Alma Mater Studiorum - Università di Bologna

DOTTORATO DI RICERCA IN

Ingegneria Energetica, Nucleare e del Controllo Ambientale

Ciclo XXVIII

settore scientifico-disciplinare di afferenza: ING/IND-19 IMPIANTI NUCLEARI

Settore Concorsuale di afferenza: 09/C2

Multiscale multiphysics coupling on a finite element platform

Presentata da Daniele Cerroni

Coordinatore Dottorato	Relatore
Chiar.mo Prof. Ing.	Chiar.mo Prof. Ing.
Vincenzo Parenti Castelli	Sandro Manservisi

Esame finale anno 2014/2015

Contents

Abstract

In recent years numerical simulations are becoming fundamental for the design of many engineering components. For this reason many multiphysical and multiscale problems are investigated by coupling different existent software created specifically for solving each single problem. However, because of the intrinsic differences among these codes, such coupling is very challenging. In this thesis we develop a computational platform that can be used to integrate different computing tools into the common framework of the SALOME platform. Inside this platform various codes are coupled through numerical libraries with the purpose of exchanging data and melting intrinsic differences. After a description of the generic code integration procedure into the numerical platform, we introduce three classes of problems where different codes have been coupled and complex computational problems are studied. In the first problem class, the computational platform is used to study a nuclear reactor system. We study the dynamics of a multiscale primary loop of a liquid metal reactor by coupling a mono-dimensional system code with the high resolution three-dimensional full scale core components models. Also we investigate a thermal-hydraulic-neutron multiphysics problem. The heat energy production in the reactor core, obtained by solving the neutron code DRAGON-DONJON, is coupled with the solution of the thermal-hydraulics conservative equations implemented in a in-house code. In the second problem class, we consider

multiscale multiphysics Fluid Structure Interaction problems implemented in different modules of the FEMUs code. The mechanics of a three-dimensional particular component of the cardiovascular system is coupled with a mono-dimensional model that takes into account the remaining parts of a simplified circulatory system. Finally, in the last class of problems, Multiphase Fluid Structure Interaction problems are investigated by coupling the solution of a multiphase fluid interface advection VOF module with a FSI solver.

CHAPTER 1

Introduction

In these years the design and the control of an increasing number of engineering devices is possible thanks to computer simulations. Recently we have seen the development of many codes that are able to simulate and reproduce complex system physics, helping the developers to better design large projects involving technologies from nuclear to aerospace field. The design of engineering facilities is a very difficult task. The different physics and scales of its components leave unsolved many questions. From the engineering point of view, basically two classes of software have been developed for this type of design: system and CFD codes. The first ones are used to study the dynamics of highly complex systems in which a huge amount of components such as pump, valves, tank, pressurizer, etc. are considered. Because of the system complexity, all the components cannot be represented in three dimensional full scale model. The computational cost of this approach would be impossible to take, so reduced and simplified models must be used. In this way the dynamics of the whole complex system can be studied but the full scale, detailed, behavior of a specific component cannot be investigated because of the high resolution of the problem. For this second type of simulation CFD codes are largely used. In these programs the governing equations of the problem are discretized and solved on a three-dimensional computational grid that well represents, from the geometrical point of view, the specific object. Then, based on these solutions, one can investigate the problem at a component level scale neglecting the dynamics of the whole system. Through the years CFD programs have been specialized in different application fields such as neutron, fluid dynamics, structural mechanics, fluid structure interaction, thermal dynamics, etc. Depending on the problem type, each code uses different discretization scheme such as Finite Volume (FV), Finite Difference (FD) and Finite Element (FE) methods. With this type of approximation one can study the specific problem at a small-scale resolution but, as the computational power increases, one would also be able to consider more complex problems which involve multiphysics phenomena where different equations should simultaneously be solved. A typical and simple example is the study of stresses determined by heat distribution in mechanical systems or components. This case, in a first approximation, can be investigated assuming a constant temperature field and evaluating locally the Young Elastic modulus considering specific temperature fields. A more realistic approach leads to solve the heat and the mechanical problem in a coupled way in order to have a better comprehension of the particular phenomena and then to use this information in component design. As we mention, nowadays, engineering codes have become very specific and, thanks to years of experience and developing, have become very accurate. However their use for studying multiphysics problems is a really difficult task. Most difficulties arise from the intrinsic differences among different parts of the software code such as input and output formats, programming languages, etc. In this situation two approaches are possible: writing a new software or trying to couple different software to solve the multiphysics problem. The first option, writing a new code, is the hardest one and requires a huge knowledge of different problems from the theoretical and practical point of view together with a large amount of time in order to reproduce the accuracy, reached through years, of long time developed codes. The second strategy, codes coupling, is trying to merge the knowledge buried into the original codes in order to better investigate specific aspects. This type of coupling is usually achieved by using scripts that launch different programs and read inputs from textual files. This approach usually requires a huge amount of computational time and leads to a compound problem that is quite hard to be controlled. A more efficient way to couple those codes is the inclusion of them into a computational platform, which is a framework that can control the execution of the different codes and, through a common interface, allows the software to exchange run-time data with high reduction of the computational cost. In order to include a code into a numerical platform it must be organized in a specific way that allows the extraction and insertion data. A common interface between the specific results and a common format must be provided. In this way the specific code can be easily integrated into the computational platform and it can be easily used by all the other software already integrated. In this work we describe a general procedure for code inclusion into a particular numerical platform. We use this computational tool to investigate complex problems such as the dynamics of the primary loop of a lead cooled nuclear reactor, the aneurysm growth into the cardiovascular system and a multiphase fluid structure interaction problem.

The work is organized as it follows. In the first Chapter we give an introduction on the specific numerical platform SALOME, which is developed and used in this work for coupling different codes. The general structure of the computational platform is explained together with a general procedure for the integration of different codes. In particular we explain how a code should be structured to be included in the SALOME platform. We focus the attention on the interfaces that must be introduced to allow the computational platform to extract and impose computational fields in the integrated codes. This capability is the essential feature so that different codes, integrated into the platform, can mutually exchange data. In this work different CFD modules are used, all them are based on Finite Element Method. The finite element notation and basic mathematical tools are introduced in Section 2.3. Several computational examples are then reported. In these examples introductory partial differential equation problems (PDE) are solved with two different simple library codes that solve the problem into different computational domains. These regions share a common surface where the computational field, evaluated by the different codes, is exchanged until convergence is reached. In the first two tests, computational grids are characterized by the same spatial dimensional while in

the third test a three-dimensional problem is coupled with a mono-dimensional one. We remark that the correct solution over the compound domain is reached through a continue exchange of data among CFD programs. This exchange of data is handled by the numerical platform that drives the two codes execution by extracting and imposing computational run-time fields.
In the second Chapter the SALOME platform is used to study the dynamics of the primary system of a LFR reactor. In the first part, an introduction over the mathematical model used for the description of the reactor core and plena is given. The energy, momentum and mass balance equations are solved on a three-dimensional domain, while the boundary conditions are dynamically set by a mono-dimensional system code that simulates the whole primary loop of the reactor. In the last part of this Section a 3D core model of a PWR reactor is coupled with a neutron code. The temperature field, evaluated with the CFD code is exchanged with the neutron code that updates the cross section according to the local temperature field. The neutron flux is then evaluated and used for defining the new power density distribution in the CFD module.
In the third Chapter the coupling of codes into the numerical platform is used to investigate another type of problem: the Fluid Structure Interaction (FSI) of a particular component of the cardiovascular system. As in the previous case, a multidimensional problem is coupled with a system code that takes into account the remaining parts of a simplified model of the circulatory system. After an introduction of the FSI problems the mathematical description of the mono-dimensional and the three-dimensional system is given together with the description of boundary conditions that allows the exchange of data between the problems. The multidimensional FSI module uses a monolithic approach which despite having a great stability, has huge computational cost. In order to reduce this computational effort, a new algorithm for solving monolithic FSI problems is introduced. Different test cases are then reported: the first is a validation for the new FSI solving algorithm while the second is a test for the stability of the coupling algorithm between the mono and multidimensional FSI module. In the last two tests we consider an aneurysm in the abdominal aorta in a steady state and a transient condition, respectively. In the last Chapter, Multiphase Fluid Structure Interaction problems are intro-

duced. Such problems are studied by coupling a FSI module with a geometric solver for the convection of the multiphase fluid interface. Because of deep differences between these two solvers, the computational grids are different and the SALOME platform is used to create an interface that projects the velocity field into the interface convection Cartesian cells. The new location of the multiphase interface is projected back into the CFD mesh and the new velocity field together with the structure domain are computed. The general mathematical description of the problem is investigated by focusing on the interface coupling these two modules. Different test cases are presented to verify the coupling algorithm.

CHAPTER 2

Numerical platform

In this Chapter we briefly describe the computational platform used in this thesis. This work modifies an existing numerical platform, the NURESAFE platform, developed by the CEA for coupling different codes in the study of a new design of light water reactors. Clearly this software is not open-source and its use is restricted to collaborative CEA studies. However the CEA platform is based on SALOME platform, which is indeed an open-source software that can be used for developing new applications with no restrictions. For these reasons we have used the open-software platform to add new codes and develop coupling interface compatible with open and not open-source codes. After a general description of the SALOME platform structure we describe the coupling procedure of a generic code into the platform. In particular we introduce the principal elements that must be developed in order to exchange data with another code that has been previously integrated into this platform.

2.1 SALOME Platform

The collaborative project NURESAFE for the development of reliable software usable for safety nuclear reactor analysis, has been funded by the European

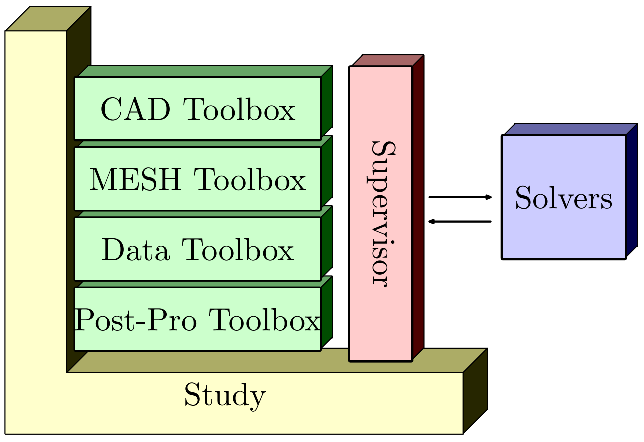


Figure 2.1: Layer structure of SALOME architecture.

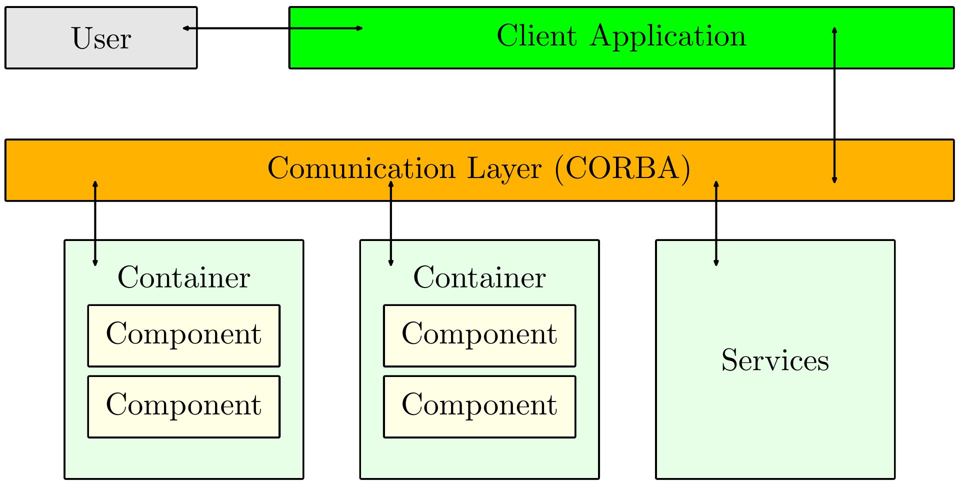


Figure 2.2: Layer structure of SALOME communication algorithm.

community based on the open-source SALOME platform. NURESAFE has been created to improve the nuclear safety by developing high level of expertise and collecting the most recent simulation tools in the nuclear field. The project has started with the development of CATHARE, NEPTUNE and TRIO_U codes as independent software, with not compatible input and output formats [1]. CATHARE is a Code for the Analysis of Thermal Hydraulics during an Accident of Reactor and safety Evaluation for LWR [2]. This system code is used mainly for PWR safety analysis, accident management and definition of plant operating procedures. It is also used to quantify conservative analysis margins and for nuclear reactor licensing [3, 4]. TRIO_U and NEPTUNE codes solve the flow equations in multidimensional geometries with particular attention to two-phase flows [5]. CATHARE, NEPTUNE and TRIO_U are developed by CEA and EDF and they can be used only under NURESAFE project agreement. The CATHARE system code, like other important nuclear codes, have been developed with an interface on SALOME platform for coupling and integration [4, 6]. At the moment the platform is based on computational tools for light water reactors but many of these codes can be adapted to a large range of applications. SALOME architecture is based on CORBA technology that uses distributed system model of computational resources. SALOME combines several software components that allow the integration of solvers and existing meshing algorithms along with the specification of physical properties. The originality of this approach is that various components could cooperate dynamically for the optimization of resources management. A sketch of the general structure of the SALOME numerical platform is shown in Figure . SALOME integrates a number of modules each having its own function: KERNEL, GUI, GEOM, SMESH, MED, YACS and Paravis module. The KERNEL module provides a common shell for all components, which can be integrated into the SALOME platform. The GUI module provides visual representation: basic widgets, viewers, etc. Very important is the GEOM module, which facilitates the construction and the optimization of geometrical models using a wide range of CAD functions. The SMESH module generates meshes on geometrical models previously created or imported by the GEOM component, ParaVis performs data visualization and post processing and finally MED allows to work with highly compressed file files.

KERNEL module. SALOME architecture is based on the concept of multilayer client/server. The distributed system model exposes all functionality of the application as objects, each of which can use any of the services provided by other objects in the system, or even objects in other systems. The architecture can also shadow the distinction between client and server because the client components can also create objects that behave in server-like roles. This type of architecture provides the most flexible solution. The distributed system architecture achieves its flexibility by encouraging (or enforcing) the definition of specific component interfaces. The interface of a component specifies to other components what services are offered by that component and how they are used. As long as the interface of a component remains constant, that component implementation can change without affecting other components. All software components (GEOM, SMESH, etc) integrated into SALOME platform implement predefined interfaces. Each component provides data for the SALOME "study" in a form of links to the specific data created and stored in the component. All components represent CORBA servers and it allows to run them on different host computers as shown in Figure 2.2. It is equally possible to create engine-independent modules that may not use CORBA at all, and can have internal data structure which can be written in pure C++ (or python). Such modules are located inside SALOME GUI process and from the point of view of the end user have no difference with standard components. These modules, which do not use the standard tools of SALOME platform, are defined on a special separated level named CAM. CAM component is the basis for new SALOME GUI and contains all basic functionality for working with modules. Another fundamental aspect of the SALOME architecture is the use of the Interface Definition Language (IDL), which specifies interfaces among CORBA components. The architecture of this platform for numerical components has several advantages. First of all the creation and modification of computation schemes may be easy, the developer can have easy access to all modeling parameters to create domain-specific tools adapted to new situations or to test new numerical algorithms. The implementation of code is simple for the user and the reuse of components is noticeably facilitated. SALOME is also able to more finely simulate phenomena that is more complex in scale by coupling different scale code allowing the investigation of a particular phenomena at different resolution.

MED module. Above all the computational tools included in platform, the MED module provides a library for storing and recovering computer data in MED format, associating numerical meshes and fields allowing the exchange between different codes and solvers. Inside SALOME these structures are exchanged between solvers at the communication level (CORBA or MPI) offering common read and write functions through MED files. The MED libraries are divided into three groups of functions: MED File, MED Memory and MED CORBA. The first group (MED File) is a C and FORTRAN API that implements mesh and read/write functions into files with med extension, these files are in HDF5 format: a data model for storing and managing data. The module supports an unlimited variety of data types, it is designed for flexible and efficient input/output and for handle complex data. The MED Memory group, which is a C++ and Python API, creates mesh and field objects in memory, the mesh creation can be done using set functions, or by loading a file. Fields are also created loading files or by initialization with user-defined functions. Finally the group of functions called MED CORBA allows distributed computation inside SALOME [7].

GEOM module. In the SALOME platform there are modules which are fundamental for multidimensional CFD computations and for system codes. The geometry module (GEOM) of SALOME is destined for: import and export of geometrical models in IGES, BREP, STEP, STL, XAO and VTK formats, construction of geometrical objects using a wide range of functions, viewing geometrical objects in the OCC viewer, transformation of geometrical objects using various algorithms, optimization of geometrical objects, viewing information about geometrical objects using measurement tools and designing shapes from pictures. It is possible to easily set parameters via the variables predefined in SALOME notebook. The GEOM module supplies data structures to implement boundary representation (BRep) of objects in 3D. In BRep the shape is represented as an aggregation of geometry within topology. The geometry is understood as a mathematical description of a shape, i.e. as curves and surfaces (simple or canonical, Bezier, NURBS, etc). The topology is a data structure binding geometrical objects together. Topology defines relationships between simple geometric entities. A shape, which is a basic topological entity, can be divided into components (sub-shapes): Vertex, a zero-dimensional shape corresponding to a point; Edge, a shape corresponding to a curve and bounded by a vertex at each extremity; Wire, a sequence of edges connected by their vertices; Face, a part of a plane (in 2D) or a surface (in 3D) bounded by wires; Shell, a collection of faces connected by edges of their wire boundaries; Solid, a finite closed part of 3D space bounded by shells and Compound solid, a collection of solids connected by faces of their shell boundaries. Complex shapes can be defined as assemblies of simpler entities.

MESH module. The main function of the MESH module of SALOME is to create meshes, import and export meshes in various formats, modify meshes with a large array of dedicated operations, create groups of mesh elements, filter mesh entities (nodes or elements) using different functionality for creating groups and applying mesh modifications and viewing meshes in the VTK viewer and, finally, get info on mesh and its sub-objects together with applying meshes quality controls. In particular computational grids can be created by meshing geometrical models previously created or imported by the GEOM component. The mesh can be constructed with a bottom-up approach by using mesh edition operations, especially extrusion and revolution and by generation of the 3D mesh from the 2D mesh. It is possible to use the variables predefined in SALOME notebook to set parameters of operations. Almost all mesh module operations are accessible via Python interface. To create a mesh on geometry, it is necessary to create a mesh object by choosing a geometrical shape produced in the GEOM module and some meshing parameters, including meshing algorithms and hypotheses specifying constraints to be taken into account by the chosen meshing algorithms. Then one can launch mesh generation by invoking Compute command. The MESH module contains a set of meshing algorithms, which are used for meshing entities (1D, 2D, 3D sub-shapes) composing geometrical objects. An algorithm represents either an implementation of a certain meshing technique or an interface to the whole meshing program generating elements of several dimensions. For meshing of 1D entities (edges) the Wire Discretization Meshing (WDM) and the Composite Side Discretization (CSD) algorithms are available. The first one (WDM) splits an edge into a number of mesh segments following an 1D hypothesis. CSD algorithm allows to apply a 1D hypothesis to a whole side of a geometrical face even if it is composed of several edges. For meshing of 2D entities (faces) two algorithm are available: Triangle (Mefisto) and Quadrangle (Mapping) which splits faces into triangular and quadrangular elements, respectively. For meshing of 3D entities (solid objects) two possible algorithm are considered. The first is the Hexahedron (i,j,k) meshing algorithm in which solids are split into hexahedral elements thus forming a structured 3D mesh. The algorithm requires that 2D mesh generated on a solid could be considered as a mesh of a box, i.e. there should be eight nodes shared by three quadrangles and the rest nodes should be shared by four quadrangles. The second meshing algorithm is the Body Fitting, where solids are split into hexahedral elements forming a Cartesian grid; polyhedral and other types of elements are generated where the geometrical boundary intersects Cartesian cells. Hypotheses represent boundary conditions which will be taken into account by meshing algorithms. The hypotheses allow you to manage the level of detail of the resulting mesh: when applying different hypotheses with different parameters you can preset the quantity or size of elements which will compose your mesh. So, it will be possible to generate a coarse or a more refined mesh. The choice of a hypothesis depends on the selected algorithm. Hypotheses are imposed during creation and edition of meshes and sub-meshes. Once created a hypotheses can be reused during creation and edition of other meshes and sub-meshes. It is possible to open a dialog to modify the parameters of a hypothesis from its context menu. This menu also provides the Unassign command that does not assign the hypothesis from all meshes and sub-meshes. Modification of any parameter of a hypothesis and its unassignment leads to automatic removal of elements generated using it.

ParaVis module. The ParaVis module is the integration of ParaView inside SALOME. SALOME uses by default the detached server mode of ParaView: the server is launched outside the main SALOME process and the ParaVis module, or the PVViewer view connects to it. Following this logic, the PVSERVER CORBA service has a very restrained role. Its only purpose is to control the start and stop of the pvserver process and provide the URL of the server, so that a client can connect to it. we remark that the CORBA engine does not provide any access to the objects or the visualization results themselves. It only serves to establish the link with the ParaView server. The latter can then be queried to retrieve those objects. A typical session sequence is as follows: start of SALOME's GUI, activation request of ParaVis, activation of the PVSERVER CORBA service, invocation of the method FindOrStart-PVServer, which launches the server process and returns its URL, and finally invocation of the standard ParaView's API to connect to the server. We remind that ParaView works in a client/server mode. In two words, a server part (the pvserver) takes care of the computations (filter, etc.) and a client part serves to control this server, and obviously visualize the final rendering. The pvserver represents the main visualization server, and can be either built-in or detached. in the first case launching ParaView suffices to activate it automatically while in the second case one has to launch the server first (possibly on another host) and then connect to it from a client.

YACS module. The YACS module is a tool to supervise execution of complex interconnected scientific applications on computer networks and clusters. Interconnected scientific applications can be seen as a collection of computational tasks that are executed in a known order. In YACS this application application is described by a calculation schema which can be defined with an XML syntax and is mainly a graph of nodes that refer to computational tasks or control structures. A calculation scheme can be built either using a graphic tool, or by editing an XML file directly, or by using an application programming interface (API) in Python. A calculation scheme is constructed based on the calculation node concept which represents an elementary calculation that can be the local execution of a Python script or the remote execution of a SALOME component service. This assembly is made by connecting input and output ports of these calculation nodes. Composite nodes: Block, Loop, Switch are used to modularize a calculation scheme and define iterative processes, parametric calculations or branches. Finally, containers can be used to define where SALOME components will be executed (on a network or in a cluster). Data exchanged between calculation nodes through ports are typed in four categories: basic types, object references, sequences and structures. User types can be defined by combining these basic elements. Many types are predefined either by YACS or by the components used such as GEOM or

SMESH.

Data exchange. This platform has been conceived not only to collect a series of codes that have been extensively used in the field of thermal hydraulics of nuclear reactors but also to harmonize them with the aim of solving complex problems by exchanging information among different codes over a common platform and on large multiprocessor architectures. SALOME can also be used as a platform for the integration of external third-party numerical codes to produce a new application with full pre and post processing management of CAD models. The integration of a code on the SALOME platform is obtained by generating an interface with functions available in the MEDMem library that allows the data transfer from the platform to the code and vice versa. Two different codes both with SALOME MEDMem interface can transfer data to the interface and then from the interface to the other code. MED supports different element shapes such as point, line, triangle, quadrangle, tetrahedron, pyramid, hexahedron, polygon and polyhedron. Each element has a different number of nodes, depending on linear or quadratic interpolation. In order to have a working platform, common input and output formats should be harmonized between different codes. This can be achieved by using SALOME as the basic platform taking care of the data exchange between codes and of the distributed computation between different clusters. In the following Section we give a description of the general procedure that can be used for the integration of a generic code into the computational platform.

2.2 Code integration

In this Section we describe the integration procedure of a CFD code into the SALOME platform with the aim of using computational modules generated with this library for multiscale coupling. The integration of an open-source code into the a numerical platform, can be divided into three major steps: the first is the generation of the code-library from the original code, the second is the generation of the MEDMem interface and finally the generation of SALOME-code interface integration. In a non open-source code such as

CATHARE, the source code is not available but usually the binary version of the library is provided, in this case the first step of generation of the code-library is no longer needed.

2.2.1 Generation of the code-library

The generation of the dynamic library from a code is pretty straightforward for modern codes since they are already built as libraries. The main code is simply a collection of call functions to libraries where the algorithms are developed. For old codes, especially in FORTRAN, sometimes a monolithic main program is developed with a few functions in support with experimental data. In this case it should be straightforward, for developers, to rebuild the code using a library structure.

2.2.2 MEDMem interface

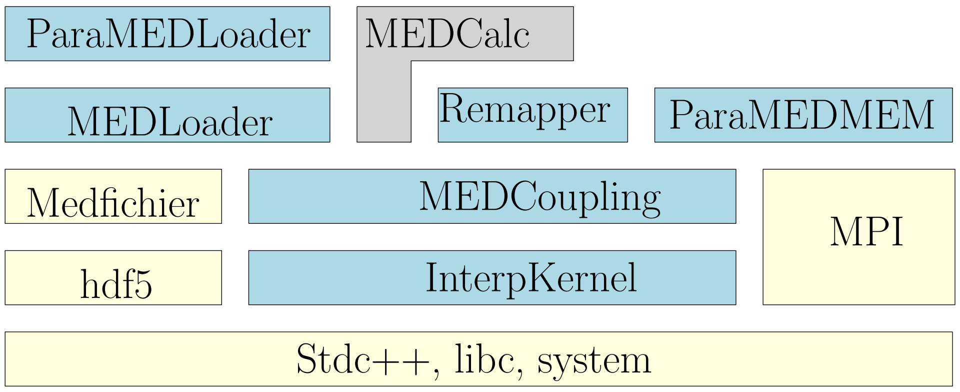


Figure 2.3: Layer structure of the packages of the library.

Simulation studies require the manipulation of meshes and fields for data processing or post processing. Corresponding computer codes can be viewed as software components accessing input meshes and fields, with specific constraints, along with parameters and producing output meshes and fields. The MED module aims at pooling operations on those items, facilitating their use by various codes involved in a simulation process. This includes making codes communicate while preserving as much as possible the integrity of their con-

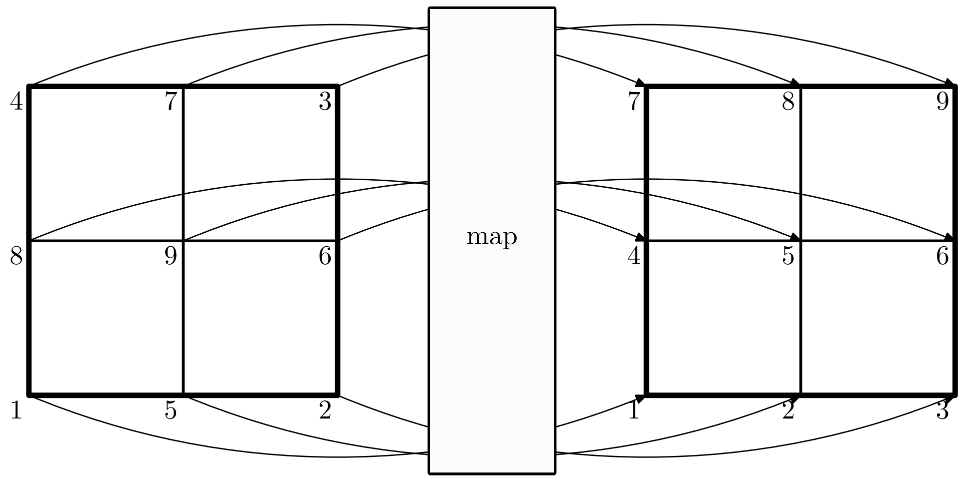


Figure 2.4: Operator that map the each node of a generic element (left) into its duplicate in med format(right).

tent. In order to fulfill its objective, the MED module includes methods for: handling meshes and fields to satisfy code input requirements, extraction of field information and projections and serialization to exchange meshes and fields between codes. The structure of the MEDMem libraries is shown in Figure 2.3, the fundamental set (blue background) consists in three atomic libraries: MEDCoupling that describes data structures used for cross process exchange of meshes and fields, MEDLoader and ParaMEDLoader that provides input output functions to the med file format and interpolation tools that provides mathematical structures and algorithms for interpolation and localization. The main services offered by MEDCoupling are the manipulation of fields and their support mesh and multidimensional interpolation on nodes, cells, Gauss points and nodes by element.

Once the code has been build as a library we need to add methods that can export the results into the MED format, in this way, in order to exchange a computational field with another code, we can just extract the solution from the MED file and project it into the different computational domain still in MED format. For this purpose we build a duplicate of the original computational grid in the med format and create a map that associates each computational node from the original mesh to one of the MED duplicate and vice versa as shown in Figure 2.4. The generated map can be used both for projecting a solution coming from the specific code into the MED support but also for transfer a computational field from the MED mesh to the code specific computational domain. The projection into the MED grid consist on a reordering of the solution according to the generated map. When the solution is prepared as just described, it can be attached to the med support and transferred to any other med support using the methods of the med API. Once the field is transferred into a different support it can be extracted and ordered according to the particular map generated for the second program.

2.2.3 SALOME interface

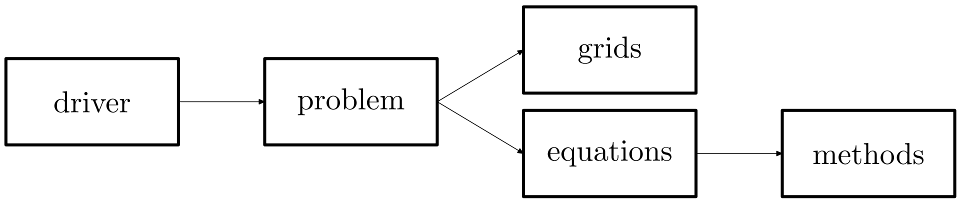


Figure 2.5: Collaboration diagram of generic CFD interface.

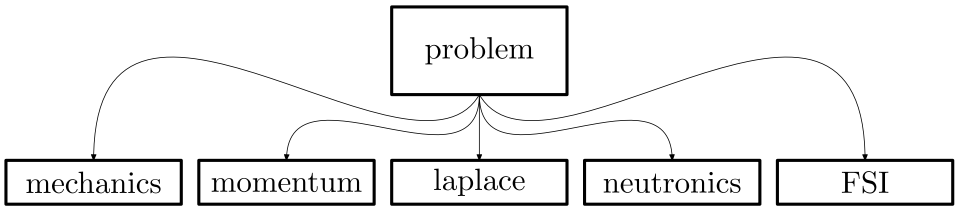


Figure 2.6: Inheritance diagram of the *problem* class.

The final step in the integration of a generic code into the numerical platform, is the construction of different methods that allow the control of the execution and the data input/output of the selected code. There are basically five classes that must be introduced in the selected code: *driver*, *problem*, *equations*, *grids* and *methods*. The *driver* class is the top level class that interface with the SALOME platform as it is shown in the collaboration diagram in Figure 2.5, where the basic structure of the code that we want to integrate into the platform is illustrated. This diagram show how the *driver* class transfers data to the *problem* class which is specialized for different physics. This class is inherited by all the physical modules used by the selected code as one can see in Figure 2.6. The *equations* class, which uses only MEDMem functions, inherits the system particular class which contains the assembly and solver of the generic code. The data from the *driver* class can be transferred into the assembly routine which is user accessible. The data should also be transferred in the opposite direction from the equations to the MEDMem interface. The data flow from *parent* to *son* class is ensured by the C++ inheritance rules wile the flow in opposite direction is defined by a *dynamic_cast* command that allows to use *son* class functions from the *parent* class. For this reason the *parent* class should be polymorphic with at least one virtual function. The *grids* class is an extension of mesh class of the particular CFD code, it contains all the routines for mesh handling together with MEDMem methods that are used to extract and manage groups of nodes and cells from the original computational grid. In particular the extension class contains the methods that allow the data transfer from one format to another. As in the previous class, data from the *driver* class can be transferred, by using a *dynamic_cast* statement, into the assembly routine which is user accessible. Data can also be transferred in the opposite directions from the grids to the MEDMem interface through *parent* to *son* class inheritance rules. All the code interfaces must have similar commands to run the program. The mesh boundary names and their flags, used for creating mesh groups, are stored in the *driver* interface class. Over these groups, by using MEDMem functions and the map between the med support and the specific mesh, we are able to extract field from the med support and project it from the MED support, coming from other SALOME platform codes, into the problem specific mesh format. The basics commands that the *problem* class must have are: *setType*, *setMesh* and *solve*. The first command sets the problem type (Navies-Stokes, energy, etc.), the second one sets and prepares the mesh which should be available in MED and code specific formats for data exchange and, finally, the *solve* command controls the solution of the discrete system. The boundary regions of the domain should be controlled for input and output. For this purpose we have to introduce class *methods* that could set and get analytic and numerical solutions in these particular zones. The interface to the selected code is obtained through the *problem* class. For this reason it should access to both *grids* and *equations* classes as shown in the collaboration diagram of the problem class in Figure 2.5.

While the *grids* class can only handle the data the specific code format, the *equations* class contains two standard maps that associate to any zone of the computational grid a *methods* so that a volumetric field can be extracted and transferred to a *methods* with a MEDMem mesh format by using the *getSource* and the *setSource* function. If the data transfer is between 3D and 1D the average source functions may be used. When codes are organized as just described exchange data is easy. We just have to create a supervisor that access the *driver* class from different problems and, after each *solving* iteration the desired field is extracted from the first *driver* and projected into the common format. This field is then projected by the second *driver* into the computational grid of the second problem and can be taken into account during its execution.

