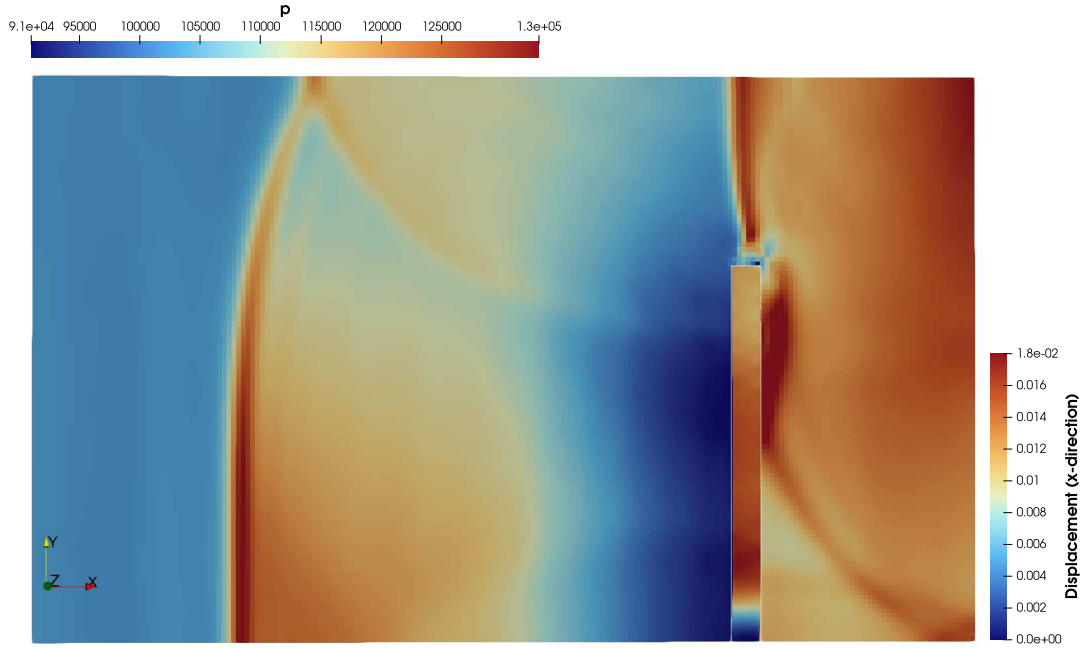


Figure 6.24: Thick-wall coupled configuration at the low-intensity shock level: pressure distribution at $t = 0.3$ s. Post-processed in ParaView.

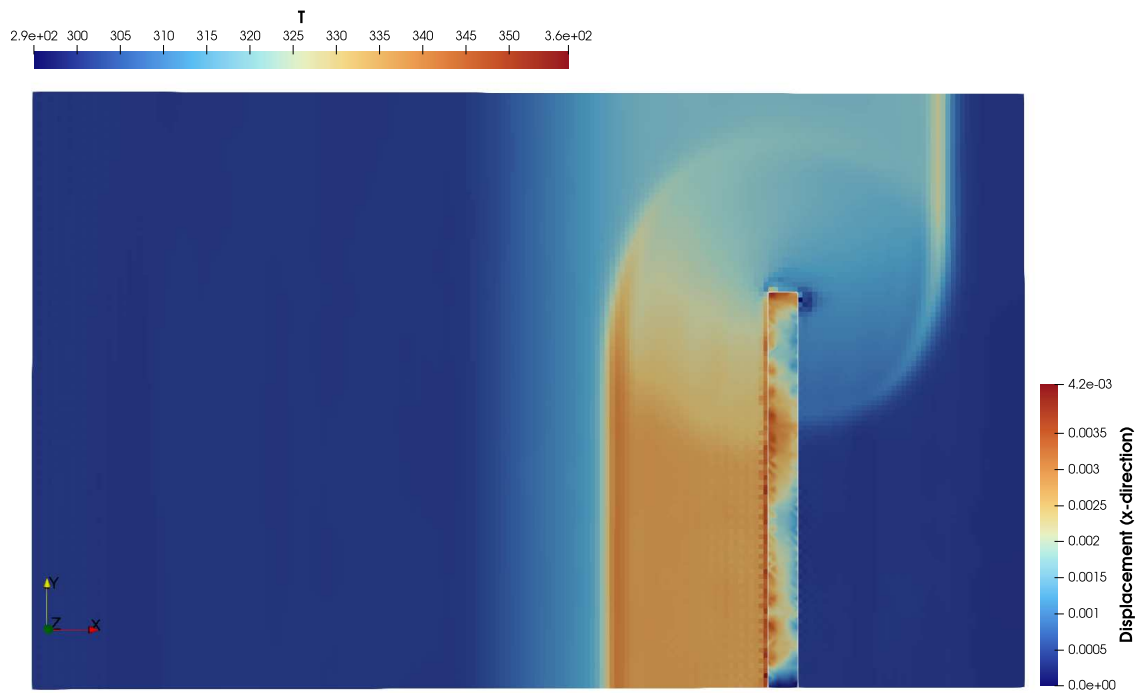


Figure 6.25: Thick-wall coupled configuration at the low-intensity shock level: temperature distribution at $t = 0.2$ s. Post-processed in ParaView.

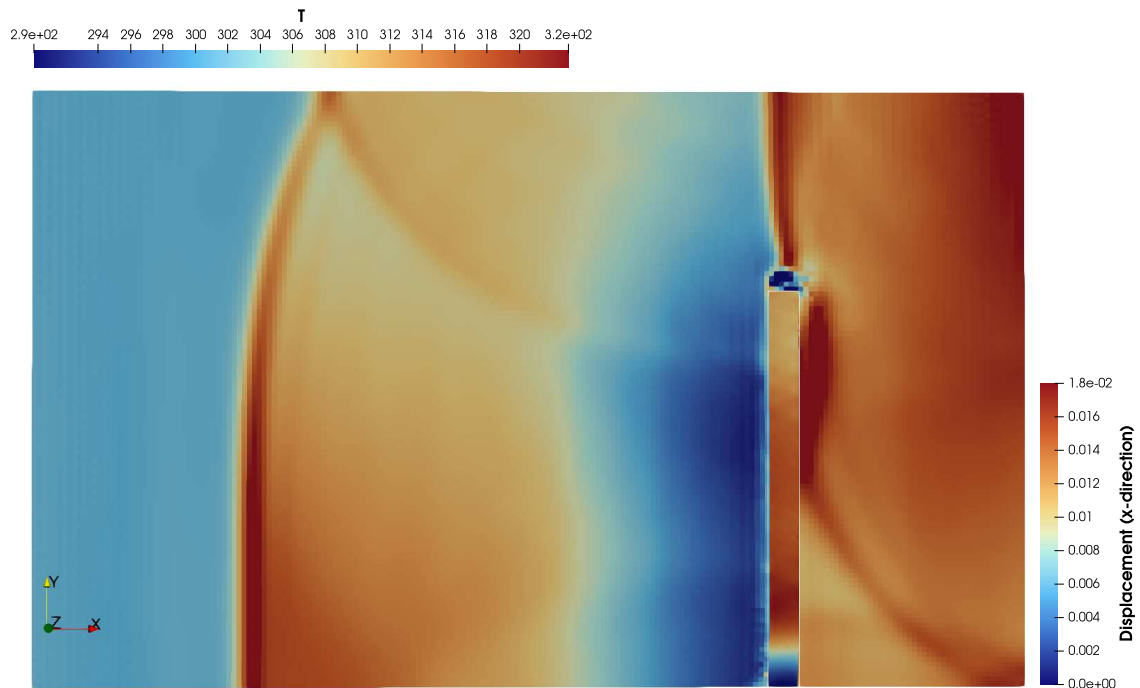


Figure 6.26: Thick-wall coupled configuration at the low-intensity shock level: temperature distribution at $t = 0.3$ s. Post-processed in ParaView.

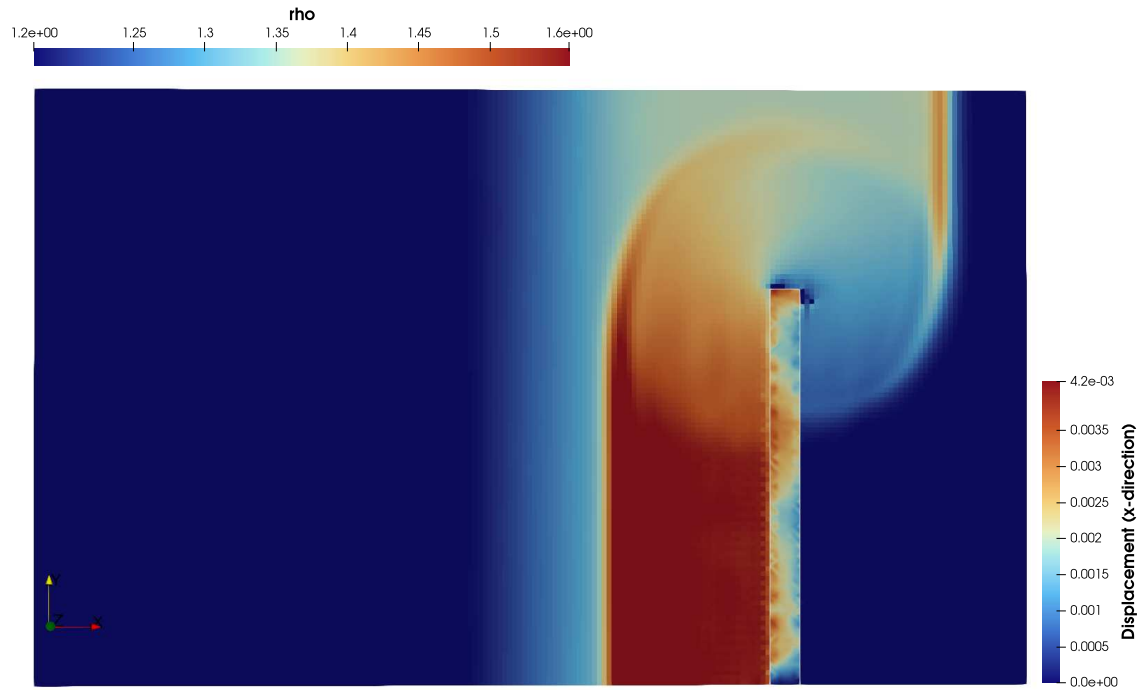


Figure 6.27: Thick-wall coupled configuration at the low-intensity shock level: density distribution at $t = 0.2$ s. Post-processed in ParaView.

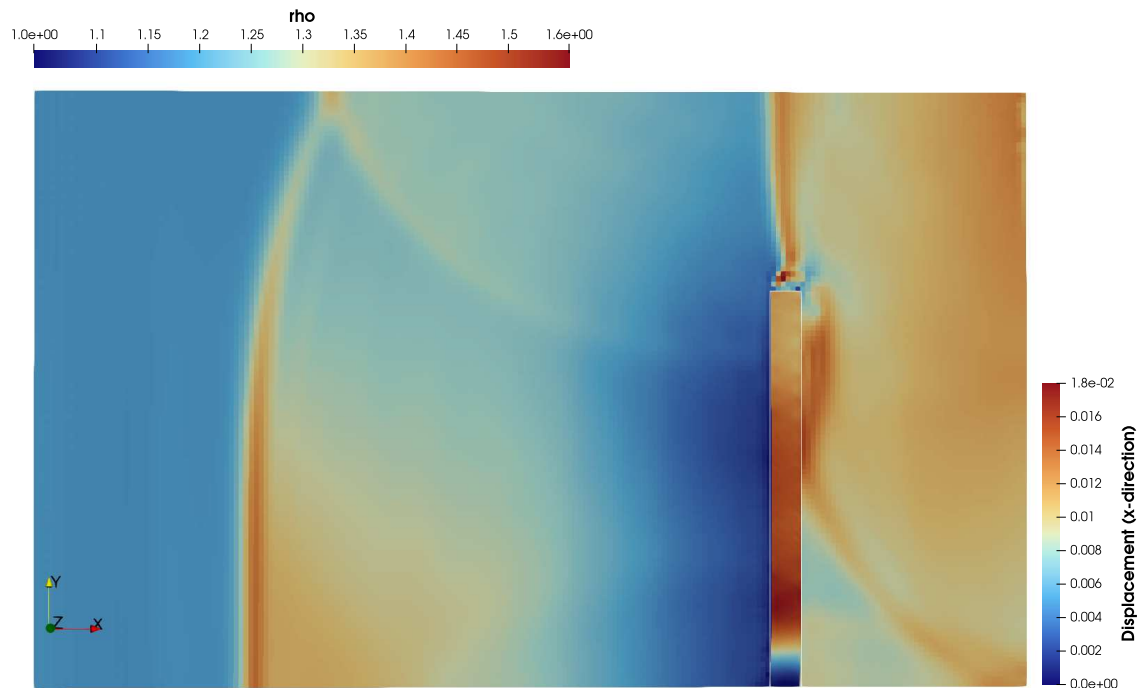


Figure 6.28: Thick-wall coupled configuration at the low-intensity shock level: density distribution at $t = 0.3$ s. Post-processed in ParaView.

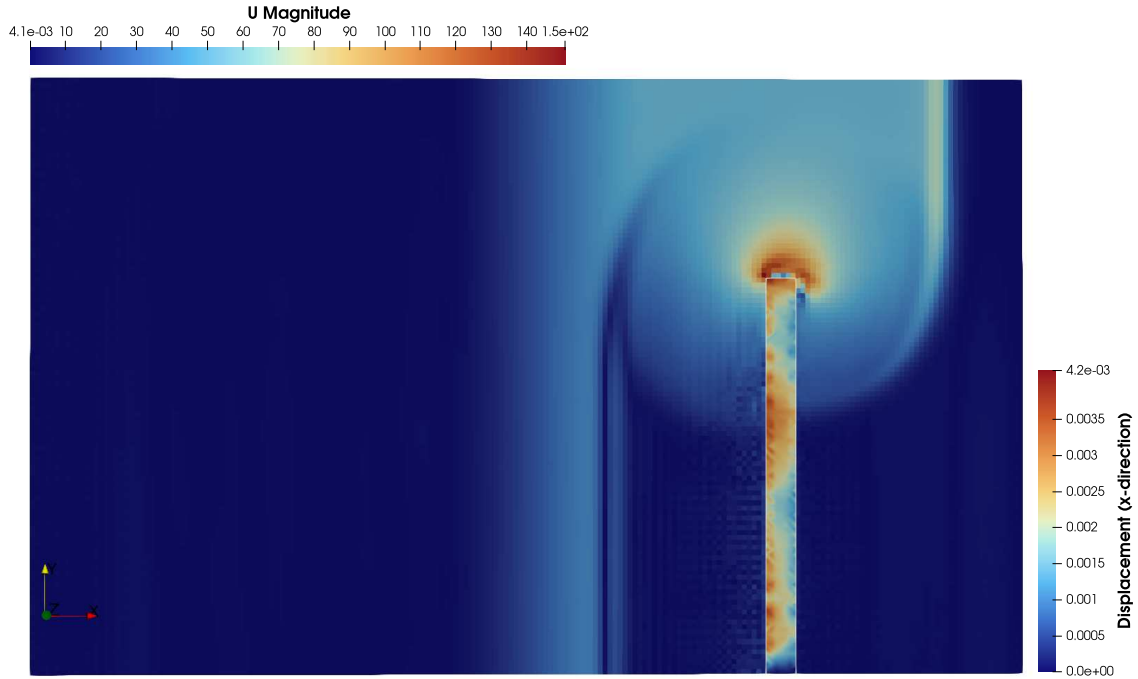


Figure 6.29: Thick-wall coupled configuration at the low-intensity shock level: magnitude of the velocity field at $t = 0.2$ s. Post-processed in ParaView.