2.3 Finite element method

The principal purpose for code coupling into the numerical platform is to couple the solution of a three-dimensional problem with a simplified but more comprehensive system. In the three-dimensional problem typically a Partial Differential Equation system (PDEs) is solved on a complex three-dimensional domain. The three classical choices for the numerical solution of PDEs are the Finite Difference Method (FDM), the Finite Element Method (FEM) and the Finite Volume Method (FVM). The FDM is the oldest and is based upon the application of a local Taylor expansion to approximate the differential equations. The FDM uses a topologically square network of lines to construct the discretization of the PDE. This is a potential bottleneck of the method when handling complex geometries in multiple dimensions. This issue motivated the use of an integral form of the PDEs and subsequently the development of the finite element and finite volume techniques. Finite element methods and, the closely related, boundary element methods nowadays belong to the standard routines for the computation of solutions to initial boundary value problems of partial differential equations. They are used in many engineering field such as elasticity and thermal-elasticity, fluid mechanics, acoustics, electromagnetic, scattering and diffusion. These methods also stimulated the development of corresponding mathematical numerical analysis. In all the computational example presented in this work, multidimensional problems are solved with a FEM method for this reason in this Section we give a general description of this method together with the notation that is used in the rest of the work.

2.3.1 Finite element space

Consider the two-point boundary value problem

$$\begin{aligned} -\frac{d^2u}{dx^2} &= f \,\text{in}\, x \in (0,1)\,, \\ u(0) = 0, &\qquad u'(1) = 0\,. \end{aligned} \tag{2.1}$$

If u is the solution and v is any (sufficiently regular) function such that $v(0) = 0$, then integration by parts yields

$$\begin{aligned} (f, v) &:= \int_0^1 f(x)v(x)dx = \int_0^1 u''(x)v(x)dx \\ &= \int_0^1 u'(x)v'(x)dx =: a(u, v)\,. \end{aligned} \tag{2.2}$$

Let us define

$$V = v \in L^2(0,1): \quad a(v, v) < \infty \wedge v(0) = 0\,. \tag{2.3}$$

L^2 is the second order Lebesgue spaces defined by

$$L^p(\Omega) := f : ||f||_{L^p(\Omega)} < \infty\,, \tag{2.4}$$

where $||f||_{L^p(\Omega)}$ is

$$||f||_{L^p(\Omega)} := \left(\int_\Omega |f(x)|^p dx \,:\, x \in \Omega \right)^{1/p}\,. \tag{2.5}$$

With the definition (2.3) we can say that the solution u to (2.1) is characterized by

$$u \in V \,\text{such that}\, a(u, v) = (f, u) \quad \forall v \in V\,, \tag{2.6}$$

which is called the variational or weak formulation of (2.1). Let $S \in V$ be any (finite dimensional) subspace. Let us consider (2.6) with V replaced by S, namely

$$u_S \in S \,\text{such that}\, a(u_S, v) = (f, v) \quad \forall v \in S\,. \tag{2.7}$$

Subtracting (2.7) from (2.6) implies

$$a(u - u_S, w) = 0 \quad \forall w \in S. \tag{2.8}$$

Equation (2.8) and its subsequent variations are the key to the success of all Ritz-Galerkin/finite-element methods. Now define

$$||v||_E = \sqrt{a(v, v)}\,, \tag{2.9}$$

for all $v \in V$, the energy norm. A critical relationship between the energy norm and inner-product is Schwarz' inequality

$$|a(v, w)| \leq ||v||_E ||w||_E \quad \forall v, w \in V, \tag{2.10}$$

then, for any $v \in S$,

$$\begin{aligned} ||u - u_S||_E^2 &= a(u - u_S, u - u_S) \\ &= a(u - u_S, u - v) + a(u - u_S, v - u_S) \\ &= a(u - u_S, u - v) \\ &\leq ||u - u_S||_E ||u - v||_E\,. \end{aligned} \tag{2.11}$$

If $||u - uS||_E \neq 0$, we can divide by it to obtain $||u - u_S||_E \leq ||u - v||_E$, for any $v \in S$. If $||u - u_S||_E = 0$, this inequality is trivial. Taking the infimum over $v \in S$ yields

$$||u - u_S||_E \leq \inf ||u - v||_E \, : \, v \in S. \tag{2.12}$$

Since $u_S \in S$, we have

$$\inf\{||u - v||_E \, : \, v \in S\} \leq ||u - u_S||_E. \tag{2.13}$$

Therefore,

$$||u - u_S||_E = \inf\{||u - v||_E \, : \, v \in S\}. \tag{2.14}$$

Moreover, there is an element (u_S) for which the infimum is attained, and we indicate this by replacing infimum with minimum. Thus, we have proved the following

$$||u - u_S||_E = \min\{||u - v||_E \,:\, v \in S\}. \tag{2.15}$$

This is the basic error estimate for the Ritz-Galerkin method, and it says that the error is optimal in the energy norm. We will use this later to derive more concrete estimates for the error based on constructing approximations to u in S for particular choices of S. Now we consider the error in another norm. Define $||v|| = (v, v)^2 = (\int_0^1 v(x)^2 dx)^{1/2}$, the L^2 (0, 1)-norm. We wish to consider the size of the error $u - u_S$ in this norm. To estimate $||u - u_S||$, we use what is known as a duality argument. Let w be the solution of

$$-\,w'' = u - u_S \,\text{on}\, [0, 1] \,\text{with}\, w(0) = w'(1) = 0\,. \tag{2.16}$$

Integrating by parts, we find

$$\begin{aligned} ||u - u_S||^2 &= (u - u_S, u - u_S) \\ &= (u - u_S, -w'') \\ &= (u - u_S, w) \\ &= (u - u_S, w - v)\,, \end{aligned} \tag{2.17}$$

for all $v \in S$. Thus, Schwarz' inequality (2.10) implies that

$$\begin{aligned} ||u - u_S|| &\le ||u - u_S||_E ||w - v||_E / ||u - u_S|| \\ &= ||u - u_S||_E ||w - v||_E / ||w''|| \end{aligned} \tag{2.18}$$

Thus, we see that the L^2-norm of the error can be much smaller than the energy norm, provided that w can be approximated well by some function in S. It is reasonable to assume that we can take $v \in S$ close to w, which we formalize in the following approximation assumption

$$\inf_{v \in S} ||w - v||_E \le \epsilon ||w''||. \tag{2.19}$$

Of course, we envisage that this holds with ϵ being a small number. Applying (2.19) yields

$$||u - u_S|| \le \epsilon ||u - u_S||_E, \tag{2.20}$$

and applying (2.19) again, with w replaced by u, gives

$$||u - u_S||_E \leq \epsilon ||u''||. \tag{2.21}$$

Combining these estimates, and recalling (2.1), yields

$$||u - u_S|| \leq \epsilon ||u - u_S||_E \leq \epsilon^2 ||u''|| = \epsilon^2 ||f||. \tag{2.22}$$

The key point of course is that $||u - u_S||_E$ is of order ϵ whereas $||u - u_S||$ is of order ϵ^2 and we have to consider a family of spaces S for which ϵ may be made arbitrarily small. In order to explain how the FEM method can represent a solution for the problem defined in (2.1) we have to introduce son functional spaces and extend the definition of the derivative. First of all we introduce the Banach space which is a normed linear space $(V, ||\cdot||)$ that is complete with respect to the metric induced by the norm, $||\cdot||$. We recall that a norm $||\cdot||$ is a function on a given linear (vector) space V with values in the non-negative real having the following properties

(i) $||v|| \geq 0 \quad \forall v \in V (||v|| = 0 \Longleftrightarrow v = 0)$

(ii) $||c \cdot v|| = 0|c| \cdot ||v|| \quad \forall c \in \mathbb{R}, v \in V$ and

(iii) $||w + v|| \leq ||w|| + ||v|| \quad \forall w, v \in V$

A norm,$||\cdot||$, can be used to define a notion of distance, or metric, $d(v, w) = ||v - w||$ for points $v, w \in V$. A vector space endowed with the topology induced by this metric is called a normed linear space. Recall that a metric space, V , is called complete if every Cauchy sequence $\{v_j\}$ of elements of V has a limit $v \in V$. The classic definition of derivative is:

$$u'(x) = \lim_{h \longrightarrow 0} \frac{u(x+h) - u(x)}{h} \tag{2.23}$$

which is a local definition, involving information about the function u only near the point x. The variational formulation developed (2.6) takes a more global view, because point wise values of derivatives are not needed and only derivatives that can be interpreted as functions in the Lebesgue space $L^2(\Omega)$ occur.

It is natural to develop a global notion of derivative more suited to the Lebesgue spaces, we can do so using a duality technique, defining derivatives for a class of not-so-smooth functions by comparing them with very-very-smooth functions. Given a domain Ω, the set of locally integrable functions is denoted by

$$L^1_{loc}(\Omega) := \{f : f \in L^1(K) \forall \text{ compact } K \in \text{ interior } \Omega\}. \tag{2.24}$$

Functions in $L^1_{loc}(\Omega)$ can behave arbitrarily badly near the boundary (not-so-smooth functions), although this aspect is somewhat tangential to our use of the space. One notational convenience is that $L^1_{loc}(\Omega)$ contains all of $C0(\Omega)$, without growth restrictions. Finally, we can say that a given function $f \in L^1_{loc}(\Omega)$ has a weak derivative, $D^\alpha_w f$, provided there exists a function $g \in L^1_{loc}(\Omega)$ such that

$$\int_\Omega g(x)\phi(x)dx = (-1)^{|\alpha|} \int_\Omega f(x)\phi^\alpha(x)dx \forall \phi \in (D)(\Omega) \tag{2.25}$$

where $(D)(\Omega)$ is the set of $C^\infty(\Omega)$ functions with compact support in Ω. We can now define the Sobolev spaces via

$$W^k_p(\Omega) := \left\{ f \in L^1_{loc} : \left(\sum_{|\alpha| \le k} ||D^\alpha_w f||^p_{L^p(\Omega)} \right) \right\}. \tag{2.26}$$

Let us introduce the bilinear form, $b(\cdot,\cdot)$, on a linear space V is a mapping $b : V \times V \longrightarrow \mathbb{R}$, a (real) inner product, denoted by $(\cdot,\cdot)$, is a symmetric bilinear form on a linear space V that satisfies

$$\begin{gathered} (v,v) \ge 0 \quad \forall v \in V \text{and} \\ (v,v) = 0 \Longleftrightarrow v = 0\,. \end{gathered} \tag{2.27}$$

A linear space V together with an inner product defined on it is called an inner-product space and is denoted by $(V,(\cdot,\cdot))$ and the quantity $||v|| := \sqrt{(v,v)}$ defines a norm in the inner-product space $(V,(\cdot,\cdot))$ and if the associated normed linear space $(V, ||\cdot||)$ is complete, then $(V,(\cdot,\cdot))$ is called a Hilbert space (H). As we introduce at the beginning of this Section, we are here interested to find an approximate solution of the PDE problem that in the general case reads

$$a(u,v) = F(v) \quad \forall v \in V. \tag{2.28}$$

For this purpose we can now build the finite-dimensional subspace $S \in V$ in a systematic, practical way. We recall Ciarlet's definition of a finite element [8] Let:

(i) $K \subseteq \mathbb{R}^n$ be a bounded closed set with non-empty interior and piecewise smooth boundary (the element domain),

(ii) $\mathcal{P}$ be a finite-dimensional space of functions on K (the space of shape functions) and

(iii) $\mathcal{N} = \{N_1, N_2, \ldots, N_k\}$ be a basis for $\mathcal{P}'$ (the set of nodal variables).

Then $(K, \mathcal{P}, \mathcal{N})$ is called a finite element. It is implicitly assumed that the nodal variables, N_i, lie in the dual space of some larger function space, e.g., a Sobolev space. In this work we consider finite elements defined on rectangles. Let $\mathcal{Q}_k = \{\sum_j c_j p_j(x) q_j(y) : p_j, q_j \text{ polynomials of degree } \leq k\}$. One can show that

$$\dim \mathcal{Q}_k = (\dim \mathcal{P}_k^1)^2 \tag{2.29}$$

where $\mathcal{P}_k^1$ denotes the space of polynomials of degree less than or equal to k in one variable.

2.4 Codes coupling on SALOME

In Section 2.2 we describe the integration of a generic code into the SALOME platform and how to build interfaces that allow the data transfer between different code libraries. In this way, by using a common computational platform, a solution, computed from a code, can be transferred to another code through these interfaces. In this Section we show three computational examples in which a classic problem is solved into different computational grids that share a boundary zone. In this zone the solution is exchanged run-time and the error, due to the field projection, is investigated. In these examples we use two different codes that have been integrated into the computational platform with the procedure explained in the Section 2.2. The first code we use is libMesh, a library that provides a framework for the numerical simulation of partial differential equations using arbitrary unstructured discretizations on serial and parallel platforms. The second code used is FEMus an in house multiphysics multigrid finite element library that can use both serial and parallel unstructured meshes. In the first and the second example, the Laplace and Navier-Stokes equations are solved, respectively. In both cases the problem is solved on a three-dimensional domain divided into two parts in which a different *driver* is considered. The solution on the common surface is exchanged between the two *driver* through the MEDMem interface. In the last test a Laplace problem is solved in a three-dimensional domain that is linked with a mono-dimensional system, the solution extracted in the common surface is then averaged over the interface and then set into the simplified problem.

2.4.1 Example 1. Laplace problem

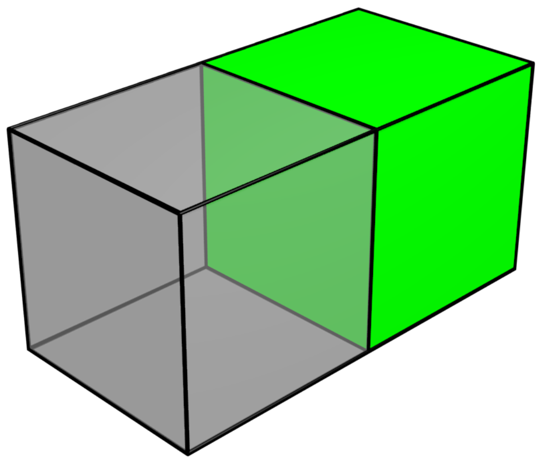


Figure 2.7: Example 1. Coupling 3D-3D geometry. Domain $\Omega_1 \cup \Omega_2$.

In this first example we consider the Laplace problem:

$$\nabla^2 u + \mathbf{v}\nabla u = f, \tag{2.30}$$

defined into the hexaedral region $[0,1] \times [0,1] \times [-1,1]$ (Ω) and characterized by a non homogeneous Dirichlet, homogeneous Neumann and homogeneous Dirichlet boundary conditions in $z=-1$, $z=1$ and the remaining boundary zones, respectively. To formulate the variational equivalent of (2.30), we define

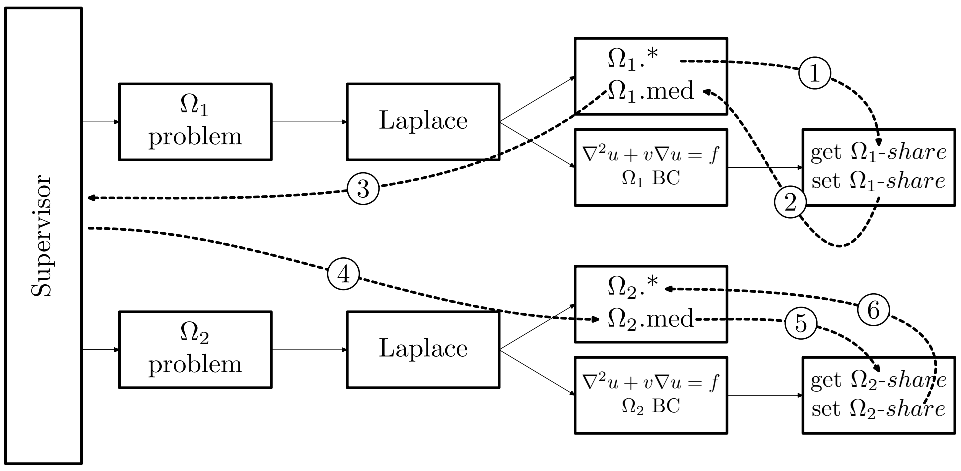


Figure 2.8: Example 1. Problem global structure.

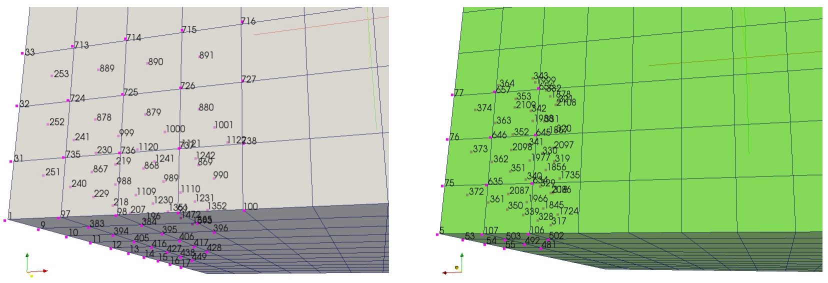


Figure 2.9: Example 1. Numbering of the same surface as Ω_1 (left) and as Ω_2 (right).

a variational space that incorporates the essential, part of boundary condition:

$$V := v \in H^1(\Omega) : v|_\Gamma = 0 \tag{2.31}$$

The appropriate bilinear form for the variational problem is determined by multiplying Poisson's equation by a suitably smooth function, integrating over Ω and then integrating by parts obtaining:

$$\begin{aligned}(f, v) &= \int_\Omega (-\nabla^2 u) v dx = \int_\Omega \nabla u \cdot \nabla v dx - \int_{\partial\Omega} v \frac{\partial u}{\partial n} u ds \\ &= \int_\Omega \nabla u \cdot \nabla v dx := a(u, v)\end{aligned} \tag{2.32}$$

The boundary term vanishes for $v \in V$ because either v o $\partial u/\partial n$ is zero on any part of the boundary. In order to verify the data transfer during the coupling we consider two separate domains one in the region $[0, 1] \times [0, 1] \times [-1, 0]$

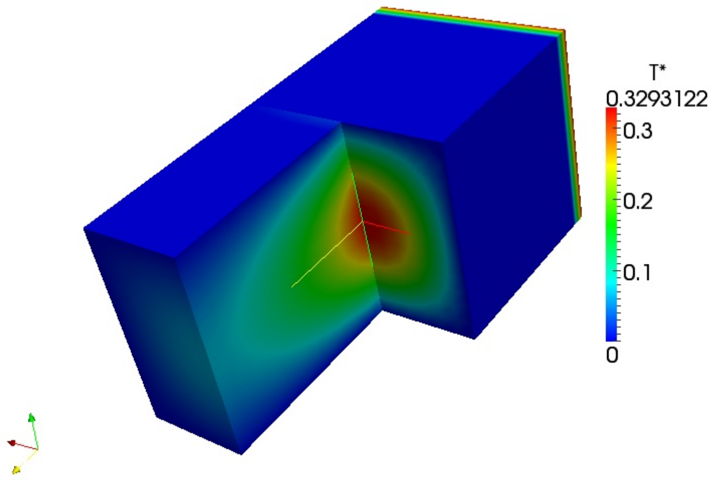


Figure 2.10: Example 1. Solution over the coupled domain.

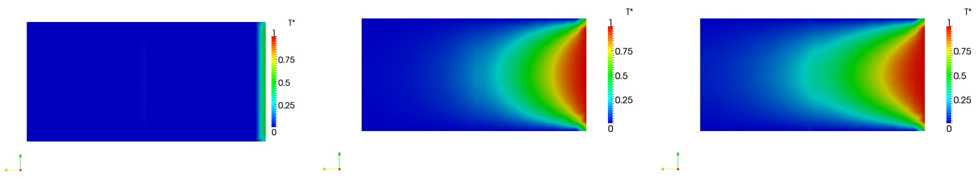


Figure 2.11: Example 1. Solution evolution over the coupled domain.

(Ω_1) and the another in $[0, 1] \times [0, 1] \times [0, 1]$ (Ω_2) as shown in Figure 2.7. The two meshes are created in a separate way and the common face has the same number of points but different numbering order. Each mesh consists of $12 \times 12 \times 12$ HEX20 quadratic elements. By dividing the complete volume Ω into two sub-volumes we have to introduce new boundary conditions into the common face, in order to close the problem. In particular we set homogeneous Neumann boundary condition in the Ω_1-problem and non homogeneous Dirichlet boundary condition in the Ω_2-problem which transfers the field computed in Ω_1-problem through the MEDMem interface. We start in parallel two different problems and continuously transfer data from the Ω_1 to the Ω_2 region until the steady state is found. In particular two libmesh Laplace problems are created on Ω_1 and Ω_2 with the proper boundary conditions in their boundary regions. A single time step iteration of the whole problem, is composed by two part: a time iteration of the Ω_1 and Ω_2 problems. After the first iteration the solution on the surface *share*, from the Ω_1 problem, is extracted and projected into the Ω_2 computational grid so that it can be set as a Dirichlet boundary condition into the Ω_2 solve step. In Figure 2.9 one can see the surface (labeled as *share*) of the computational domain where the data transfer is allowed with solver global numbering. The libmesh interface creates a map to connect this global numbering to the surface mesh *share* of the region Ω_1 and another surface map *share* of the region Ω_2. The surface mesh, in MEDMem representation, has the same number of nodes but different numbering. It is necessary therefore at the beginning of the computation to search point by point the correspondence to build the map. Once the map is computed all field values on the surface can be passed to the problems surface and vice versa. The detailed work flow and structure of the problem and how the field can be exchanged between the drivers, is shown in Figure 2.8: after a *solve* iteration of the Ω_1 problem the solution over the surface *share* is extracted, with the dedicated *method* (step 1), from the original computational grid. With another *method* this computational field is then projected into the duplicated Ω_1.med file (step 2). The supervisor can access all the data into the different problems and project the solution just extracted, using the MEDMem library, from the Ω_1.med into the Ω_2.med duplicated mesh (steps 3-4). Now the second problem can extract the solution field from the its med duplicate (step 5) and project it into its specific computational grid (step 6) using its class *method*, the field evaluated on the surface *share* can now be taken into account during the *solve* iteration of the Ω_2 problem. The solution over the coupled domain is shown in Figure 2.10 where the domain is cut along the common interface. Finally in Figure 2.11 the evolution of the solution over the coupled domain is reported. For high resolutions and small time steps the solution is continuous and the interface discontinuity in the derivative that can be seen at the initial time steps vanish. The data transfer from the nodes of the common face of the domain Ω_1 to Ω_2 seems efficient and reliable.

2.4.2 Example 2. Navier-Stokes problem

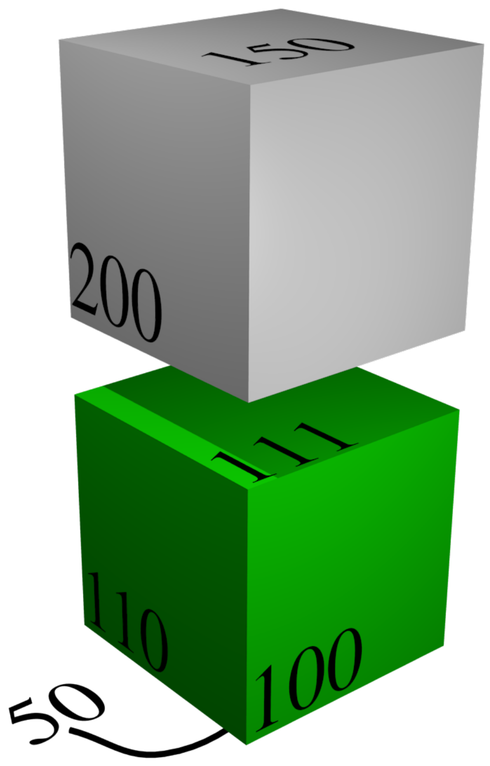


Figure 2.12: Example 2. Coupling 3D-3D geometry for Navier-Stokes equation. Domain $\Omega_1 \cup \Omega_2$.

The coupling of a scalar over a surface common region between two codes has been shown in the previous example. Now we consider a data transfer for a vector in the Navier-Stokes equation which reads

$$\frac{\partial \rho\, \mathbf{v}}{\partial t} + \nabla \cdot (\rho \mathbf{v}\mathbf{v}) = -\nabla p + \nabla \cdot \bar{\tau} + \rho \mathbf{g}\,, \tag{2.33}$$

with the proper boundary conditions for closing and solving the problem. We consider the geometry shown in Figure 2.12 which is similar to the geometry presented in the previous case, with motion along the vertical z-axis. The boundary has no slip boundary condition on the *110*, *100* and *200* face. On the *111* face we set non-homogeneous Neumann in the Ω_1 problem and non-homogeneous Dirichlet boundary conditions in the Ω_2 problem imposing the continuity of velocity field computed in the down problem. In the inlet *50*

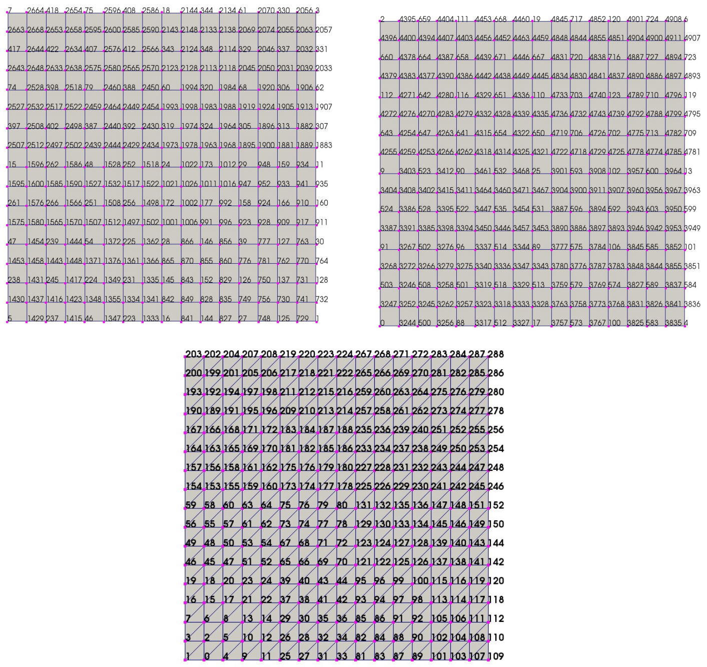


Figure 2.13: Example 2. Numbering of the same surface in MEDmesh (bottom) and in MGMesh as Ω_1 (left top) and as Ω_2 (right top).

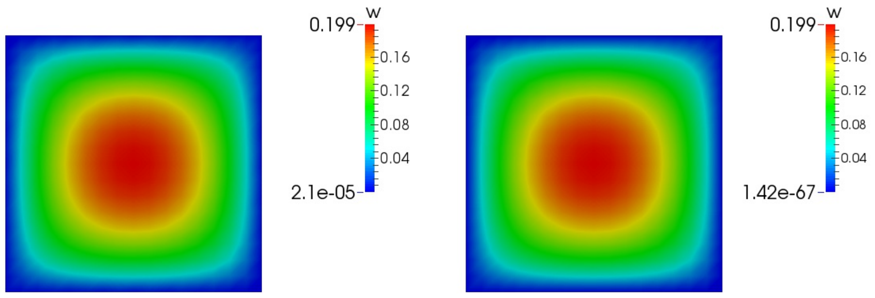
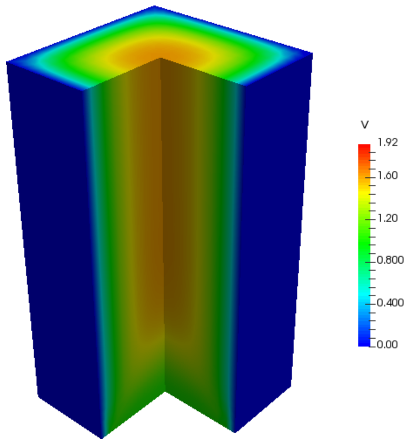


Figure 2.14: Example 2. Solution over the coupled interface as Ω_1 (left) and Ω_2 (right).

we impose standard velocity boundary conditions with a constant field. In the surface *150* outflow conditions are imposed. Again we start in parallel two different problems on multiprocessor machines and the program structure is similar to the one presented in the previous example. In this case on the interface *111* of the problem on Ω_2 we set a Dirichlet boundary condition



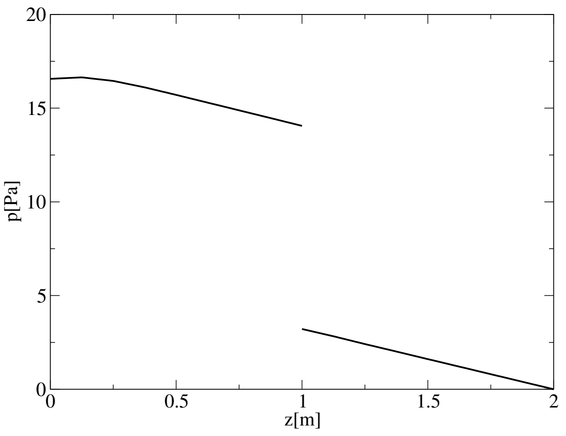
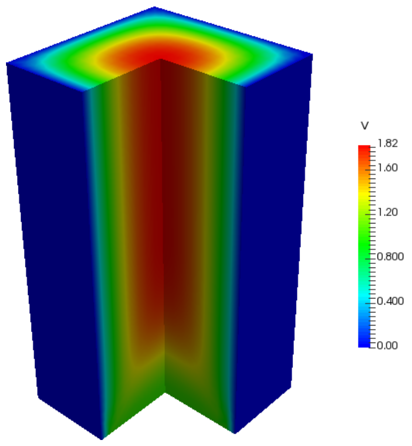


Figure 2.15: Example 2. Axial component of the velocity on the left and pressure over the axis on the right at $t = 0.1s$.



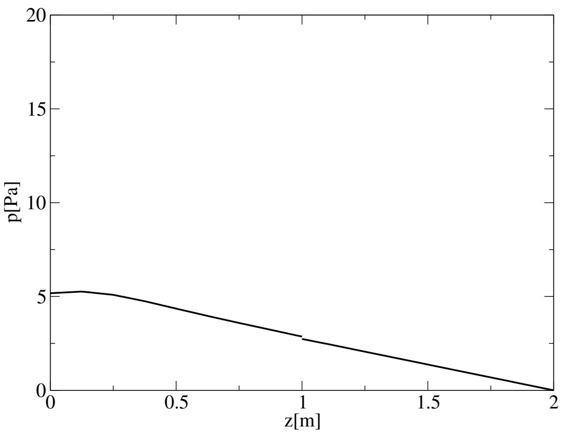


Figure 2.16: Axial component of the velocity on the left and pressure over the axis on the right at $t = 0.3s$.

imposing the boundary velocity field $\mathbf{v}$ of the problem on Ω_1 which is passed through the MEDMem interface function. In order to introduce the variational formulation of the problem we can write the system 2.33 in a more compact form which reads

$$\begin{aligned} -\nabla^2\mathbf{v} + \nabla p &= -R(\mathbf{v}\cdot\nabla\mathbf{v} + \frac{\partial\mathbf{v}}{\partial t})\,, \\ \nabla\cdot\mathbf{v} &= 0\,, \end{aligned} \tag{2.34}$$

in $\Omega(\mathbb{R}^d)$, where $\mathbf{v}$ denotes the fluid velocity and p denotes the pressure. The expression $\mathbf{v}\cdot\nabla\mathbf{v}$ is the vector function whose i^{th} component is $\mathbf{v}\cdot\nabla\mathbf{v}_i$. These

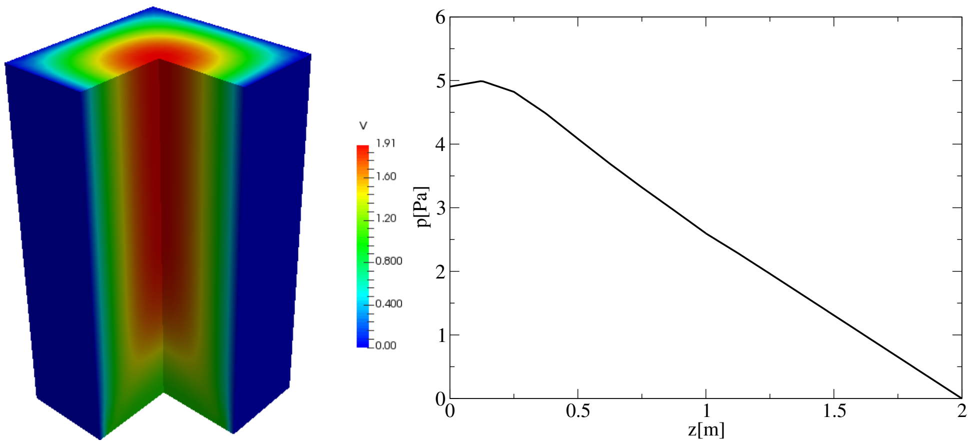


Figure 2.17: Axial component of the velocity on the left and pressure over the axis on the right at $t = 0.5s$.

equations describe both two and three dimensional flows ($d = 2$ and 3, respectively); in the case of two dimensions, the flow field is simply independent of the third variable, and the third component of $\mathbf{v}$ is correspondingly zero. The parameter R in (2.34) is the Reynolds number and when this is very small, the equations reduce to the Stokes equations. Numerical techniques for solving (2.34) often involve different issues relating separately to the solution of the Stokes (or Stokes-like) equations and to the discretization of the convection term that R multiplies. We will focus here on particularly simple time stepping schemes, putting emphasis on the affect this has on the particular form of the corresponding Stokes-like equations. A complete variational formulation of (2.34) takes the form

$$\begin{aligned} a(\mathbf{v},\phi) + b(\mathbf{v},\phi) + R(c(\mathbf{v},\mathbf{v},\phi) + (\mathbf{v}_t,\phi)_{L^2}) &= 0 \quad \forall \phi \in V \\ b(\mathbf{v},q) &= 0 \quad \forall q \in \Pi \end{aligned} \tag{2.35}$$

where

$$a(\mathbf{v},\phi) := \int_\Omega \sum_{i=1}^{d} \nabla \mathbf{v}_i \cdot \nabla \phi_i \,, \tag{2.36}$$

and

$$b(\mathbf{v},\phi) = -\int_\Omega (\nabla \cdot \mathbf{v}) q dx \,. \tag{2.37}$$

Choosing $V = H^1(\Omega)^n$ and $\Pi = \{q \in L^2(\Omega : \int_\Omega q dx = 0)\}$ the solution of (2.35) is unique [9]. When solving the problem on Ω_1 the pressure field that is imposed on *111* is the one evaluated solving the Ω_2 problem and also passed using the interface MED function some non linear iteration, in which the computational fields are exchanged between the two problems, are needed to enforce the continuity of all the computational field at the domains interface. In Figure 2.13 one can see the surface (labeled as *111*) of the computation domain where the data transfer is allowed with different global numbering. The *FEMus* interface creates a map to connect this global numbering to the surface mesh *111* of the region Ω_1 and another surface map *111* of the region Ω_2. The surface mesh in MEDMem representation has the same number of nodes but different numbering. It is necessary therefore to search point by point the correspondence. Once the map is computed all field values on the surface can be passed to the different solvers. The axial velocity over the coupled interface on the region Ω_1 and Ω_2 at a certain time step can be seen on the left and right of Figure 2.14, respectively. No differences can be noted in this case. We remark that in this case the data are transferred points by points and differences are negligible with respect to the case in which the data are transferred cells by cells. From Figure 2.15 to Figure 2.17 one can see the solution over the coupled domain $\Omega_1 \cup \Omega_2$ at different time steps, in particular in the left part of these Figures the axial velocity field is shown, while in the right part the axial pressure profile is reported. We can notice that, although no non-linear iterations were performed, the initial discontinuity in the pressure field (Figure 2.15) is highly reduced after one time step (Figure 2.16) and vanished after few time steps (Figure 2.17). No discontinuities can be observed in the axial velocity field during the simulation.

2.4.3 Example 3. FEMus-libMesh coupling

The multiscale coupling of the temperature over a 2D surface connecting a 3D module and 1D module can be done easily by using SALOME MEDMem library. In this library many functions are available for averaging, finding maximum or minimum of any field over a surface. We consider the geometry shown in Figure 2.18. The equations considered here are the energy and momentum balance. The FEMus module is used for the 3D problem while

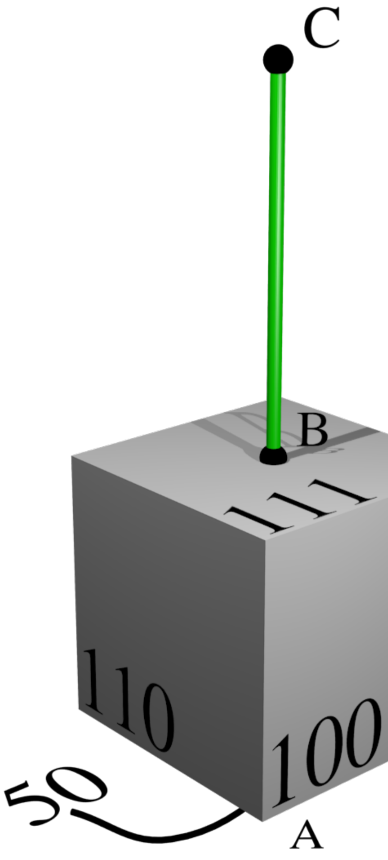


Figure 2.18: Example 3. Coupling 3D-1D geometry. Domain $\Omega_1 \cup \Gamma_2(\overline{BC})$.

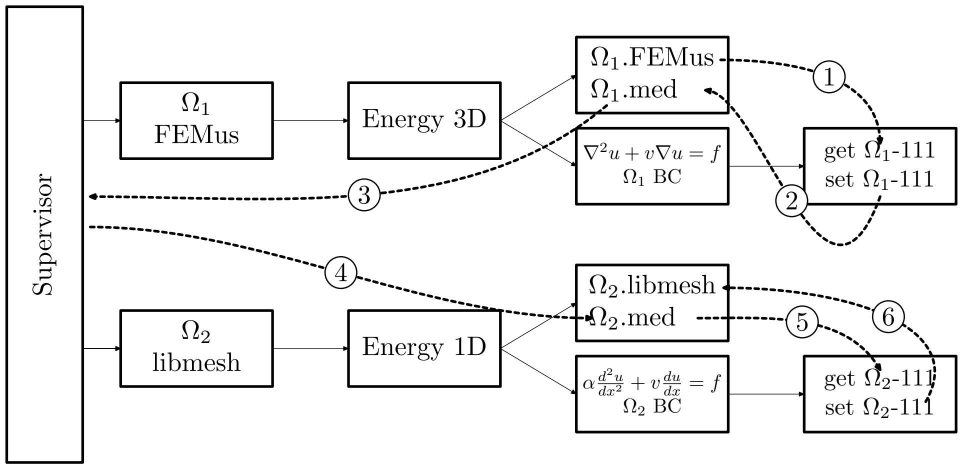


Figure 2.19: Example 3. Coupling 3D-1D geometry. Domain $\Omega_1 \cup \Gamma_2(\overline{BC})$.

the mono-dimensional domain is handle by libMesh. The motion in the 3D domain is along the z-axis from A to B. The 1D region is attached to the

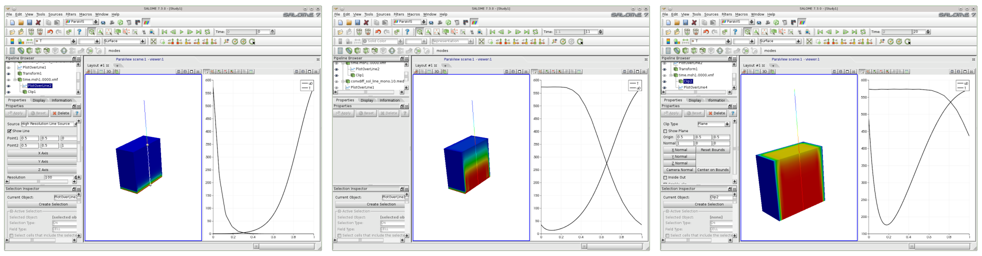


Figure 2.20: Example 3. Evolution over the coupled domain at different time steps. On the left the three-dimensional solution and on the right the solution over the mono-dimensional line BC.

three-dimensional region at the point B. The mono-dimensional domain goes from B to C over a unitary length.

The boundary has Dirichlet boundary conditions on the *100*, *50* and *110* faces. On the *110* face (B) we set Neumann in the down problem and non-homogeneous Dirichlet boundary conditions at the point B problem imposing the average temperature computed in the down problem. In the inlet *50* we impose constant Dirichlet boundary conditions with a constant field. At the end of the line (C) Dirichlet boundary conditions are imposed as at the face (A). We start in parallel two different problems on multiprocessor machines: a FEMus problem is solved in the down-region and a libMesh problem in the line region. The FEMus and the libMesh problems are coupled. In this case on the point B of the 1D-problem we set a Dirichlet boundary condition imposing the boundary field of the 3D-problem. Since the point B is a single point we need to take the average of the surface using the averaging MEDMem function before imposing the Dirichlet boundary condition over the first point of the line domain. The overall structure of the program is shown in Figure 2.19, the first step is the extraction of the temperature field from the 111 surface (step 1) and its projection into the med duplicated (step 2). Now the supervisor can access the temperature field (step 3) and through the MEMmem library can average this field over the selected surface. The supervisor pass the average temperature to the MED grid duplicated of the libmesh problem (step 4) and finally the proper method set the average temperature in the libmesh problem. This procedure is performed after a *solve* iteration of the multidimensional problem and before the libmesh *solve*. In Figure 2.20 the evolution of the solution over the coupled domain is reported. The 1D solution follows the 3D dimensional solution as expected. From left top to bottom right the solution is reported for different time steps. In each figure on the left is reported the three-dimensional domain while on the right the temperature along the central axis of the cube domain (segment $\overline{AB}$) and along the mono-dimensional domain ($\overline{BC}$). The temperature at B in the mono-dimensional domain is always below the central point temperature of the coupling surface.

CHAPTER 3

Nuclear Reactor analysis

The primary loop of a nuclear reactor is a complex system composed by a huge number of components. The simulation of the entire loop, due to its complexity, requires the use of simplified models while the investigation of the behavior of a specific part requires complex and multidimensional models. In this Chapter we investigate the thermodynamics of different nuclear reactors systems in a multiscale and multiphysics framework. Different computational tools, that can exchange data thanks to their inclusion into the computational platform SALOME, are used to study different aspects of the problem. The Chapter is organized as follows, after an introduction into the nuclear energy production we give the geometrical description of a specific nuclear reactor core. We study the reactor dynamics at different resolution scales (3D, 1D, etc.) and we introduce the different mathematical models that are used together with the structures needed to interface them. Some computational examples are then presented. In the firsts two of them, we study the dynamics of the entire primary loop of the nuclear system coupling a system code with a full-scale model of the core, in particular two different coupling techniques are investigated. In the last example, we couple the CFD model of the reactor core with a neutron code and the specific energy production in core is evaluated taking into account the local temperature field. In all cases codes can exchange data thanks to their inclusion into the numerical platform SALOME.. Concerning the CFD modules that are used, we remark that they are in-house developed and solve the balance equations using a standard Galerkin Finite Element Method based on the Taylor-hoods elements.

3.1 Introduction

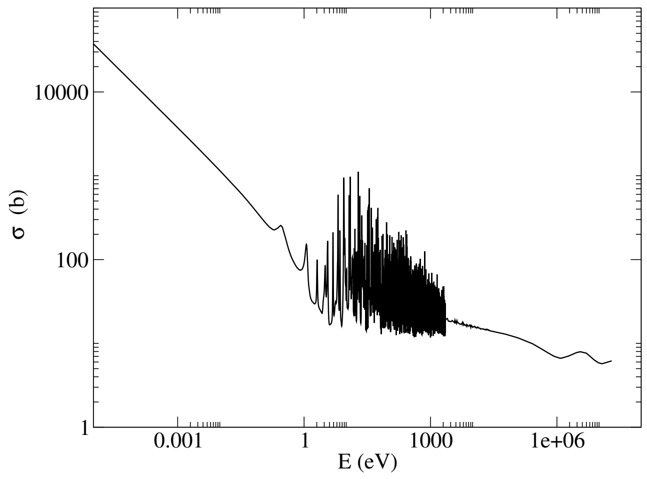
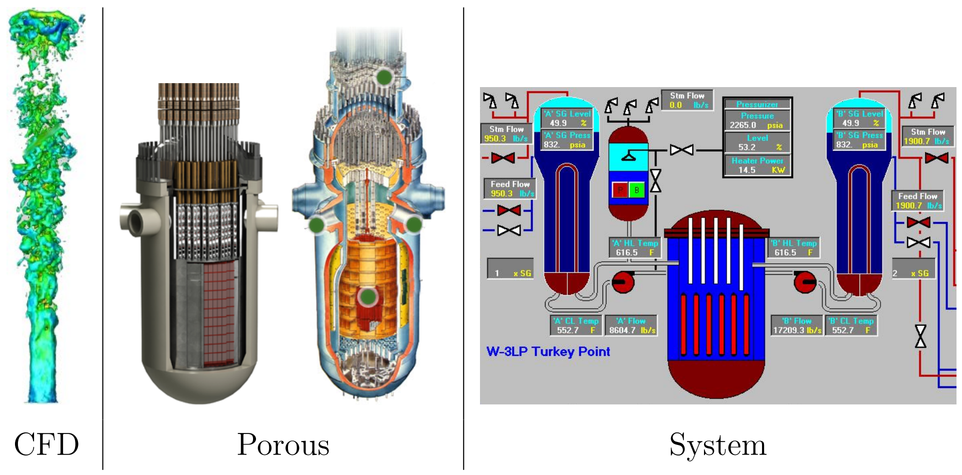


Figure 3.1: Total cross section for ^{235}U.



Stm Flow 950.3 lb/s
'A' SG Level 49.9 %
'A' SG Press 832. psia
Feed Flow 950.3 lb/s
1 x SG
Stm Flow 0.0 lb/s
Pressurizer
Pressure 2265.0 psia
Level 53.2 %
Heater Power 14.5 KW
'A' HL Temp 616.5 F
'B' HL Temp 616.5 F
'A' CL Temp 552.7 F
'A' Flow 8604.7 lb/s
'B' Flow 17209.3 lb/s
'B' CL Temp 552.7 F
'B' SG Level 49.9 %
'B' SG Press 832. psia
Stm Flow 1900.7 lb/s
Feed Flow 1900.7 lb/s
2 x SG

Figure 3.2: CFD (3D), porous (3D) and system scale (1D).

The reactor physics characteristics associated with the interaction of neutrons with all the material components of a nuclear reactor influence, and in some cases determine, the design and operation of the system. One of the most important parameter is the neutron reaction rate with ^{235}U or other fissile materials since it can be used for the evaluation of fission rate and the consequent heat production in the specific core. This heat source determines the power output of the reactor and strongly influences material temperatures which in turn often determine the reactor's safe operating conditions. The reaction rate is determined from the material specific neutron cross sections, which is the probabilities of various reactions occurring when neutrons interact with the material nuclei. As an example, the energy dependent cross section for a neutron reacting with ^{235}U is shown in Figure 3.1. This parameter is the microscopic cross section, usually denoted by σ, which applies to an individual nucleus and has units of area and it is a problem independent property of the nuclide. The macroscopic cross section applies to the specific material and it is usually interpreted as the probability of interaction per cm of distance of a free neutron. This cross section, usually denoted Σ, it is problem dependent and is related to the microscopic cross section through the relationship

$$\Sigma_x(E) = \sum_{i=1}^{I} N_i \sigma_i(E) \tag{3.1}$$

where N_i represents the atom densities in the target and x represents a reaction type. We can now describe the transport of neutrons through material with the time independent neutral particle Boltzmann transport equation

$$\begin{aligned}\nabla \cdot \mathbf{J}(\mathbf{r}, E) + \Sigma_t(\mathbf{r}, E)\Phi(\mathbf{r}, E) =& \frac{\chi(\mathbf{r}, E)}{k_{eff}} \int_0^\infty dE' \nu(E') \Sigma_f(\mathbf{r}, E)\Phi(\mathbf{r}, E) \\ &+ \int_0^\infty dE' \Sigma_s(\mathbf{r}, E' \to E)\Phi(\mathbf{r}, E)\end{aligned} \tag{3.2}$$

where $\Phi(\mathbf{r}, E)$, $\mathbf{J}(\mathbf{r}, E)$ are the flux and current as a function of position and energy, respectively and $\chi(\mathbf{r}, E)$ is the distribution of fission neutrons in position and energy. This equation gives the balance of the different reactions that can occur with neutrons at a given point and energy, along with the k-effective eigenvalue, denoted by k_{eff}. Each of the terms in the equation (3.2) denotes a particular loss mechanism for the neutrons (on the left hand side of the equation) or gain mechanisms for neutrons (on the right-hand side). Computer codes that solve the time-independent, steady-state neutron flux can be divided into two categories: deterministic and stochastic. Stochastic (Monte Carlo) methods solve for the neutron flux by simulating particle transport rather than by numerically solving the Boltzmann transport equation (3.2). Monte Carlo particles simulations use an algorithm in which decisions faced by a neutron during its lifetime in the reactor are are determined from the mathematical probability densities. Deterministic methods involve the numerical subdivision of the independent variables of space, energy and direction into computational subdivisions, with a subsequent reformulation of the continuous-variable Boltzmann Equation (3.2) into a set of discrete variable equations. Specialized deterministic computer codes solve these coupled linear algebra equations for the neutron flux in each phase cell, and the desired flux integrals are approximated by summations over the appropriate cells to get the engineering parameters (including k-effective) of interest in the analysis. The most common deterministic methods are: discrete ordinates and diffusion theory. The discrete ordinates method subdivides all three independent dimensions space, energy, and direction. Space is divided into a regular one, two or three-dimensional grid, energy utilizes the multigroup method and direction is handled by calculating the flux only in particular directions (discrete ordinates). This is the slowest, but most accurate, of the deterministic methods. It is generally used for pin-cell and fuel assembly calculations for situations for which regular geometries either apply or can be reasonable approximated. The diffusion theory method subdivides only space and energy ans simplifies the directional dependence by assuming a nearly isotropic flux. The spatial treatment again utilizes a regular grid of spatial elements connected to their immediate neighbors and the energy treatment again employs the multigroup approach. This is the simplest and fastest of the deterministic methods. The primary use of diffusion theory is for full-reactor calculations in which reactor assemblies have been homogenized, so that strong absorbers and fission/scattering sources have been mathematically spread out. A detailed description of the reactor physics and analysis can be found in [10, 11, 12].

The technology used to control a fission chain reaction evolves during these years, in particular nuclear reactor designs are usually categorized by generations: Generation I, II, III, III+, and IV. The key attributes characterizing the development and deployment of nuclear power reactors illuminate the essential differences between the various generations of reactors. Three generations of nuclear power systems, derived from designs originally developed for naval use beginning in the late 1940s, are operating worldwide today. Generation I refers to the prototype and power reactors that launched civil nuclear power. This generation consists of early prototype reactors from the 1950s and 1960s, such as Shippingport (1957–1982) in Pennsylvania, Dresden-1 (1960–1978) in Illinois, and Calder Hall-1 (1956–2003) in the United Kingdom. The only remaining commercial Gen I plant, the Wylfa Nuclear Power Station in Wales, was scheduled for closure in 2010. However, the UK Nuclear Decommissioning Authority announced in October 2010 that the Wylfa Nuclear Power Station will operate up to December 2012. Generation II refers to a class of commercial reactors designed to be economical and reliable. Designed for a typical operational lifetime of 40 years, 2 prototypical Gen II reactors include pressurized water reactors (PWR), CANada Deuterium Uranium reactors (CANDU), boiling water reactors (BWR), advanced gas-cooled reactors (AGR), and Vodo-Vodyanoi Energetichesky Reactors (VVER). The Generation II BWR primary coolant system, the steam water mixture, first enters the steam separators after exiting the core. After subsequent passage through a steam separator and dryer assembly located in the upper portion of the reactor vessel, dry saturated steam flows directly to the turbine. Saturated water, which is separated from the steam, flows downward in the periphery of the reactor vessel and mixes with the incoming main feed flow from the condenser. This combined flow stream is pumped into the lower plenum through jet pumps mounted around the inside periphery of the reactor vessel. The jet pumps are driven by flow from recirculation pumps located in relatively small-diameter ($50cm$) external recirculation loops, which draw flow from the plenum just above the jet pump discharge location. The primary coolant system of a PWR consists of a multi-loop arrangement arrayed around the reactor vessel. Higher power reactor ratings are achieved by adding loops of identical design. Designs of two, three, and four loops have been built with three and four loop reactors being the most common. In a typical four loop configuration, each loop has a vertically oriented steam generator and coolant pump. The coolant flows through the steam generator within an array of U tubes that connect the inlet and outlet plena located at the bottom of the steam generator. The system pressurizer is connected to the hot leg of one of the loops. Gen II systems began operation in the late 1960s and comprise the bulk of the world's 400+ commercial PWRs and BWRs. These reactors, typically referred to as light water reactors (LWRs), use traditional active safety features involving electrical or mechanical operations that are initiated automatically and, in many cases, can be initiated by the operators of the nuclear reactors. Some engineered systems still operate passively (for example, using pressure relief valves) and function without operator control or loss of auxiliary power. The design of such reactors require relatively large electrical grids, have a defined safety envelope based on Western safety standards, and produce significant quantities of used fuel that require ultimate disposition in a high-level waste repository or reprocessing as part of a partially or fully closed fuel cycle. The economics of existing Gen II plants and of those under construction or in the planning stage are generally favorable, particularly in Asia. The extraordinary events unfolding at the Fukushima Daiichi and Daini nuclear power plants are being assessed by the technical and regulatory experts both in the United States and across the world. At press time, a full assessment is not possible. We have learned that an increased safety focus will likely be required and, at a minimum, will focus on four safety systems: BWR containment structures, common-mode emergency core cooling capability resulting from loss of emergency backup power, the performance of mixed-oxide fuel in Gen II reactor designs and critical safety analyzes of the various extant used fuel cooling pool designs. Generation III nuclear reactors are essentially Gen II reactors with evolutionary, state-of-the-art design improvements. These improvements are in the areas of fuel technology, thermal efficiency, modularized construction, safety passive systems, and standardized design. Improvements in Gen III reactor technology have aimed at a longer operational life, typically 60 years of operation, potentially to greatly exceed these years, prior to complete overhaul and reactor pressure vessel replacement. The Westinghouse 600 MW advanced PWR (AP-600) was one of the first Gen III reactor designs. On a parallel track, GE Nuclear Energy designed the Advanced Boiling Water Reactor (ABWR) and obtained a design certification from the NRC. The first of these units went online in Japan in 1996. Other Gen III reactor designs include the Enhanced CANDU 6, which was developed by Atomic Energy of Canada Limited (AECL); and System 80+, a Combustion Engineering design. Only four Gen III reactors, all ABWRs, are in operation today. Generation III+ reactor designs are an evolutionary development of Gen III reactors, offering significant improvements in safety over Gen III reactor designs certified by the NRC in the 1990s. Manufacturers began development of Gen III+ systems in the 1990s by building on the operating experience of the American, Japanese, and Western European LWR fleets. Perhaps the most significant improvement of Gen III+ systems over second-generation designs is the incorporation in some designs of passive safety features that do not require active controls or operator intervention but instead rely on gravity or natural convection to mitigate the impact of abnormal events. The inclusion of passive safety features, among other improvements, may help expedite the reactor certification review process and thus shorten construction schedules. These reactors, once on line, are expected to achieve higher fuel burn up than their evolutionary predecessors (thus reducing fuel consumption and waste production). Nuclear scientists have left implementation of the Gen III+ and SMR designs and have instead focused on nuclear alternatives commonly called Gen IV that still require considerable fundamental research. Conceptually, Gen IV reactors have all of the features of Gen III+ units, as well as the ability, when operating at high temperature, to support economical hydrogen production, thermal energy off-taking, and perhaps even water desalination. These reactors are cooled by a wide variety of coolants liquid metal sodium, lead or lead-bismuth, helium, and super critical carbon dioxide gas, liquid salt, and super critical water. They were identified through an international evaluation based on a set of sustainability, safety, economic, proliferation resistance, and physical protection criteria. Their individual development is led by the interested countries that coordinate their efforts through the Generation IV International Forum[13, 14]. The missions of these reactors include production of electricity, process heat including hydrogen as well as waste management by transmutation of actinides (neptunium, plutonium, americium, and curium). Principal reactor characteristics are therefore the average neutron energy and primary system outlet temperature. Among all the rectors type, the Lead Fast Reactor (LFR) is one of the most promising options. The central part of a LFR consists of several components and their accurate simulation cannot be performed in three-dimensional space with the actual computational power. For this reason it seems reasonable a multiscale approach where different components are studied on different scales based on a well defined computing resource plan. As shown in Figure 3.2, one can see three representative different scales: DNS - CFD, porous and system scale. The scale where the Navier-Stokes can be solved in order to define completely the physics of the system is called DNS or Direct Numerical Simulation scale. The CFD scale is the scale where the usual Computational Fluid Dynamics codes can approximate satisfactory the system evolution. This includes turbulence models that usually cannot be simulated in the DNS scale. In the porous model scale the geometrical details are so numerous that a detailed simulation cannot be possible in the DNS scale. In this scale the introduction of a unique fictitious porous material with average properties must be considered. Finally, in the system scale, all the components are considered through a mono-dimensional or zero-dimensional model in order to study the global behavior of several components against desired plant control. System codes for nuclear reactors have been developed for many years and some of them, such as RELAP and CATHARE are certified by international agencies since they provides transient solutions reliable for safety analysis. However they cannot be used for studying three-dimensional behavior, so they cannot reproduce all the three-dimensional physical phenomena in the core and plenum regions. A multiscale strategy combines the CFD approach in the region of interest with time dependent boundary conditions computed by system codes. Further modeling is necessary in the core region since the real geometry of a reactor core is made up of an extremely large number of fuel rods, packed in several fuel assemblies. Instead of simulating each flow sub-channel between the pins, the core is considered to be a porous media, where average velocity and temperature fields are computed. The three-dimensional porous module is equipped with appropriate power heat and pressure loss sources that reproduce the space dependent characteristics of the system. In particular the thermal source and the pressure losses can be loaded by data files at the assembly level in all the space domain by using neutron computations and according to the mechanical structural design of the reactor. The lower and upper plenum can be simulated with a traditional CFD approach, as there is no fine geometry to take into account. In order to respect the balance of momentum and energy through the 3D-CFD and 3D-porous media the velocity field is discontinuous due to reduction of the fluid cross area. Appropriate interfaces are also introduced in order to solve the three-dimensional and mono-dimensional equations through a unique non-linear coupled solver.

In the following Sections after a geometrical description of a selected reactor core we describe the different mathematical models that are used in the different zones of the reactor primary loop together with several computational examples.

3.2 Geometrical model

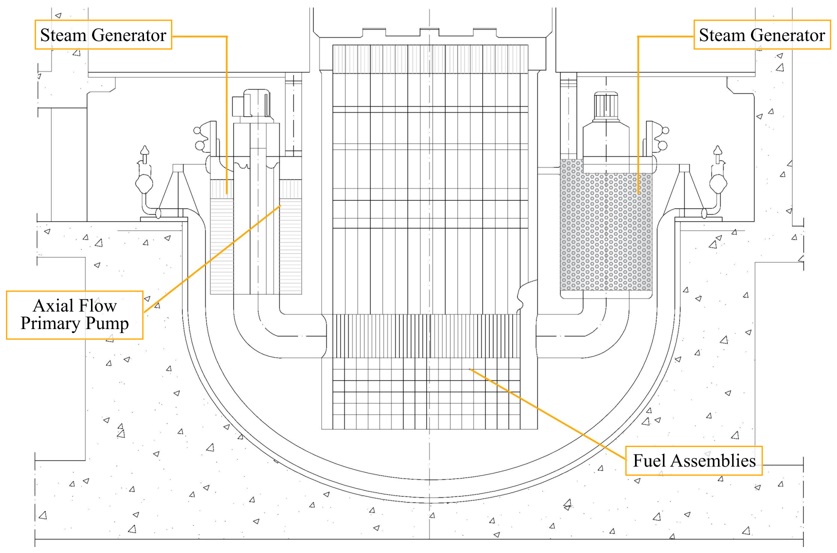


Figure 3.3: Schematic of the reactor.

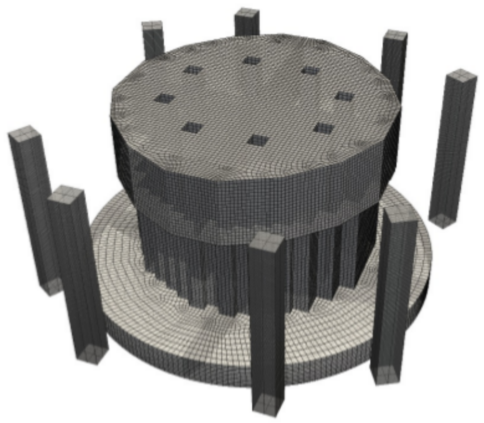
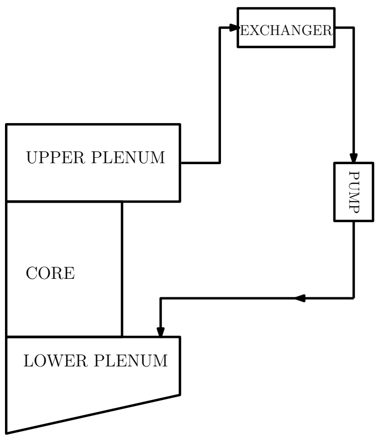



Figure 3.4: On the left, computational domain. On the right, modules used in the multiscale LFR modelization.

Parameter	Value
Power [MWe]	600
Conversion Ratio	~ 1
Thermal efficiency [%]	42
Primary coolant	Lead
Circulation type	Forced
Core inlet temperature [K]	673.15
Core outlet temperature [K]	753.15
Fuel	MOX (Nitrides)
Fuel cladding material	T91 (aluminized)
Peak cladding temperature [K]	823.15
Fuel pin diameter [mm]	10.5
Active height/diameter [m]	0.9/4.32
Primary pumps	8 integrated in the SG
Working fluid	Water-superheated at 18 MPa, 450°C

Table 3.1: Main characteristics of the ALFRED reactor.

The LFR model considered in this part, is the type-pool LFR reactor ALFRED, shown in Figure 3.3. The nuclear reactor is characterized by a very compact design and its main characteristics are illustrated in Table 3.1. One important feature of this reactor is that it uses lead as basic coolant. This

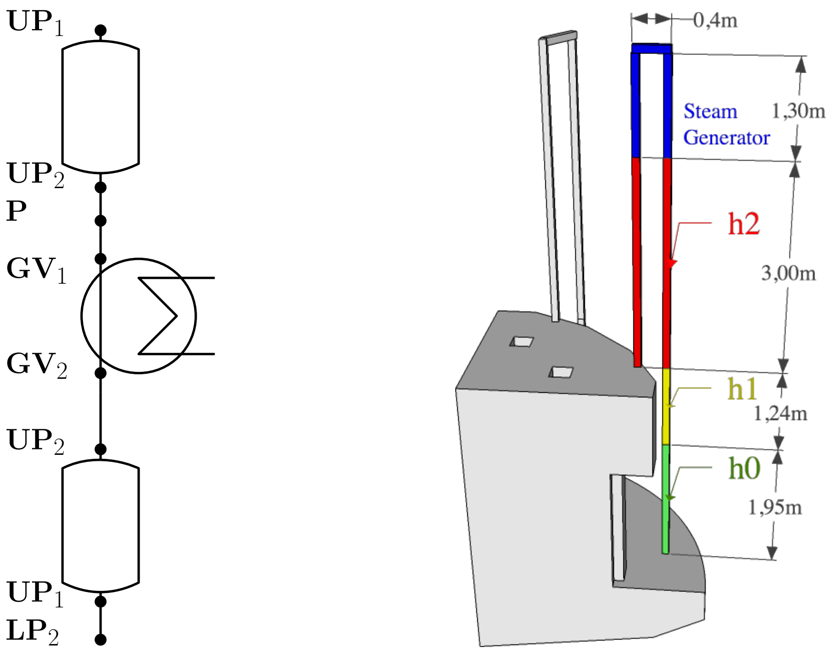


Figure 3.5: Schematic of primary loop with heat exchanger.

is due to its good chemical behavior and its thermodynamic properties. The main disadvantages of this coolant are the high solidification temperature of $600.15K$ and the oxidation and corrosion behavior. The thermal power of one unit at nominal conditions is $1500MW$ at a total mass flow rate of about $126000Kg/s$. The LFR vessel has 8 primary loops with 8 steam generators (SG) and 8 integrated primary pumps. At nominal conditions the coolant enters the core at $T = 673.15K$ to reach the outlet at temperature of about $753.15K$. For the secondary loop superheated water-steam is used to reach a thermal efficiency of about 43%. This converts the $1500MW$ heat power into $600MW$ of electrical power. A detailed description of the ALFRED reactor is presented in [15]. As shown on the right part of Figure 3.4 we consider the LFR system basically divided into three regions: heat exchange loop (1D-porous), plenum (3D-CFD) and core (3D-porous) region. The computational mesh is shown in the left part of Figure 3.4. If we set the zero vertical co-ordinate at the lowest point, the core region goes from $1.3m$ to $3.24m$. The active core (upper core) where heat is generated ranges between $H_{in} = 2.25m$ and $H_{out} = 3.15m$. Below the core we have the lower plenum with the inlet between 0 and $H_{lp} = 1.3m$. The lower plenum has an approximate hemispherical form with the lowest region at $H_{bot} = 0$ (reference point). Above the core for a total height of $H_{up} = 1.24m$ there is the upper plenum with the coolant outlet. The generation of this mesh is not trivial, every assembly must be exactly discretized by entire cells. In this way all the data attributes related to a given assembly may be easily associated. For details we refer to [16]. In the design of this LFR reactor there are 8 primary loops, each of them contains a steam generator and a pump. We model each of this reactor segment as a 1D-porous module. The schematic of this module is defined in the left part of Figure 3.5 where one can find the upper plenum from UP_1 to UP_2, the steam generator from GV_1 and GV_2 together with the pump P. The point with label LP_2 indicates the entrance in the lower plenum. Since the interface between the 3D and 1D module is defined by a single point, the primary loop starts at the lower point of the upper plenum outlet and ends at the upper point of the lower plenum inlet as one can see on the right part of Figure 3.5 where the seal path of the primary loop are shown.