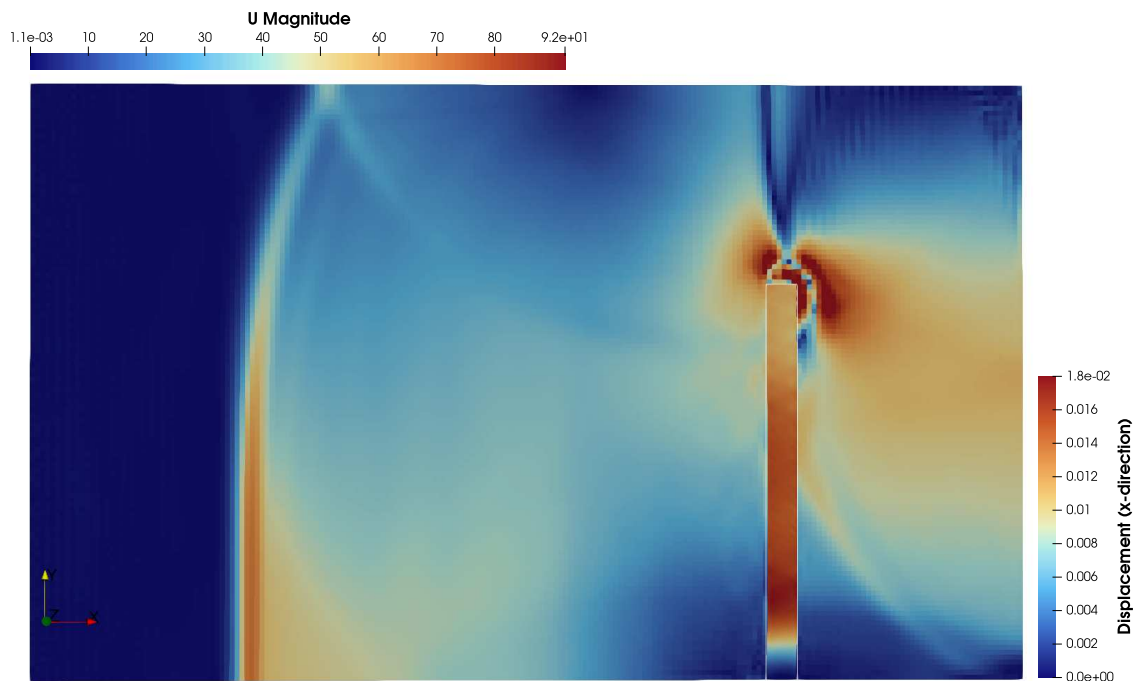


Figure 6.30: Thick-wall coupled configuration at the low-intensity shock level: magnitude of the velocity field at $t = 0.3$ s. Post-processed in ParaView.

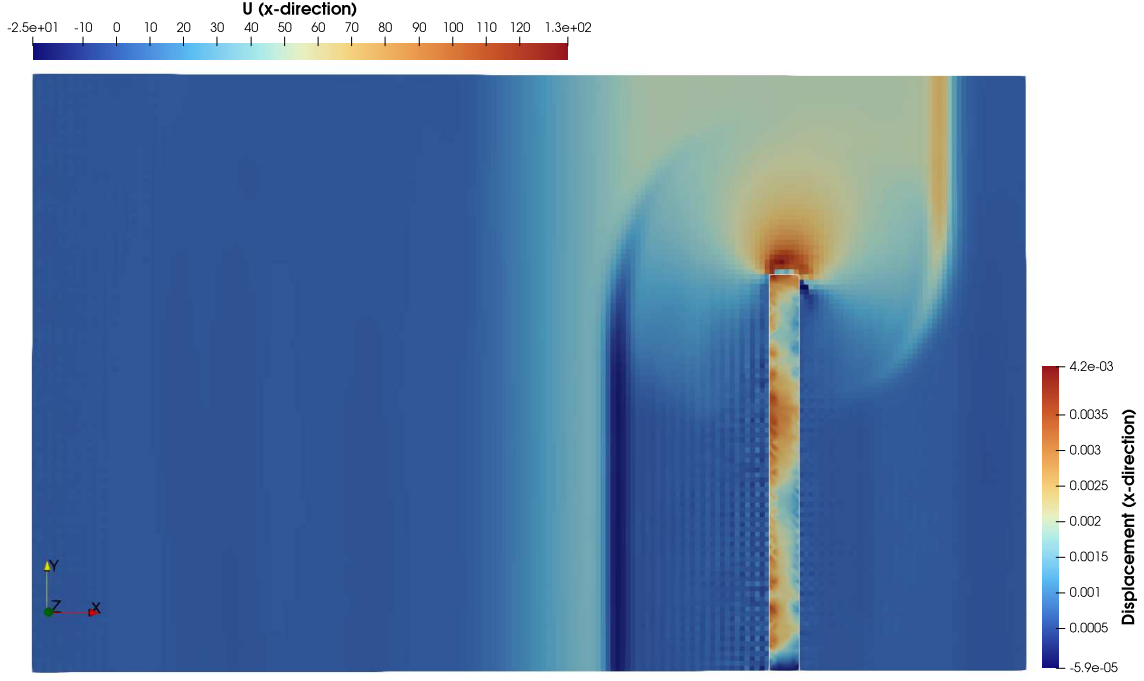


Figure 6.31: Thick-wall coupled configuration at the low-intensity shock level: flow velocity along the x -direction at $t = 0.2$ s. Post-processed in ParaView.

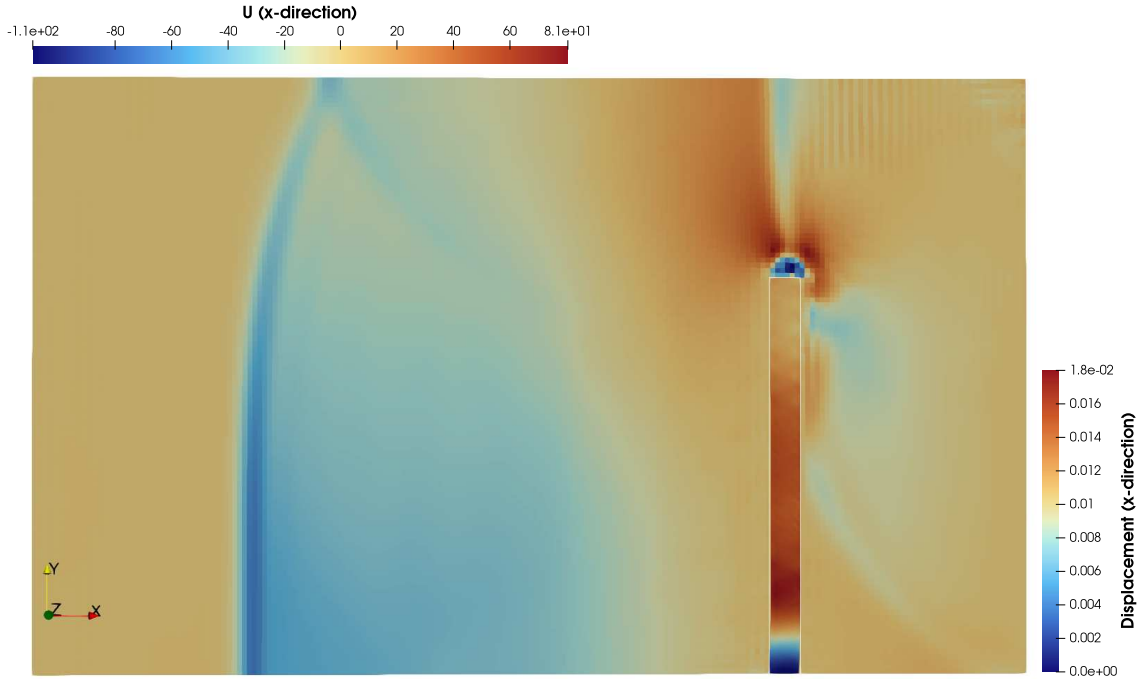


Figure 6.32: Thick-wall coupled configuration at the low-intensity shock level: flow velocity along the x -direction at $t = 0.3$ s. Post-processed in ParaView.

Analysis of the displacement time histories recorded by the three watchpoints reveals that, although coupling completes successfully, the structural response exhibits oscillations even prior to the shock wave impact; these are attributable to the comparable order of magnitude between the displacement amplitude and the numerical instabilities intrinsic to the non-linear constitutive response of hyperelastic materials. The comparatively irregular structural response, relative to that observed in Chapter 5, reflects the preliminary nature of this configuration, which remains under active investigation rather than representing a fully optimized setup. The results nonetheless establish a solid basis for the

subsequent **thin-wall** configurations, confirming that the selected material model sustains the imposed loading without failure. Figure 6.33 illustrates the three watchpoint displacement histories along the x -direction, post-processed in MATLAB, and Table 6.6 summarizes the recorded displacement values at selected instants $t = 0.015$ s, $t = 0.02$ s, and $t = 0.03$ s.

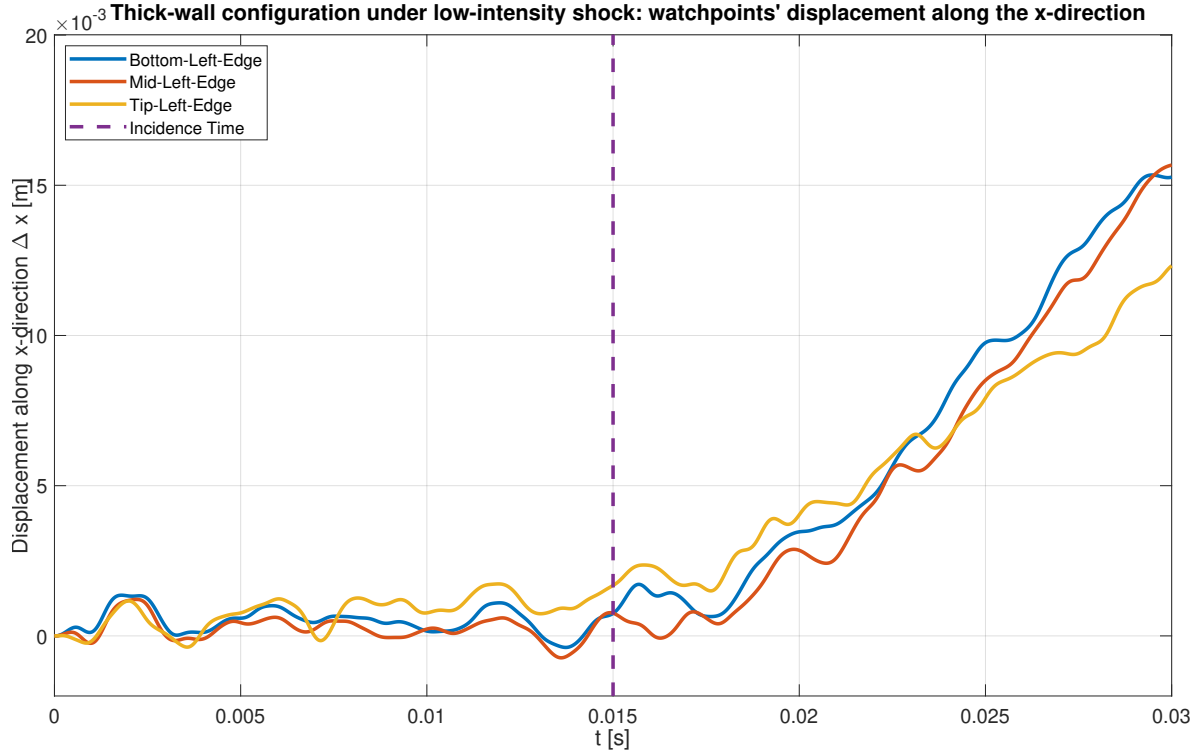


Figure 6.33: **Thick-wall** coupled configuration at the **low-intensity** shock level: overlaid displacement time histories along the x -direction for comparison across all three watchpoints. Post-processed in MATLAB.

	Interface watchpoints: displacement along x -direction [m]		
Time [s]	Bottom-Left-Edge	Mid-Left-Edge	Tip-Left-Edge
0.015	$7.4 \cdot 10^{-4}$	$7.6 \cdot 10^{-4}$	0.0017
0.02	0.0035	0.0029	0.004
0.03	0.0153	0.0157	0.0123

Table 6.6: **Thick-wall** coupled configuration at the **low-intensity** shock level: displacement comparison along the x -direction, for selected time instants, across all three watchpoints.

6.4.2 Effect of wall thickness: thin-wall configuration at a low-intensity shock level

By reducing the wall to one third of the baseline value along the x -direction while leaving its boundary conditions and the fluid domain entirely unchanged, the second configuration isolates the effect of wall thickness on the coupled structural response. The setup features a 0.1 m thick structural wall characterized by the same reference material model as detailed in Listing 6.4, and the shock wave properties reported in Table 6.5 remain unaltered.

Analogous field snapshots to those presented in Section 6.4.1 are reported below (Figures 6.34 through 6.43) for the present configuration. To assess the influence of structural coupling, these results may also be compared against the uncoupled fluid response reported in Section 6.1.3.

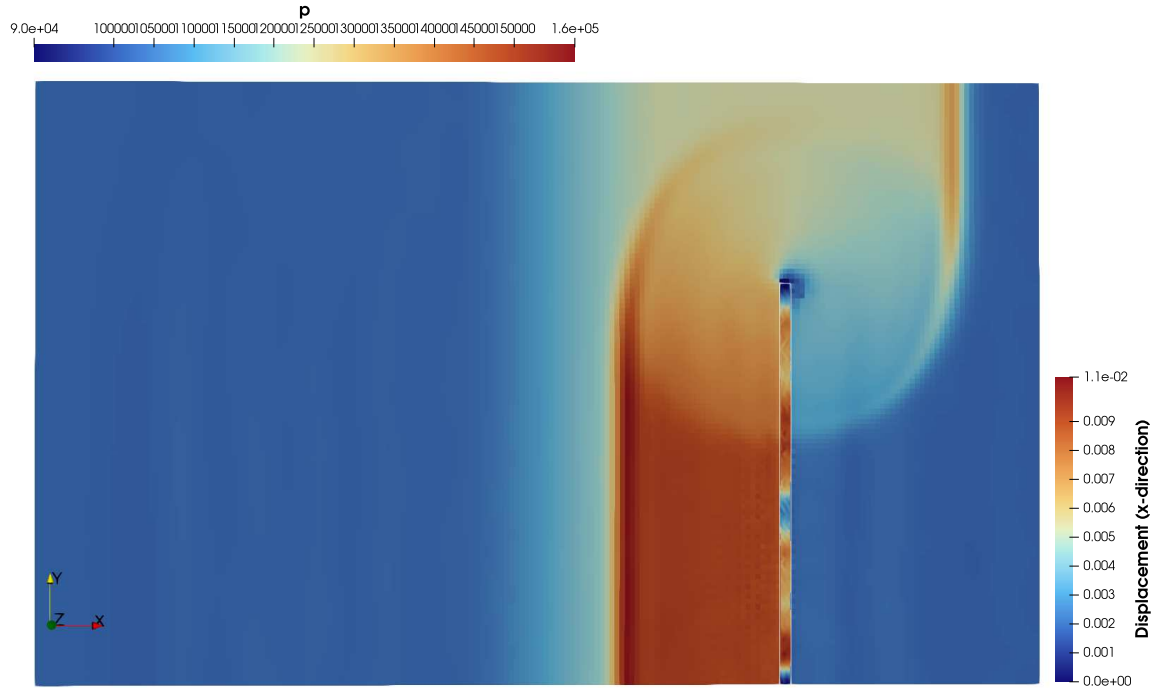


Figure 6.34: Thin-wall coupled configuration at the low-intensity shock level: pressure distribution at $t = 0.2$ s. Post-processed in ParaView.

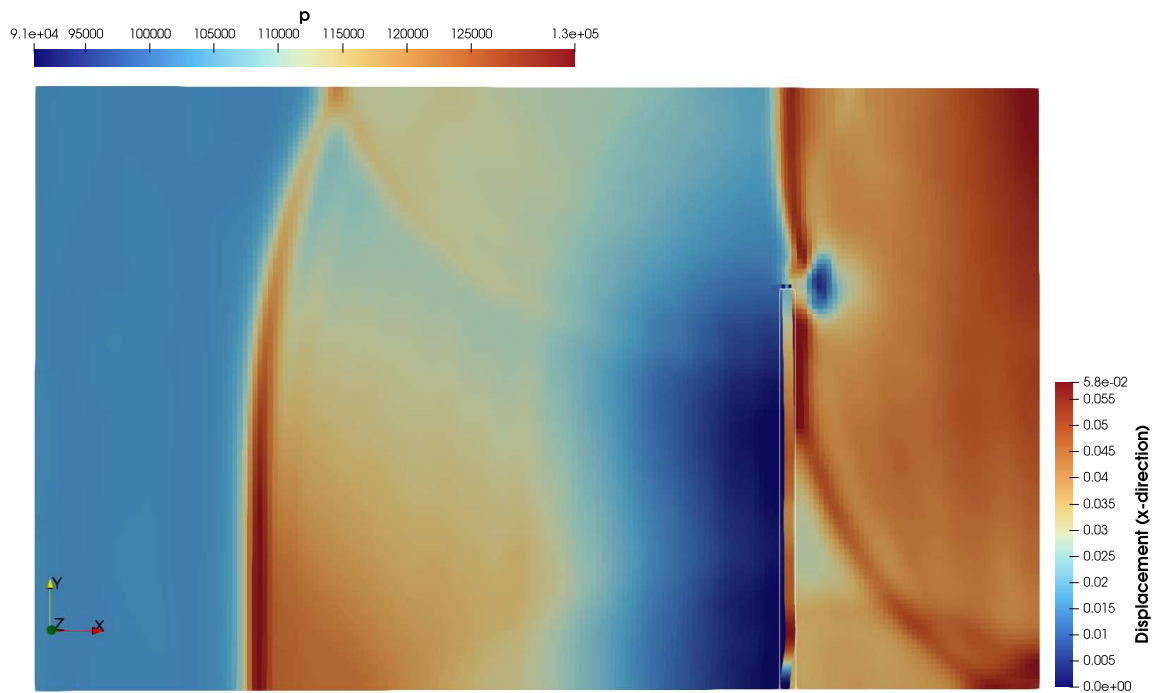


Figure 6.35: Thin-wall coupled configuration at the low-intensity shock level: pressure distribution at $t = 0.3$ s. Post-processed in ParaView.

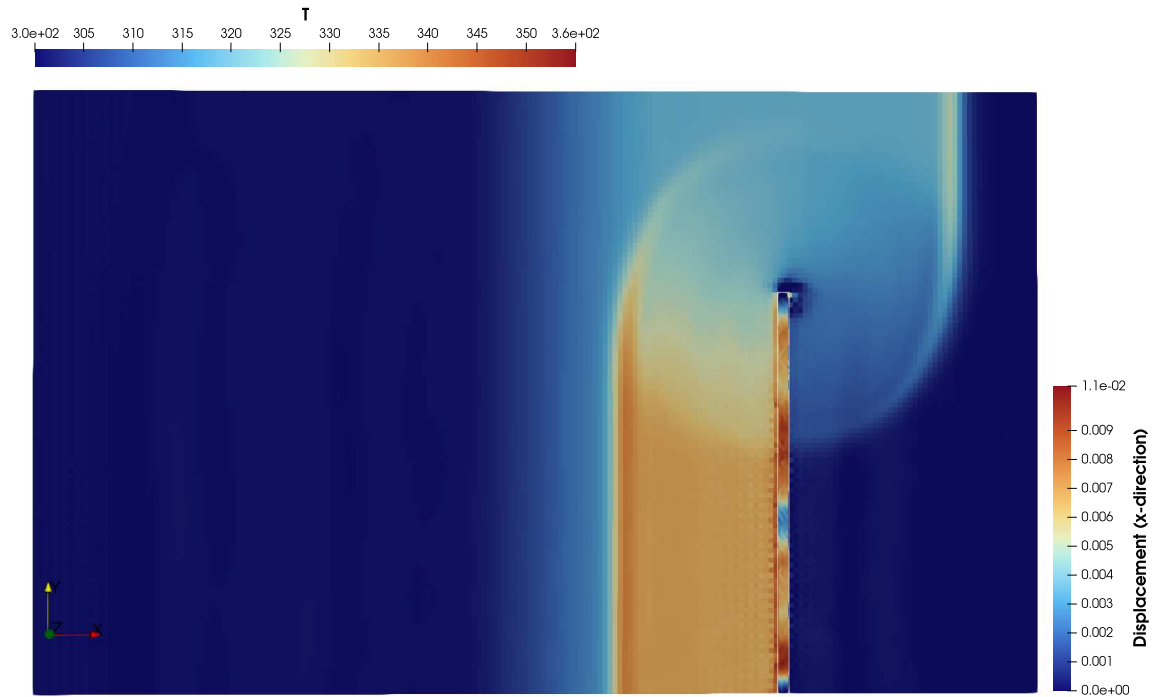


Figure 6.36: Thin-wall coupled configuration at the low-intensity shock level: temperature distribution at $t = 0.2$ s. Post-processed in ParaView.

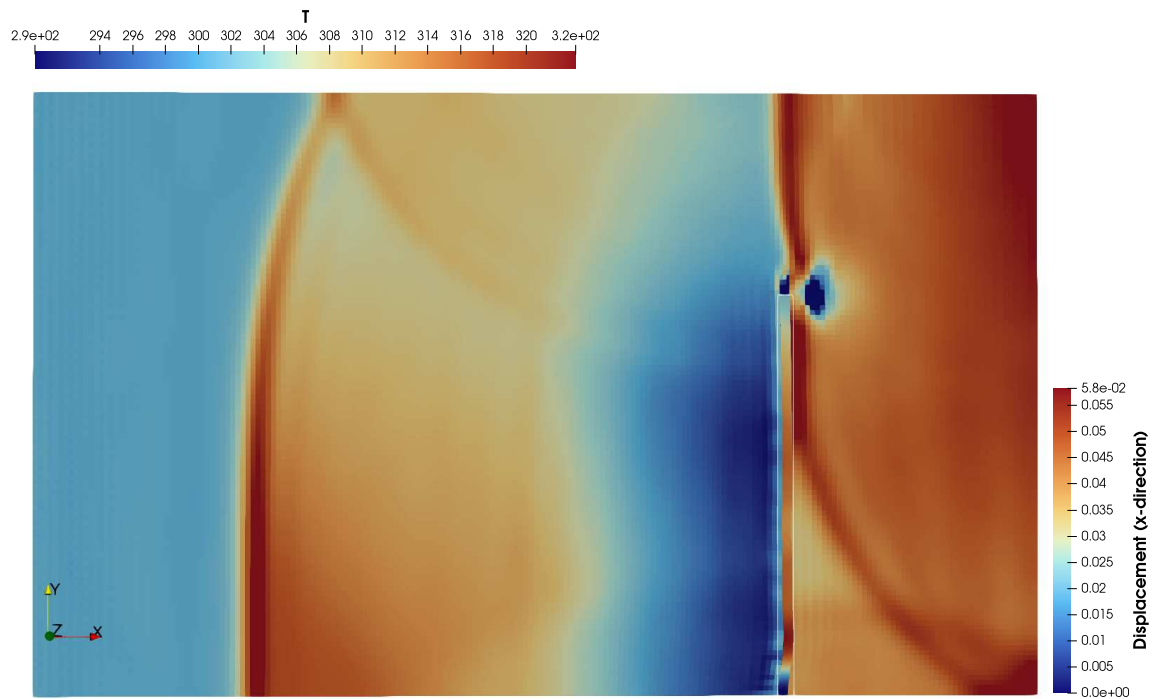


Figure 6.37: Thin-wall coupled configuration at the low-intensity shock level: temperature distribution at $t = 0.3$ s. Post-processed in ParaView.

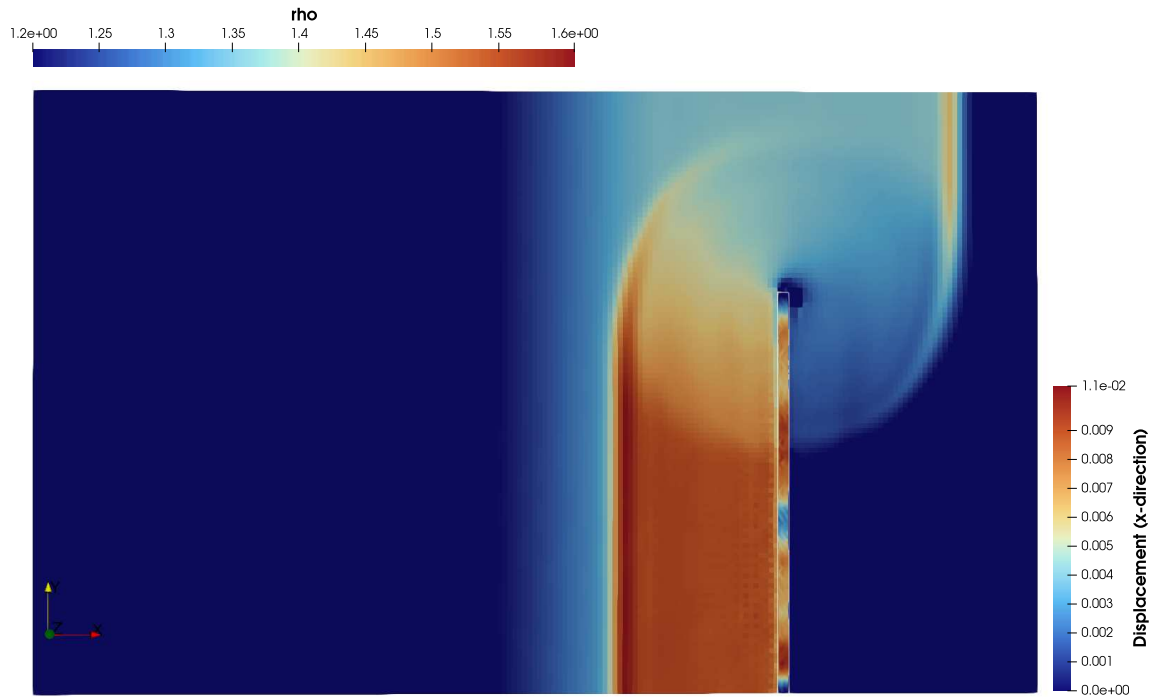


Figure 6.38: Thin-wall coupled configuration at the low-intensity shock level: density distribution at $t = 0.2$ s. Post-processed in ParaView.

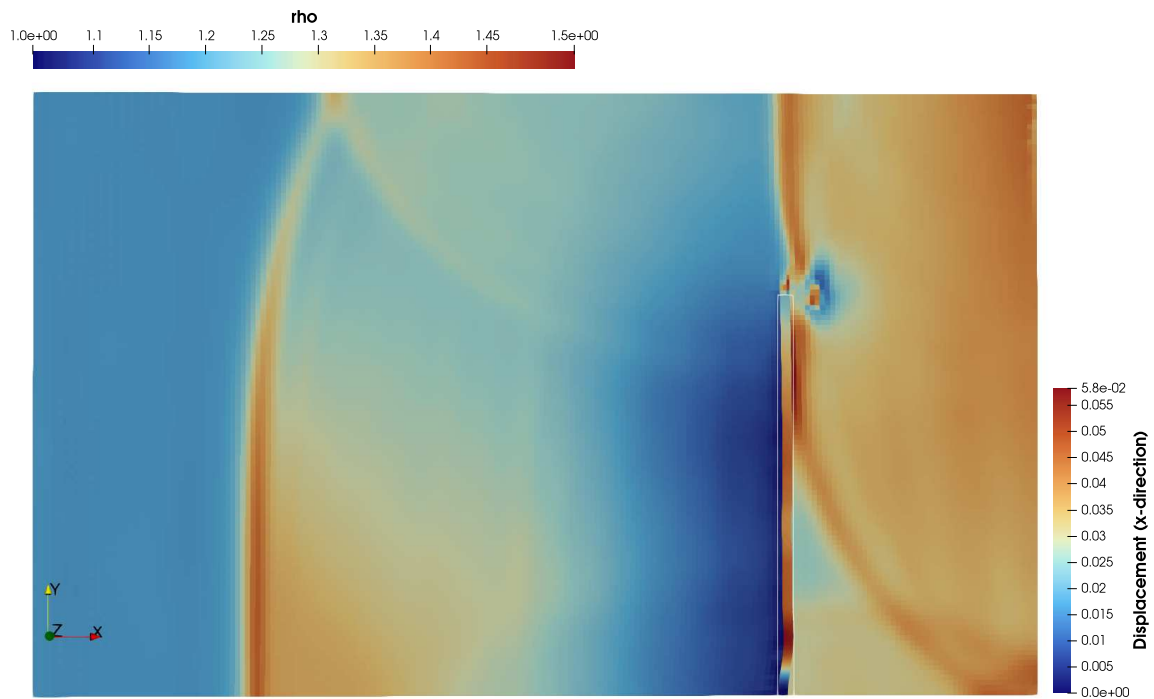


Figure 6.39: Thin-wall coupled configuration at the low-intensity shock level: density distribution at $t = 0.3$ s. Post-processed in ParaView.

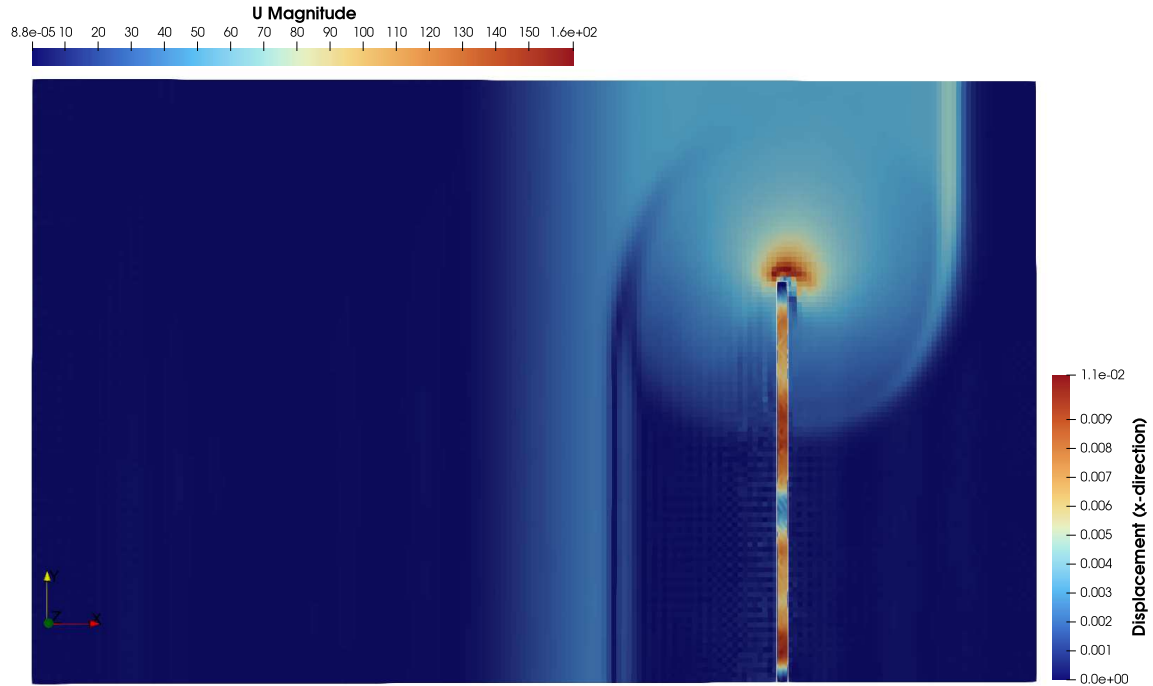


Figure 6.40: Thin-wall coupled configuration at the low-intensity shock level: magnitude of the velocity field at $t = 0.2$ s. Post-processed in ParaView.

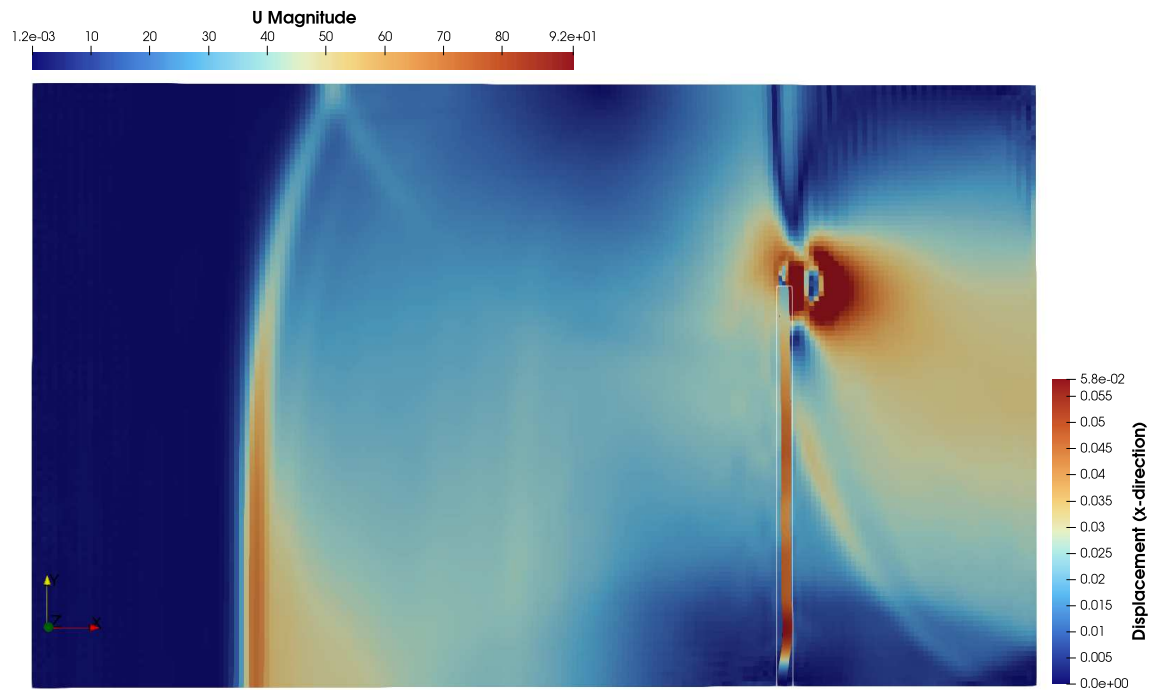


Figure 6.41: Thin-wall coupled configuration at the low-intensity shock level: magnitude of the velocity field at $t = 0.3$ s. Post-processed in ParaView.