3.3 3D-CFD scale

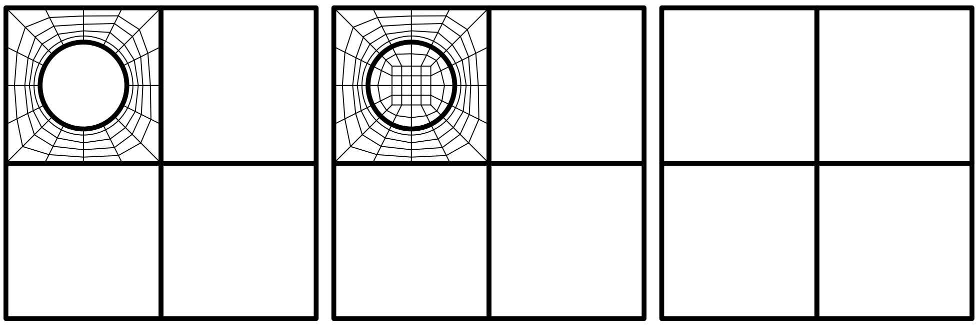


Figure 3.6: Two level solution scheme: geometrical fine (left), fine (middle) and coarse (right) level.

On 3D-CFD scale the three-dimensional conservative incompressible equations in velocity, pressure and enthalpy (or temperature) field $(\mathbf{v}, p, h(T))$ over a domain Ω with boundary Γ are the following

$$\nabla \cdot (\rho\, \mathbf{v}) = 0\,, \tag{3.3}$$

$$\frac{\partial \rho\, \mathbf{v}}{\partial t} + \nabla \cdot (\rho \mathbf{v}\mathbf{v}) = -\nabla p + \nabla \cdot \bar{\tau} + \rho \mathbf{g}\,, \tag{3.4}$$

$$\frac{\partial \rho\, h}{\partial t} + \nabla \cdot (\rho \mathbf{v}\, h) = \Phi + \nabla \cdot \left(\frac{k_f}{C_p} \nabla h \right) + \dot{Q}\,, \tag{3.5}$$

where ρ is the density, $\mathbf{g}$ the gravity acceleration vector, C_p is the pressure specific heat, k_f the effective heat conductivity, $\dot{Q}$ the volume heat source and Φ the dissipative heat term. The enthalpy h is defined by

$$h = C_p T\,, \tag{3.6}$$

with C_p constant in the liquid region. The use of h or T is completely indifferent. We use the temperature T as a main variable but we refers to enthalpy when the liquid and the solid phase are present at the same time. We assume pressure solution in the space $P(\Omega)$, velocity in $\mathbf{V}(\Omega)$ and temperature in $H(\Omega)$. For details see [16, 1, 17]. The viscous stress tensor $\bar{\tau}$ is defined by

$$\bar{\tau} = 2\mu_f \bar{D}(\mathbf{u})\,, \quad D_{ij}(\mathbf{u}) = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)\,,$$

where μ_f is the liquid viscosity. For details see [1, 17]. The incompressibility constraint (3.3) is assumed to be valid even if the density is supposed to be dependent on temperature. These equations should be averaged in space depending on the scale. In the simulation of the primary loop of the LFR reactor, the active core and the lower and upper plena are modeled with 3D models, while the rest of principal remaining parts such as pumps, pool, steam generator etc. are taken into account with simplified models. In many components the simulation of internal flows is very complex and simplifying assumptions must be introduced, for example let us consider the core case where there are hundreds of assemblies and each assembly has hundreds of fuel rods. The complexity of this geometry asks for domain homogenization. As shown in Figure 3.6 on the left one can see the geometrical fine grid of the fluid domain where the rod fuel is defined by the circle in the middle of the square region. The multiscale procedure extends the original solution from the fluid region to the solid zone and then projects it on the coarse grid considering a new mixture material.

3D-porous mass equation In order to illustrate this multiscale procedure one can consider the mass conservation equation in strong (left) and variational (right) form over the domain Ω

$$\nabla \cdot \rho \mathbf{v} = 0 \qquad \int_{\Omega} \nabla \cdot \rho\, \mathbf{v}\, \psi\, d\mathbf{x} = 0\,. \tag{3.7}$$

In the first step the domain is extended $\Omega \to \widehat{\Omega}$ and the (3.7) written as

$$\int_{\widehat{\Omega}} \nabla \cdot \rho \mathbf{v}\, \psi\, d\mathbf{x} = 0\,, \tag{3.8}$$

with density and velocity field extended on $\widehat{\Omega}$. Now we project the equation on the fine grid over test base function $\psi_h(k)$

$$\int_{\widehat{\Omega}_h} \nabla \cdot \rho \mathbf{v}_h\, \psi_h(k)\, d\mathbf{x} = 0$$

and consider the solution $(\widehat{\mathbf{v}}, \widehat{p}, \widehat{T})$ on the coarse grid with test functions $\widehat{\psi}_h(j)$.

If the fine test base function $\psi_h(k)$ is a complete set in the coarse grid then we have

$$\widehat{\psi}_h(j) = \sum_k C_{kj}\, \psi_h(k) \qquad \widehat{\mathbf{v}}_h = \sum_j \mathbf{v}_j \sum_k C_{kj}\, \psi_h(k)\,.$$

The sum over the equation involving $\widehat{\mathbf{v}}_h$ on a coarse level becomes

$$\sum_k C_{kj} \int_{\widehat{\Omega}_h} \nabla \cdot \rho \mathbf{v}_h\, \psi_h(k)\, d\mathbf{x} = \int_{\widehat{\Omega}_h} \nabla \cdot \rho \mathbf{v}_h\, \widehat{\psi}_h(j)\, d\mathbf{x} = 0$$

or

$$\int_{\widehat{\Omega}_h} \nabla \cdot \rho(\mathbf{v}_h + \widehat{\mathbf{v}}_h - \widehat{\mathbf{v}}_h)\, \widehat{\psi}_h(j)\, d\mathbf{x} = 0\,.$$

Therefore the incompressibility constraint can be written as

$$\int_{\widehat{\Omega}_h} \nabla \cdot \rho\, \widehat{\mathbf{v}}_h\, \widehat{\psi}_h(j)\ d\mathbf{x} = \int_{\widehat{\Omega}_h} P^c_{ef}(\widehat{\mathbf{v}}_h,\, \mathbf{v}_h)\, \widehat{\psi}_h(j)\, d\mathbf{x} \quad \forall \widehat{\psi}_h(j)\,,$$

with the mass fine-coarse transfer operator P^c_{ef} defined by

$$P^c_{ef}(\widehat{\mathbf{v}}_h, \mathbf{v}_h) = \nabla \cdot \rho(\widehat{\mathbf{v}}_h - \mathbf{v})\,.$$

Usually one assumes

$$R^c_{ef} = \int_{\widehat{\Omega}_h} P^c_{ef}(\widehat{\mathbf{v}}_h,\, \mathbf{v}_h)\, \widehat{\psi}_h(j)\, d\mathbf{x} \approx \int_{\widehat{\Omega}_h} \nabla \cdot \zeta\, \rho \widehat{\mathbf{v}}_h\, \widehat{\psi}_h(j)\, d\mathbf{x} \quad \forall \widehat{\psi}_h(j)\,,$$

where ζ is the fraction of the structural material in the volume. In final form we can write

$$\int_{\widehat{\Omega}_h} \nabla \cdot (1-\zeta)\rho\, \widehat{\mathbf{v}}_h\, \widehat{\psi}_h(j)\; d\mathbf{x} = 0\,. \tag{3.9}$$

The (3.9) implies that the velocity in the porous media is $(1-\zeta)\,\widehat{\mathbf{v}}_h$ while the real velocity in the fluid is $\widehat{\mathbf{v}}_h$. The velocity $(1-\zeta)\,\widehat{\mathbf{v}}_h$ is called reduced velocity. The liquid mass flux is $\rho\,\widehat{\mathbf{v}}_h \cdot \mathbf{n} A_f$ with A_f the fluid area and $\mathbf{n}$ the unit normal. If we add the area A_m of the structural material then the liquid mass flux through the porous media area $A = A_f + A_m$ is $\rho\,\widehat{\mathbf{v}}_h \cdot \mathbf{n} A$ which is $(1-\zeta)$ times greater than the real one.

3D-porous momentum equation In a similar way we can write the momentum equation as [18, 16]

$$\begin{aligned}\int_\Omega \frac{\partial\, \rho\, \widehat{\mathbf{v}}_h}{\partial t} \cdot \widehat{\phi}_{hj}\, d\mathbf{x} + \int_\Omega (\nabla \cdot \rho\, \widehat{\mathbf{v}}_h \widehat{\mathbf{v}}_h) \cdot \widehat{\phi}_{hj}\, d\mathbf{x} - \\ \int_\Omega \widehat{p}_h\, \nabla \cdot \widehat{\phi}_h(k)\, d\mathbf{x} + \int_\Omega \bar{\bar{\tau}}_h : \nabla \widehat{\phi}_{hj}\, d\mathbf{x} - \int_\Omega \rho \mathbf{g} \cdot \widehat{\phi}_{hj}\, d\mathbf{x} = \\ \int_\Omega R^m_{cf}(p_h, \mathbf{v}_h, \widehat{p}_h, \widehat{\mathbf{v}}_h) \cdot \widehat{\phi}_{hj}\, d\mathbf{x}\,,\end{aligned}$$

where the fine-coarse transfer operator $R^m_{cf}(p_h, \mathbf{v}_h, \widehat{p}_h, \widehat{\mathbf{v}}_h)$ is defined by

$$R^m_{cf}(p_h, \mathbf{v}_h, \widehat{p}_h, \widehat{\mathbf{v}}_h) = P^m_{cf}(\widehat{p}_h - p_h, \widehat{\mathbf{v}}_h - \mathbf{v}_h) + K^m_{cf}(\mathbf{v}_h)\,,$$

where

$$\int_\Omega P^m_{cf}(\widehat{p}_h - p_h, \widehat{\mathbf{v}}_h - \mathbf{v}_h) \cdot \widehat{\phi}_{hj}\, d\mathbf{x} \tag{3.10}$$

is the volume fine-coarse term and

$$\int_\Omega K^m_{cf}(\mathbf{v}_h) \cdot \widehat{\phi}_{hj}\, d\mathbf{x} = \int_\Gamma (-p_h\, \vec{n} + \tau_h \cdot \vec{n}) \cdot \widehat{\phi}_{hj}\, d\mathbf{s} \tag{3.11}$$

the pressure term. For details see [18, 16, 1, 17].

3D-porous energy equation For the energy equation, by applying the same procedure, we have [18, 16]

$$
\begin{aligned}
&\int_\Omega \frac{\partial \rho\, C_p\, \widehat{T}_h}{\partial t}\, \widehat{\varphi}_{hj}\, d\mathbf{x} + \int_\Omega \widehat{\varphi}_{hj}\, \nabla \cdot \rho\, C_p \widehat{\mathbf{v}}_h\, \widehat{T}_h\, d\mathbf{x} \\
&\int_\Omega k\, \nabla \widehat{T}_h \cdot \nabla \widehat{\varphi}_{hj}\, d\mathbf{x} - \int_\Omega Q_h\, \widehat{\varphi}_{hj}\, d\mathbf{x} = \int_\Omega R^e_{cf}(T_h)\, \widehat{\varphi}_{hj}\, d\mathbf{x}
\end{aligned}
$$

where the energy source from the fine scale is R^e_{cf}, i.e.,

$$
\begin{aligned}
&R^e_{cf}(T_h) = S^e_{cf}(T_h) + P^e_{cf}(\widehat{T}_h - T_h, \widehat{\mathbf{v}}_h - \mathbf{v}_h) + T^e_{cf}(\widehat{\mathbf{v}}_h, \mathbf{v}_h) \\
&\int_\Omega S^e_{cf}(T_h)\, \widehat{\varphi}_{hj}\, d\mathbf{x} = \int_\Gamma k\, (\nabla T_h \cdot \vec{n})\, \widehat{\varphi}_{hj}\, d\mathbf{x} \quad \text{fuel heat source} \\
&\int_\Omega P^e_{cf}(\widehat{T}_h - T_h, \widehat{\mathbf{v}}_h - \mathbf{v}_h)\, \widehat{\varphi}_{hj}\, d\mathbf{x} = \text{volume term.}
\end{aligned}
$$

For details see [18, 16]. This leads to the 3D-porous system of equations strong form to take the following form

$$
\nabla \cdot r\, \rho\, \widehat{\mathbf{v}} = 0\,, \tag{3.12}
$$

$$
\frac{\partial r\, \rho\, \widehat{\mathbf{v}}}{\partial t} + (\nabla \cdot r\, \rho\, \widehat{\mathbf{v}}\widehat{\mathbf{v}}) = -\nabla(r\, \widehat{p} + \widehat{p}_l) + \nabla \cdot (r\, (\widehat{\tau})) + r\rho\mathbf{g}\,, \tag{3.13}
$$

$$
\frac{\partial\, r\, \rho\, C_p\, \widehat{T}}{\partial t} + \nabla \cdot (r\, \rho C_p \widehat{\mathbf{v}}\, \widehat{T}) = \nabla \cdot \left[r \left(k + \frac{\mu_t}{Pr_t} \right) \nabla \widehat{T} \right] + r\, Q + W\,, \tag{3.14}
$$

where $r = 1 - \zeta$. We remark that the main terms coming from the fine grid are the surface terms $W(\mathbf{x})$ and $\widehat{p}_l$. We note that the system (3.12-3.14) resembles the system of conservative equations for two-phase flow.

3.3.1 Plenum model $->$ 3D-CFD module

In the reactor there are two plena: the upper plenum and the lower plenum located below and above the core, respectively. The module that describes a plenum module is basically a standard 3D-CFD module. In these regions, the coolant flows in a quite open three-dimensional domain and the coolant state, defined by velocity $\mathbf{v}$, pressure p and temperature T, can be determined by: the mass conservation equation Eq. (3.3), the momentum conservation equation Eq. (3.4) and the energy equation Eq. (3.5). In this region the density, the viscosity, the thermal conductivity and specific thermal capacity

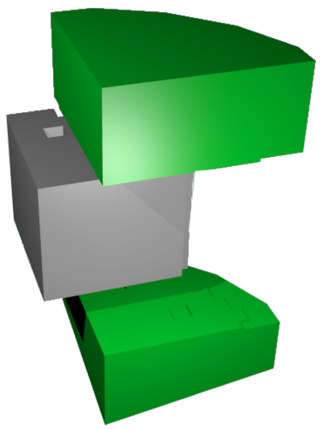


Figure 3.7: Schematic of the plenum volumes (in green).

can be a function of temperature. For example the density can be a function defined as $\rho = \rho(T) = a + bT$. The equations (3.3-3.5) are stabilized with a standard up-wind for finite element method (FEM).

3.3.2 CORE model (3D-porous module)

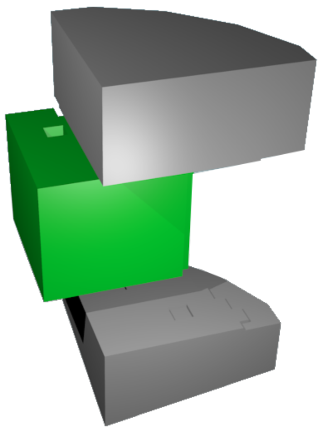


Figure 3.8: Core model. Schematic of the core volume (in green).

Mathematical model In the core region the geometry is so complex and detailed that a porous model approximation is necessary since a direct simulation of the velocity, pressure and temperature distributions is not possible.

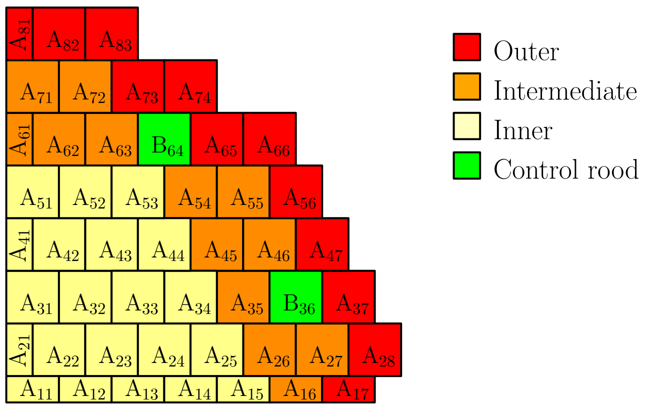

A_{81} A_{82} A_{83}
A_{71} A_{72} A_{73} A_{74}
A_{61} A_{62} A_{63} B_{64} A_{65} A_{66}
A_{51} A_{52} A_{53} A_{54} A_{55} A_{56}
A_{41} A_{42} A_{43} A_{44} A_{45} A_{46} A_{47}
A_{31} A_{32} A_{33} A_{34} A_{35} B_{36} A_{37}
A_{21} A_{22} A_{23} A_{24} A_{25} A_{26} A_{27} A_{28}
A_{11} A_{12} A_{13} A_{14} A_{15} A_{16} A_{17}
🟥 Outer
🟧 Intermediate
⬜ Inner
🟩 Control rood

Figure 3.9: Core model. Core power distribution.

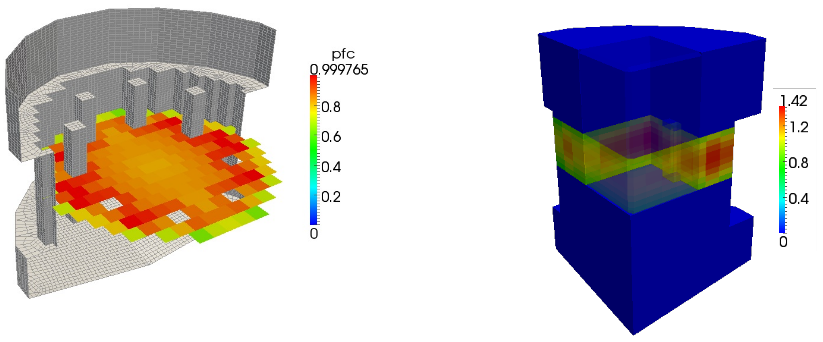


Figure 3.10: Core model. Computational vertical and horizontal core power distributions.

The 3D-porous module consists of the following equations: the mass conservation equation defined in (3.12), the momentum conservation equation defined in (3.13) and the energy equation defined in (3.14). As shown in Figure 3.8 the 3D-porous module is connected with the 3D-CFD modules of the lower plenum and upper plenum. At these interfaces the fluid must flow from one module to the other and conserve the flow momentum. The 3D-CFD module in the plenum is different from the 3D-porous module since in the porous medium consists of fluid and solid material. This issue is going to be discussed in the interface model Section.

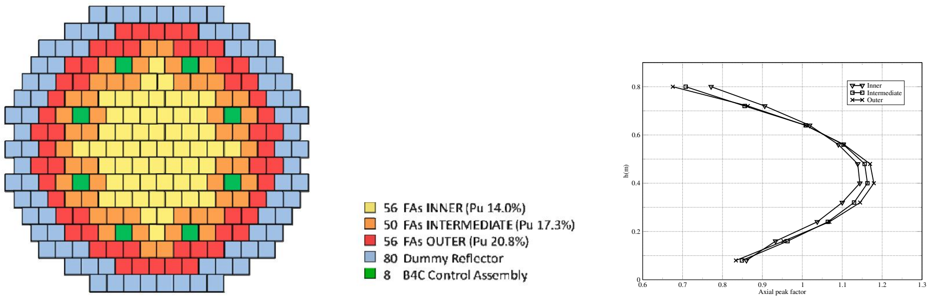
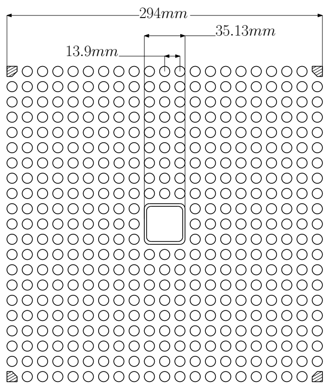


Figure 3.11: Core model. Reactor fuel distribution, on the left, and axial peak factor, on the right.



	area (m^2)
Pin area	370.606×10^{-4}
Corner box area	5.717×10^{-4}
Central box beam	2.092×10^{-4}
Channel central area	12.340×10^{-4}
Coolant area	473.605×10^{-4}
Assembly area	864.360×10^{-4}
Coolant/Assembly ratio	0.5408

Figure 3.12: Schematic of the assembly on the left and its main characteristics in the Table on the right.

Fuel distribution The space heat power distribution is defined in two steps. First one must define the average power value and then the fuel peak factors. The average value heat source $\dot{q}_v$ should be set independently of a space-dependent configuration. In order to obtain the assembly averaged specific $\dot{q}_v$ and linear $\dot{q}_l$ heat power, the total core power is to be divided over $N_a - 8$. We obtain

$$\dot{q}_l = \frac{\dot{Q}}{(N_a - 8)L_{core}}$$

where A_{core} is

$$(N_a - 8)A_{assembly}$$

$$\dot{q}_v = \frac{\dot{Q}}{(N_a - 8)L_{core}A_{assembly}},$$

Inner		Intermediate		Outer	
A11/2*	8.606	A16/2*	10.775	A17/2*	10.036
A12/2*	8.801	A25	10.188	A27	10.908
A13/2*	9.043	A26	10.269	A28	7.842
A14/2*	9.301	A34	9.943	A37	8.465
A15/2*	9.560	A36	9.249	Aa7	8.520
A21/2*	8.678	A45	10.132	A55	10.962
A22	8.759	A46	9.399	A56	8.621
A23	8.943	A52	9.696	A65	9.766
A24	9.112	A53	9.857	A66	7.776
A3l	8.811	A54	10.400	A72	9.746
A32	8.921	A62	9.410	A73	8.833
A33	9.071	A64	9.057	A74	7.699
A41/2*	8.850	A71	9.318	A81/2*	8.029
A42	8.903			A82	7.806
A43	9.043			A83	6.922
A44	9.218				
A5l	8.790				
A61/2*	8.725				

	Power		Pavg	
	$MWth$	# FA	$MWth$	ffrad
Inner	501.41	56	8.95	1.07
Intermediate	489.23	50	9.78	1.10
Outer	491.60	56	8.78	1.25
Total	1482.24	162	9.15	

Table 3.2: Core model. Horizontal fuel distribution. The value labeled with star (*) are referred to the entire assembly.

where $\dot{Q}$ is the total heat thermal power, L_{core} the active core length. Then the space configuration of the heat source is taken into account by using peak factors. The power distribution over the horizontal quarter section of the core for the ALFRED reactor is shown in Figure 3.9. Each fuel assembly consists of a $n_p \times n_p$ pin lattice. The overall number of assembly positions in the core is N_a. Eight of these positions are dedicated to house special control rods and therefore the global number of fuel assemblies is N_{a-8}. For details one can see [19, 20]. The transverse core area is approximately circular but not axial symmetric. Therefore, in order to predict accurately the behavior of the reactor, a three-dimensional simulation should be performed. The model sets the fuel assemblies in three radial zones: N_{a1} fuel assemblies in the inner zone, N_{a2} fuel assemblies in the intermediate zone and the remaining N_{a3} fuel assemblies in the outer one. We label the assemblies as in Figure 3.9. The first row is labeled $A1$-i for $i = 1, \dots, 8$, the second row $A2$-i for $i = 1, \dots, 7$ and so on. We remark that the fuel assembly configuration is not based on a Cartesian grid but rather on a staggered grid. The power distribution factors, i.e. the power of each single fuel assembly over the average fuel assembly power, are mapped based on their row-column location. In order to obtain the horizontal power distribution of Figure 3.10 we must be reported the pick factors in the double-indexed array in which every assembly in the quarter of reactor is denoted by two indexes, both ranging from 0 to 7. The fuel area is divided into different zones, classified as inner core (I), outer core (O), control zone with no fuel (C) and reflector area (D). These are written in the a matrix with the corresponding horizontal power factor reported as

$$
\begin{array}{c}
A1 \\ A2 \\ A3 \\ A4 \\ A5 \\ A6 \\ A7 \\ A8 \\ \\
\end{array}
\begin{array}{c}
\begin{bmatrix}
fp_{A11} & fp_{A12} & fp_{A13} & fp_{A14} & fp_{A15} & fp_{A16} & fp_{A17} & 0 \\
fp_{A21} & fp_{A22} & fp_{A23} & fp_{A24} & fp_{A25} & fp_{A26} & fp_{A27} & fp_{A28} \\
fp_{A31} & fp_{A32} & fp_{A33} & fp_{A34} & R & fp_{A36} & fp_{A37} & 0 \\
fp_{A41} & fp_{A42} & fp_{A43} & fp_{A44} & fp_{A45} & fp_{A46} & fp_{A47} & 0 \\
fp_{A51} & fp_{A52} & fp_{A53} & fp_{A54} & fp_{A55} & fp_{A56} & 0 & 0 \\
fp_{A61} & fp_{A62} & R & fp_{A64} & fp_{A65} & fp_{A66} & 0 & 0 \\
fp_{A71} & fp_{A72} & fp_{A73} & fp_{A74} & 0 & 0 & 0 & 0 \\
fp_{A81} & fp_{A82} & fp_{A83} & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \\
\begin{matrix} 1 & \quad 2 & \quad 3 & \quad 4 & \quad 5 & \quad 6 & \quad 7 & \quad 8 \end{matrix}
\end{array} .
$$

We remark that in the control area the pick factor is set to zero. The vertical power factor can be assumed to have any distribution $g(z)$. This discrete profile is written as

$$
\begin{array}{ccc}
\left[\begin{matrix} Z_1 \\ Z_2 \\ Z_3 \\ Z_4 \\ Z_5 \\ Z_6 \end{matrix}\right. & \begin{matrix} g_I(Z_1) \\ g_I(Z_2) \\ g_I(Z_3) \\ g_I(Z_4) \\ g_I(Z_5) \\ g_I(Z_6) \end{matrix} & \left.\begin{matrix} g_O(Z_1) \\ g_O(Z_2) \\ g_O(Z_3) \\ g_O(Z_4) \\ g_O(Z_5) \\ g_O(Z_6) \end{matrix}\right] \\
height & I & O
\end{array} , \tag{3.15}
$$

where Z_i are the discrete vertical coordinates and $g_I(z)$ and $g_O(z)$ are the values in the INNER and OUTER zones. In particular for the ALFRED reactor each fuel assembly consists of a $n_p \times n_p = 21 \times 21$ pin lattice and the coolant/assembly ratio r is 0.548. The design of the assembly and its characteristics are shown in Figure 3.12. Each assembly has a square section with a side length of $L = 0.294m$ and this completely defines the horizontal core structure. For The overall number of assembly positions in the core is 170[15, 19, 20]. Eight of these positions are dedicated to house special control rods and therefore the global number of fuel assemblies is 162. The transverse core area is approximately circular but not axial symmetric. However, we can argue from the left part of Figure 3.11 that two symmetry planes passing through the reactor axis can be identified so that only a quarter domain has to be taken into account for the simulations. The model design distributes the fuel assemblies in three radial zones: $N_{a1} = 56$ fuel assemblies in the inner zone, $N_{a2} = 62$ fuel assemblies in the intermediate zone and the remaining $N_{a2} = 44$ fuel assemblies in the outer one. The power distribution factors, i.e. the power of a fuel assembly over the average fuel assembly power, are mapped in the right part of Figure 3.11 and the Table 3.2. The maximum power factor is 1.17, while the minimum is 0.74. In the computational model the following constant matrix of fuel pick factor is assumed

$$\begin{array}{c|cccccccc|}
A1 & 0.941 & 0.962 & 0.989 & 1.017 & 1.045 & 1.178 & 1.097 & 0. \\
A2 & 0.949 & 0.958 & 0.978 & 0.996 & 1.114 & 1.123 & 1.193 & 0.857 \\
A3 & 0.963 & 0.975 & 0.992 & 1.087 & R & 1.011 & 0.925 & 0. \\
A4 & 0.967 & 0.973 & 0.989 & 1.008 & 1.108 & 1.028 & 0.931 & 0. \\
A5 & 0.961 & 1.060 & 1.078 & 1.137 & 1.198 & 0.943 & 0. & 0. \\
A6 & 0.954 & 1.029 & R & 0.990 & 1.068 & 0.850 & 0. & 0. \\
A7 & 1.019 & 1.066 & 0.966 & 0.842 & 0. & 0. & 0. & 0. \\
A8 & 0.878 & 0.853 & 0.757 & 0. & 0. & 0. & 0. & 0. \\
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8
\end{array}$$

Pressure loss source In a similar way the pressure drop distribution, due to the spacer grids and the inlet and outlet of the core region, can be reported in a matrix and in a vector. The matrix defines the pressure losses in each assembly as show in the top part of Table 3.3. The vertical pressure loss factor can be assumed to have any distribution $q(z)$. This discrete profile is written as shown in the bottom part of Table 3.3. where Z_i are the discrete vertical coordinates and $q0$ and $q1$ are the values in the INNER and OUTER zones. The pressure loss distribution defined by the spacer grids, shown in Figure 3.13, can be easily written by defining the modulus of the vector function $|\bar{\beta}|$

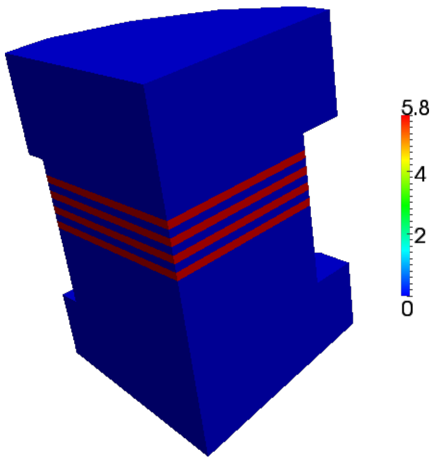


Figure 3.13: Pressure loss source due to spacer grids.

A1	β_{A1_1}	β_{A1_2}	β_{A1_3}	β_{A1_4}	β_{A1_5}	β_{A1_6}	β_{A1_7}	β_{A1_8}
A2	β_{A2_1}	β_{A2_2}	β_{A2_3}	β_{A2_4}	β_{A2_5}	β_{A2_6}	β_{A2_7}	β_{A2_8}
A3	β_{A3_1}	β_{A3_2}	β_{A3_3}	β_{A3_4}	β_{A3_5}	β_{A3_6}	β_{A3_7}	β_{A3_8}
A4	β_{A4_1}	β_{A4_2}	β_{A4_3}	β_{A4_4}	β_{A4_5}	β_{A4_6}	β_{A4_7}	β_{A4_8}
A5	β_{A5_1}	β_{A5_2}	β_{A5_3}	β_{A5_4}	β_{A5_5}	β_{A5_6}	β_{A5_7}	β_{A5_8}
A6	β_{A6_1}	β_{A6_2}	β_{A6_3}	β_{A6_4}	β_{A6_5}	β_{A6_6}	β_{A6_7}	β_{A6_8}
A7	β_{A7_1}	β_{A7_2}	β_{A7_3}	β_{A7_4}	β_{A7_5}	β_{A7_6}	β_{A7_7}	β_{A7_8}
A8	β_{A8_1}	β_{A8_2}	β_{A8_3}	β_{A8_4}	β_{A8_5}	β_{A8_6}	β_{A8_7}	β_{A8_8}
	1	2	3	4	5	6	7	8

Z_1	$q_I(Z_1)$	$q_O(Z_1)$
Z_2	$q_I(Z_2)$	$q_O(Z_2)$
Z_3	$q_I(Z_3)$	$q_O(Z_3)$
Z_4	$q_I(Z_4)$	$q_O(Z_4)$
Z_5	$q_I(Z_5)$	$q_O(Z_5)$
Z_6	$q_I(Z_6)$	$q_O(Z_6)$
height	I	O

Table 3.3: Radial and axial pressure loss peak factor on the top and the bottom, respectively.

where the discrete pressure drops are defined by

$$\frac{1}{2}\bar{\beta}(\mathbf{x}) \cdot \mathbf{v}|\mathbf{v}| \,. \tag{3.16}$$

If we assume the flow to be approximately vertical then $\bar{\beta}(\mathbf{x}) \cdot \mathbf{v}|\mathbf{v}|/2 = |\bar{\beta}||\mathbf{v}|^2/2$. This is the result of the usual mono-dimensional approximation.