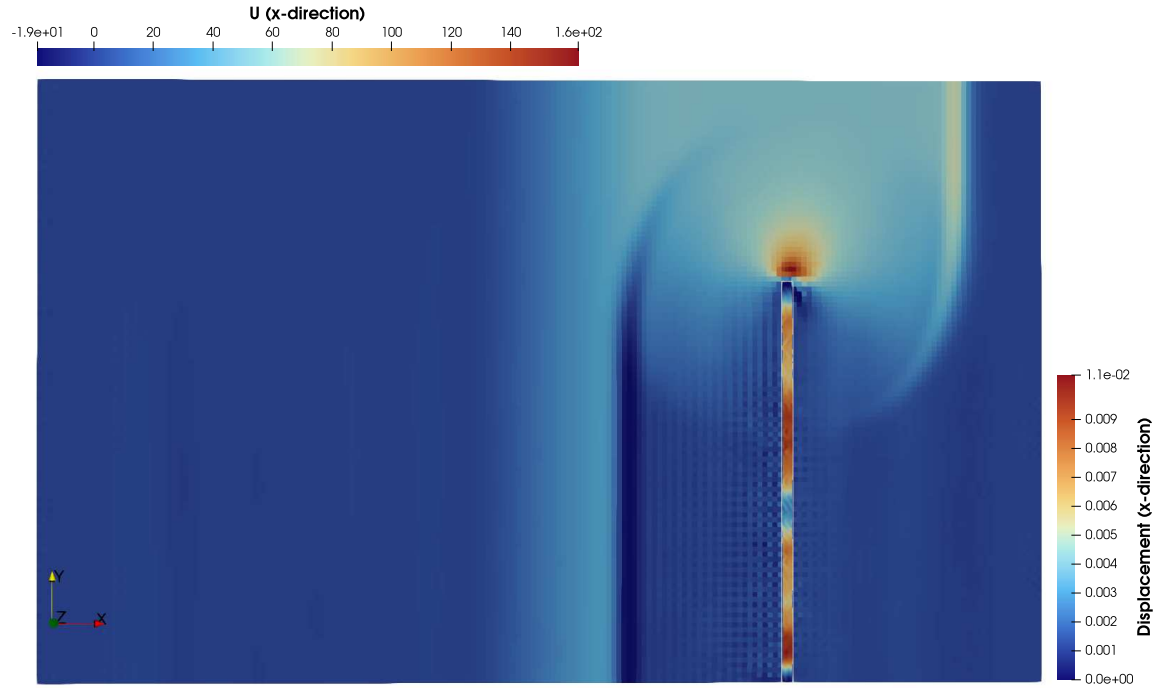


Figure 6.42: Thin-wall coupled configuration at the low-intensity shock level: flow velocity along the x -direction at $t = 0.2$ s. Post-processed in ParaView.

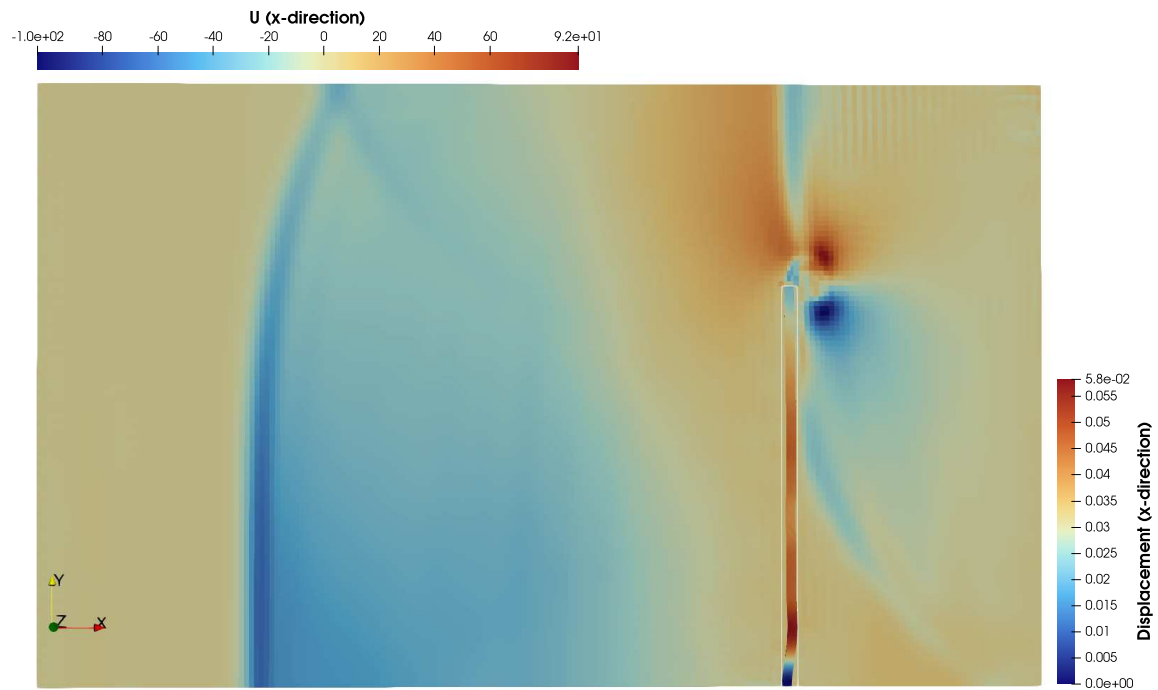


Figure 6.43: Thin-wall coupled configuration at the low-intensity shock level: flow velocity along the x -direction at $t = 0.3$ s. Post-processed in ParaView.

Figures 6.44 through 6.47 provide a comparative overview of the two **low-intensity** configurations, alongside a combined plot overlaying all three watchpoints' time histories of the displacement along the x -direction. Furthermore, Tables 6.7 and 6.8 summarize the recorded displacement values at selected instants $t = 0.015$ s, $t = 0.02$ s, and $t = 0.03$ s.

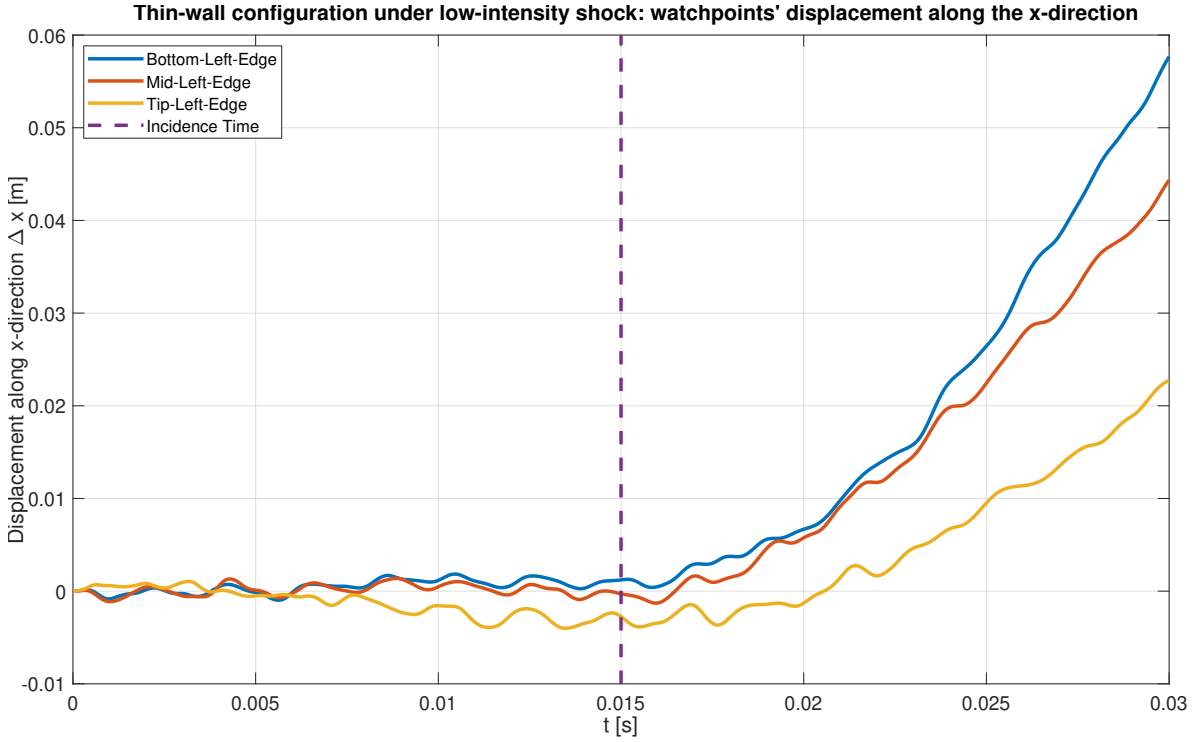


Figure 6.44: **Thin-wall** coupled configuration at the **low-intensity** shock level: overlaid displacement time histories along the x -direction for comparison across all three watchpoints. Post-processed in MATLAB.

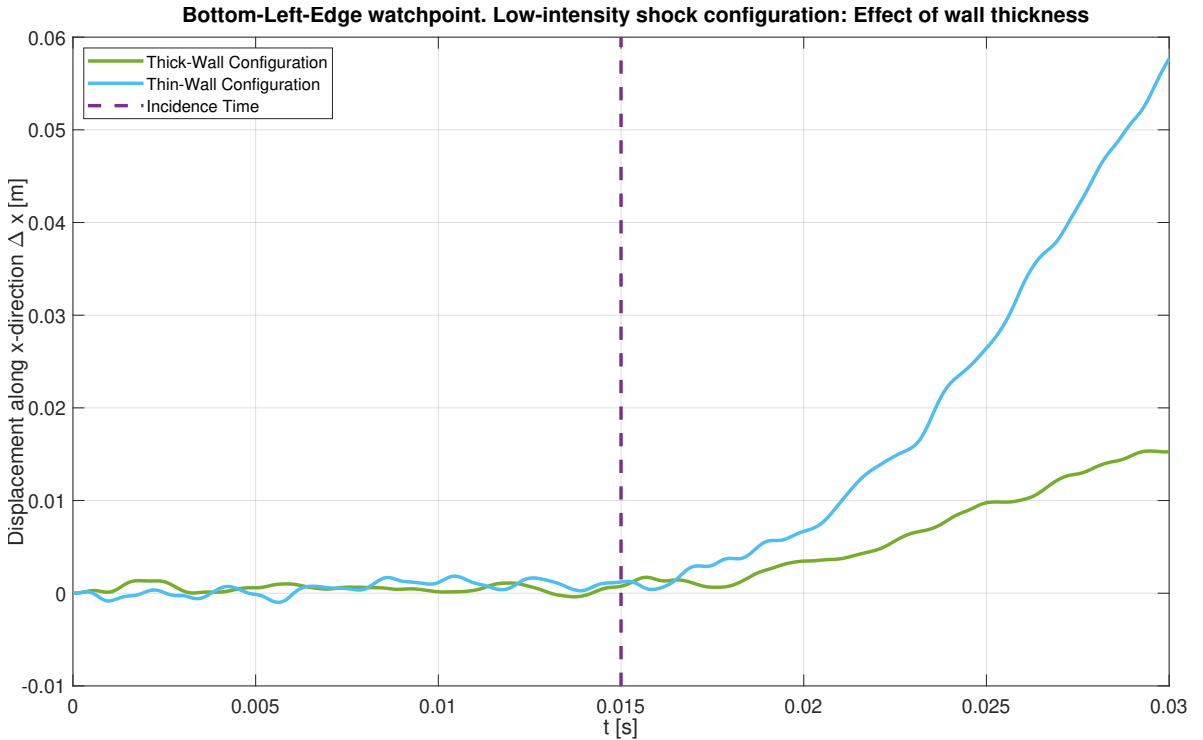


Figure 6.45: **Bottom-Left-Edge** watchpoint: displacement time history along the x -direction. Investigation on the effect of wall thickness. Post-processed in MATLAB.

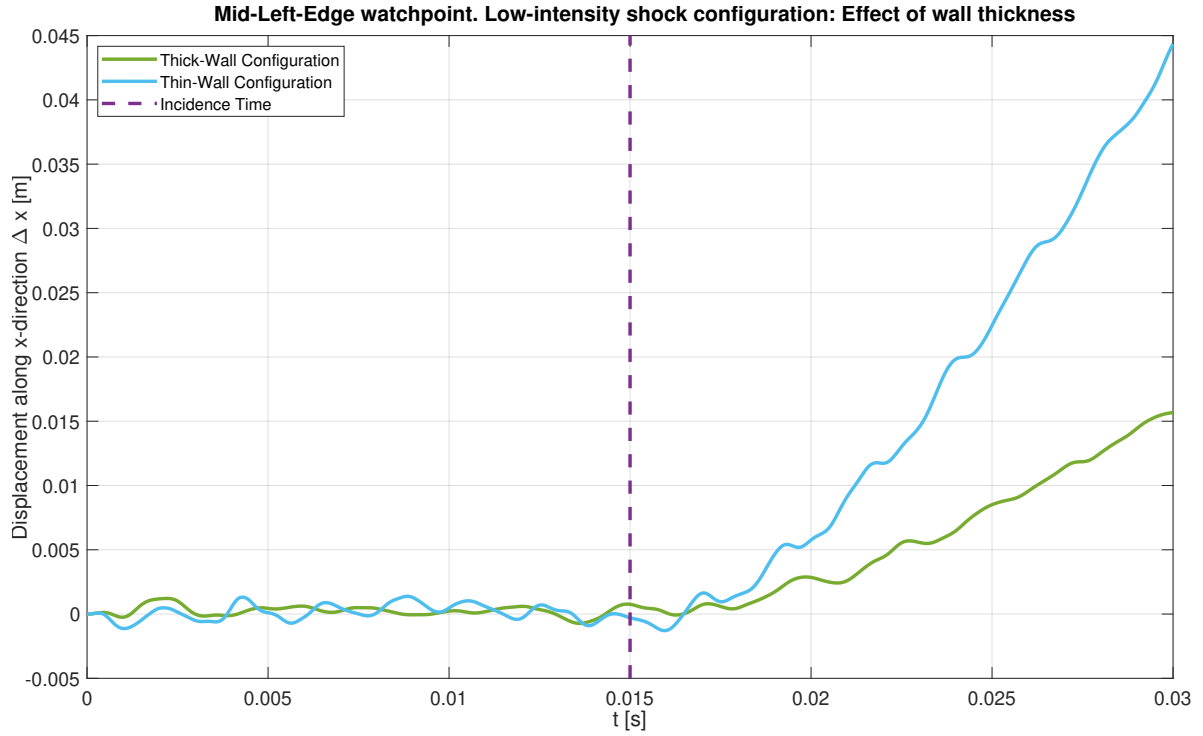


Figure 6.46: Mid-Left-Edge watchpoint: displacement time history along the x -direction. Investigation on the effect of wall thickness. Post-processed in MATLAB.