3.4 1D-CFD scale

In order to derive the mono-dimensional set of conservation equations we introduce the variational form of equations (3.3-3.5). The variational form of the incompressible mass conservation equation can be obtained by multiplying by $\psi \in P(\Omega)$ the (3.3) as

$$\int_\Omega \psi \, \nabla \cdot \ \mathbf{v} \, d\mathbf{x} = 0 \quad \forall \psi \in P(\Omega) \, . \tag{3.17}$$

In a similar way the momentum and energy equation in variational form can be obtained by multiplying the (3.4) and (3.5) by $\phi \in \mathbf{V}(\Omega)$ and $\varphi \in H(\Omega)$, respectively. For the momentum equation we have

$$\begin{aligned} \int_\Omega \frac{\partial \rho \mathbf{v}}{\partial t} \cdot \phi \, d\mathbf{x} + \int_\Omega (\nabla \cdot \ \rho \mathbf{v}\mathbf{v}) \cdot \phi \, d\mathbf{x} = - \int_\Gamma (p\vec{n} - \bar{\tau} \cdot \vec{n}) \cdot \phi \, d\mathbf{s} + \\ \int_\Omega p \, \nabla \cdot \phi \, d\mathbf{x} - \int_\Omega \bar{\tau} : \overline{\nabla \phi} \, d\mathbf{x} + \int_\Omega \rho \mathbf{g} \cdot \phi \, d\mathbf{x} \qquad \forall \phi \in \mathbf{V}(\mathbf{\Omega}) \, . \end{aligned} \tag{3.18}$$

For the energy equation we have

$$\begin{aligned} \int_\Omega \frac{\partial \rho C_p T}{\partial t} \varphi \, d\mathbf{x} + \int_\Omega \nabla \cdot (\rho \mathbf{v} C_p T) \, \varphi \, d\mathbf{x} = \int_\Omega \Phi \, \varphi \, d\mathbf{x} \\ - \int_\Omega k_f \, \nabla T \cdot \nabla \varphi \, d\mathbf{x} + \int_\Omega \dot{Q} \varphi \, d\mathbf{x} + \int_\Gamma (k_f \nabla T \cdot \vec{n}) \, \varphi \, d\mathbf{s} \quad \forall \varphi \in \mathbf{H}(\Omega) \end{aligned} \tag{3.19}$$

The surface integrals are defined by the boundary conditions. The test functions ψ, ϕ, φ are the weight to average the equation over the scale.

If we consider a domain discretization and the corresponding subspaces of $P(\Omega)$, $\mathbf{V}(\Omega)$ and $H(\Omega)$ parametrized by the element length h, the system (3.17-3.19) can be identified with its numerical approximation. We remark that setting $P(\Omega) = P_h \subset L^2_0(\Omega)$, $\mathbf{V}(\Omega) = \mathbf{X}_h \subset \mathbf{H}^1(\Omega)$ and $H(\Omega) = X_h \subset H^1(\Omega)$ gives finite dimensional solution with the Finite Element Method (FEM). If one sets $P(\Omega) = P_h \subset L^2_0(\Omega)$, $\mathbf{V}(\Omega) = \mathbf{X}_h \subset \mathbf{L}^2(\Omega)$ and $H(\Omega) = X_h \subset L^2(\Omega)$ then one has a finite dimensional solution with the Finite Volume Method (FVM). As $h \to 0$ we have the continuous solution, for details the interested reader can see [18, 16, 1, 17]. The 1D-CFD module used in FEMUs is obtained by integrating (3.17-3.19) with mono-dimensional weight functions. Briefly one can set $\psi = \psi(s)$, $\phi = \phi(s)$ and $\varphi = \varphi(s)$ implying that the test functions are only a function of the mono-dimensional coordinate s. This module is properly used for mono-dimensional flows such as channels with single fluid.

1D-CFD mass equation Let A be a surface perpendicular to the center-line. Over the surface A we define average density and average velocity as

$$\bar{\rho} = \frac{\int_A \rho \, dA}{A} \qquad \bar{v} = \frac{\int_A \rho \, v \, dA}{\bar{\rho} A} \,. \tag{3.20}$$

With this definition we write

$$\int_0^L \psi(s) \frac{\partial}{\partial s} (\bar{\rho} \, \bar{v} \, A) \, ds = \int_0^L \psi(s) \, S_s \, ds \quad \forall \psi \in P(0, L) \,, \tag{3.21}$$

where S_s is the mass source from surface integral. Usually $S_s \approx 0$.

1D-CFD momentum equation In a similar way for the average quantities ($\bar{\rho}$, $\bar{v}$, $\bar{p}$) and $\phi \approx \phi(s)$ the momentum equation becomes

$$\begin{aligned} \int_0^L \bar{\rho} \, A \, (\frac{\partial}{\partial t} \bar{v}) \, \phi(s) \, ds + \int_0^L \bar{\rho} \bar{v} \, A \, (\frac{\partial}{\partial s} \bar{v}) \, \phi \, ds = \int_0^L \bar{p} \, A \frac{\partial}{\partial s} \phi \, ds \\ \int_0^L A \bar{\rho} \mathbf{g} \cdot \widehat{i}_s \, \phi \, ds + \int_0^L \phi(s) \, (M_s + M_v) \, ds \qquad \forall \phi \in V(0, L) \,, \end{aligned} \tag{3.22}$$

where M_s is from surface integral and M_v from volume contributions. Usually $M_s \approx - k \, \frac{\rho}{2} \, \bar{u} |\bar{u}|$ (pressure loss). The volume contribution M_v consists of several terms. Is is easy to see that $M_v = M_{v,\tau}(\bar{\tau}) + M_{v,vv}(\bar{\mathbf{v}\mathbf{v}} - \bar{v}\bar{v})$ with obvious definition of the terms $M_{v,\tau}$ and $M_{v,vv}$. For details see [18, 16].

1D-CFD energy equation For the Energy equation the average quantities involved are ($\bar{\rho}$, $\bar{v}$, $\bar{p}$, $\bar{T}$) and the equation becomes

$$\begin{aligned} \int_0^L \frac{\partial \, A \, \bar{\rho} \, C_p \, \bar{T}}{\partial t} \varphi(s) \, ds + \int_0^L (\frac{\partial}{\partial s} \, (A \, \bar{\rho} \, \bar{v} C_p \, \bar{T}) \, \varphi \, ds = \\ - \int_0^L \dot{Q}_s \varphi \, d\mathbf{x} + \int_0^L \dot{Q}_v \varphi \, ds \quad \forall \varphi \in \mathbf{H}(\Omega) \,. \end{aligned} \tag{3.23}$$

We have a contribution from the boundary and a contribution from the volume. From surface integrals we define Q_s (heat exchange through surfaces). From volume integral (volume interaction) we define

$$Q_v = Q_{v,q}(\bar{\tau}) + Q_{v,vv}(\bar{\mathbf{vT}} - \bar{v}\bar{T}) + Q_{v,\Phi}(\Phi) \tag{3.24}$$

with $Q_{v,q}$ internal heat conduction and $Q_{v,vv}$ turbulent term. For details see [18, 16].

1D-CFD scale model In standard form (non variational form) the 1D-CFD scale model, defined by (3.21-3.23), can be written as follows

$$\frac{\partial}{\partial s}\big(\bar{\rho}\,\bar{v}\,A\big) = 0\,. \tag{3.25}$$

$$\frac{\partial}{\partial t}A\bar{\rho}\bar{v} + \bar{\rho}\,A\,\bar{v}\frac{\partial}{\partial s}\big(\,\bar{v}\,\big) + \,A\,\frac{\partial}{\partial s}\,P = -\,k\,\frac{\bar{\rho}}{2}\,\bar{v}|\bar{v}| + \bar{\rho}\,A\,g_s + M_v\,. \tag{3.26}$$

$$\frac{\partial}{\partial t}A\,\bar{\rho}\,C_p\,\bar{T} + \frac{\partial}{\partial s}\,A\,\bar{\rho}\,\bar{v}\,C_p\,\bar{T} = \rho\,A\,\bar{v}\,g_s + \bar{Q}_s + \bar{Q}_v\,. \tag{3.27}$$

The first equation states that the mass flux $\dot{m} = \bar{\rho}\,\bar{v}\,A$ is constant in space. The second equation is the transient Bernoulli equation and with no sources ($M_s = M_v = 0$) it states that $P + \rho v^2/2 + \rho g_s$ is constant in space.

3.5 Interfaces

3.5.1 3D-CFD $<->$ 3D-porous interface

As shown in Figure 3.14 the three-dimensional CFD and porous interfaces are the inlet and the outlet of the core. The core inlet, shown in yellow, is the interface between the lower plenum and the core while the core outlet is the interface between the upper plenum and the core. At these interfaces the fluid must flow from one module to the other and conserve the momentum. The 3D-CFD module in the plenum is different from the 3D-porous module since in the porous model the fluid and solid material are together inside the considered

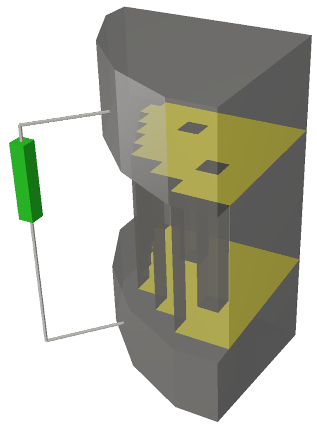


Figure 3.14: 3D-CFD $<->$ 3D-porous interfaces in a reactor model schematic diagram.

volume. The fluid flow is clearly reduced with respect to the total area. The flow reduction is defined by the volume ratios

$$\zeta = \frac{V_m}{V_{tot}} \quad r = 1 - \zeta \,, \tag{3.28}$$

where ζ and r are the ratio between no-fluid material and total volume and between fluid and total volume, respectively. The problem of the continuity of the velocity field through the interface 3D-CFD and 3D-porous modules can be solved by defining a new velocity field variable

$$\widehat{\mathbf{v}}^* = \widehat{\mathbf{v}} \qquad \text{3D-CFD}\,, \tag{3.29}$$

$$\widehat{\mathbf{v}}^* = r\,\widehat{\mathbf{v}} \qquad \text{3D-porous}\,. \tag{3.30}$$

With this definition the system (3.12-3.13) becomes

$$\nabla \cdot \rho\,\widehat{\mathbf{v}}^* = 0\,, \tag{3.31}$$

$$\frac{\partial \rho\,\widehat{\mathbf{v}}^*}{\partial t} + (\nabla \cdot \rho\,\widehat{\mathbf{v}}^*\widehat{\mathbf{v}}) = -\nabla(\widehat{p} + \widehat{p}_l) + \nabla \cdot (\widehat{\tau}^*) + \rho \mathbf{g}\,. \tag{3.32}$$

The system (3.31-3.32) is valid in the plenum and in the core region and maintain velocity and pressure fields continuous. The temperature field remains continuous at the 3D-CFD $<->$ 3D-porous interface and therefore the equation (3.14) can be applied in both regions with the appropriate value of r and the corresponding coefficients. With this approach the use of a unique equation in both region allows a strong and robust coupling between the velocity pressure and temperature fields.

3.5.2 3D $<->$ 1D interface

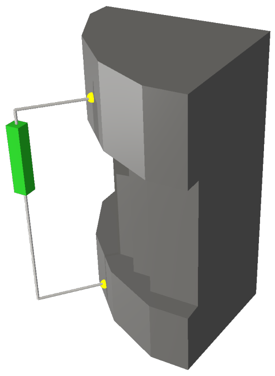


Figure 3.15: 3D-CFD $<->$ 1D-porous interfaces in a reactor model schematic diagram.

As shown in Figure 3.15 the 3D-CFD $<->$ 1D-porous interfaces are the inlet and the outlet of the plenum regions. The lower plenum inlet, shown as yellow sphere, is the interface between the lower plenum and the primary loop while the plenum outlet is the interface between the upper plenum and the primary loop. At these interfaces the fluid must flow from one-dimensional module to a three-dimensional one. In this interface the mass, momentum and the temperature field must be conserved. There are many sophisticated techniques to define a numerical algorithm able to identify the values to set on the interfaces for example one can use algorithms based on Mortar or Lagrangian multiplier. The one-dimensional module is essentially a hyperbolic differential equation and therefore it requires boundary conditions only in inflow regions. The interface 1D-porous $<->$ 3D-CFD that links the 1D-porous module to 3D-CFD module of the lower plenum is an outflow region for the reactor and therefore it does not require boundary conditions. The surface of the lower plenum is an inlet region for the core/plenum system and therefore the average temperature, velocity and pressure must set as boundary conditions. This is a very challenging situations. From the mono-dimensional primary loop we have the mass flux (or velocity in the normal direction), pressure and temperature. In three-dimensional domains if the vector of the velocity is fully specified on boundaries then the pressure is determined. The pressure can be specified only if the normal component of the velocity field is not imposed. Imposing pressure with brute force on the inlet surface leads to large oscillations and velocity discontinuities. For this reason we compute, with a non linear algorithm, the velocity field that matches the pressure of the plenum and the primary loop. The velocity field is kept continuous and the two pressures (in the plenum and in the primary loop) matches iteratively. This approach substantially reduces the time step of the core/plenum system to 1/4 or 1/5 of the step of the primary loop.

3.6 Test 1. Direct coupling

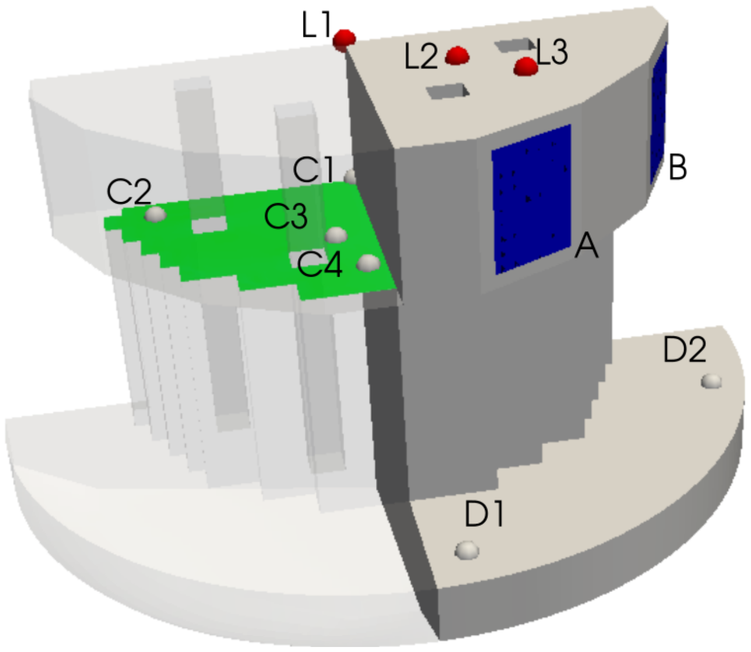


Figure 3.16: Test 1. Geometry and reference points.

In the first test we study the evolution of the primary loop of a LFR when the circulation pumps are switched off and the heat source of the reactor is kept in nominal working conditions. This test allows us to evaluate the nature of the natural convection and the temperature that can be reached in the system under this particular operating conditions. The principal part of the primary loop, such as the core and the plena, are modeled with three-dimensional models, while the remains part, pumps, heat exchanger, etc. are considered with a simplified mono-dimensional module

We use the in-house developed code (FEMUs) and the open-source code libmesh to solve the multidimensional and the mono-dimensional part, respectively. The two modules are integrated into the computational platform with the procedure explained in the Section 2.2. In this test the two computational domains do not overlap and they share only two junction points where the boundary conditions are updated with the data coming from the other code, thanks to the MEDmem interface as described in 2.4.3. The performance of this one-way coupling between two codes is analyzed. In Figure 3.16 we see the reactor and its reference points. The left and right outlet surface of the upper plenum are labeled with A and B, respectively. The points over the inlet of the lower plenum are indicated with D. have D_1 below the A surface and D_2 below the B surface. The points on the horizontal plane at $z = 3\,m$ in the core are labeled by C. The point C_1 is located in the assembly $A11$, the point C_3 in the assembly $A54$, C_2 in $A26$ and C_4 in $A62$. The points L_1, L_2 and L_3 indicate the location of the vertical lines along the diagonal of the upper plenum. In this test the mono-dimensional loop start from the upper plenum of the reactor and finish in the lower plenum. We remark that the two meshes do not overlap so new boundary conditions must be introduced as explained in Section 3.5.2. The mono and the three-dimensional problem are solved with finite element codes that are coupled through the MED interface of the computational platform as exposed in the Chapter 2.

3.6.1 Initial conditions

In this test the initial conditions of the reactor is the steady solution in fully working condition with core average velocity of about $\bar{v} = 1.56 m/s$. The corresponding average reduced velocity $\bar{v}^*$ is $\simeq 0.85$ with $r = 0.548$. The non-

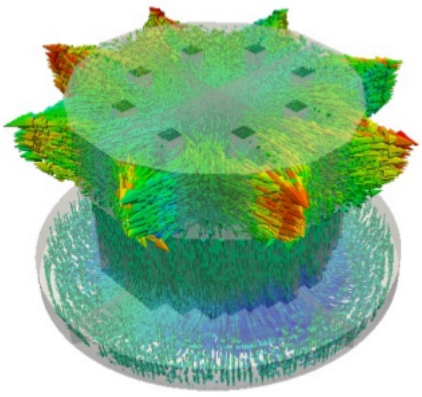


Figure 3.17: Test 1. Reduced velocity field in the reactor and plenum at $t = 0$.

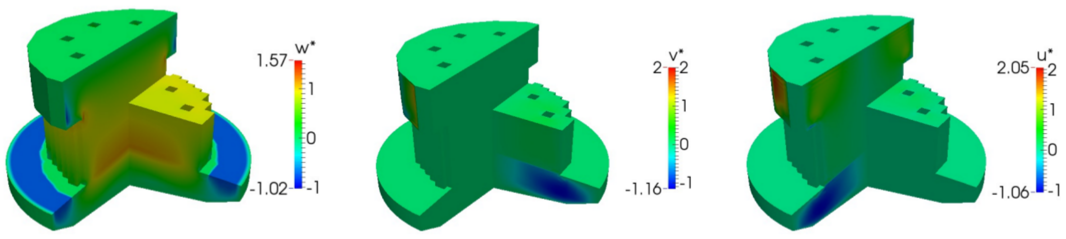


Figure 3.18: Test 1. The w,v and u-component of the reduced velocity field in the reactor at $t = 0$.

dimensional mass flux $m^* = m/\rho_0 A$ in the left outlet of the upper plenum is 1.24886 and the corresponding mass flux $m \simeq 16150 Kg/s$ ($A \simeq 1.23m$ and $\rho_0 = 10563 Kg/m^3$). Since there are 4×2 exits the total mass flux is $\simeq 128000 Kg/s$. We remark that the mass flux of the left and right outlet of the upper plenum are slightly different due to the fact that the reactor is not symmetric with respect to the diagonal. The velocity field of the initial solution is shown in Figure 3.17. In Figures 3.18 the three components w,v and u of the reduced velocity field are shown in the various part of the reactor at the initial time. The u and v component of the velocity field have a strong component in the lower plenum and almost vanishing inside the core to gain again its value in the upper plenum. In Figure 3.18 we note that the inlet velocity of the lower plenum is vertical and uniform with value set to $w = w^* = -1.02m/s$. The inlet velocity boundary conditions are defined by the primary loop velocity

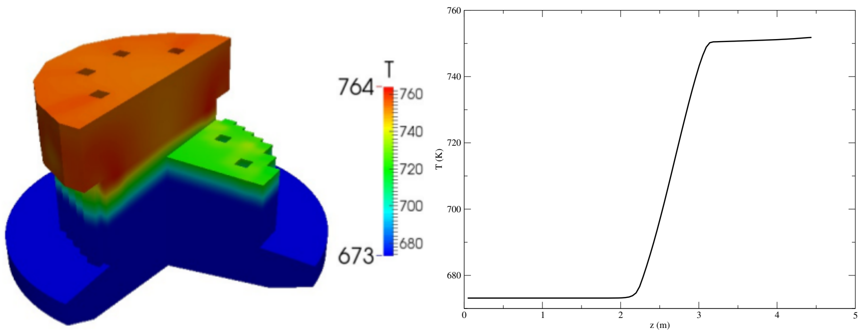


Figure 3.19: Test 1. Temperature distribution in the reactor (left) and along the vertical at the center of the reactor (right) at time $t = 0$.

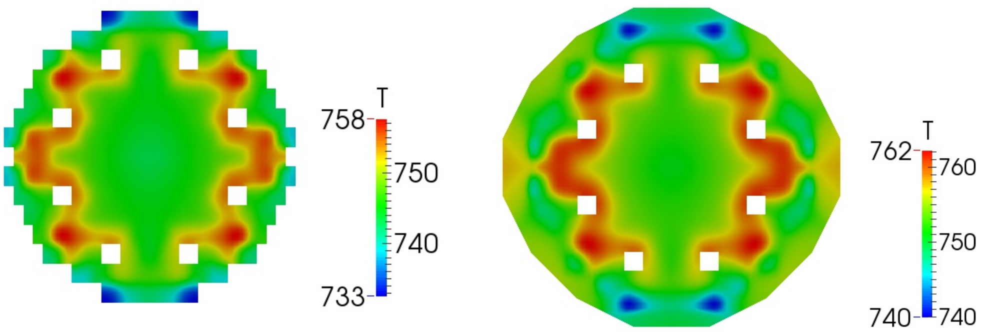


Figure 3.20: Test 1. Temperature distribution over the plane at $z = 3m$ (left) and $z = 3.3m$ (right) at time $t = 0$.

based on pressure values. The primary loop variables are defined as a unique point since the 1D-porous module is mono-dimensional. In Figure 3.19 the temperature distribution in the reactor at the initial time is shown. On the left of Figure 3.19 the temperature distribution is over all the reactor. The average temperature T_{in} over the inlet is $673.15K$ and average temperature in the left and right outlet of the upper plenum is $T_{out} = 754.9K$ for a inlet/outlet temperature difference of $81.75K$. In Figure 3.20 the temperature distribution over the plane at $z = 3m$ (left) and $z = 3.3m$ (right) at time $t = 0$ sec are shown. Over the plane at $z = 3m$ (core exit) the maximum temperature is near the assembly with higher peak factor. Over the plane obtained by a cut at $z = 3.3m$ the maximum temperature $(762K)$ is in the region between

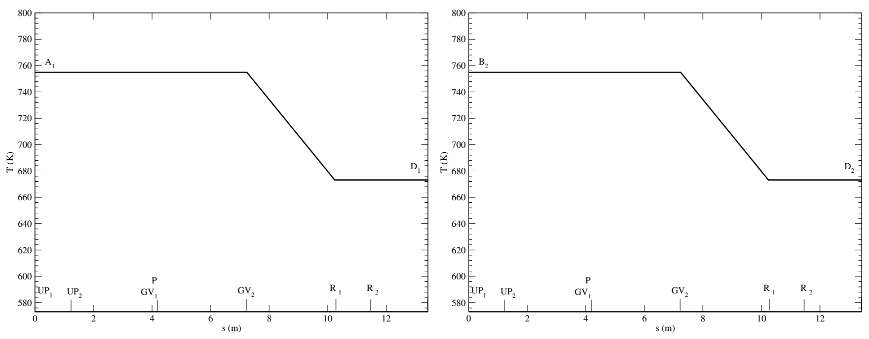


Figure 3.21: Test 1. Temperature distribution in the primary loop *A*-*D* (left) and *B*-*D* (right) at time $t = 0$.

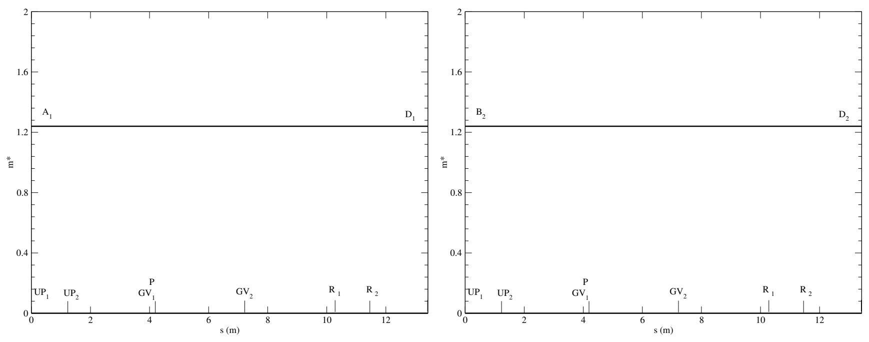


Figure 3.22: Test 1. Non-dimensional mass flux m^* in the primary loop *A*-*D* (left) and *B*-*D* (right) at time $t = 0$.

the left and right exit of the upper plenum where a stagnation zone does not allow efficient cooling. In Figures 3.21-3.23 the temperature, the pressure and the mass flow rate initial conditions are shown in the primary loop for the branch *A*-*D* (left) and *B*-*D* (right) as a function of the local mono-dimensional coordinate system. The GV_1 and GV_2 labels refer to the steam generator. In the steam generator area GV_1-GV_2 the gravity is neglected.

3.6.2 Reactor evolution in natural convection flow

After the initial time the pumps are switched off and the system is moved only by natural convection. The distribution of power remains the same as shown in Figure 3.24. In Figure 3.25 one can see the average temperature $\bar{T}$ and non-dimensional mass flux m^* on left exit surface A and on the right exit surface B

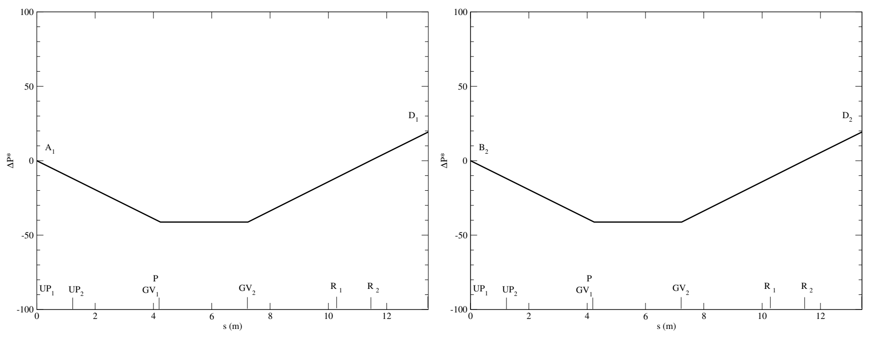


Figure 3.23: Test 1. Non-dimensional ΔP^* in the primary loop A-D (left) and B-D (right) at time $t = 0$.

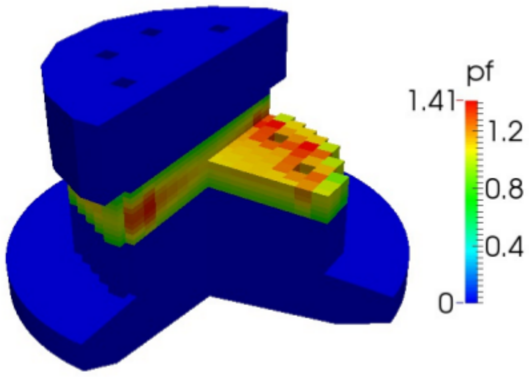


Figure 3.24: Test 1. Power distribution for $t > 0$.

as a function of time. The average temperature is reported on the left where, after the pumps are switched off, it easy to note that there is a substantial increase to reach constant value after $30s$. The difference between the surface A and B is not large but it implies that the system is not symmetric with respect to the diagonal and it is fully three-dimensional. The non-dimensional mass flux $m^* = m/\rho_0 A$ is reported on right of Figure 3.25. We recall that the area $A \simeq 1.23m$ and $\rho_0 = 10563Kg/m^3$. Due to the lack of pump pressure the mass flux drops to a constant value, which is 1/3 of the nominal mass flux, after approximately after $30s$. In Figure 3.26 the temperature and the reduced w-component of the velocity field is reported on the point D_1 and D_2 as a function of time. The points D_1 and D_2 are located in the inlet of the lower plenum. The values are the same since the boundary conditions are imposed uniformly from the primary loop. We recall that $m^* = v^*$ if the

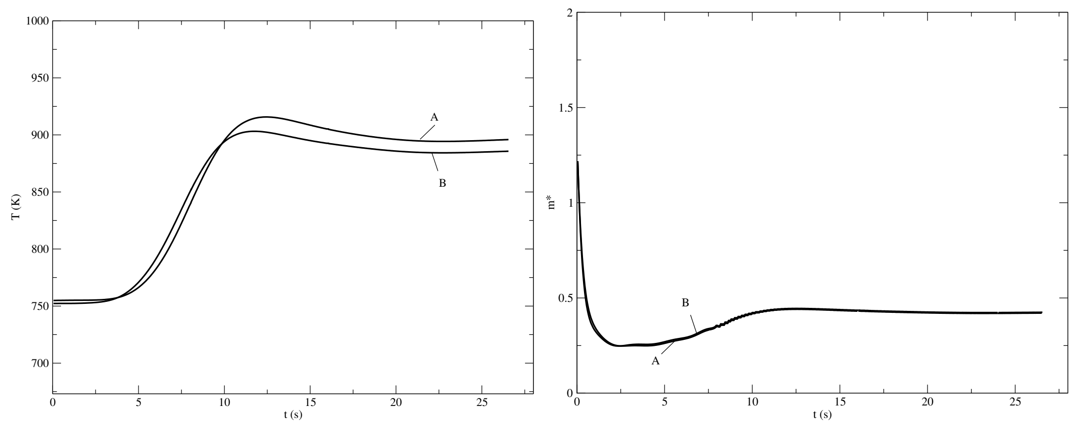


Figure 3.25: Test 1. Average temperature (left) and mass flux $m^* = m/\rho_0 A$ (right) on left exit surface A and on the right exit surface B as a function of time.

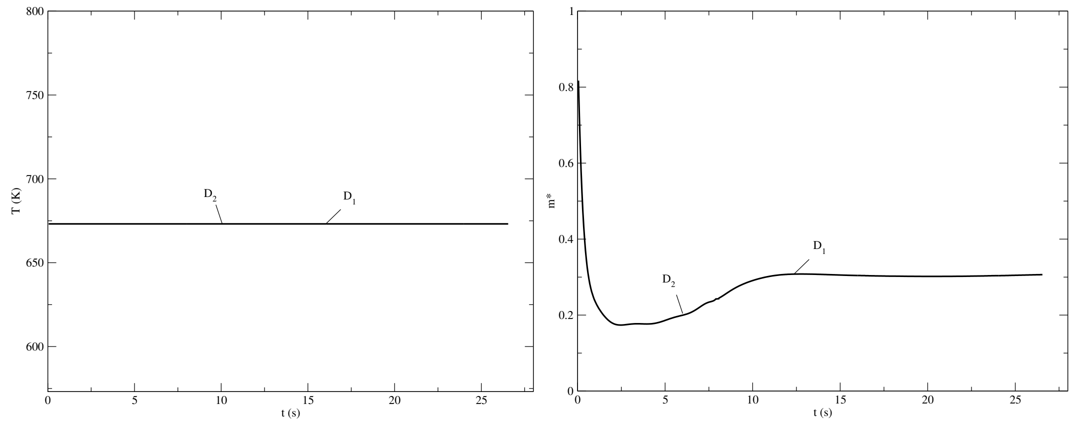


Figure 3.26: Test 1. Temperature (left) and velocity w^* (right) on D_1 and D_2 as a function of time.

temperature is the reactor inlet temperature $673.15K$ and the density is the reference density ρ_0. The non dimensional pressure drop $\Delta P^* = p^* - p_{ref}$ on D_1 and D_2 is shown as a function of time in the left part of Figure 3.27. The non-dimensional pressure is defined by $p/\rho_{ref}\, u_{ref}$ where $\rho_{ref} = \rho_0 = \rho(673.15K)$. The reference pressure p_{ref} is the value of the pressure at the lowest point of the upper plenum exit. The non-dimensional pressure Δp^* initially at 7.3 drops to approximately 1.1. This implies that the pressure loss ΔP initially t $0.76bar$ drops to $0.12bar$. In this case we neglect the accidental pressure losses

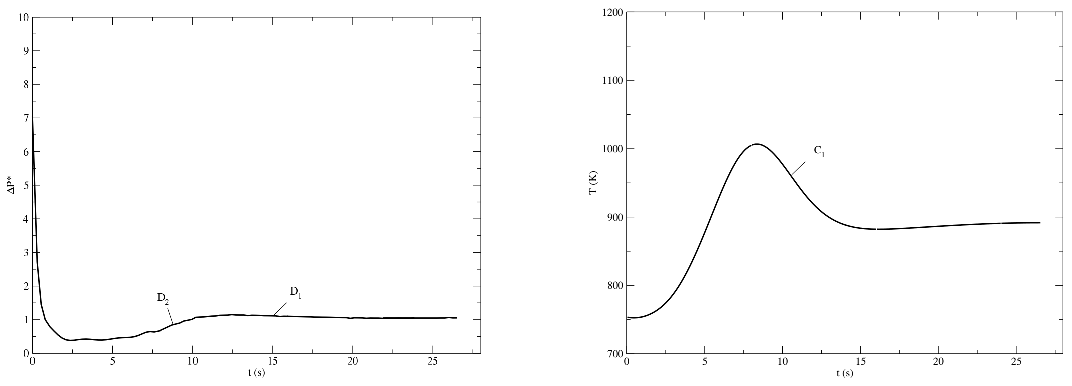


Figure 3.27: Test 1. On the left, non dimensional pressure drop ΔP^* on D_1 and D_2 as a function of time. On the right, the temperature profile at the point C_1 as a function of time.

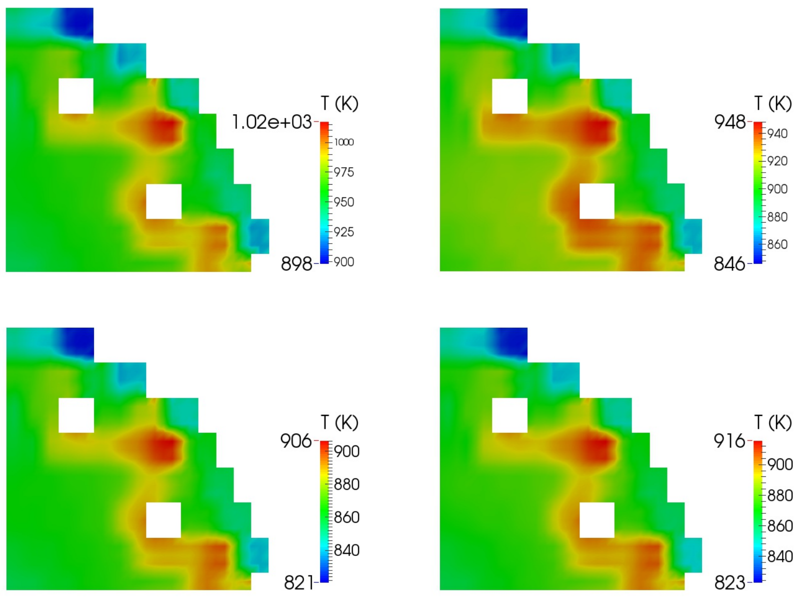


Figure 3.28: Test 1. Temperature profile on section at $z = 3.0m$ (defined by C_1, C_2, C_3, C_4) for $t =$ 8, 10, 16 and 26 s.

due to reactor grids. In the right part of Figure 3.27 the temperature is shown over the plane at $z = 3m$ of the reactor at the point C_1 as a function of time. The point C_1 is at the center of the reactor. We note that in this point the temperature reach $1000K$ to come back to $900K$ after $15sec$. In Figure 3.28

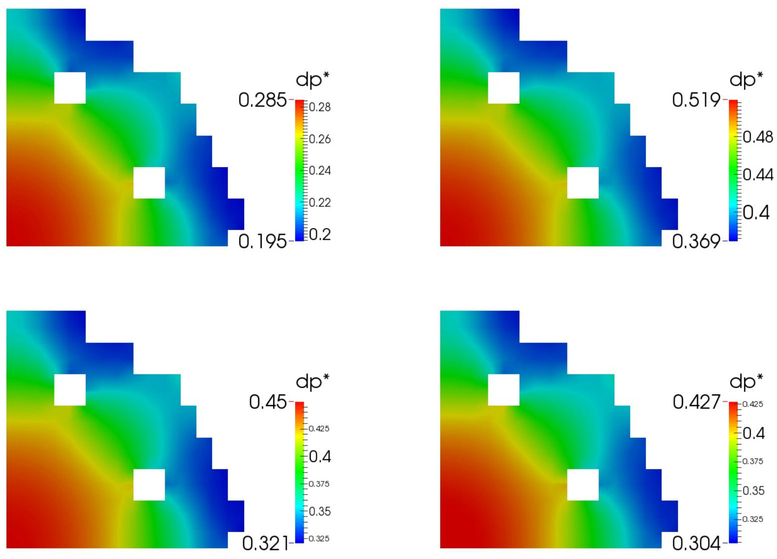


Figure 3.29: Test 1. Pressure profile on section at $z = 3.0m$ (defined by $C1, C2, C3, C4$) for $t = 8$, 10, 16 and 26 s.

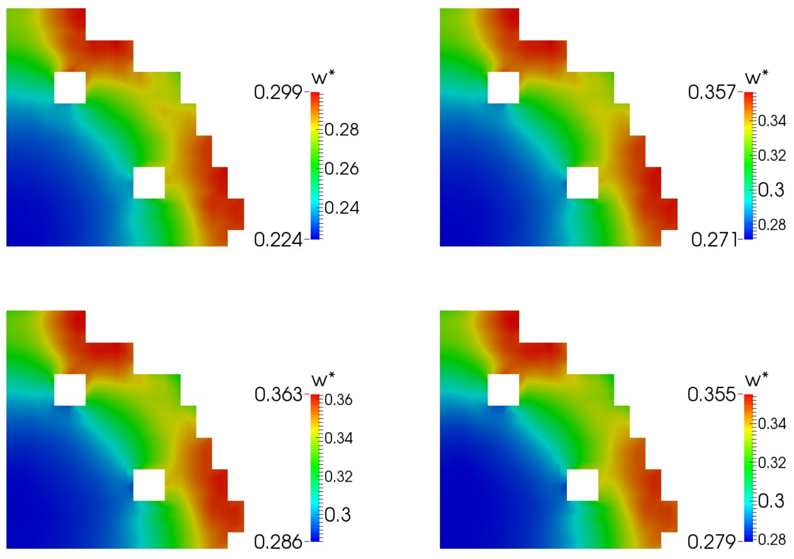


Figure 3.30: Test 1. Velocity profile on section at $z = 3.0m$ (defined by $C1, C2, C3, C4$) for $t = 8$, 10, 16 and 26 s.

one can see the temperature over the plane section at $z = 3.0m$ for $t = 8$, 10, 16 and 26 sec. In time there is a large initial temperature oscillation and then

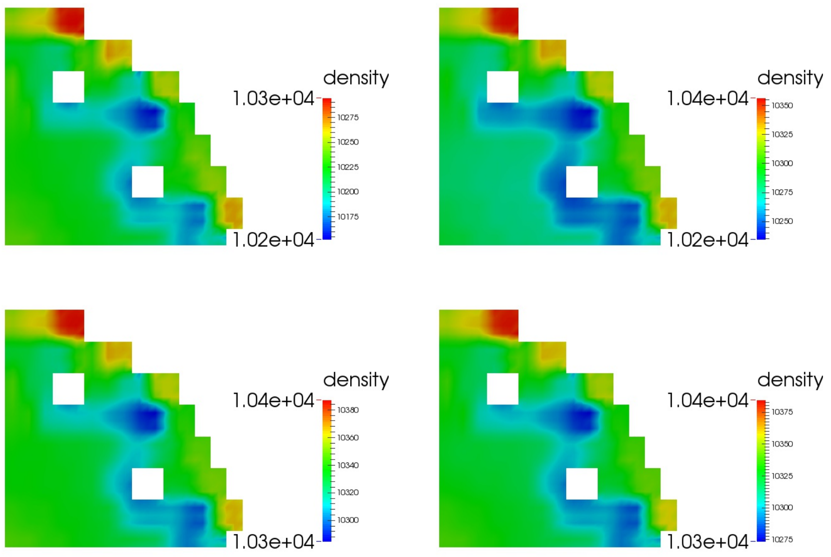


Figure 3.31: Test 1. Density profile on section at $z = 3.0m$ (defined by $C1, C2, C3, C4$) for $t = 8$, 10, 16 and 26 s.

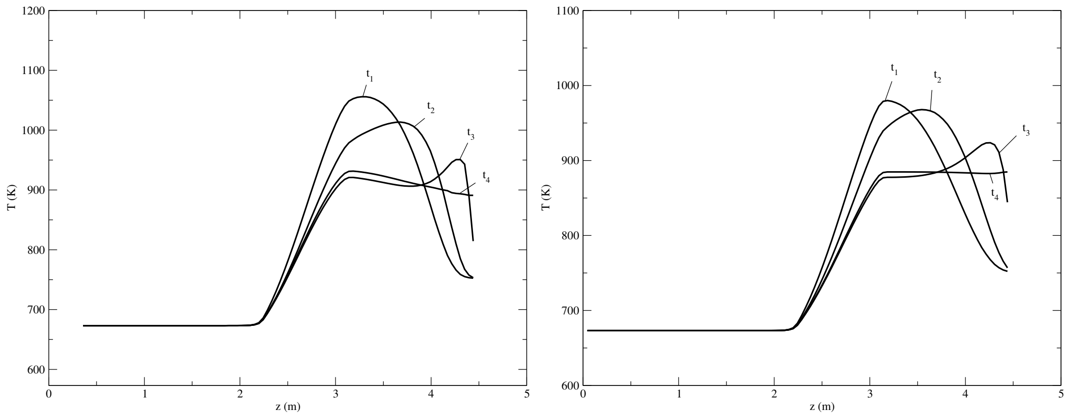


Figure 3.32: Test 1. Temperature profile on a vertical line passing through the point C_3 on the right, and C_1 on the left, for $t = 8$ (t_1), 10 (t_2), 16 (t_3) and 26 (t_4) s.

small fluctuations around an asymptotic distribution. The initial fluctuation leads to very high temperature above $1000K$ after approximately $8s$. In Figure 3.28 it is easy to note that in all these oscillations the distribution of the fuel assembly defines the hot and cold spots of the reactor. At $t = 8$ more than hundred degree difference can be found between different regions. However at $t = 26s$ there is still a ninety degree difference between the hot and cold spots over the section. In Figures 3.29-3.30 the pressure and velocity distributions over the plane section at $z = 3.0m$ are shown for $t = 8$, 10, 16 and $26s$. Large pressure variations imply the unstable nature of the natural convection flow. For each temperature distribution corresponds a distribution of density.The density profile on section at $z = 3.0m$ for $t = 8$, 10, 16 and 26 sec is shown in Figure 3.31. On the right and left part of Figures 3.32 the temperature profile on a vertical line passing through the point C_3 and C_1 are shown, respectively, for $t = 8$ (t_1), 10 (t_2), 16 (t_3) and $26s$ (t_4).

3.6.3 Primary loop evolution in natural convection flow

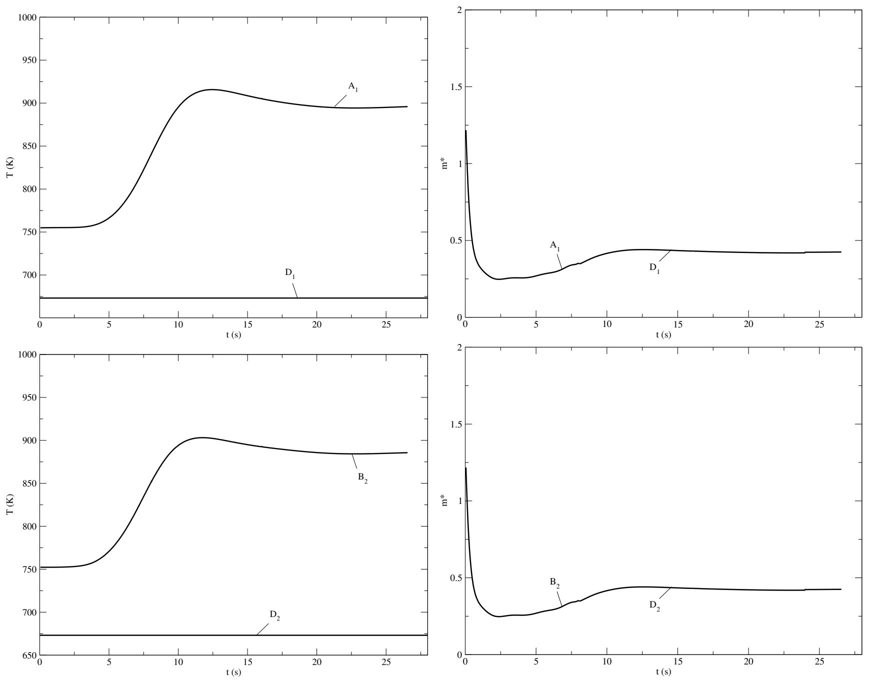


Figure 3.33: Test 1.Temperature (left) and non-dimensional mass $m^* = m/\rho_0 A$ (right) distribution in the primary loop$A$-D on the top and in the primary loop B-D on the bottom, and as a function of time.

In Figures 3.33-3.34 one can see the evolution of the primary loop witch is composed by two branches: A-D and B-D. The reactor is not completely

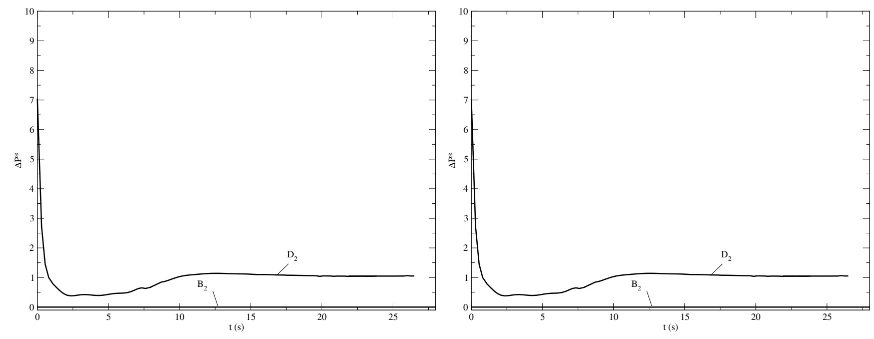


Figure 3.34: Test 1. Pressure distribution in the primary loop A-D (left) and B-D (right) at time $t = 0$.

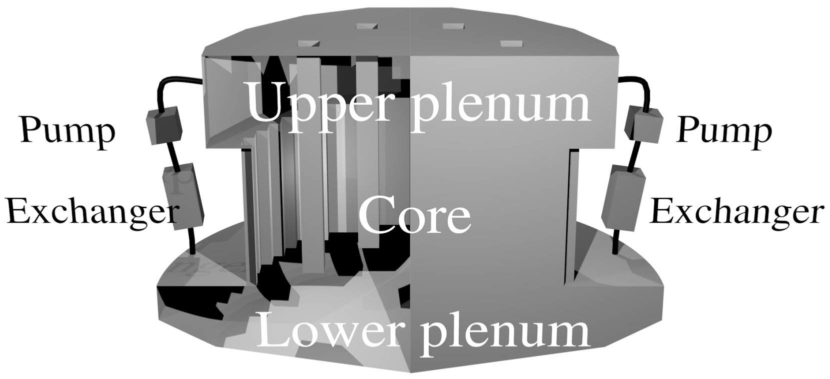
symmetric and there are differences between the loop A-D and B-D. However the differences are very small and can be ignored according with the solutions shown in the top and the bottom part of Figure 3.21. In the top part of Figure 3.33 one can see the inlet and the outlet of the primary loop labeled A_1 (B_2) and D_1 (D_2), respectively. The mass fluxes from all the loops add up to the inlet of the lower plenum. In Figure 3.34 the pressure at the inlet and outlet of the primary loop is shown for the branch A-D on the left and B-D on the right. The pressure from the A-D loop defines the velocity in the left part of the plenum inlet while the pressure of the B-D loop defines the velocity in the right one. If the pressure is the same on both branches then the inlet velocity is uniform. Otherwise greater is the pressure greater is the velocity in the corresponding area. The large oscillation in the pressure and, as a consequence in the velocity field, are imposed into the inlet region of the multidimensional domain, this oscillation leads to a large instability and several non linear iteration are needed in order to ensure the convergence of the resolution. Different coupling scheme can be considered in order to obtain a more robust algorithm.

3.7 Test 2. Defective coupling

In this Section another technique for code coupling, which can be classified as a defective method, is tested. The mono-dimensional system equation is solved taking into account he entire primary loop Γ. The mono-dimensional circuit gives the boundary conditions to the three-dimensional problem and the three-dimensional code corrects the system solutions through the appropriate sources in the momentum and energy equations. The sources, located in the overlapping region Γ_2, are feedback control devices that increase or decrease the source intensity based on the variable state matching at the 1D and 3D interface S_1 and S_2. For the three-dimensional part the FEMUs code is used while the mono-dimensional system is modeled with CATHARE.

3.7.1 General description

In this test we couple a three-dimensional model for a LFR reactor which consists of a core, an upper and a lower plenum with a mono-dimensional primary loop defined through an input deck of the CATHARE code. On the top of Figure 3.35 a general sketch of the reactor and primary loop is shown. Details about the coupling between the mono and three-dimensional domain geometry are illustrated on the bottom. In particular on the left bottom the mono-dimensional mesh is shown. This circuit consists of three pieces: a point volume (UPLENUM), and two *AXIAL* modules (CORE and LOOP). In the mono-dimensional circuit the fluid flows from the *VOLUME* module and moves downward then horizontally and finally vertically to enter again into the *VOLUME* module. On the right bottom the three-dimensional domain, which consists of the reactor plena and core, is drew together with the mono-dimensional one. In this coupled geometry, the fluid flows from the Upper Plenum FEMUs module and moves downward from point P_1 to P_2 through the LOOP CATHARE *AXIAL* module then enters to the Lower Plenum FEMUs module. The three-dimensional region is overlapped to the mono-dimensional one along the horizontal LOOP and the vertical CORE module. The mono-dimensional mesh can be generated by using GUITHARE, the graphical user interface on Salome platform. An overall sketch is shown in Figure 3.36. As said before in the mono-dimensional circuit the fluid exits from the point volume VOL enters the axial DOWNWARD, flows into the UPWARD branch and returns to the volume point VOL. There is only one circuit



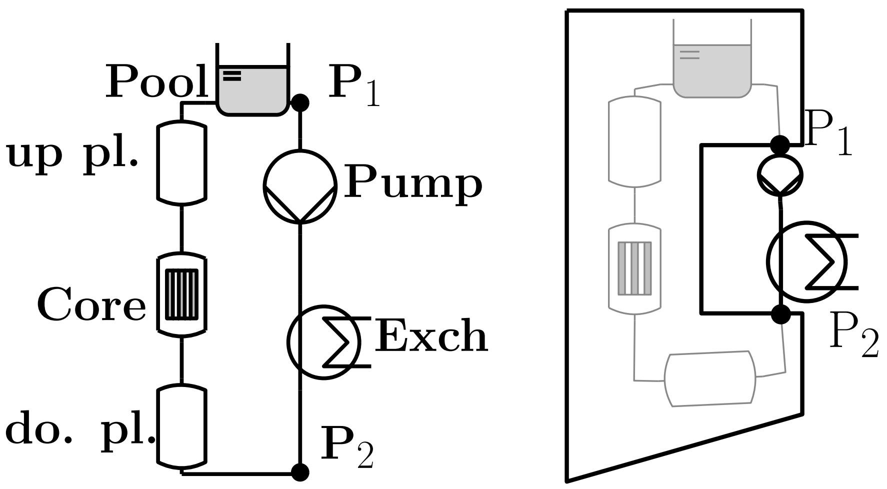
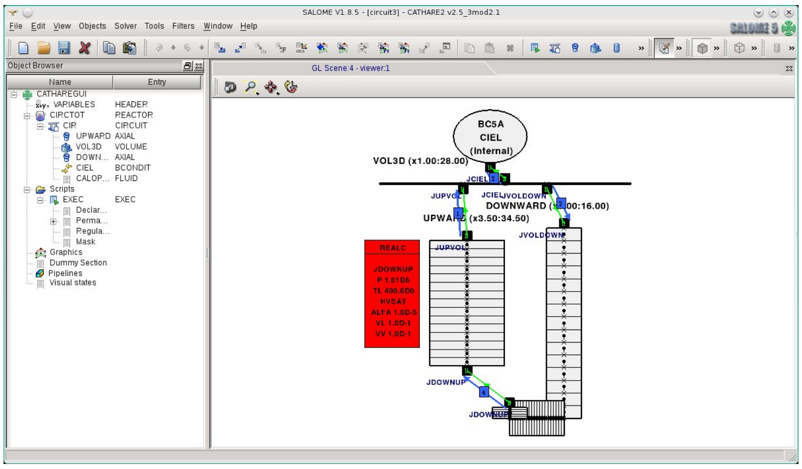


Figure 3.35: Test 2. On the top sketch of the coupling. On the bottom the mono-dimensional circuit (on the right) and the coupling 3D-1D domain geometry (on the left).

which consists of 4 modules: UPWARD, VOL3D, DOWNWARD, CIEL. The UPWARD and DOWNWARD are axial mono-dimensional modules. VOL3D is a 0-dimensional volume module and CIEL is a BCONDIT module which is used to define pressure boundary condition in VOL3D. The key word CALOPORT is needed to identified the fluid. We define at the beginning of the file to have lead as a coolant fluid. VOL3D is a *VOLUME* module (0-dimensional) with 3 junctions JUPVOL (USTREAM), JVOLDOWN (DSTREAM) and JCIEL (DSTREAM). The USTREAM and DSTREAM label indicates the circuit orientation. The geometry of VOL3D is defined by the key. We have two points



Name	Entry
CATHAREGUI	
VARIABLES	HEADER
CIRCTOT	REACTOR
CIR	CIRCUIT
UPWARD	AXIAL
VOL3D	VOLUME
DOWN...	AXIAL
CIEL	BCONDIT
CALOP...	FLUID
Scripts	
EXEC	EXEC
Declar...	
Perma...	
Regula...	
Mask	
Graphics	
Dummy Section	
Pipelines	
Visual states	

Figure 3.36: Test 1. Sketch of the primary loop components.

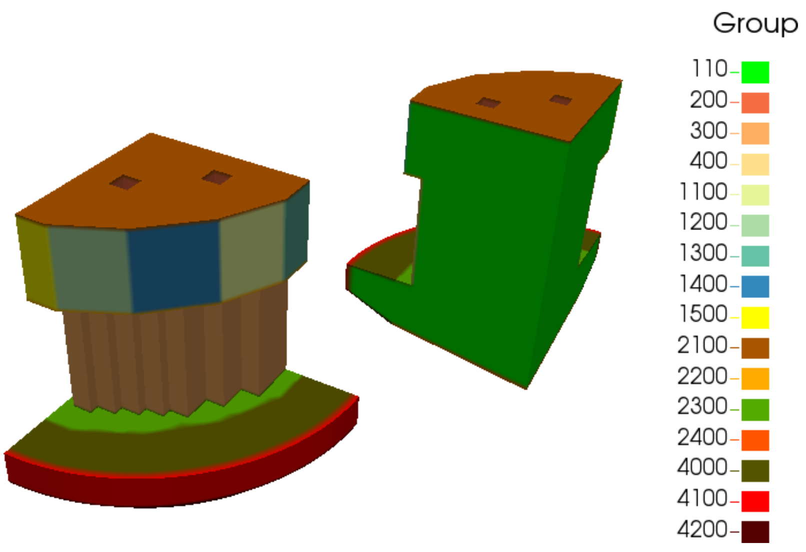


Figure 3.37: Test 2. Geometry and main mesh groups.

at level $z = 0$ and $z = 2m$ with constant section of $2.14m^2$. The junction JCIEL is connect to the *BCONDIT* CIEL module. The boundary condition model for CIEL is BC5A. The BC5A type of boundary conditions is an external outlet where one must specify only the pressure P. The pressure P is defined over three intervals of time $[0, 0.1]$, $[0.1, 100]$ and $[100, 1.E + 11]$. In these three intervals the pressure is assumed to be constant at 1.01×10^5bar by the REALLIST command. DOWNWARD is an *AXIAL* module with two junctions: JVOLDOWN (USTREAM) and JDOWNUP (DSTREAM). This

location	name	component
inlet	4000	Lower Plenum
outlet 1	1200	Upper plenum
outlet 2	110	Upper plenum
top reactor	2100	Upper plenum
bottom	4100	Lower plenum
wall plenum 0	4200	Lower plenum
wall plenum 1	1300	Upper plenum
wall plenum 2	1400	Upper plenum
wall plenum 3	1500	Upper plenum
ctrl rods 1	200	core
ctrl rods 2	300	core
ctrl rods 3	400	core

Table 3.4: Test 2. Table of geometry and mesh groups.

component consists of 4 points: DOW01 ($x = 0$), DOW02 ($x = 7$), DOW03 ($x = 9$) and DOW04 ($x = 9.4$). It stars at the point DOW01 with 3 segments with 15 20 and 5 elements, respectively. The section, perimeter and size of the pipe are define with the key directive GEOM. The UPWARD component is an AXIAL connected between two junctions: JDOWNUP (USTREAM) and JUPVOL (DSTREAM). The USTREAM and DSTREAM label indicates the circuit orientation. The components has two points: UPW01 ($x = 9.4$) and UPW02 =($x = 16$). We remark that we have kept a unique coordinate system along the line for the UPWARD and DOWNWARD branch. There a unique segment with 15 element in the upward vertical ($COS = 1$) direction. The geometry is the same as the other $AXIAL$ one. The pump model, which is located at the coordinate $x = ix\_p$, is very simple. In the pump point we impose a constant pressure with DP= $.75 \times 10^5$Pa is the pump gain in pressure. The CATHARE code is a two-phase code and even if we use only fluid coolant initial and boundary conditions for the other phase should be given. In this case we assume gas with the same temperature and velocity with gas fraction irrelevant (ALFA=1.0D-5). In order to exchange data from the three-dimensional simulation to the mono-dimensional one we need to define interface in the 3D problem, in Table 3.4 a brief list of marked surfaces, and in Figure 3.37 the corresponding between the number and the location can be seen. From Figure 3.37 we can see the location of the groups defined from mesh generator. With the first and second line the groups 1200 and 1100 are associated with outlet of the reactor. We recall that in this reactor we have multiple exits. However on the top surface we need to transfer data for the temperature and pressure fields. The bottom surface is the inlet of the reactor where we need to exchange temperature, velocity and pressure. In order to set the temperature we use a feedback algorithm. Let T_{outr} be the average temperature that flows out from the three-dimensional reactor and t_{21}/h_{21} be the temperature/enthalpy on the other side of the interface on the CATHARE mono-dimensional mesh (point 2). The temperature t_{21} is corrected until it reaches T_{outr}. The pressure coupling is obtained by imposing the three-dimensional average pressure drop $P[0] - P0[0]$ to the mono-dimensional part of the circuit between the point 1 and point 2. In this case the average is obtained simply by weighting the pressure value on the node number on the interfaces. In this way we correct the mono-dimensional drop with the correct one. On the inlet section of the reactor we need to set temperature and velocity field. Since we do not know the field distribution at the reactor we assume a constant distribution. The temperature T_1 from the point 1 of CATHARE should be imposed to the inlet reactor. The boundary conditions for temperature and velocity are imposed directly by using the FEMUs interface functions *write_Boundary_value* directly on the old solution vector *x_old*. The reactor temperature T_{outr} on the outlet is compute as average $0.5(T_{o1} + T_{o2})$ on the surface 110 and 110 The average value $0.5(P_{o1} + P_{o2})$ can be storage as reference top pressure $P[0]$ for the tree-dimensional reactor. At the bottom we have a single face and therefore the three-dimensional reference pressure at inlet is $P0[0] = P_{in}$.

3.7.2 Initial conditions

In order to compute the solution at nominal condition we start the computation from uniform initial velocity condition $v = 0m/s$ and temperature $T = 400C$.

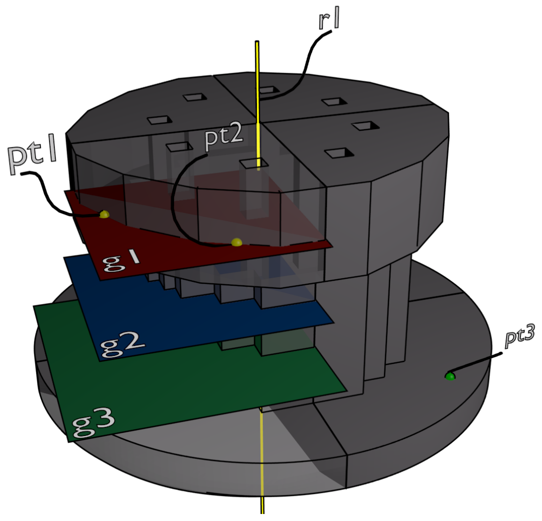


Figure 3.38: Test 2. Global domain overview and reference elements.

We apply an inlet velocity of $0.1m/s$ and the value of the pump pressure $DP = 0.75 \times 10^5 Pa$. The pressure from the pump increases the velocity until this matches the total pressure losses of the circuit with velocity and temperature transient. The transient is rather long and CPU consuming. Even if CATHARE cannot be run in parallel in this configuration the FEMUs problem can be run in a multiprocessor cluster. In this way each processor runs a CATHARE copy. This is not very efficient but the CATHARE solving time can be considered almost negligible in comparison with the three-dimensional code solver. However FEMUs and CATHARE codes can run independently with a different number of time steps. If one is interested only the steady state solution in working condition the code CATHARE can be run until the velocity has reached the steady state saving a lot of the computational time. In next parts we report the results for the three and mono-dimensional domain when the steady state is reached. We also report the evolution of the key points, shown in Figure 3.38, that define the interface between the different codes. The evolution of the solution in these points let us investigate the numerical behavior of the feedback control at the single point and of the defective coupling approach described in the previous Section.

3.7.3 Reactor evolution

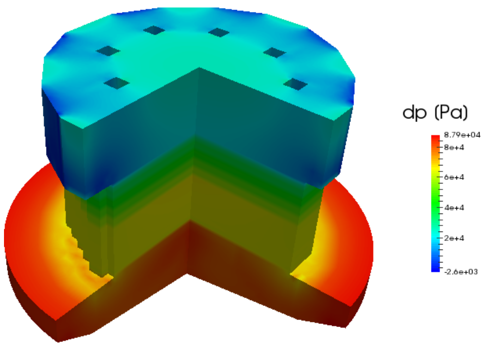


Figure 3.39: Test 2. FEMUs partial pressure field in the three-dimensional domain.

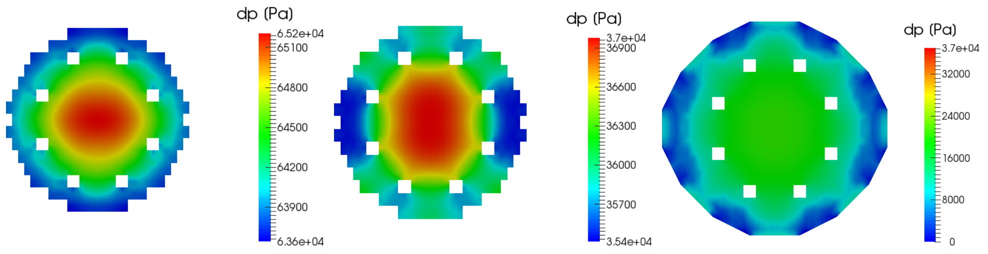


Figure 3.40: Test 2. FEMUs partial pressure field over different planes for $z = 2$, 3 and 3.4.

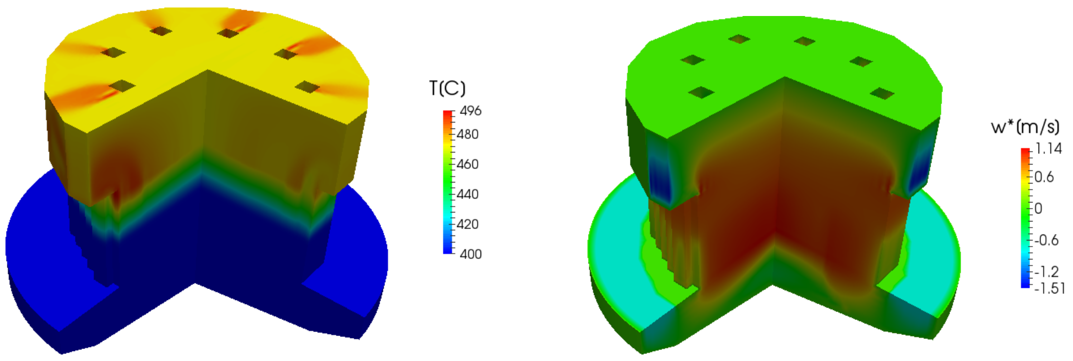


Figure 3.41: Test 2. FEMUs temperature field, on the left and normalized velocity field w^*, on the right, in the three-dimensional domain.

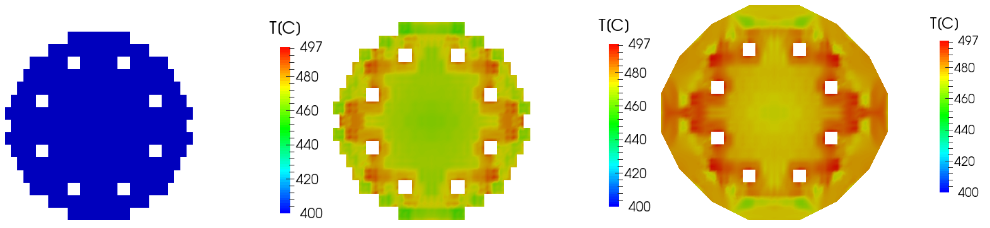


Figure 3.42: Test 2. FEMUs temperature field at $z = 2$, 3 and $3.4m$.

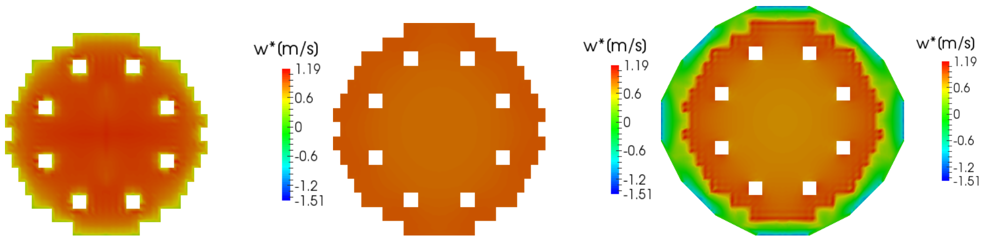


Figure 3.43: Test 2. FEMUs normalized velocity field w^* at $z = 2$, 3 and 3.4.