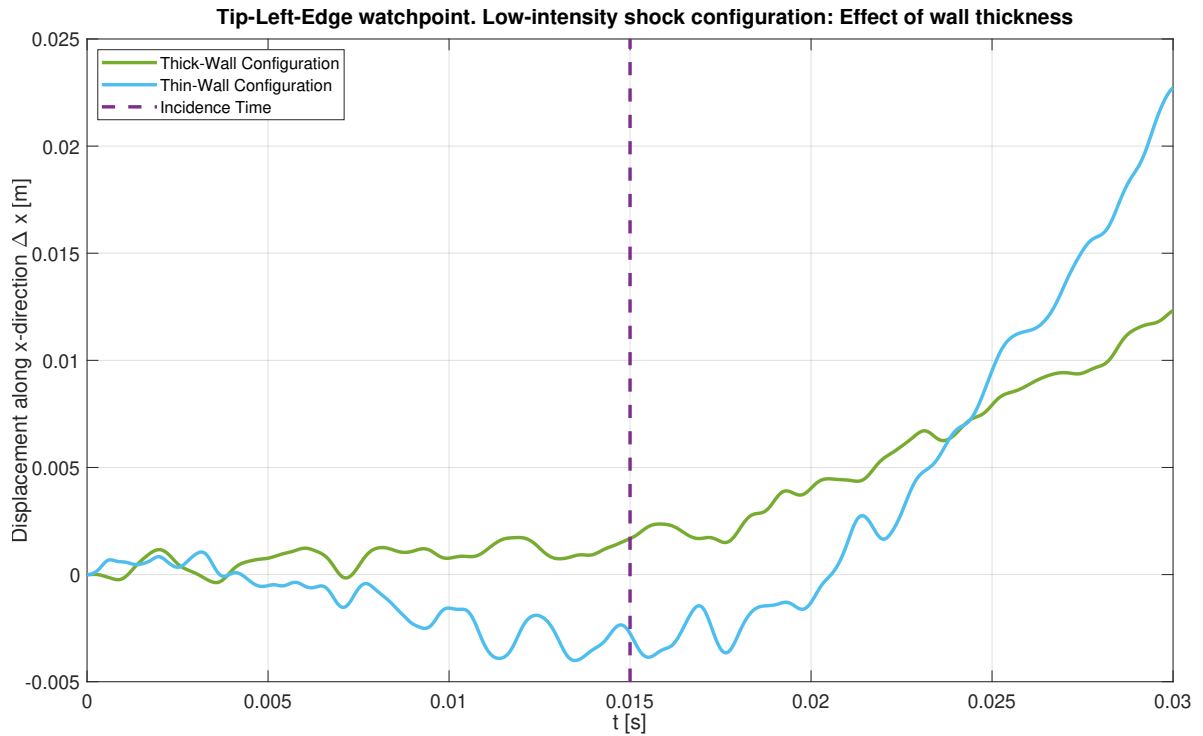


Figure 6.47: Tip-Left-Edge watchpoint: displacement time history along the x -direction. Investigation on the effect of wall thickness. Post-processed in MATLAB.

The negative values registered at the tip of the wall indicate that, in addition to a predictably larger displacement resulting from the wall's reduced stiffness while subjected to an identical loading condition, the wall tends to deform into a concave curvature, with its lower portion advancing ahead of the tip. Nevertheless, all displacements analyzed remain physically negligible with respect to the wall span and the domain characteristic dimensions, indicating that the reduced thickness has yet to be pushed to its structural limit.

	Interface watchpoints: displacement along x -direction [m]		
Time [s]	Bottom-Left-Edge	Mid-Left-Edge	Tip-Left-Edge
0.015	0.0012	$-2.8 \cdot 10^{-4}$	-0.0027
0.02	0.0067	0.0057	0.0013
0.03	0.0576	0.0443	0.0227

Table 6.7: **Thin-wall** coupled configuration at the **low-intensity** shock level: displacement comparison along the x -direction, for selected time instants, across all three watchpoints.

	Interface watchpoints: displacement along x -direction [m]					
	Bottom-Left-Edge		Mid-Left-Edge		Tip-Left-Edge	
Time [s]	Thick-Wall	Thin-Wall	Thick-Wall	Thin-Wall	Thick-Wall	Thin-Wall
0.015	$7.4 \cdot 10^{-4}$	0.0012	$7.6 \cdot 10^{-4}$	$-2.8 \cdot 10^{-4}$	0.0017	-0.0027
0.02	0.0035	0.0067	0.0029	0.0057	0.004	0.0013
0.03	0.0153	0.0576	0.0157	0.0443	0.0123	0.0227

Table 6.8: **Low-intensity** shock level: displacement comparison, for selected time instants, across all three watchpoints locations. Investigation on the effect of wall thickness.

6.4.3 Effect of blast wave intensity: thin-wall configuration at a high-intensity shock level

As the concluding characterization, the **thin-wall** configuration is tested under a more severe loading condition, obtained by doubling the blast wave intensity with respect to the **low-intensity** test cases. With all other parameters and the implementation of both domains left unchanged, the corresponding 300 000 Pa blast wave implementation is reported in Listing 6.3.

Under this loading condition, the recorded displacements increase substantially in magnitude, with the structural oscillations preceding the shock wave impact (associated with the non-linear hyperelastic constitutive response) becoming negligible by comparison. This confirms that the coupled structural response is no longer of comparable order to the underlying numerical instabilities, and it is therefore correctly resolved rather than obscured by numerical noise.

Since the fluid domain upstream pressure, initialized as the `volScalarFieldValue` `p` overpressure via `system/setFieldsDict`, has been increased to reflect the higher shock intensity, the corresponding shock wave properties yielded by the normal shock relations are updated accordingly, as reported in Table 6.9.

High-intensity blast wave: upstream properties	
Mach Number []	1.64
Temperature [K]	425
Density [kg/m ³]	2.57
Shock Propagation Velocity [m/s]	675.91
Instant of Shock Incidence [s]	0.008

Table 6.9: Upstream properties of the **high-intensity** blast wave, derived from the upstream pressure p_{blast} using the normal shock relations.

As in the previous configuration, the blast wave travel time is reproduced with reasonable accuracy: impact occurs at $t \approx 0.013$ s, yielding a relative error of 62.5 % against the theoretical estimate. This is attributable to the solver's known limitations in resolving sharp pressure gradients as noted in Section 6.4.1, which become substantially more pronounced under the increased blast intensity.

The ParaView post-processing results shown in Figure 6.48 reveal a discrepancy between the predicted shock Mach number and propagation velocity and the corresponding simulated values. The velocity field dataset immediately prior to the impact ($t = 0.013$ s) yields a maximum value of $U \approx 200$ m/s, corresponding to a relative error of 70.4 % with respect to the theoretical estimate. This discrepancy is likely attributable to numerical smearing of the shock front: the discontinuity is distributed over a widening region as the shock propagates rather than resolved as a sharp jump, consequently lowering the peak velocity captured within any single cell relative to the true post-shock state.

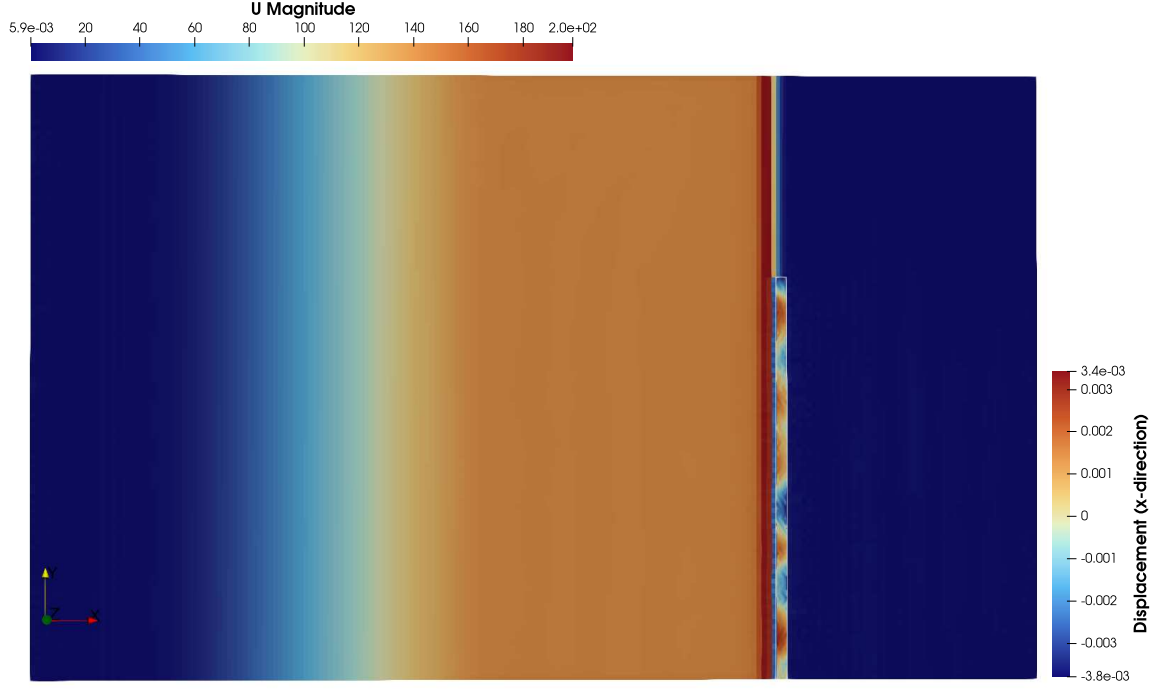


Figure 6.48: **High-intensity** shock level: magnitude of the velocity field at $t = 0.013$ s. Post-processed in ParaView.

Analogous field snapshots to those presented in Sections 6.4.1 and 6.4.2 are reported below (Figures 6.49 through 6.58) for the present configuration. As previously suggested, these results may additionally be compared against the uncoupled fluid response reported in Section 6.1.3 to isolate the influence of structural coupling from intensity effects.

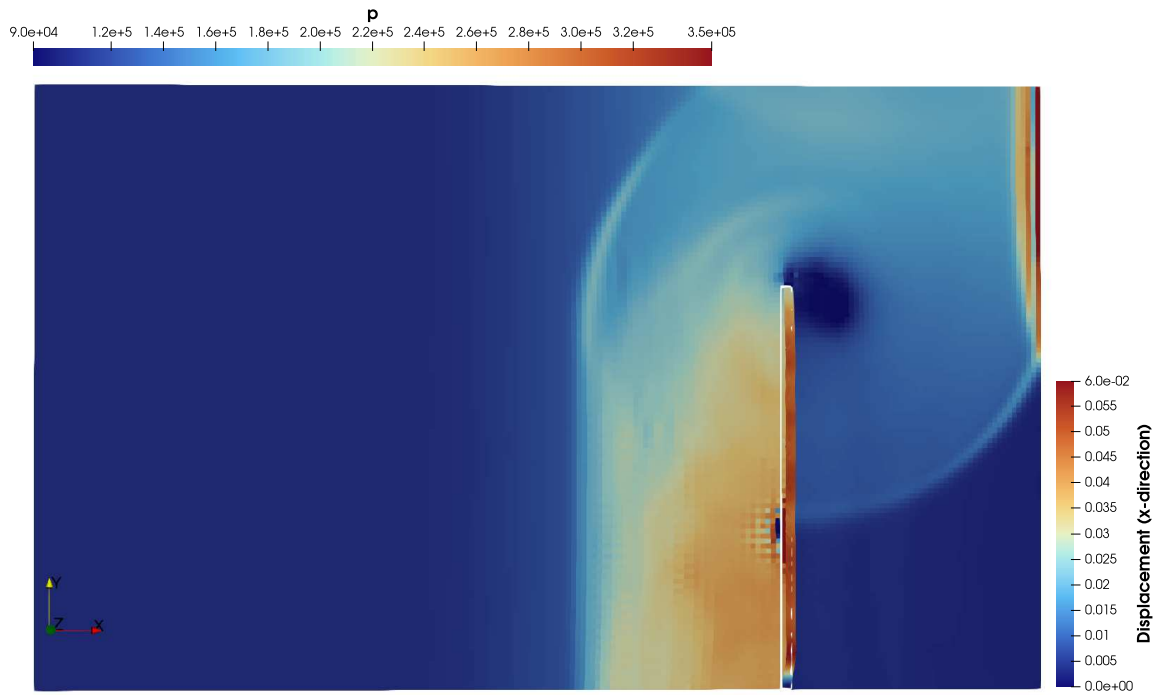


Figure 6.49: Thin-wall coupled configuration at the high-intensity shock level: pressure distribution at $t = 0.2$ s. Post-processed in ParaView.

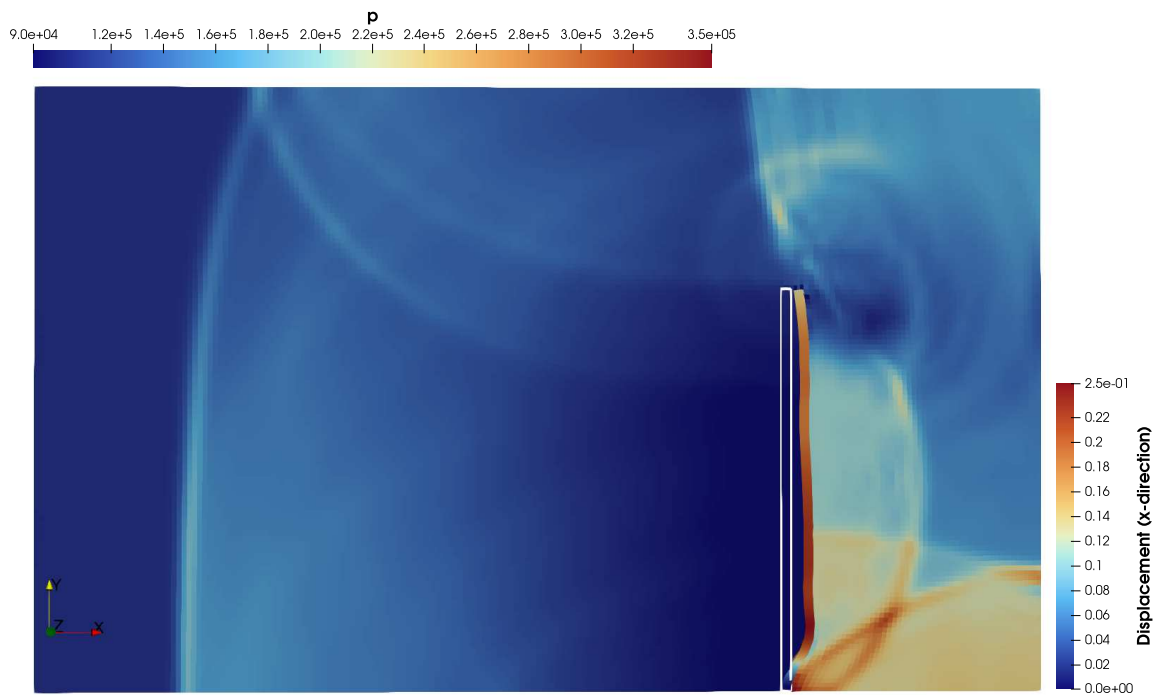


Figure 6.50: Thin-wall coupled configuration at the high-intensity shock level: pressure distribution at $t = 0.3$ s. Post-processed in ParaView.

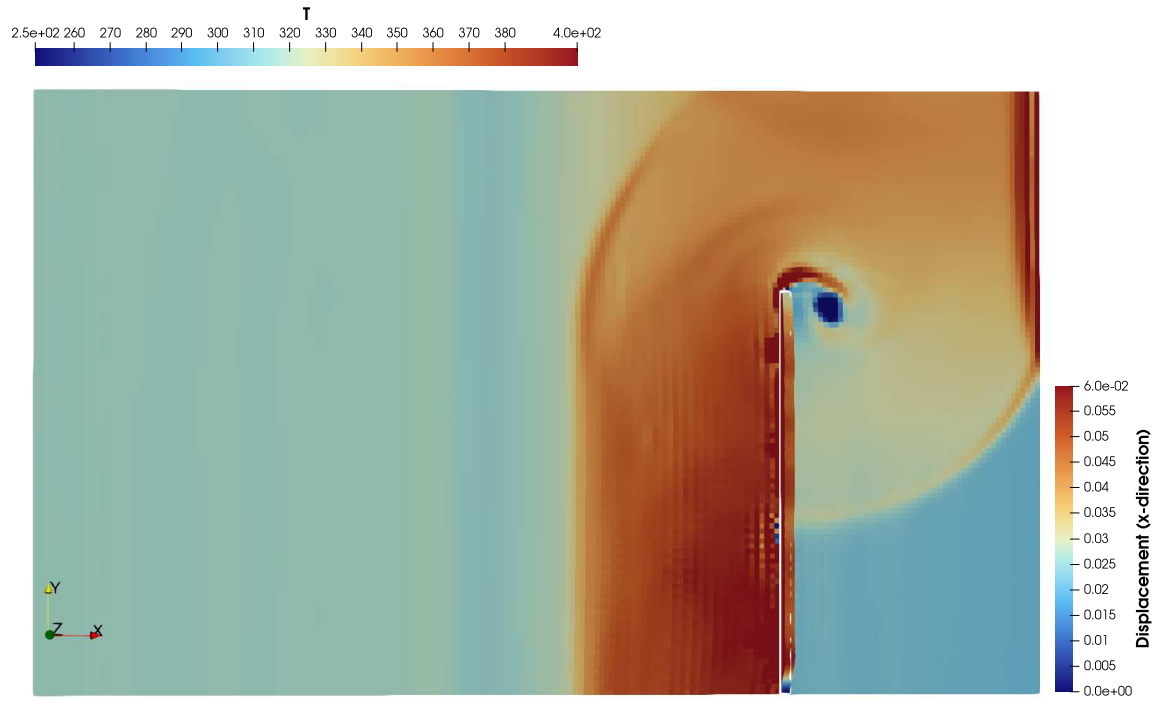


Figure 6.51: Thin-wall coupled configuration at the high-intensity shock level: temperature distribution at $t = 0.2$ s. Post-processed in ParaView.

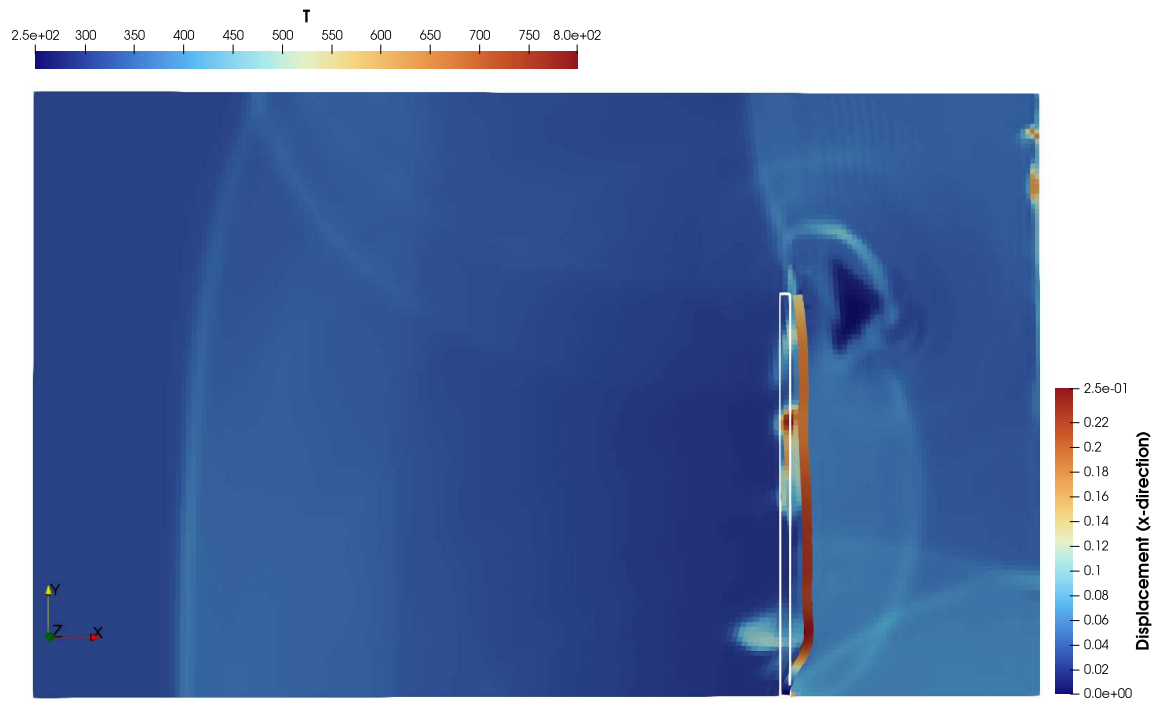


Figure 6.52: Thin-wall coupled configuration at the high-intensity shock level: temperature distribution at $t = 0.3$ s. Post-processed in ParaView.

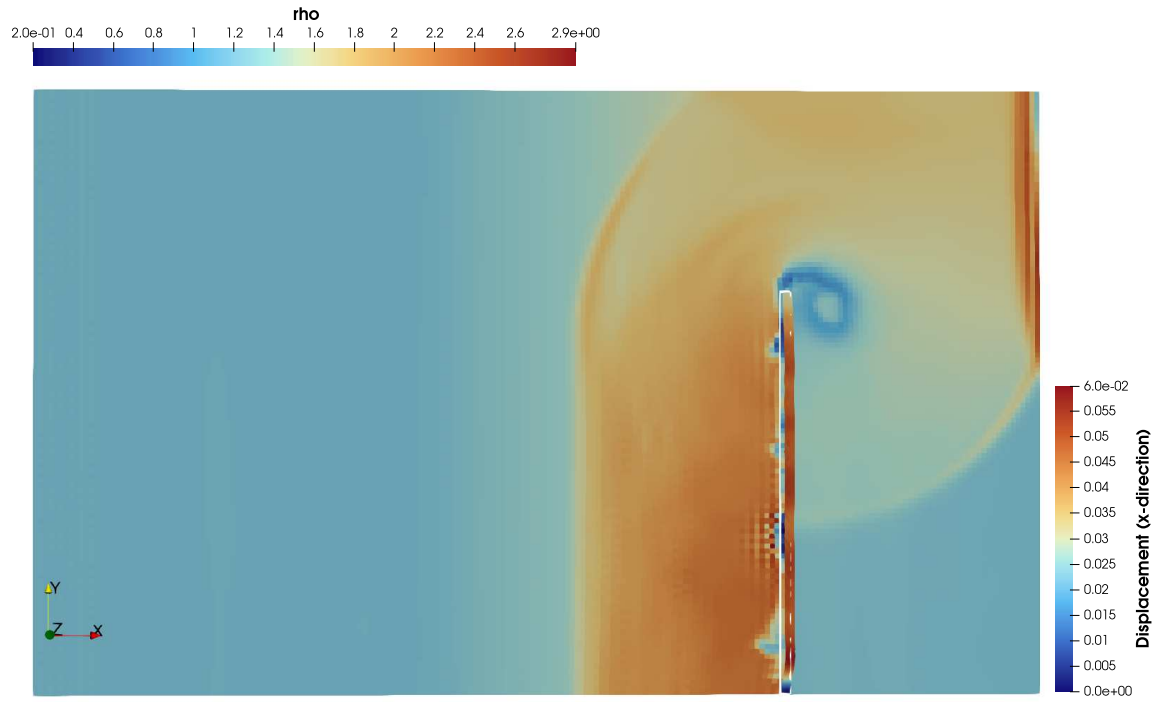


Figure 6.53: Thin-wall coupled configuration at the high-intensity shock level: density distribution at $t = 0.2$ s. Post-processed in ParaView.

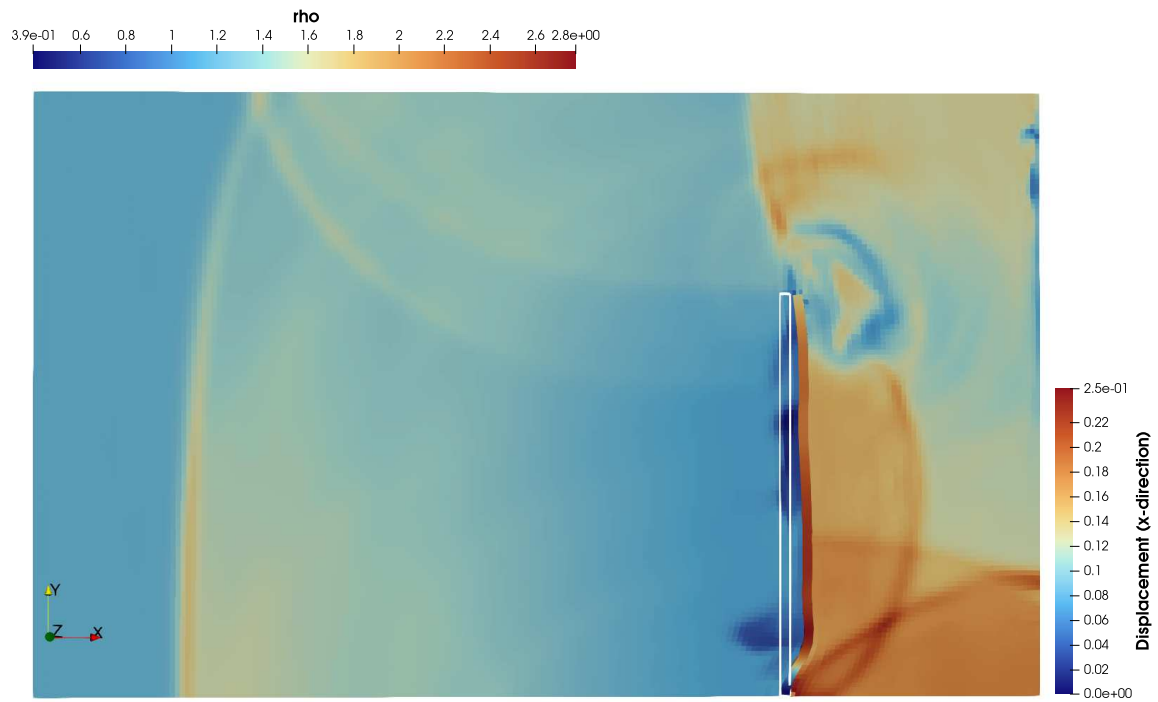


Figure 6.54: Thin-wall coupled configuration at the high-intensity shock level: density distribution at $t = 0.3$ s. Post-processed in ParaView.

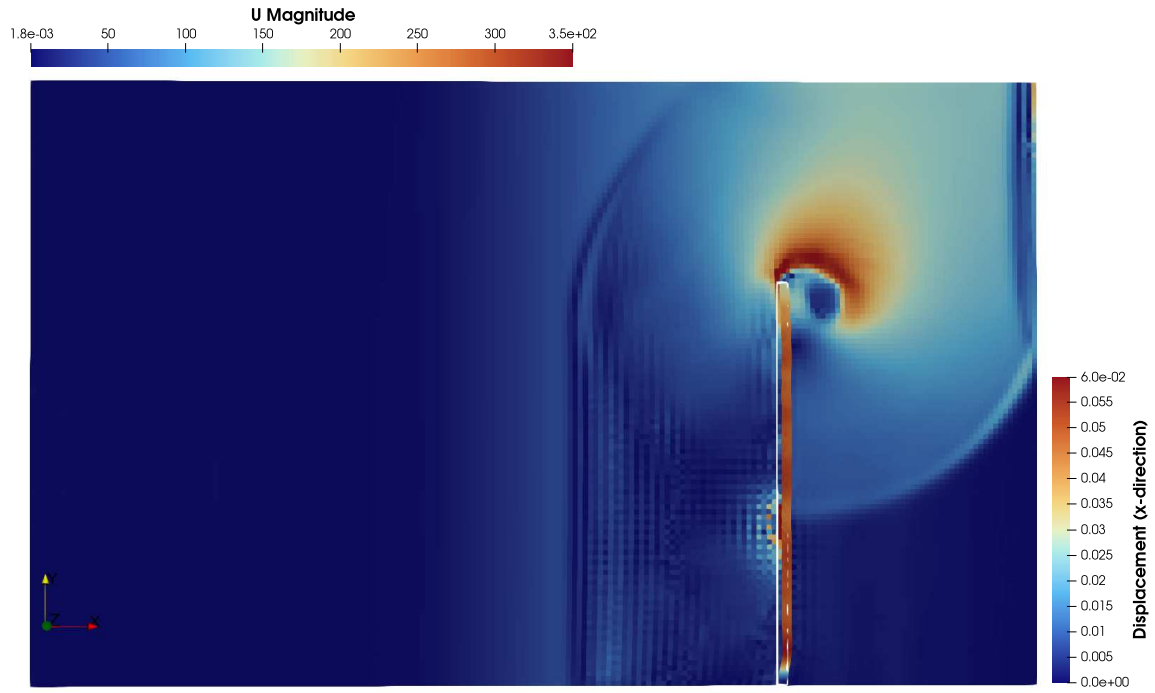


Figure 6.55: Thin-wall coupled configuration at the high-intensity shock level: magnitude of the velocity field at $t = 0.2$ s. Post-processed in ParaView.

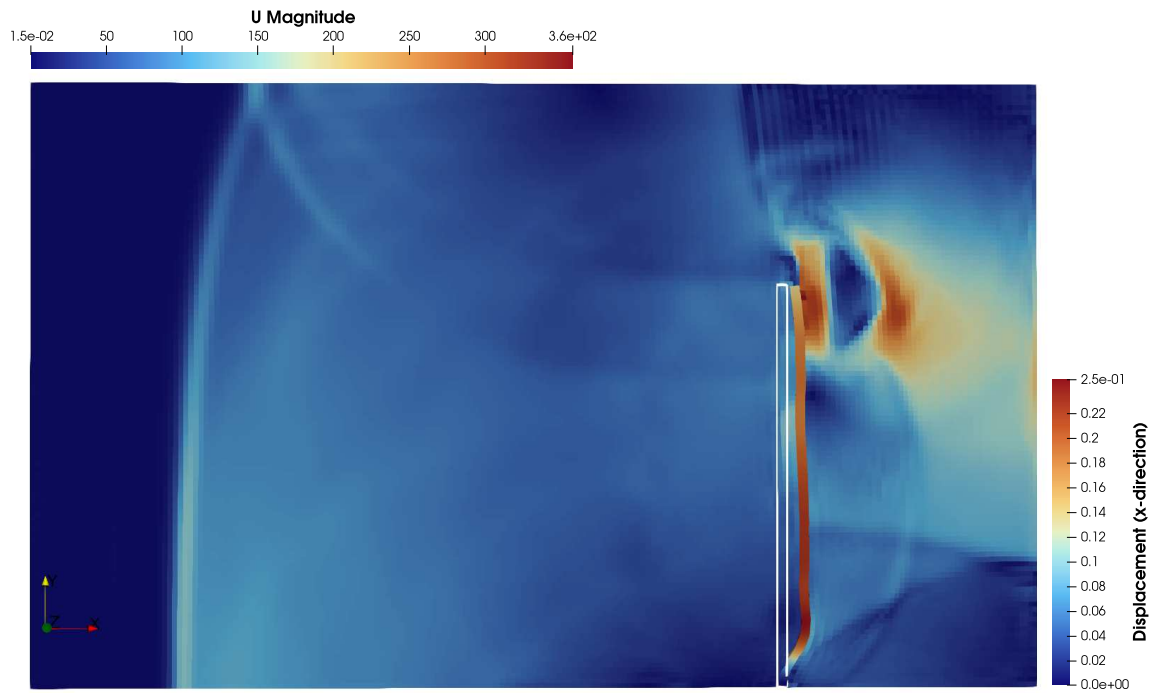


Figure 6.56: Thin-wall coupled configuration at the high-intensity shock level: magnitude of the velocity field at $t = 0.3$ s. Post-processed in ParaView.

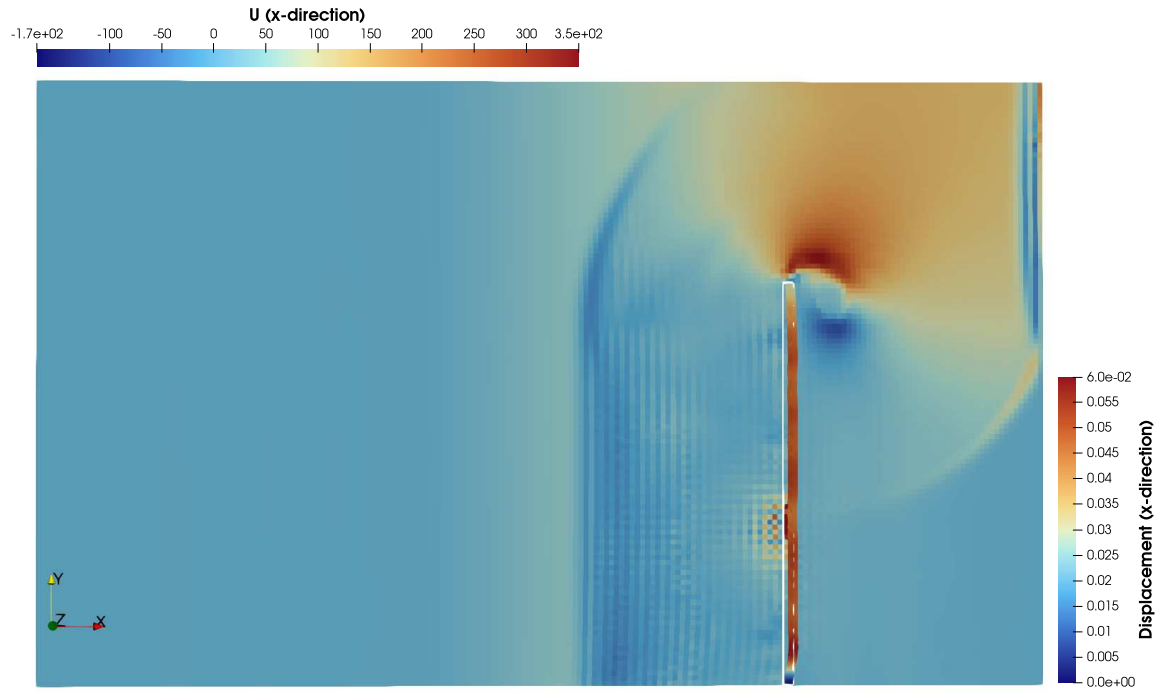


Figure 6.57: Thin-wall coupled configuration at the **high-intensity** shock level: flow velocity along the x -direction at $t = 0.2$ s. Post-processed in ParaView.