The three-dimensional solutions are obtained by solving the FEMUs modules. The boundary conditions are enforced by the CATHARE mono-dimensional code at the inlet of the reactor. The real distribution of the velocity and temperature at the inlet is a two-dimensional profile that cannot be known. Only the average value of temperature and velocity fields are defined and therefore we impose a constant field. The pressure field (p) is reported in Figures 3.39-3.40. p in a Navier-Stokes system can be always written as

$$p = -\rho g(z - z_0) + p' \tag{3.33}$$

by including the potential of the gravity term inside the pressure gradient. ρ is the density, g the gravity acceleration, z the vertical coordinate and z_0 the reference pressure point. The quantity $p' = dp$ is called partial pressure and it is defined as the difference between the total pressure and the pressure gravity term. Over the 3D/1D coupling interface the definition of the location of z_0, where the reference pressure should be imposed, is not a trivial problem. The reactor takes the pressure imposed from the CATHARE code at the outlet of the *AXIAL* CORE module. Since the outlet of the reactor is over different values of the coordinate zeta it is difficult to impose constant pressure over all the outlet surfaces. The decomposition in (3.33) solves this problem fixing the pressure at a well defined height. On the Figure 3.39 the partial FEMUs pressure field $dp = p'$ in the three-dimensional domain is shown. In Figure 3.40 details on the same pressure field over different planes for $z = 2$, 3 and 3.4 are reported. On the left part of Figure 3.41 the FEMUs temperature field T is shown over the whole three-dimensional domain. In Figure 3.42 one can see details on the same temperature field over horizontal sections for $z = 2$, 3 and 3.4 the planes shown in Figure 3.38. Inside the three-dimensional reactor the temperature increases and its pattern over these sections is clearly determined by the fuel distribution ($z = 2$). The velocity field inside the reactor is not continuous since we have the plenum and the core that satisfy different form of the Navier-Stokes equations. In the core the assembly contains fluid and solid material. The geometric dimensions are real, comprehensive of the solid material, while in the simulation we allow the fluid to flow to occupy the whole assembly section. For this reason in order to maintain the same mass flow in the core and in the plenum, over different flowing area the velocity field should be discontinuous. In order to have a continuous variable to solve we consider the normalized velocity field $\mathbf{v}^* = r\mathbf{v}$ where r is the assembly occupancy factor. Substantially we rewrite the Navier-Stokes equation with the mass flow variable. On the right part of Figure 3.41 the vertical component w^* of the FEMUs normalized velocity field v^* is shown over the whole three-dimensional domain. In Figure 3.43 one can see details on the same w^* field over horizontal sections for $z = 2$, 3 and 3.4.

3.7.4 Primary loop evolution

We show the results for the mono-dimensional mesh and for the whole circuit. In Figure 3.44 the key points of the mono-dimensional mesh are shown. In red the UPLENUM CATHARE *VOLUME* module and in blue the LOOP *AXIAL* module. The CORE *AXIAL* module is in black. In Figure 3.45 the pressure along the line $r1$ of Figure 3.38, is shown on the left. In this Figure we report the partial pressure p', namely the pressure without the gravity contribution. The gravity term is dominant and, if shown, it covers the accidental pressure losses. We can see two large jumps in pressure due to the inlet and outlet

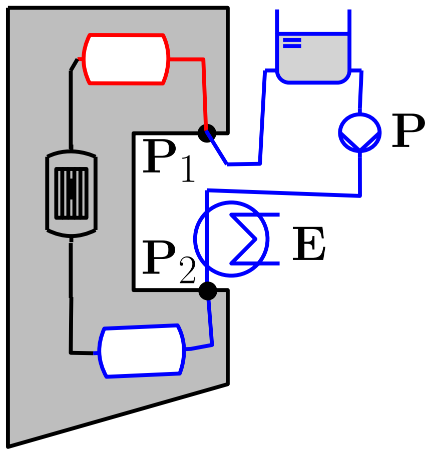


Figure 3.44: Test 2. Reference points in the mono-dimensional mesh.

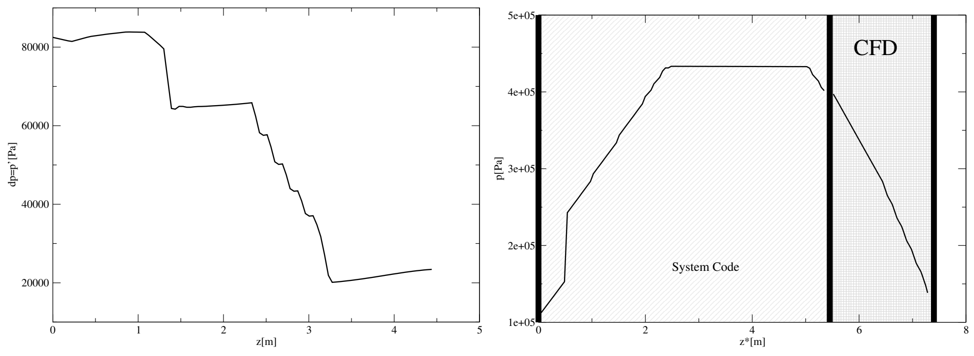


Figure 3.45: Test 2. Total pressure p^* along the central line of the three-dimensional domain (left) and along the plenum-core-plenum-primary loop (right).

grids. In particular the inlet grid has also the duty to steer the flow in the vertical direction. It is possible also to see the pressure losses of the four internal grids. On the right the total pressure in the circuits is shown. The mono-dimensional coordinate starts at the point P_1, defined in Figure 3.44, enters the reactor lower plenum at $x = 1.95m$ and levels horizontally from $2.35m$ and $5m$. At $x = 5.4m$ enters the core and reaches the upper plenum at $7.35m$. Between $x = 1.95m$ and $7.35m$ this mesh overlaps with the three-dimensional mesh. On the right of Figure 3.45 the initial pressure at $x = 0m$ is imposed by the BCONDIT CATHARE module and increases linearly due to the gravity term. The pressure has a jump, defined by the pump, and then

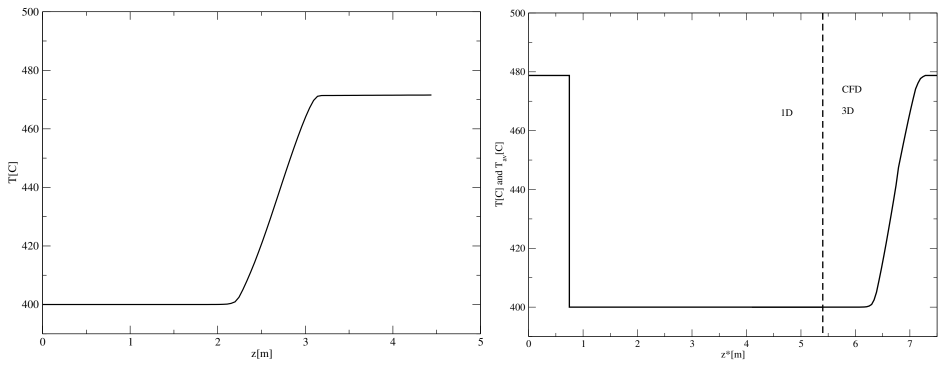


Figure 3.46: Test 2. Temperature along the central line of the three-dimensional domain (left) and average temperature along the plenum-core-plenum-primary loop (right).

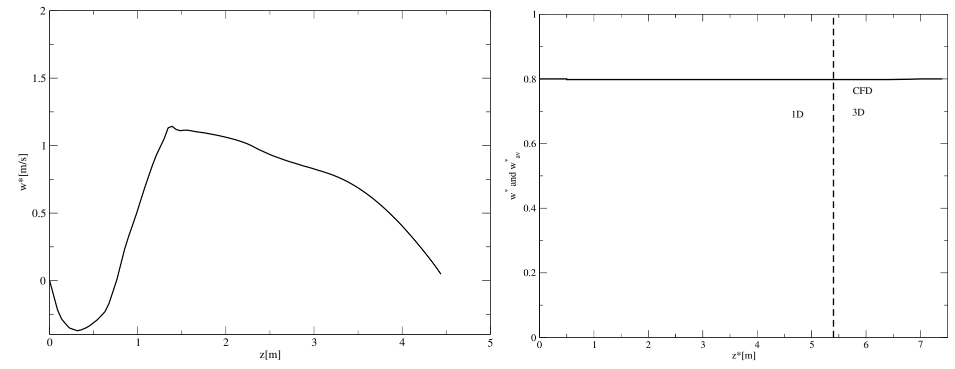


Figure 3.47: Test 2. Normalized velocity along the central line of the three-dimensional domain (left) and average normalized velocity along the plenum-core-plenum-primary loop (right).

it starts to increase linearly again. At the bottom the pressure line is almost horizontal since only the distributed pressure losses are taken into account. In the last part the circuit reaches the core where the mono-dimensional pressure behavior is substituted with the three-dimensional one. On the left of Figure 3.46 the temperature along the central line of the three-dimensional domain is shown. On the right the average temperature along the circuit is reported. The average temperature at exit of the the reactor is about $478C$. The fluid flow goes through the pump and the heat exchanger. In the exchanger the temperature drops to 400C. The temperature remains the same until it reaches the core. In the core we plot the average temperature of the three-dimensional simulation. Finally the normalized average velocity is shown in Figure 3.47. On the left the normalized vertical velocity for a case of open assembly is reported. On the left the normalized velocity on the circuit and its average on the three-dimensional domain are shown.

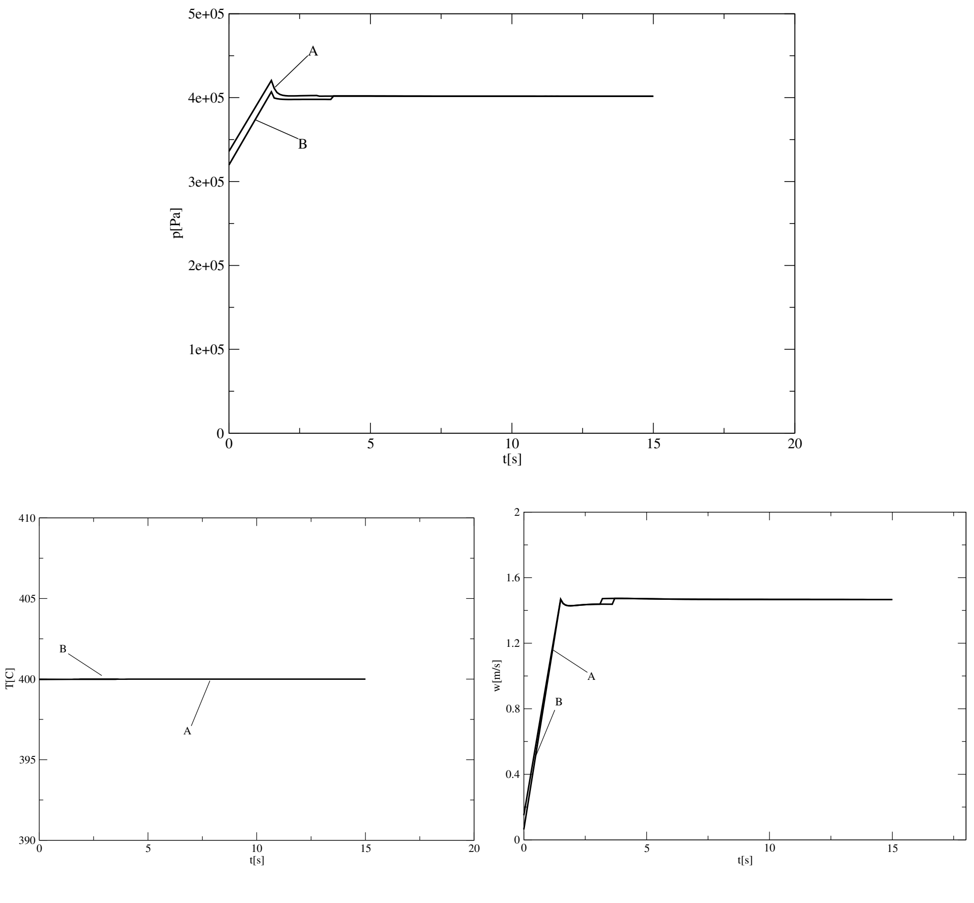


Figure 3.48: Test 2. Evolution of the state variables at the interface 1D/3D P_2: circuit (A) and three-dimensional (B) solution values.

Solution at the coupling interfaces. In Figures 3.48- 3.49 pressure, temperature and the velocity field at the 1D and 3D interfaces are shown as a function of time when no internal iterative algorithm is used. Since we use a defective coupling approach every state variable is matched by using sources in the mass, momentum and energy equation. These sources are in form of

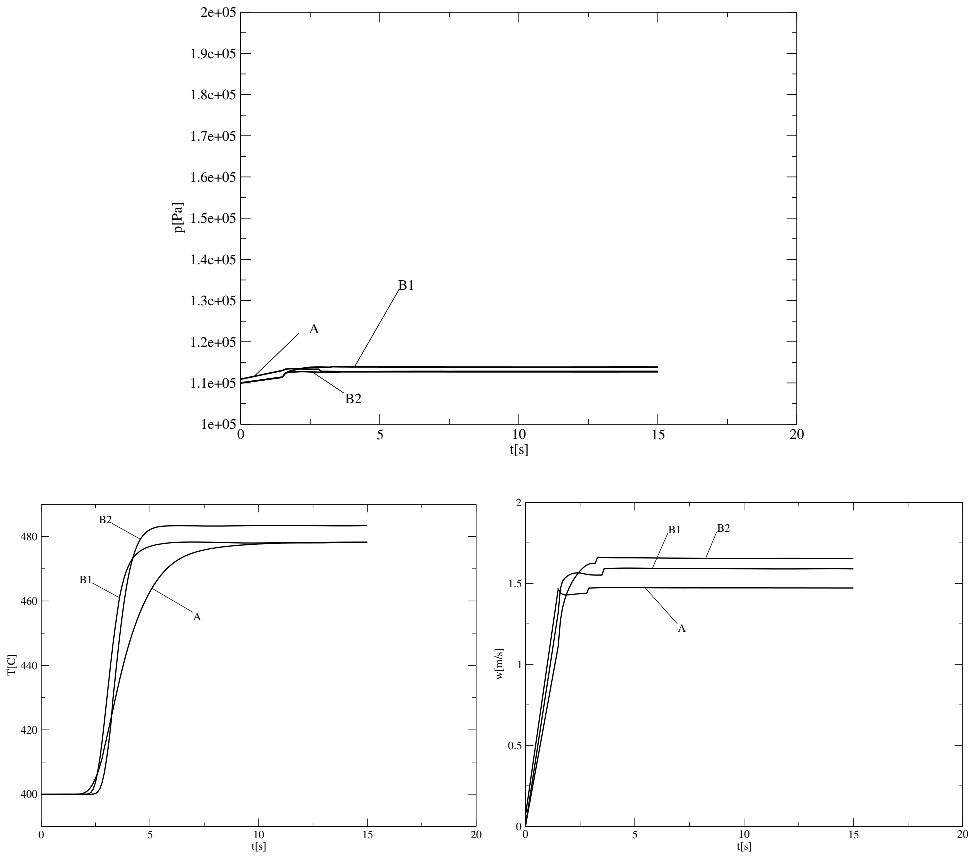


Figure 3.49: Test 2. Evolution of the state variables at the interface 1D/3D P_1: circuit (A) and three-dimensional (B) values.

feedback terms, namely

$$S_i^{k+1} = S_i^k + w_i(\phi_{3D} - phi_{1D}) \,, \tag{3.34}$$

where k indicates the k-step of the iterative algorithm. The index i indicates the equation where the source is applied ($i = 0$ mass, $i = 1$ momentum, $i = 2$ energy). The under-relaxation parameter w_i can be tuned to get fast convergence. The state variable from the mono-dimensional solution is indicated as ϕ_{1D} and for the three-dimensional domain as ϕ_{3D}. For the k that tends to infinity S_i^k tends to S_i that implies $\phi_{3D} = phi_{1D}$. However if a steady solution is considered the number of iterations for this feed back control algorithm can be set to one. In this case we say that no internal iterative algorithm is used. In Figure 3.49 the evolution of the state variables at the interface 1D/3D P_2

are shown. On the top we have the pressure and on the bottom the average temperature and velocity. The interface P_2 is the inlet of the reactor. Over this interface there is no need of the feedback algorithm on temperature and mass flow since they are simply boundary conditions imposed from the circuit to the three-dimensional simulation. During the evolution, the three-dimensional values follow the transient determined by the pump. However the pressure is determined with the feedback control algorithm. As one can see the matching is almost perfect. The difference is due to the fact that the transfer of data in these two codes is passed with one step delay. In Figure 3.48 the evolution of the state variables at the interface 1D/3D P_2 is shown. The state variables are pressure on the top and average temperature and velocity on the bottom. The interface P_2 is the outlet of the reactor. The reactor has multiple exits which are the points $pt1 - pt2$ of Figure 3.38, where the state variables takes no constant values. The value imposed to the mono-dimensional simulation is imposed by using sources located in the CORE $AXIAL$ module that overlaps the three-dimensional domain. The pressure is set quickly at the beginning of the evolution. The two exits B_1 and B_2 and the mono-dimensional inlet keep the fixed pressure dictated by the BCONDIT module. At $t = 0$ the temperature is $T = 400C$. This point starts to feel the increasing in temperature $t \approx 3$s when the reactor warm up. The average temperature increases till reach its steady value. Two points in two different exits B_1 and B_2 are shown together with the average temperature imposed to the circuit. The same happens for the velocity. The pump pressure defines the circuit flow rate which increases till it reaches the steady state at the mean velocity of approximately 1.48m/s. It is worth while to note how smooth is the numerical convergence at the interface when the coupling defective approach is used. If compared with the simple coupling boundary approach used in previous reports we can say that this approach is more computational efficient.

3.8 Test 3. Neutron CFD coupling

In this last example, we focus our attention on the energy production into the core region. In all previous tests, we investigate the dynamics of the entire primary loop of a LFR reactor, under the assumption that the power distribution in the reactor remain constant. In this condition the average heat rate production and its peak factor is an input for the problem and cannot change during the simulation. With this approach the effect of the temperature field on the cross section of the nuclear fuel is neglected. In this third test we investigate the thermal-hydraulics behavior of a PWR nuclear reactor evaluating the power generation distribution taking into account the local temperature field. The neutron code used in this example is DONJON-DRAGON, an open source framework developed by the Institute of Nuclear Engineering of the École Polytechnique de Montréal that as been integrated into the numerical platform SALOME. The software is composed by two modules: DRAGON and DONJON. The DRAGON code [21] results from a concerted effort to unify inside a single code various numerical techniques and calculation methods that can be used to solve the neutron transport equation (3.2) both with the collision probability technique and the method of characteristics. The lattice code DRAGON is divided into modules which transfer information through well defined and documented data structures. The geometry module represents a versatile geometry analysis technique available in DRAGON, it can process 2D clusters geometry as well as Cartesian and hexagonal lattices of cells in both two and three dimensions. The second code (DONJON) contains a collection of modules that solve the neutron multigroup diffusion equation on realistic complex reactor core modeling. It includes all the functions that characterize a reactor core. DRAGON performs transport calculations using a microscopic library based on different nuclear libraries and then, the diffusion code DONJON, is used to compute fluxes and multiplication factor. The computer codes DRAGON and DONJON are set up in a modular form which allows the user to break up his calculation in procedures having a smaller number of steps. Relevant data can then be easily exchanged from one process to the other through hierarchical data files and/or its sequential export facilities. The standard calculation procedure we carry is in two steps. In the first one we perform critical transport calculations on the transport model and generate a consistent set of multigroup properties such as the various cross sections and diffusion coefficients for each different material. In the second step we introduce these nuclear properties in the DONJON full core model and compute the macroscopic flux distribution and the multiplication factor of the core. The DRAGON generated macroscopic properties are stored in COMPO files. This type of file has been developed to unify output storage and to be able to keep macroscopic as well as microscopic cross sections with a variable number of energy groups and eventually for different steps of evolution. The COMPO files will be directly accessed in the DONJON computation to ensure adequate communications between transport and diffusion calculations. One of the most important properties needed for simulating the operating reactor is its temperature reactivity coefficients and DONJON calculations can determine the temperature reactivity coefficients of the different components, in temperature ranges. The dominant effect is due to the coolant water which has an important negative reactivity. The effects of fuel temperature are not negligible because of the Doppler reactivity caused by large presence of $^{2}38$U and the low conductivity of the ceramic fuel. The execution of DONJON is controlled by the generalized GAN driver [22] which it is modular and has been integrated into the SALOME platform so that it can be easy controlled by a supervisor so that the actual temperature field in the core region, evaluated using the same FEMUs module used in the previous examples, is exchanged, through a MEDmem interface, to the DONJON code. The material macroscopic cross sections are then updated according the new temperature field using the COMPO object previously generated with the DRAGON and, the consequent neutron flux, is computed. From the updated neutron flux the new peak factor is evaluated and projected, again through the MEDmem interface, into the FEMUs problem that evaluate the new temperature field.

3.8.1 General Description

The geometry of the reactor used in this example, is show in Figure 3.50, we consider an annular reactor characterized by a diameter (d) of $3.23m$ and an height (a) of $6.6m$. The active part of the core start form $c = 2.5m$ and ends at $b = 4.5m$. The coolant (water) flows from the lower inlet region to the outlet on the top. The models and the algorithms that are used for solving

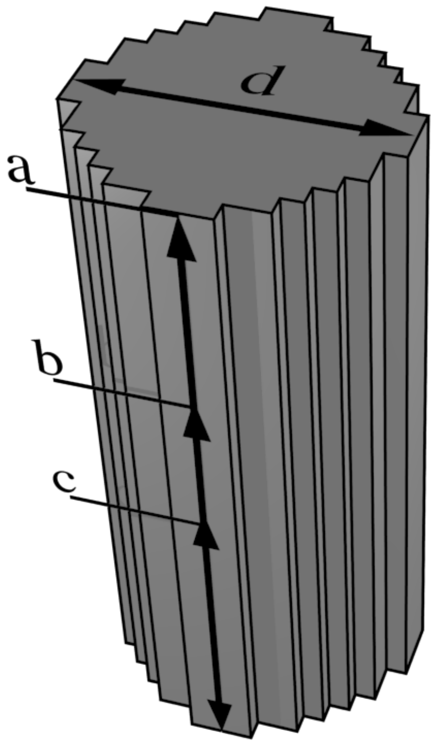


Figure 3.50: Test 3. Geometry of the reactor.

the mass, momentum and energy balance equations from the 3D-CFD scale point of view, are similar to the ones described in Sections 3.2-3.3 they only differs for some geometrical aspect, for this reason in this Section we focus our attention in the description of the active zone of the reactor explaining how it has been considered into the neutron code. The first step in the execution of the neutron code is the generation of the macroscopic cross section library needed for solving the multigroup diffusion equation, this operation is performed by the code DRAGON which solve the transport equation in a single pin fuel element taking into account its particular geometry and composition. The input parameter for this specific case are reported in Table 3.8.1. The transport equation is integrated over a lot of tiny energy interval and the results are then condensed into the energy groups considered into the multigroup diffusion diffusion. The macroscopic cross section are evaluated for different fuel compositions and for different average temperatures according to a range defined by the user. This operation allows the interpolation of the macro-

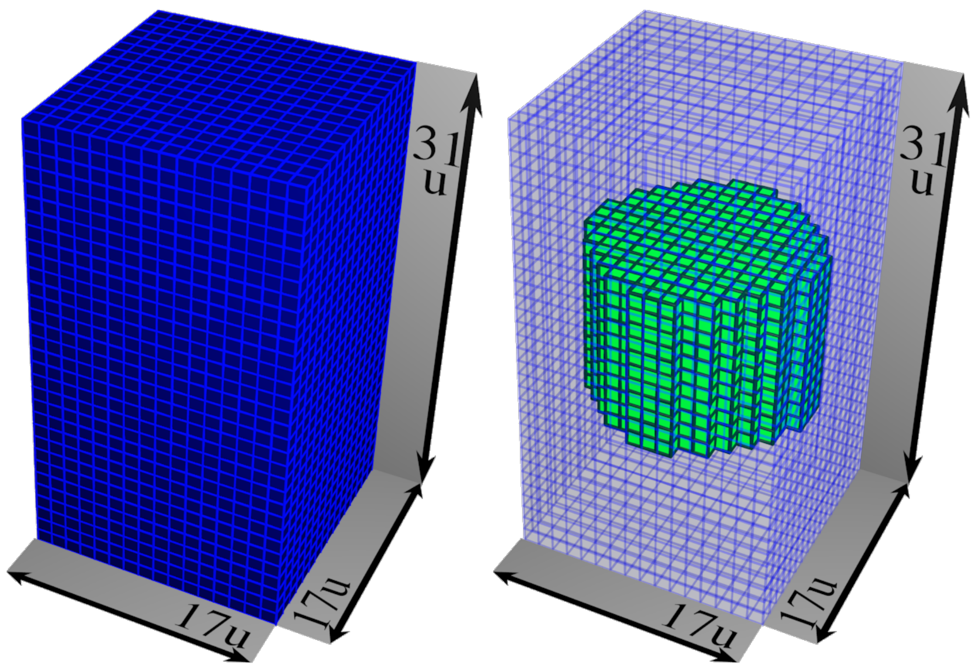


Figure 3.51: Test 3. DONJON computational grid on the left and active zone of reactor core on the right.

r_4	$0.41cm$		
r_1	$\sqrt{0.5}r_4$	N_{UOX}^{U5}	0.037%
r_2	$\sqrt{0.8}r_4$	N_{MOX}^{U5}	0.0025%
r_3	$\sqrt{0.95}r_4$	$\%Pu\ r_1$	0.075%
R_{iTG}	$0.56cm$	$\%Pu\ r_2$	0.049%
R_{eTG}	$0.62cm$	$\%Pu\ r_3$	0.028%
R_{iG}	$0.418cm$	T_f	$800.0K$
R_{eG}	$0.48cm$	T_m	$600.0K$
lp	$1.26cm$	ρ_m	$0.659g/cm^3$
ws	$0.04cm$		

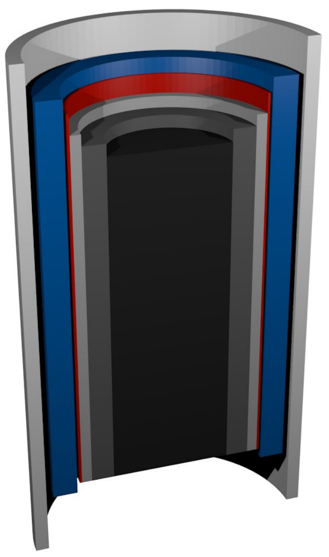


Figure 3.52: Test 3. Fuel pin geometry on the right and composition parameters on the left.

scopic cross section between the range values that has been set. We remark that this first step, the generation of macroscopic cross section library, depends

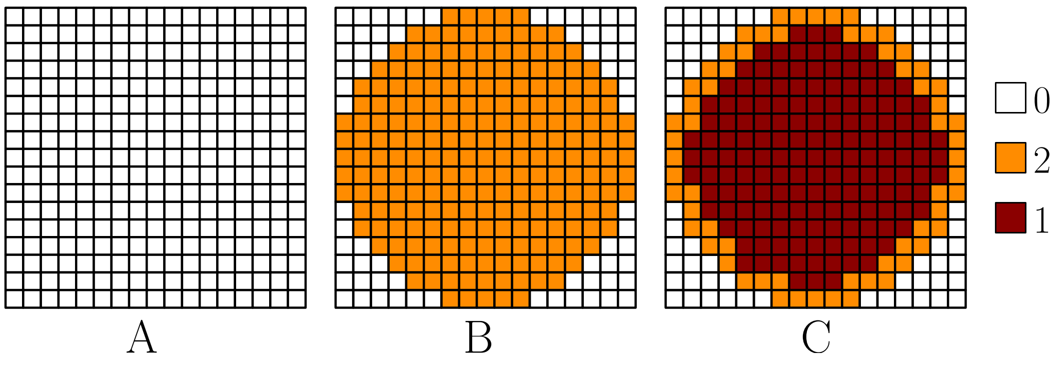


Figure 3.53: Test 3. Three possible mix (A, B and C) of materials 1, 2 and 3 in a reactor plane.

only on the fuel pin composition and geometry, the possibility to evaluate the cross section in a range of composition and temperature is meant for avoiding the generation of this library every time iteration during a simulation. DONJON, which is the program the solves the multigroup diffusion problem, is the software that gets the new temperature field coming from the CFD module, update the cross section according with the local temperature and evaluate the new neutron flux and so the energy production in the active core region. Since the DONJON code can handle only Cartesian structured mesh we have to embed the computational domain shown in Figure 3.50 into a Cartesian three-dimensional grid specified by a characteristic length u. In our case we choose $u = 0.215m$ and the computational domain is show in left part of Figure 3.51. In the right part of the same Figure we mark the active core region in the domain, in this zone the neutron flux and the peak factor are evaluated considering the particular composition of every fuel element of the computational grid. In particular the domain is composed by 31 sections made of 17×17 cells and the composition of each cell is specified in a 17×17 matrix. In this example we consider three material types: 0 is a dummy material that mark a zone in which the neutron flux is zero and must not be evaluated, 1 is the fuel material which has the composition shown in the left part of Figure 3.8.1 and finally 2 is the reflector material. In the right part of Figure 3.8.1 a sketch of the fuel pin element is reported, from that Figure one can appreciate the geometrical complexity that can be considered in the evaluation of the macroscopic cross sections. The firsts and the lasts 10 planes of the neutron computational grid, has the A configuration show in Figure 3.53, the 11^{th} and 22^{th} planes has the B configuration show in Figure 3.53 and finally the remaining planes, the active core, has the C configuration shown in Figure 3.53. Now that we have specified the composition of all the elements of the computational grid we can solve the two groups diffusion equation evaluating the neutron flux inside the domain. Concerning the CFD module geometry, the computational domain for this problem is shown on Figure 3.50. A porous three dimension module is used in the active zone of the reactor ($c < z < b$) as described in Section 3.3. The coolant flows inside the domain from the inlet surface on the bottom part of the geometry with a fixed axial velocity of $1m/s$, and exit from the top outlet surface where an homogeneous Neumann condition is imposed. In the lateral surfaces the flow is allowed only in the axial dimension. Regarding the energy balance problem the refrigerant enter into the domain with a fixed temperature of $500C$ while in all the remaining zones an homogeneous Neumann condition is imposed. At the initial time step the temperature and the axial velocity field are set in the whole domain to a value of $500C$ and $1m/s$, respectively.

3.8.2 Interface

As in the previous cases the interface between the neutron and CFD module is created by the construction of a duplicated mesh of the two computational domains in the MEDmem format and creating a map between them. In this example the two meshes have different geometry as we can see in Figure 3.54. In particular in the left part of that Figure the two computational domains are shown, the bounding box mark the Cartesian structured mesh mesh and the intern grid is the CFD mesh. In the right part of Figure 3.54 we can see the overlapping between the active core zone in the Cartesian mesh and the CFD domain. The map between the two MEDmem meshes couple all the elements of the CFD computational grid in the nearest cell into the Cartesian grid. After a *solve* iteration of the CFD problem the temperature field is taken from the problem ad projected, through the MEDmem meshes duplicates and the

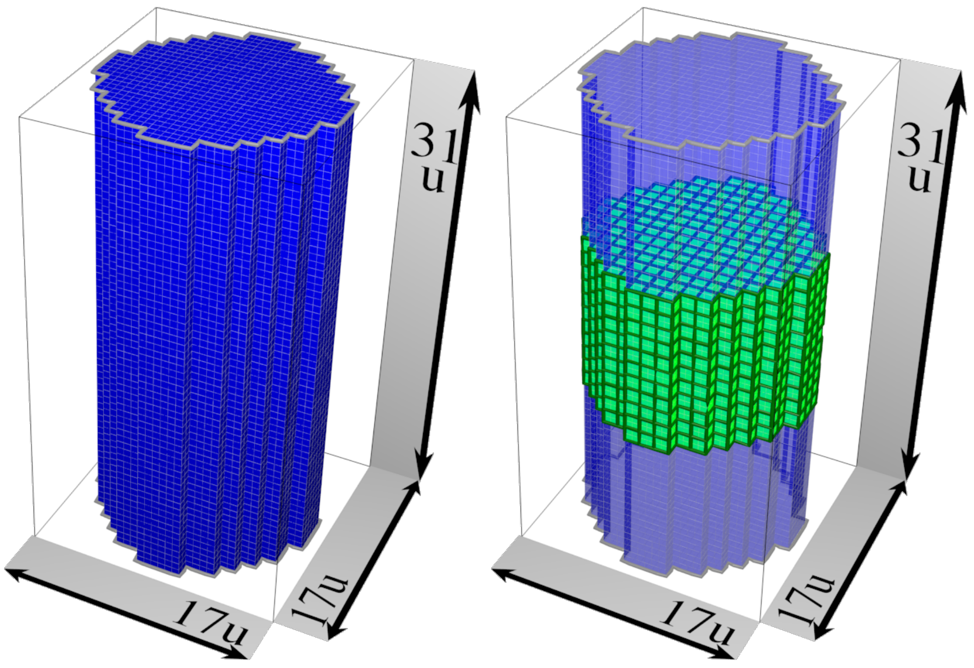


Figure 3.54: Test 3. FEMUs (solid color) and DONJON (outline box) meshes. Whole meshes on the left and heated zone on the right.

map operator, to the neutron code. The neutron flux is then calculated by DONJON updating the cross section according according to the new temperature field. From the updated neutron flux the new peak factor is evaluated and projected back to the CFD code which can evaluate the new temperature field.