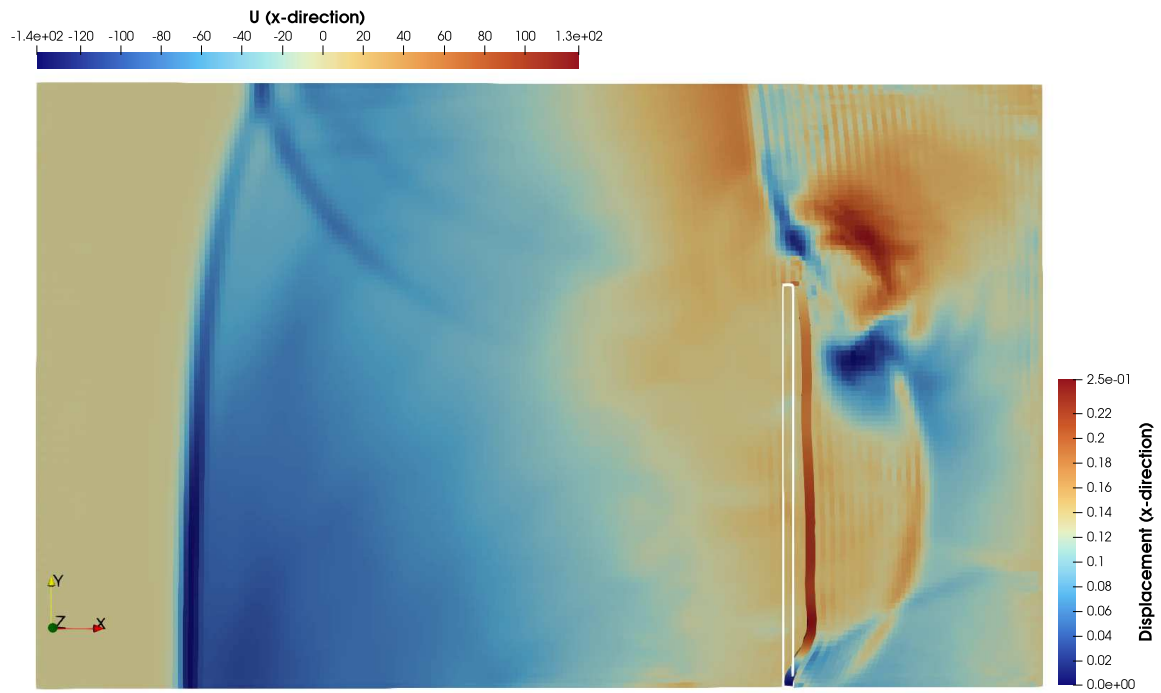


Figure 6.58: Thin-wall coupled configuration at the **high-intensity** shock level: flow velocity along the x -direction at $t = 0.3$ s. Post-processed in ParaView.

A visual assessment of the configuration at $t = 0.03$ s indicates that the wall is substantially more bent relative to the **low-intensity** test case. This observation is supported by the post-processed watchpoint time histories reported in the figures and tables below, which provide both a comparative overview of the two **thin-wall** configurations across all monitored locations and a combined plot overlaying all three displacement time histories along the x -direction for the **high-intensity** test case. The overlaid plot in particular evidences a pronounced curling deformation, with the maximum deflection occurring closer to the constraint and displacement remaining comparatively smaller at the tip across its full range.

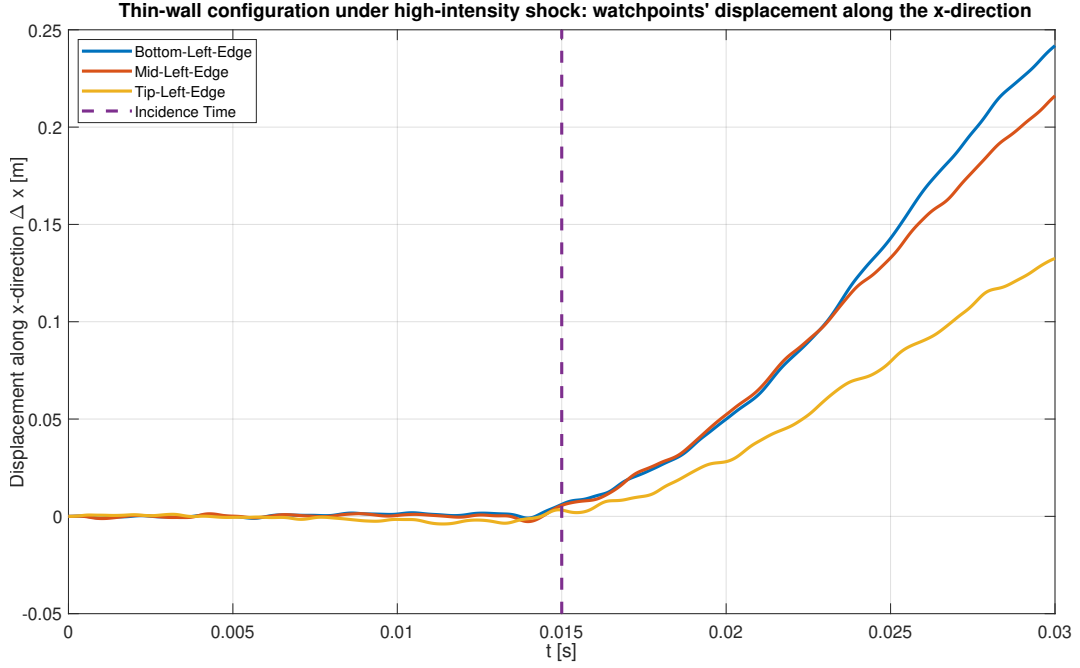


Figure 6.59: **Thin-wall** coupled configuration at the **high-intensity** shock level: overlaid displacement time histories along the x -direction for comparison across all three watchpoints. Post-processed in MATLAB.

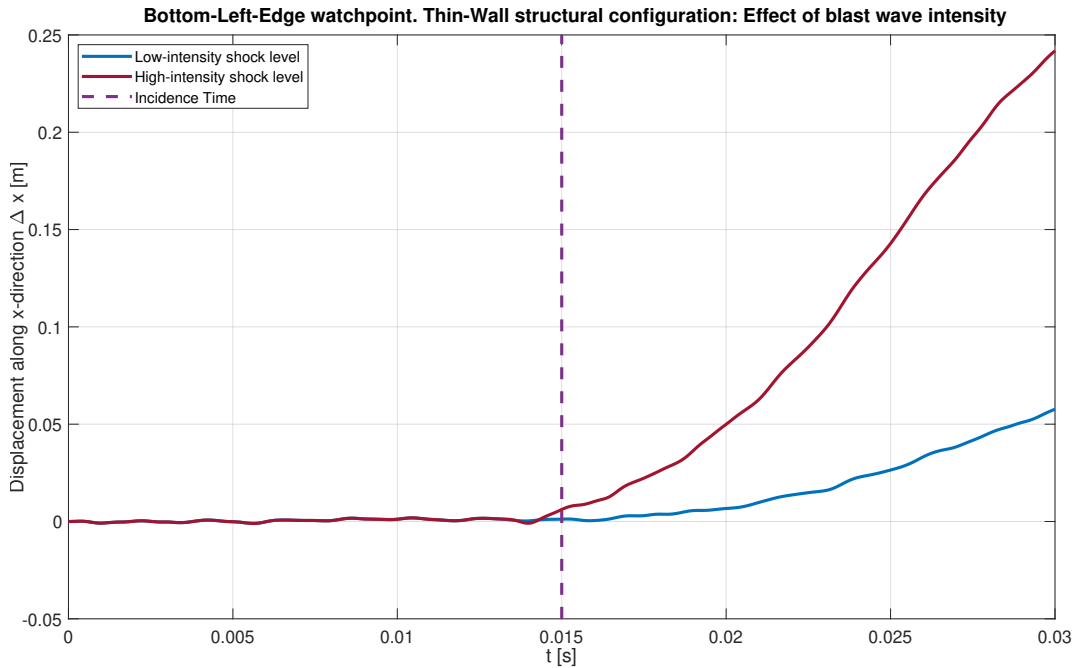


Figure 6.60: **Bottom-Left-Edge** watchpoint: displacement time history along the x -direction. Investigation on the effect of blast wave intensity. Post-processed in MATLAB.

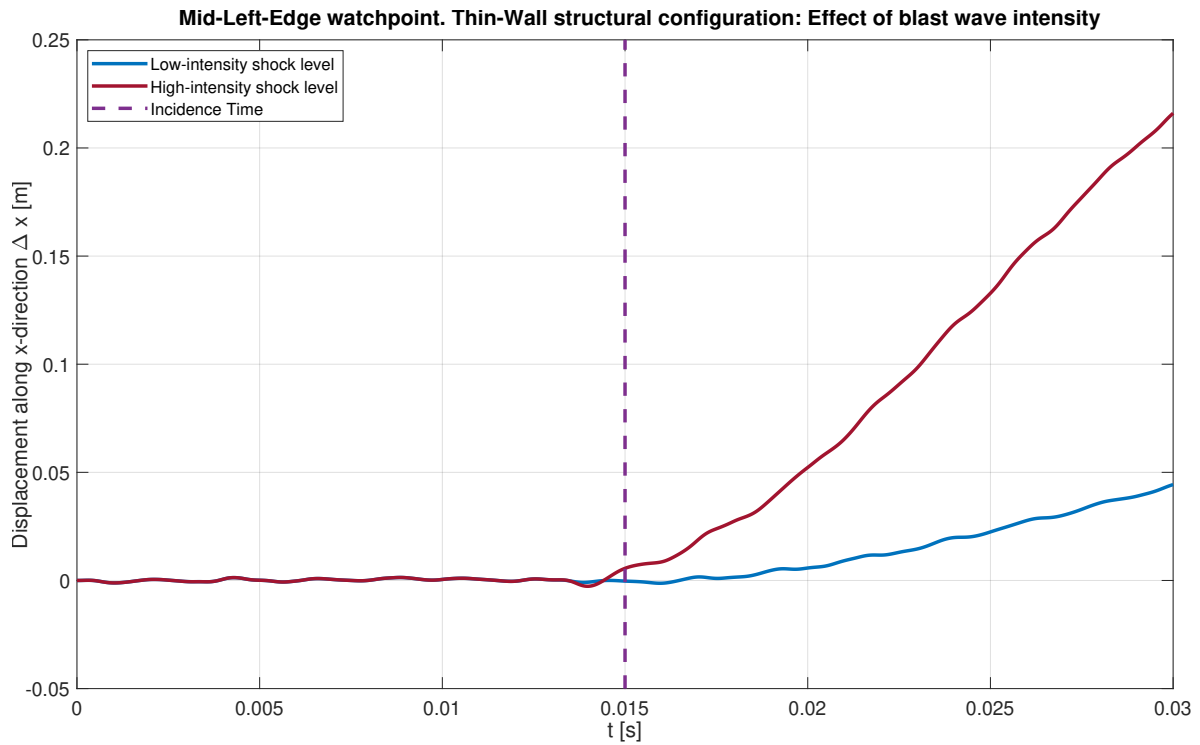


Figure 6.61: Mid-Left-Edge watchpoint: displacement time history along the x -direction. Investigation on the effect of blast wave intensity. Post-processed in MATLAB.

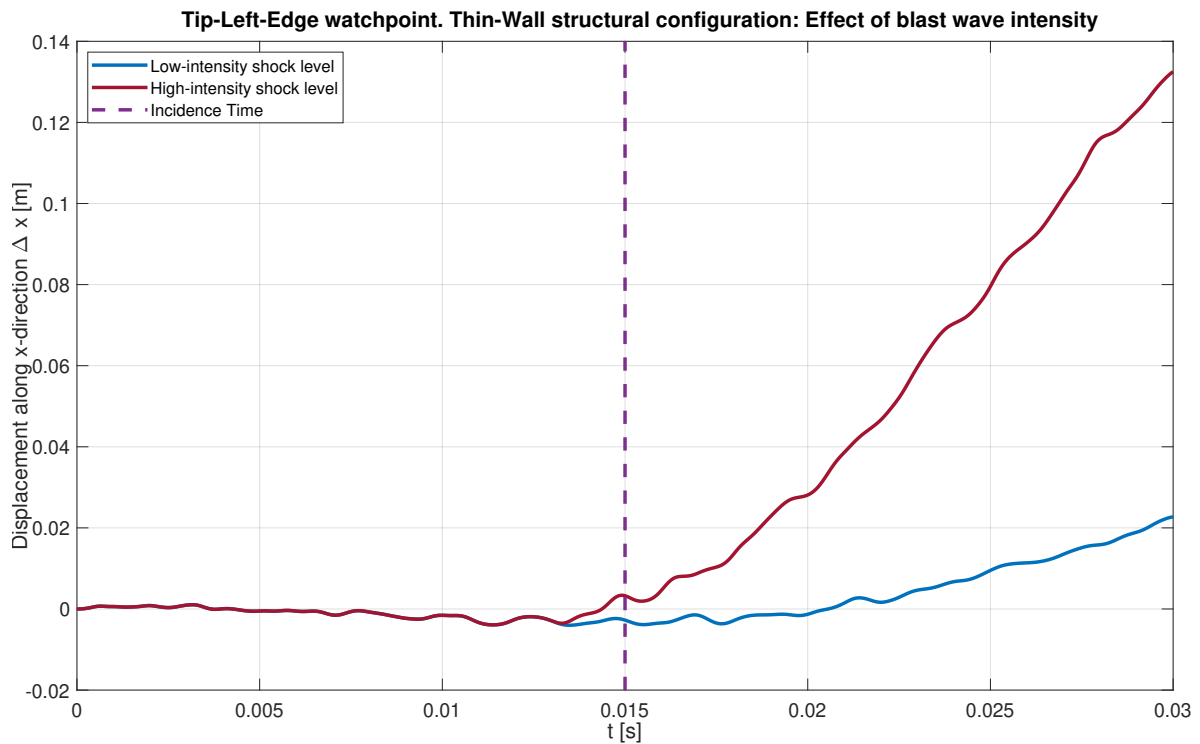


Figure 6.62: Tip-Left-Edge watchpoint: displacement time history along the x -direction. Investigation on the effect of blast wave intensity. Post-processed in MATLAB.

	Interface watchpoints: displacement along x -direction [m]		
Time [s]	Bottom-Left-Edge	Mid-Left-Edge	Tip-Left-Edge
0.015	0.0061	0.0055	0.0033
0.02	0.0498	0.0521	0.0281
0.03	0.2417	0.2159	0.1324

Table 6.10: **Thin-wall** coupled configuration at the **high-intensity** shock level: displacement comparison along the x -direction, for selected time instants, across all three watchpoints.

	Interface watchpoints: displacement along x -direction [m]					
	Bottom-Left-Edge		Mid-Left-Edge		Tip-Left-Edge	
Time [s]	Low-Intensity	High-Intensity	Low-Intensity	High-Intensity	Low-Intensity	High-Intensity
0.015	0.0012	0.0061	$-2.8 \cdot 10^{-4}$	0.0055	-0.0027	0.0033
0.02	0.0067	0.0498	0.0057	0.0521	0.0013	0.0281
0.03	0.0576	0.2417	0.0443	0.2159	0.0227	0.1324

Table 6.11: **Thin-Wall** structural configuration: displacement comparison, for selected time instants, across all three watchpoints locations. Investigation on the effect of blast wave intensity.

In conclusion, the present configuration remains in need of significant refinement given the anomalous trends and inconsistencies observed throughout the current chapter: the fluid solver’s known limitations in representing sharp discontinuities remain a likely source of the discrepancies discussed above, and further numerical work would help establish whether the observed wall behavior reflects the true physical response or is instead an artifact of the current setup. More specifically, a dedicated test campaign employing an equivalent structural participant modeled with a different solver, with the aim of replicating the previously adopted validation strategy, would serve as a benchmark for the corresponding material characterization and indicate whether additional refinement of the OpenRadioss configuration is warranted. Nevertheless, these results already constitute a promising preliminary outcome, as they represent the first successful demonstration of the previously undocumented coupled configuration under the conditions considered. Table 6.12 reports the computational cost associated with each case, which remains relatively low across all configurations. Despite the fluid solver’s limitations and the current uncertainties pertaining to the OpenRadioss characterization, increasing numerical complexity and pursuing further investigation into convergence behavior, predictability characterization and benchmark validation would remain computationally affordable as considerably larger `clockTime` values could be reached without becoming prohibitive.

	Low-Intensity Thick-Wall	Low-Intensity Thin-Wall	High-Intensity Thin-Wall
Elapsed Time [s]	5 680.49	6 302.60	5 745.62
Elapsed Time [hh:mm:ss]	1:34:40	1:45:02	1:35:45

Table 6.12: Total computational cost for each configuration of the **Blast Wave Impact against a Structural Wall** test case.

Taken together, these findings establish a robust foundation for the continued development and refinement of partitioned fluid–structure interaction frameworks through the incorporation of OpenRadioss as a structural solver within the preCICE coupling environment. Having succeeded in characterizing previously unexplored multiphysics phenomena, this groundwork supports the future application of OpenRadioss to increasingly complex numerical scenarios.

Chapter 7

Conclusions and outlook

This thesis establishes the first successful coupling of OpenFOAM and OpenRadioss through systematic exploration of simulation configurations and targeted debugging efforts of the OpenRadioss adapter prototype, supported by the preCICE and OpenRadioss development teams.

The work was developed as a structured investigation aimed at integrating the OpenRadioss software within the preCICE ecosystem for partitioned fluid-structure interaction applications, thereby addressing the absence of prior documentation concerning implementations of this kind. Accordingly, the test cases configured in this research were designed to serve as an original reference, thus establishing validated procedures to guide future implementations. Building on this methodology, the thesis was structured around two independent configurations, each targeting a distinct aspect of the coupling’s validation objective: the **Pulsating Flow over Flexible Flap** case enabled quantitative assessment against reference data provided by Chourdakis et al.[40] and Schott et al.[39], while the **Blast Wave Impact against a Structural Wall** case yielded the first characterization of a previously unexplored structural response in synchronized coupled interactions.

The opening chapters of this thesis reviewed the fundamental principles of fluid mechanics and structural dynamics to establish the foundation supporting the applications discussed, while the fundamentals of numerical coupling underlying partitioned frameworks within preCICE were subsequently addressed to outline coupling strategies, data mapping at the shared interface, and configuration architecture governing software participation. Based on these theoretical foundations, the two test cases central to this research were implemented, investigated, and thoroughly discussed in the final chapters of this manuscript. The positive post-processing results observed for the **Pulsating Flow over Flexible Flap** simulation enabled validation of the established coupling against two independent references: a partitioned benchmark reported by Chourdakis et al.[40] and monolithic results provided by Schott et al.[39]. The same coupling approach was then extended to characterize the **Blast Wave Impact against a Structural Wall** case.

Owing to its underlying numerical formulation, the pressure-based fluid solver selected for this study (i.e. `rhoPimpleFoam`) failed to accurately represent sharp pressure discontinuities encountered during the **Blast Wave Impact against a Structural Wall** test case; this consequently produced spurious oscillations and physical inconsistencies in the resulting fluid domain, without yet precluding its use as a preliminary basis to characterize the fluid response under interaction with a medium of different physical nature. As a direction for future work and a potential improvement upon the present limitation, the adoption of a density-based solver (e.g. `rhoCentralFoam`) extended to support dynamic mesh morphing should be investigated as a means of improving shock-capturing accuracy, in the pursuit of a more robust and higher-fidelity numerical characterization.

Further refinement and development of the OpenRadioss adapter beyond the preliminary stages assessed with this work may additionally extend its range of applicability within the preCICE coupling environment. Support for checkpoint writing, which would enable the selection of implicit coupling schemes and the consequent relaxation of time-step size constraints, could only be achieved through targeted modifications to the software’s source code itself. This would allow the solver’s internal state to be restored in full upon convergence failure within a specific coupling time window, and permit the coupling iteration to be repeated with updated boundary conditions without a complete simulation restart. Beyond checkpoint-writing capabilities, extending the adapter’s support to two-dimensional coupling scenarios requires further investigation in collaboration with the OpenRadioss development team. This follows from a concern, raised by the developers themselves, that shell elements discretizing a planar geometry are handled internally as three-dimensional elements (complete with thickness and 3×3 tensor operations) instead of being treated as two-dimensional (plane-strain) cases. Since no validated plane-strain reference could be provided within the timeframe of this study, the configuration of this class of problems remains an open question worth addressing in future releases; nevertheless, to make these limitations available for further consideration, they were reported via the software’s official GitHub repository.

In conclusion, the results presented in this thesis demonstrate the first successful configuration of a previously unexplored coupled scenario despite the initial immaturity of some of the tools involved and persistent limitations that prevented certain cases from being configured. While a dedicated test

campaign involving an equivalent structural solver could benchmark and help explain the anomalous trends observed throughout the final chapter of this work, these findings establish a robust foundation for the continued development and refinement of partitioned fluid–structure interaction frameworks incorporating OpenRadioss within the preCICE coupling environment, paving the way for future applications in increasingly complex numerical scenarios.

Bibliography

- [1] preCICE Documentation. (2026). Terminology. In: *Fundamentals*. <https://precice.org/fundamentals-terminology>
- [2] Weller, H. G., Tabor, G., Jasak, H., & Fureby, C. (1998). A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in Physics*, 12(6), 620–631. <https://doi.org/10.1063/1.168744>
- [3] Jasak, H. (2009). OpenFOAM: Open source CFD in research and industry. *International Journal of Naval Architecture and Ocean Engineering*, 1(2), 89–94. <https://doi.org/10.2478/ijnaoe-2013-0011>
- [4] Stallman, R. M. (2002). The GNU Project. In *Free software, free society: Selected essays of Richard M. Stallman* (pp. 17–32). GNU Press.
- [5] OpenFOAM User Guide. (2026). <https://www.openfoam.com/documentation/user-guide>
- [6] Okoye, O. C., & Bolaji, B. O. (2020). Physics of fluid motion. In *Elsevier eBooks* (pp. 1–22). <https://doi.org/10.1016/b978-0-12-817949-9.00009-8>
- [7] Moukalled, F., Mangani, L., & Darwish, M. (2015). Mathematical description of physical phenomena. In *The Finite Volume Method in Computational Fluid Dynamics. Fluid mechanics and its applications* (pp. 43–83). https://doi.org/10.1007/978-3-319-16874-6_3
- [8] Houghton, E. L., Carpenter, P. W., Collicott, S. H., & Valentine, D. T. (2017b). Compressible flow. In *Elsevier eBooks* (pp. 255–327). <https://doi.org/10.1016/b978-0-08-100194-3.00004-3>
- [9] Moukalled, F., & Darwish, M. (2015). Fluid Flow Computation: Compressible Flows. In: *The Finite Volume Method in Computational Fluid Dynamics. Fluid Mechanics and Its Applications*, (pp. 655–690). https://doi.org/10.1007/978-3-319-16874-6_16
- [10] Mangani, L., Sanz, W., & Darwish, M. (2016). Comparing the performance and accuracy of a pressure-based and a density-based coupled solver. In *16th International Symposium on Transport Phenomena and Dynamics of Rotating Machinery (ISROMAC-16)*, Honolulu, United States. <https://hal.archives-ouvertes.fr/hal-01894391>
- [11] Holzmann, T. (2019). The numerical algorithms: SIMPLE, PISO and PIMPLE. In *Mathematics, Numerics, Derivations and OpenFOAM[®]*, (pp. 93–122). <https://Holzmann-cfd.de>
- [12] Wang, K., Jin, G., Ye, T., Liu, H., Chen, Y., & Zhang, B. (2025b). A high order and efficient acoustic solver using parallel finite volume method. *Physics of Fluids*, 37(4). <https://doi.org/10.1063/5.0268914>
- [13] Koetsenruijter, D. (2023). DSM research exercise. *ResearchGate*. https://www.researchgate.net/publication/371180076_DSM_research_exercise
- [14] Nobels, F. S. J. (2025). V2004-WP3.1: OpenRadioss as a reliable replacement for LS-DYNA. In *Report No. TNO 2024 R11057A*. TNO. <https://openradioss.org/wp-content/uploads/2025/08/TNO-2024-R11057A.pdf>
- [15] Haug, E. (1981). Engineering safety analysis via destructive numerical experiments. *Engineering Transactions*, 29(1), (pp. 39–50). <https://et.ippt.pan.pl/index.php/et/article/view/1990>
- [16] Haug, E., Scharnhorst, T. & Du Bois, P. (1986): FEM-Crash Berechnung eines Fahrzeugfrontaufpralls. In *VDI Berichte*, 613, (pp. 479–505).
- [17] OpenRadioss Community. (2024). Tools/openRadioss_gui [Repository]. In *OpenRadioss/Tools*. GitHub. https://github.com/OpenRadioss/Tools/tree/main/openradioss_gui
- [18] preCICE Tutorials. (2026) Perpendicular Flap. <https://precice.org/tutorials-perpendicular-flap>
- [19] Bonet, J., & Wood, R. D. (2008). HYPERELASTICITY. In: *Cambridge University Press eBooks*, (pp. 155–187). <https://doi.org/10.1017/cbo9780511755446.007>

- [20] Trobentar, B., Glodež, S., & Zafošnik, B. (2014). Gear tooth deflection of spur polymer gears. In *Elsevier eBooks*, (pp. 129–137). <https://doi.org/10.1533/9781782421955.129>
- [21] Eltobgy, H., Refaee, M. S., Aboshok, M. A., & Emadel-Din, A. (2024). Influence of Concrete Jacketing on the Performance of Steel Columns under Blast induced Progressive Collapse. *Journal of Engineering Research*, 8(4), 18. <https://doi.org/10.13140/rg.2.2.35565.29920>
- [22] Chen, L., Li, T., Liu, L., Wang, W., Du, X., & Wang, W. (2024). Isogeometric analysis of hyperelastic material characteristics for calcified aortic valve. *Computer Modeling in Engineering & Sciences*, 139(3), 2773–2806. <https://doi.org/10.32604/cmes.2024.046641>
- [23] Radioss User Guide. (2026). Ogden Materials. In: *Explicit Structural Finite Element Analysis: Materials*. https://help.altair.com/hwsolvers/rad/topics/solvers/rad/explicit_structural_fea_hyperelastic_ogden_materials_r.htm
- [24] Logan, D. L. (2007). Three-Dimensional Stress Analysis. In: *A first course in the finite element method*, (pp. 490–514). ISBN: 0-534-55298-6.
- [25] Chourdakis, G., Davis, K., Rodenberg, B., Schulte, M., Simonis, F., Uekermann, B., Abrams, G., Bungartz, H.-J., Yau, L. C., Desai, I., Eder, K., Hertrich, R., Lindner, F., Rusch, A., Sashko, D., Schneider, D., Totounferoush, A., Volland, D., Vollmer, P., & Koseomur, O. Z. (2022). preCICE v2: A sustainable and user-friendly coupling library. *Open Research Europe*, 2, 51. <https://doi.org/10.12688/openreseurope.14445.2>
- [26] preCICE Documentation. (2026). Non-matching time step sizes. In: *Couple your code: Step 5*. <https://precice.org/couple-your-code-time-step-sizes>
- [27] Chourdakis, G. (2017). A general OpenFOAM adapter for the coupling library preCICE. *Master's thesis*. Department of Informatics, Technical University of Munich. https://www.researchgate.net/publication/321758082_A_general_OpenFOAM_adapter_for_the_coupling_library_preCICE
- [28] preCICE Documentation. (2026). Introduction to configuration. In: *Configuration: Basics*. <https://precice.org/configuration-introduction>
- [29] preCICE Documentation. (2026). Config visualization. In: *Tooling: config visualization*. <https://precice.org/tooling-config-visualization>
- [30] preCICE Documentation. (2026). Configure the OpenFOAM adapter. In: *Provided adapters: OpenFOAM*. <https://precice.org/adapter-openfoam-config>
- [31] preCICE Tutorials. (2026) Partitioned elastic beam. <https://precice.org/tutorials-partitioned-elastic-beam>
- [32] Albert, A. (2026). Schneider - Bautabellen für Ingenieure: mit Berechnungshinweisen und Beispielen.
- [33] OpenRadioss. (2026). *Coupling: preCICE adapter reads the wrong data type*. Issue #4565. GitHub. <https://github.com/OpenRadioss/OpenRadioss/issues/4565>
- [34] OpenRadioss. (2026). *Fix preCICE adapter reading wrong data type for displacements*. Pull request #4566. GitHub. <https://github.com/OpenRadioss/OpenRadioss/pull/4566>
- [35] preCICE. (2024). *Add more watchpoint plots and source files*. Pull request #507. GitHub. <https://github.com/precice/tutorials/pull/507#issuecomment-2043120253>
- [36] OpenRadioss. (2026). *preCICE coupling: time step size is not correctly adjusted to minimum allowed*. Issue #4699. GitHub. <https://github.com/OpenRadioss/OpenRadioss/issues/4699>
- [37] elisa-santoro. (2026). *FSI OpenFOAM - OpenRadioss Coupling: Results with vs without a Radioss Fixed Time Step*. Issue #35. GitHub. <https://github.com/elisa-santoro/openradioss-cases/issues/35>
- [38] OpenRadioss. (2026). *preCICE coupling: bug fix in timestep reduction*. Pull request #4706. GitHub. <https://github.com/OpenRadioss/OpenRadioss/pull/4706>

- [39] Schott, B., Ager, C., & Wall, W. A. (2019). Monolithic cut finite element-based approaches for fluid-structure interaction. *International Journal for Numerical Methods in Engineering*, 119(8), 757–796. <https://doi.org/10.1002/nme.6072>
- [40] Chourdakis, G., Schneider, D., & Uekermann, B. (2023). OpenFOAM-preCICE: Coupling OpenFOAM with External Solvers for Multi-Physics Simulations. *OpenFOAM® Journal*, 3, 1–25. <https://doi.org/10.51560/ofj.v3.88>
- [41] Santoro, E. (2026). *pulsating-flow-over-flexible-flap-3d-case2*. GitHub repository: openradioss-cases. GitHub. <https://github.com/elisa-santoro/openradioss-cases/tree/master/pulsating-flow-over-flexible-flap-3d-case2>
- [42] Gerstenberger, A., & Wall, W. A. (2010). An embedded Dirichlet formulation for 3D continua. *International Journal for Numerical Methods in Engineering*, 82(5), 537–563. <https://doi.org/10.1002/nme.2755>
- [43] Santoro, E. (2026). *blast-problem-coupled-1n5bar-M20-rubberWall*. GitHub repository: openradioss-cases. GitHub. <https://github.com/elisa-santoro/openradioss-cases/tree/master/blast-problem-coupled-1n5bar-M20-rubberWall>
- [44] Santoro, E. (2026). *blast-problem-coupled-3bar-M20-rubberWall*. GitHub repository: openradioss-cases. GitHub. <https://github.com/elisa-santoro/openradioss-cases/tree/master/blast-problem-coupled-3bar-M20-rubberWall/case1-100mmWall>
- [45] Villani, V., & Lavallata, V. (2020b). Entanglement Locking in the Unique Elasticity of Polydimethylsiloxane Rubbers. *Macromolecular Chemistry and Physics*, 221(10). <https://doi.org/10.1002/macp.201900497>
- [46] Lu, S., Li, Y., & Suo, T. (2025). Design and blast mitigation of solid pyramid structure for personnel. *Scientific Reports*, 15(1), 34961. <https://doi.org/10.1038/s41598-025-18833-x>

Acknowledgements

The results achieved in this work are the product of a collective effort, made possible by the significant support and contributions of all those involved over the course of this ambitious project. I would like to thank Prof. Andrea Ferrero and Prof. Dario Pastrone, my supervisors from Politecnico di Torino, for giving me the chance to pursue this opportunity and supporting me throughout the collaboration with the Bundeswehr University Munich.

I am sincerely grateful to Prof. Markus Klein and Dr. Josef Hasslberger for hosting me at the Institute of Applied Mathematics and Scientific Computing during my time in Munich. Their continued assistance, expertise, and insight have been crucial over the months I spent conducting this research.

I would also thank Dr. Georgios Michaloudis and Dr. Matthias Andrae, from the Institute of Mechanics and Statics, for their help and guidance as I worked through unfamiliar concepts and challenges inherent to structural FEM analysis, and the OpenRadioss development team for the technical support they provided via GitHub throughout the fault-reporting process.

A heartfelt and special thanks goes to Gerasimos Chourdakis and Carme Homs-Pons from the Institute for Parallel and Distributed Systems at the University of Stuttgart: thank you for the time and dedication invested in my progress through the countless emails and scheduled meetings, the assistance provided and guidance offered whenever needed, and the tireless effort devoted toward ensuring a positive research experience.

Finally, my deepest gratitude goes to my family and friends for the continued encouragement and invaluable support offered throughout every stage of my studies. Questo traguardo è tanto mio quanto vostro. ♡