3.8.3 Core evolution

The temperature field in the initial condition is set to a constant value of $500C$ while the axial velocity field to a constant value of $1m/s$. The velocity field remains equal to the initial field during the whole simulation because of the boundary condition and the geometry of the problem. For these reasons is not reported. Due to the energy production in the active zone of the core, the temperature increases from the initial value until the steady state is reached. The temperature field at different time steps t_1, t_2 and t_3 (0.1, 2.5 and $10s$, respectively) over the 3D domain is shown in Figure 3.55. We can notice that

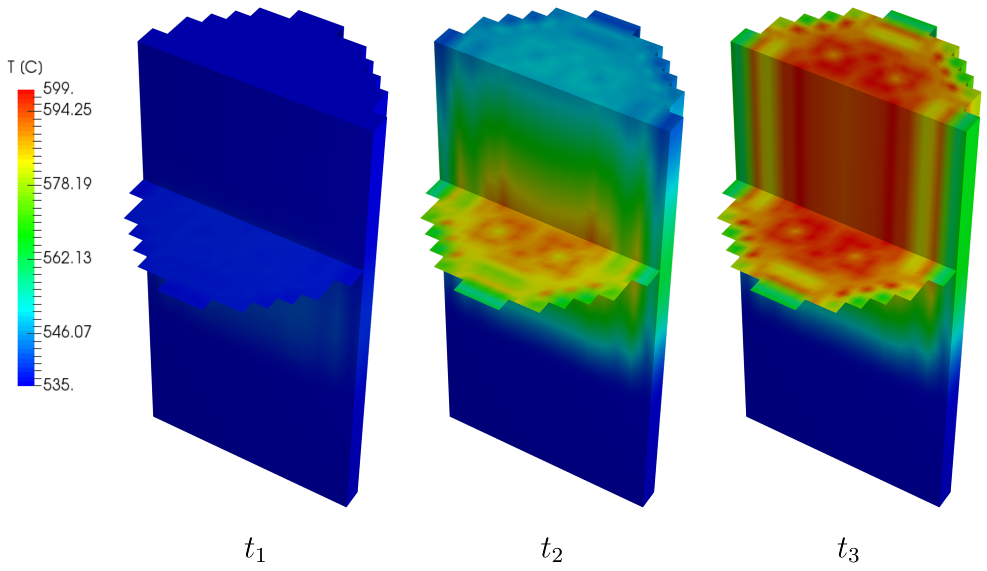


Figure 3.55: Test 3. Temperature overview at different time steps t_1, t_2 and t_3.

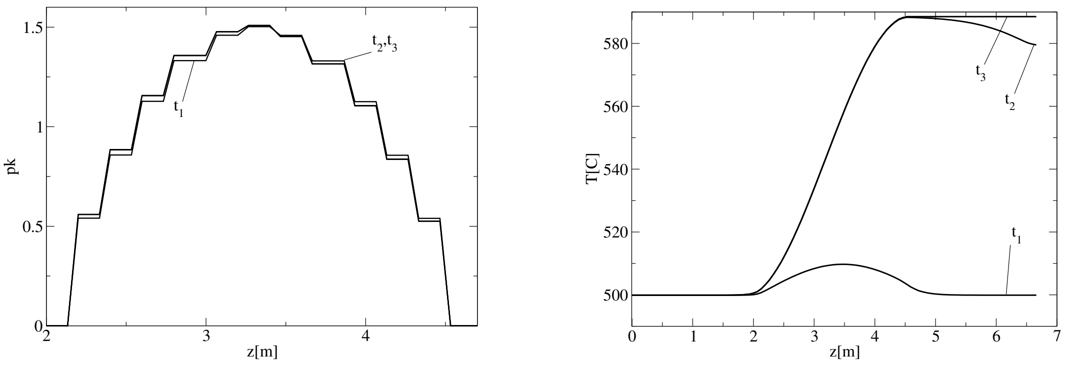


Figure 3.56: Test 3. Neutron flux and temperature profile over the center line of the domain at different time steps.

the temperature field increases in the active zone of the reactor in the initial time steps (t_1 and t_2) and then, because of the velocity field, all the upper zone is heated until the steady state is reached (t_3). In Figure 3.56 the peak factor (on the left) and the temperature field (on the right) over the center line of the domain at different time steps (t_1, t_2 and t_3) is shown. We can notice that as the temperature field increase the peak factor i changes because of the different cross section. This effect cannot be considered without the usage of the neutron code.

CHAPTER 4

Fluid structure Interaction Problems

In this Chapter after an introduction, we present the mathematical formulation of the fluid structure problem in a multiscale framework and then the algorithm is used to investigate the behavior of different components in the cardiovascular system. The Fluid Structure Interaction (FSI) module has been developed in house and it has been validated against the most common FSI benchmarks for large deformation problems [23] such as the one proposed by Turek [24] and the ones proposed by Bathe [25]. The system code is also an in house program and, as in the previous Chapter, the coupling of the modules is achieved with the computational platform SALOME following the procedure explained in the Chapter 2. In particular using the MEDmem libraries the velocity field is projected into a simplified model of the circulatory system which sets the inlet velocity for the multidimensional system. The software that are used are in-house codes and have been integrated into the SALOME platform following the procedure explained in the Chapter 2. This Chapter is organized as follows: After the mathematical description of both the multidimensional and the mono-dimensional FSI problems the MED interfaces needed for coupling the two codes are described. Since the solution of a monolithic FSI problem is highly CPU-time expensive a decoupling algorithm is used and is largely explained in Section 4.3. Four tests cases are then reported: in the first the decoupling algorithm is validated with a benchmark case in which a pipe like system is considered, in the second test a similar FSI problem is coupled, through the MEDmem interface, with the simplified mono-dimensional circulatory system code. Finally an aneurysm in a blood vessel is considered and the dynamics of the system is investigated in the case of a stationary and transient aneurysm in Test 3 and 4, respectively.

4.1 Introduction

Fluid Structure Interaction (FSI) is a class of problems with mutual dependence between the fluid and structural mechanics parts. The flow behavior depends on the shape of the structure and its motion, and the motion and deformation of the structure depend on the fluid mechanics forces acting on the structure. We see FSI almost everywhere in engineering, sciences, and medicine, and also in our daily lives. The FSI effects become more significant and noticeable when the dependence between the influence and response becomes stronger. The fluttering of aircraft wings[26], flapping of an airport windsock[27], deflection of wind-turbine blades[28], inflation of auto mobile airbags[29], dynamics of spacecraft parachutes[30], rocking motion of ships[31], pumping of blood by the ventricles of the human heart[32], accompanied by the opening and closing of the heart valves, and blood flow and arterial dynamics in cerebral aneurysms, lubrication studies [33], are all FSI examples. In engineering applications, FSI plays an important role and influences the decisions that go into the design of systems of contemporary interest. Therefore, truly predictive FSI methods, which help address these problems of interest, are in high demand in industry, research laboratories, medical fields, space exploration, and many other contexts[34]. The inherently non-linear and time-dependent nature of FSI makes it very difficult to use analytical methods in this class of problems. Only a handful of cases have been studied analytically, where simplifying assumptions have been invoked to arrive at closed-form solutions of the underlying partial differential equations. While we see some use of analytical methods in solution of fluid-only or structure-only problems, there are very few developments in solution of FSI problems. In contrast, there have been significant advances in computational FSI research, especially in recent decades, in both core FSI methods forming a general framework and special FSI methods targeting specific classes of problems. Computational methods, which are robust, efficient, and capable of accurately modeling in 3D FSI with geometrically complex configurations at full spatial scales, have been the focal point of these advances. The challenges involved in computational FSI can be categorized into three areas: problem formulation, numerical discretization, and fluid–structure coupling. The problem formulation takes place at the continuous level, before the discretization. In a typical single-field mechanics problem, such as a fluid-only or structure-only problem, one begins with the classic set of governing differential balance equations in the problem domain and a set of boundary conditions on the domain boundary. The domain may or may not be in motion. The situation is more complicated in an FSI problem. The sets of differential equations and boundary conditions associated with the fluid and structure domains must be satisfied simultaneously. The domains do not overlap, and the two systems are coupled at the fluid–structure interface, which requires a set of physically meaningful interface conditions. These coupling conditions are the compatibility of the kinematics and traction at the fluid–structure interface. The structure domain is in motion and, in most cases, its motion follows the material particles, or points, which constitute the structure. This is known as the Lagrangian description of the structural motion. As the structure moves through space, the shape of the fluid sub-domain changes to conform to the motion of the structure. The motion of the fluid mechanics domain needs to be accounted for in the differential equations and boundary conditions. There are two major classes of methods for this, which are known in the discrete setting as the non-moving-grid and moving-grid approaches. Furthermore, the motion of the fluid domain is not known a priori and it is a function of the unknown structural displacement. This makes FSI a three-field problem, where the third unknown is the motion of the fluid domain. All the issues related to the numerical discretization of a single-field problem, such as the accuracy, stability, robustness, speed of execution, and the ability to handle complex geometries, are likewise present in an FSI problem. The additional challenges in FSI come from the discretization at the fluid–structure interface. The most flexible option is, of course, to have separate fluid and structure discretizations for the individual sub-problems, the so called partitioned approach, which results in non matching meshes at the interface. In this case, one needs to ensure that, despite the non matching interface meshes, the fluid and structure have the correct coupling of the kinematics and traction. Another option is to used a monolithic approach in which we have a matching discretizations at the fluid–structure interface. In this case, the satisfaction of the FSI coupling conditions is much less challenging. However, this choice leads to a lack of flexibility in the discretization choices and mesh refinement levels for the fluid and structure sub-problems. That flexibility becomes increasingly important as the complexity of the fluid–structure interface geometry increases. On the other hand, there are situations where having matching discretizations at the interface is the most effective approach. There are two major classes of FSI coupling techniques: loosely-coupled and strongly-coupled, which are also referred to as staggered or partitioned and monolithic, respectively. Monolithic coupling often refers to strong coupling with matching interface discretizations. In loosely-coupled approaches, the equations of fluid mechanics, structural mechanics, and mesh moving are solved sequentially. For a given time step, a typical loosely-coupled algorithm involves the solution of the fluid mechanics equations with the velocity boundary conditions coming from the extrapolated structure displacement rate at the interface, followed by the solution of the structural mechanics equations with the updated fluid mechanics interface traction, and followed by the solution of the mesh moving equations with the updated structural displacement at the interface. This enables the use of existing fluid and structure solvers, a significant motivation for adopting this approach. In addition, for several problems the staggered approach works well and is very efficient. However, convergence difficulties are encountered sometimes, most-commonly when the structure is light and the fluid is heavy, and when an incompressible fluid is fully enclosed by the structure. In strongly-coupled approaches, the equations of fluid, structure, and mesh moving are solved simultaneously, in a fully-coupled fashion. The main advantage is that strongly-coupled solvers is that they are more robust with respect to the loosely-coupled ones [35, 36] and many of the problems encountered with the partitioned approaches are avoided. However, strongly-coupled approaches necessitate writing a fully-integrated FSI solver, virtually precluding the use of existing fluid and structure solvers. There are three categories of coupling techniques in strongly-coupled FSI methods: block-iterative, quasi-direct, and direct coupling. The methods are ranked according to the level of coupling between the blocks of the left-hand-side matrix [37, 38, 39]. In all three cases, iterations are performed within a time step to simultaneously converge the solutions of all the equations involved. Another computational challenge in some FSI applications is the need to accommodate very large structural motions. In this case, one needs a robust mesh moving technique and the option to periodically regenerate the fluid mechanics mesh (i.e., re-mesh) to preserve the mesh quality and consequently the accuracy of the FSI computations. The re-meshing procedure requires the interpolation of the solution from the old mesh to the new one. Re-meshing and data interpolation are also necessary for fluid-only computations over domains with known motion. The difference between that and FSI is that the re-meshing can be precomputed in fluid-only simulations, while in the case of FSI the fluid mechanics mesh quality depends on the unknown structural displacements, and the decision to re-mesh is made during run-time. Above all the applications of FSI problem, the biomedical field is becoming of great interest [40, 41, 42, 32]. In particular we assist to a huge increase of FSI studies of different part of the cardiovascular system. The general motivation for cardiovascular modeling is the prevalence of cardiovascular diseases, the single largest cause of death worldwide, which is responsible for more than half of mortality in the developed countries. For example, atherosclerosis is responsible for both heart attacks and stroke. It is a complex disease that generally takes decades to develop. There is widespread study of its origins, its treatment and, hopefully, its reversal. Of the many risk factors that have been identified blood cholesterol and triglyceride levels, smoking, obesity, genetics, etc., only haemodynamic and mechanical factors can explain the focal nature of the disease. A better understanding of these factors is essential to our understanding of the disease[43]. Another example is the studying of an aneurysm which is a gradual dilation of an arterial segment over a period of years. The aneurysm wall stretches and becomes thinner and weaker than normal arterial walls. Consequently, untreated aneurysms can rupture causing massive haemorrhage, except in the brain where rupture leads to possibly lethal vasospasm. The plastic deformation of the arterial wall is associated with structural changes in the connective tissue. Modeling and simulations of blood flow through arterial grafts, through reconstructed vascular segments and through vessels with implanted medical devices, have been carried out by many groups in order to provide knowledge of flow behavior and the applied stress fields. These investigations are useful to optimize surgical procedures or the design of medical devices. Most of the studies have been performed in idealized geometries but there is a growing interest in computations carried out in realistic geometries determined from medical imaging[43]. Many problems of the investigated problems, such as composite materials, porous media, turbulent transport in high Reynolds number flows, etc., have multiscale solutions. A complete analysis of these problems is extremely challenging since a direct numerical solution of the multiscale problems is difficult even with modern supercomputers due to different characteristic temporal and spatial scale of problem. Direct solutions provide quantitative information of the physical processes at all scales, on the other hand, from an engineering point of view, it is often sufficient to predict the macroscopic properties of the multiscale systems, such as the effective conductivity, elastic moduli, permeability and eddy diffusivity. Therefore, it is desirable to develop a method that captures the small scale effect on the large scales, but does not require to solve all the small scale features.

In the previous Chapter we show how coupling different codes into the computational platform SALOME can be used to investigate the dynamics of the primary loop of a nuclear reactor. Another example where this coupling is useful, is into biomedical applications, where usually the system of interest is a component of the vascular circulatory system and, in order to study the dynamics of that problem, one has to take into account the effects of the entire connected loop. Because of the complexity of the system it is not possible consider the circulatory system as a unique three-dimensional domain and it is indeed necessary to use multiscale models. In particular a multiscale technique allows us to use multidimensional complex model to represent some part of the circulatory system, such as a valve, and a simplified mono-dimensional model for the rest of the system. The coupling between those modules represents a fundamental topic with regard to stability issues. Multiscale approach is clearly extremely attractive and can be achieved using the computational platform in the development of any complex system simulation. This Chapter is organized as follows: In the first Section we introduce the mathematical description of a Fluid Structure Interaction problem in particular in the first part we derive the monolithic formulation of the FSI set of balance equation (mass, balance). In the second part we introduce the mono-dimensional simplified model for the cardiovascular system and in the third part we give a description of the interfaces used to couple those two modules. In this case the 3D and 1D modules are both in-house code that have been integrated into the computational platform SALOME according to the procedure described in 2.2. In the second Section there is a description of the solution algorithm for the monolithic 3D FSI problem. Due to the large displacement involved into the simulation, the monolithic approach represent the best solution in terms of numerical stability [36, 38], but maybe the worst int term of computational cost. In order to reduce the cost we use a new projection scheme with an iterative penalty correction algorithm which is deeply investigated into Section 4.3. In the third Section we show some numerical results for different test cases. In particular the first one is a validation test for the new penalty projection algorithm where we reproduce the results of the classic FSI benchmark proposed by Bathe in [25]. In the second Test we couple a 3D blood vessel model with the simplified mono-dimensional model of the cardiovascular system in order to test the robustness and the stability of the coupling between the different-scales problems. Finally, in the last two Tests, the presence of an aneurysm into the cardiovascular system is investigated in both a steady state and a transient conditions. In both cases the aneurysm is modeled with a multidimensional FSI model while the rest of the simplified cardiovascular system is taken into account with a mono-dimensional system code.

As in the previous Chapter, in all the examples the two modules exchange computational fields, through the computational platform, during the execution.

4.2 Mathematical Model

4.2.1 FSI problem

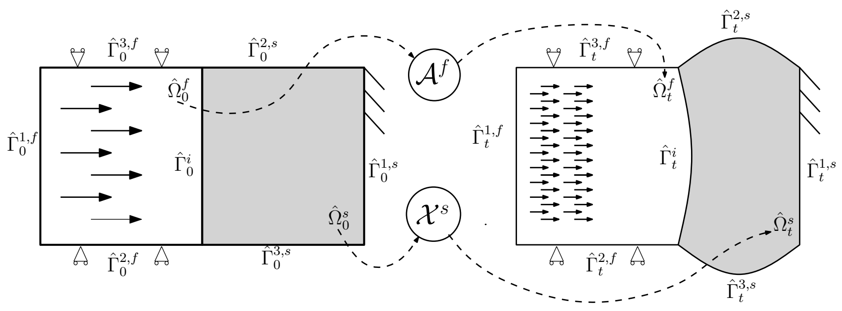


Figure 4.1: Reference and current configuration where a vessel wall interacts with a fluid.

In this Section we introduce the mathematical model for a generic fluid structure interaction problem. We denote by $H^s(\mathcal{O})$, $s \in \Re$, the standard Sobolev space of order s with respect to the set $\mathcal{O}$, which is either the flow domain Ω, or its boundary Γ, or part of its boundary. Whenever m is a non negative integer, the inner product over $H^m(\mathcal{O})$ is denoted by $(f,g)_m$ and (f,g) denotes the inner product over $H^0(\mathcal{O}) = L^2(\mathcal{O})$. Hence, we associate with $H^m(\mathcal{O})$ its natural norm $\|f\|_{m,\mathcal{O}} = \sqrt{(f,f)_m}$. For $1 \leq p < \infty$ the Sobolev space $W^{m,p}(\mathcal{O})$ is defined as the closure of $C^\infty(\mathcal{O})$ in the norm

$$\|f\|^p_{W^{m,p}(\mathcal{O})} = \sum_{|\alpha| \leq m} \int_{\mathcal{O}} |\Big(\frac{\partial}{\partial x}\Big)^\alpha f(x)|^p \, dx \, .$$

The closure of $C_0^\infty(\mathcal{O})$ under the norm $\|\cdot\|_{W^{m,p}(\mathcal{O})}$ will be denoted by $W_0^{m,p}(\mathcal{O})$. Whenever possible, we will neglect the domain label in the norm. For vector-valued functions and spaces, we use boldface notation. For example, $\mathbf{H}^s(\Omega) = [H^s(\Omega)]^n$ denotes the space of $\Re^n$-valued functions such that each component belongs to $H^s(\Omega)$. Of special interest is the space

$$\mathbf{H}^1(\Omega) = \left\{ v_j \in L^2(\Omega) \,\middle|\, \frac{\partial v_j}{\partial x_k} \in L^2(\Omega) \quad \text{for } j,k = 1,2 \right\}$$

equipped with the norm $\|\vec{v}\|_1 = (\sum_{k=1}^{2} \|v_k\|_1^2)^{1/2}$. For details concerning the function spaces we have introduced, one may consult [44, 8].

In an ordinary FSI problem we consider a mechanical system composed by a laminar Newtonian fluid region and a solid one which defines a moving domain Ω_t. A schematic geometry of the problem is shown in Figure 4.1. Let Ω_t^f and Ω_t^s be the fluid and the solid region at $t \in (0,T]$, respectively. At $t = 0$ the fluid and solid region are defined by $\hat{\Omega}_0^f$ and $\hat{\Omega}_0^s$. Let $\Gamma_t^i = \bar{\Omega}_t^f \cap \bar{\Omega}_t^s$ and $\hat{\Gamma}_0^i = \bar{\Omega}_0^f \cap \bar{\Omega}_0^s$ be the interface where solid and fluid interact. $\Gamma_t^k, k = 1,2,3$ and $\hat{\Gamma}_0^k, k = 1,2,3$ are defined to be the remaining external boundaries at $t \in (0,T]$ and $t = 0$, respectively. The evolution of the solid and fluid domain $\hat{\Omega}_0^f$ and $\hat{\Omega}_0^s$ are defined by

$$\mathcal{X}^s : \hat{\Omega}_0^s \times \mathbb{R}^+ \to \mathbb{R}^3 \,,$$
$$\mathcal{A}^f : \hat{\Omega}_0^f \times \mathbb{R}^+ \to \mathbb{R}^3 \,,$$

such that the range of $\mathcal{X}^s(\cdot,t)$ and $\mathcal{A}^f(\cdot,t)$ define Ω_t^s and Ω_t^f, respectively. $\mathcal{X}^s$ maps any material point $\hat{\mathbf{x}}_0^s$ from the given fixed reference configuration $\hat{\Omega}_0^s$ to the current solid material configuration Ω_t^s. The solid displacement is then defined as

$$\hat{\mathbf{u}}^s(\hat{\mathbf{x}}_0^s, t) = \mathcal{X}(\hat{\mathbf{x}}_0^s, t) - \hat{\mathbf{x}}_0^s \,. \tag{4.1}$$

The mapping $\mathcal{A}^f$ is such that $\mathcal{A}^f(\hat{\mathbf{x}}_0^f, t) = \hat{\mathbf{x}}_0^f + \hat{\mathbf{u}}^f(\hat{\mathbf{x}}_0^f, t)$, where $\hat{\mathbf{u}}^f(\hat{\mathbf{x}}_0^f, t)$ is defined as an arbitrary extension operator over the fluid domain $\hat{\Omega}_0^f$ and given by

$$\hat{\mathbf{u}}^f(\hat{\mathbf{x}}_0^f, t) = \text{Ext}(\hat{\mathbf{u}}^s|_{\hat{\Gamma}_0^i}) \quad \text{in} \quad \hat{\Omega}_0^f \,. \tag{4.2}$$

The extension operator used to evaluate the fluid region displacement is the harmonic or Laplace operator. Other similar operators can be employed as described in [45, 46, 47, 48]. The velocity $\hat{\mathbf{w}}^f$ is defined by

$$\hat{\mathbf{w}}^f = \frac{\partial \hat{\mathbf{u}}^f}{\partial t} \quad \text{in} \quad \hat{\Omega}_0^f \,. \tag{4.3}$$

This quantity represents the velocity in terms of the reference coordinate $\hat{\mathbf{x}}_0^f$. The behavior of the fluid is described by the Navier-Stokes equations for incompressible flows. For details the interested reader can also see [49, 50, 51, 52].

$$\begin{aligned}
&\rho^f \frac{\partial \mathbf{v}^f}{\partial t}\bigg|_{\tilde{\mathcal{A}}} + \rho^f \left(\mathbf{v}^f - \mathbf{w}^f\right) \cdot \boldsymbol{\nabla} \mathbf{v}^f - \boldsymbol{\nabla} \cdot \boldsymbol{\sigma}^f = \mathbf{0} && \text{in} \quad (0,T) \times \Omega_t^f\,, \\
&\boldsymbol{\nabla} \cdot \mathbf{v}^f = \mathbf{0} && \text{in} \quad (0,T) \times \Omega_t^f\,, \\
&\mathbf{v}^f|_{t=0} = \mathbf{v}_0 && \text{in} \quad \hat{\Omega}_0^f\,, \\
&\mathbf{v}^f|_{\Gamma_{t,D}^{1,f} \cup \Gamma_{t,D}^{2,f}} = \mathbf{g}^f && \text{in} \quad (0,T)\,, \\
&\boldsymbol{\sigma}_f \cdot \mathbf{n}^f|_{\Gamma_{t,N}^{1,f} \cup \Gamma_{t,}^{2,f}} = \mathbf{h}^f && \text{in} \quad (0,T)\,,
\end{aligned} \tag{4.4}$$

where ρ^f is the constant density, $\mathbf{v}^f$ is the fluid velocity, $\tilde{\mathcal{A}}$ denotes the ALE application that maps the reference fluid configuration $\hat{\Omega}_0^f$ onto the current fluid configuration Ω_t^f and $\mathbf{w}^f$ denotes the fluid domain velocity. $\mathbf{n}$ is the unit normal vector that points outward from the boundary $\partial\Omega_t^f$ and $\mathbf{g}^f$, $\mathbf{h}^f$, $\mathbf{v}_0$ are given data. The flow state variables in the incompressible case are the pressure p^f and the velocity $\mathbf{v}^f$. The contribution of external forces such as gravity is assumed to be negligible. The constitutive relation for the stress tensor in the Newtonian incompressible case reads

$$\boldsymbol{\sigma}^f = -p^f \mathbf{I} + \boldsymbol{\tau}^f = -p^f \mathbf{I} + 2\mu^f \boldsymbol{\epsilon}\left(\mathbf{v}^f\right)\,, \tag{4.5}$$

where μ^f is the dynamic viscosity of the fluid, p^f the Lagrange multiplier associated to the incompressibility constraint and $\boldsymbol{\epsilon}\left(\mathbf{v}^f\right)$ the strain rate tensor defined as

$$\boldsymbol{\epsilon}\left(\mathbf{v}^f\right) = \frac{1}{2}\left(\boldsymbol{\nabla}\mathbf{v}^f + \left(\boldsymbol{\nabla}\mathbf{v}^f\right)^{\mathrm{t}}\right)\,. \tag{4.6}$$

The total time derivative is related to the adopted reference systems. The governing equations for structural mechanics are the following momentum equations

$$\rho^s \frac{\partial \mathbf{v}^s}{\partial t} - \boldsymbol{\nabla} \cdot \boldsymbol{\sigma}^s(\mathbf{u}^s) = \mathbf{0} \qquad \text{in} \quad \Omega_t^s\,, \tag{4.7}$$

where ρ^s is the density of the solid material, $\mathbf{v}^s$ is the velocity field and $\boldsymbol{\sigma}^s$ its Cauchy stress tensor, which is a function of the solid region displacement $\mathbf{u}^s$. Since the constitutive law for the solid stress tensor is expressed in terms of displacements one must solve both the balance equations (4.7) and the kinematic relation

$$\mathbf{v}^s = \frac{\partial \mathbf{u}^s}{\partial t}\,. \tag{4.8}$$

For the reference configuration we can introduce the right Cauchy-Green deformation tensor $\mathbf{C}$ as

$$C_{ij} = F_{ki}F_{kj} \qquad \forall\, i,j = 1,\ldots,3\,, \tag{4.9}$$

where $\mathbf{F}$ is the deformation gradient tensor defined by $\mathbf{F} = \mathbf{I} + \boldsymbol{\nabla}\mathbf{u}^s$. In a similar way in the current configuration we can introduce the left Cauchy-Green deformation tensor, $\mathbf{b}$, as

$$b_{ij} = F_{ik}F_{jk} \qquad \forall\, i,j = 1,\ldots,3\,. \tag{4.10}$$

According with this notation we can now express the Cauchy stress tensor, $\boldsymbol{\sigma}^s$, as [49]

$$\sigma^s_{ij} = \frac{2}{J}\left[b_{ij}\left(I\, b_{ij} - b_{im}b_{mj}\right)\frac{J\delta_{ij}}{2}\right]\begin{pmatrix}\frac{\partial W}{\partial I}\\ \\ \frac{\partial W}{\partial II}\\ \\ \frac{\partial W}{\partial J}\end{pmatrix}\,, \tag{4.11}$$

where $I = C_{ii}$, $II = 1/2\left(I - C_{ij}C_{ji}\right)$ are the first and second invariant of the right Cauchy-Green strain tensor $\mathbf{C}$ and J its determinant. The quantity $W = W(I, II, J)$ is the strain energy of the system which depends on the constitutive law of the considered material. For example for a Neo-Hookian material, with respect to the current configuration, the energy function is defined by

$$\begin{aligned} W(I,J) =& \frac{1}{2}\mu_s\left(J^{-2/3}\mathrm{tr}\mathbf{C} - 3\right) + \\ & \frac{1}{2}\left(\lambda + \frac{2}{3}\mu_s\right)\left(\frac{1}{2}(J^2-1) - \ln J\right)\,. \end{aligned} \tag{4.12}$$

In the case of incompressible solid, the third invariant is equal to one so the energy density function becomes

$$W(I,J) = \frac{1}{2}\mu_s\left(\mathrm{tr}\mathbf{C} - 3\right) \tag{4.13}$$

and the Cauchy stress tensor is defined by

$$\boldsymbol{\sigma}^s = -p^s\mathbf{I} + \boldsymbol{\sigma}^{s*}\,, \tag{4.14}$$

where $\boldsymbol{\sigma}^{s*}$ is the tensor obtained by using the equations (4.11) and (4.13). The problem defined by (4.4)-(4.7) is not well posed since we have not yet prescribed any boundary conditions at the interface Γ^i_t. The coupling between the fluid and the solid model determines the missing boundary conditions, which consist of imposing the continuity of velocity and stress at the interface Γ^i_t as

$$\mathbf{v}^f|_{\Gamma^i_t} = \mathbf{v}^s|_{\Gamma^i_t}\,, \tag{4.15}$$

$$\boldsymbol{\sigma}^f \cdot \mathbf{n}^f|_{\Gamma^i_t} + \boldsymbol{\sigma}^s \cdot \mathbf{n}^s|_{\Gamma^i_t} = \mathbf{0}\,. \tag{4.16}$$

In order to write the weak formulation of the coupled problem, let us consider the following functional spaces

$$\begin{aligned}
\mathbf{V}^t &= \{\boldsymbol{\phi} \in \mathbf{H}^1(\Omega^f_t) : \boldsymbol{\phi}|_{\Gamma^{1,f}_{t,D}\cup\Gamma^{2,f}_{t,D}} = \mathbf{0}\}\,,\\
\mathbf{V}^t_g &= \{\boldsymbol{\phi} \in \mathbf{H}^1(\Omega^f_t) : \boldsymbol{\phi}|_{\Gamma^{1,f}_{t,D}\cup\Gamma^{2,f}_{t,D}} = \mathbf{g}^f\}\,,\\
Q^t &= L^2(\Omega^f_t)\,,\\
\mathbf{M}^0 &= \{\boldsymbol{\psi} \in \mathbf{H}^1(\hat{\Omega}^s_0) : \boldsymbol{\psi}|_{\hat{\Gamma}^{1,s}_{0,D}\cup\hat{\Gamma}^{2,s}_{0,D}\cup\hat{\Gamma}^3_{0,D}} = \mathbf{0}\}\,,\\
\mathbf{M}^0_g &= \{\boldsymbol{\psi} \in \mathbf{H}^1(\hat{\Omega}^s_0) : \boldsymbol{\psi}|_{\hat{\Gamma}^{1,s}_{0,D}\cup\hat{\Gamma}^{2,s}_{0,D}\cup\hat{\Gamma}^3_{0,D}} = \mathbf{g}^s\}\,,\\
D^0 &= L^2(\hat{\Omega}^s_0)\,.
\end{aligned}$$

In addition, let us introduce the following bilinear form

$$a^f(\mathbf{v^f}, \boldsymbol{\phi}) = \int_{\Omega^f} \boldsymbol{\tau}^f(\mathbf{v}^f) : \boldsymbol{\nabla}\phi\, d\mathbf{x} = \mu(\boldsymbol{\nabla}\mathbf{v}^f + (\boldsymbol{\nabla}\mathbf{v}^f)^T, \boldsymbol{\nabla}\boldsymbol{\phi})\,, \tag{4.17}$$

where we denote with $\boldsymbol{\tau}^f$ the fluid viscosity tensor. The variational formulation of the fluid equations can be obtained through the usual method by multiplying the equations (4.4) with appropriate test functions, performing integrations on the whole domain and keeping into account the boundary and interface conditions. This procedure leads, for the velocity field $\mathbf{v} \in \mathbf{V}^t_g$ and pressure

$p \in Q^t$, to the following fluid momentum equation

$$\rho^f \left(\left. \frac{\partial \mathbf{v}^f}{\partial t} \right|_{\tilde{\mathcal{A}}}, \boldsymbol{\phi} \right) + a(\mathbf{v}^f, \boldsymbol{\phi}) - \rho^f((\boldsymbol{\nabla} \cdot \mathbf{w}^f)\mathbf{v}^f, \boldsymbol{\phi}) + \rho^f(\left(\mathbf{v}^f - \mathbf{w}^f\right) \cdot \boldsymbol{\nabla}\mathbf{v}^f, \boldsymbol{\phi}) =$$
$$(p^f, \boldsymbol{\nabla} \cdot \boldsymbol{\phi}) + \int_{\Gamma_t^i} (\boldsymbol{\sigma}^f \cdot \mathbf{n}^f) \cdot \boldsymbol{\phi} \, d\gamma + \int_{\Gamma_N^f} \mathbf{h}^f \cdot \boldsymbol{\phi} \, d\gamma \,, \tag{4.18}$$
$$(q, \boldsymbol{\nabla} \cdot \mathbf{v}^f) = 0 \,,$$
$$\mathbf{v}^f|_{t=0} = \mathbf{v}_0^f \,,$$

for all $\boldsymbol{\phi} \in \mathbf{V}^t$ and $q \in Q^t$. In a similar way, we define the following bilinear form

$$a^s(\mathbf{u}^s, \boldsymbol{\psi}) = (\boldsymbol{\sigma}^s(\mathbf{u}^\mathbf{s}), \boldsymbol{\nabla}\boldsymbol{\psi}) \,. \tag{4.19}$$

By following the procedure briefly described above, we obtain at each time t, for the velocity $\mathbf{u}^s \circ \mathcal{X}^s \in \mathbf{M}_g^0$ and pressure $p^s \circ \mathcal{X}^s \in \mathbf{D}^0$, the following weak formulation for the solid problem

$$\rho^s \left(\frac{\partial^2}{\partial t^2} \mathbf{u}^s, \boldsymbol{\psi} \right) + a^s(\mathbf{u}^s, \boldsymbol{\psi}) - (p^s, \boldsymbol{\nabla} \cdot \boldsymbol{\psi}) = \int_{\Gamma_t^i} (\boldsymbol{\sigma}^s \cdot \mathbf{n}^s) \cdot \boldsymbol{\psi} \, d\gamma + \int_{\Gamma_N^s} \mathbf{h}^s \cdot \boldsymbol{\psi} \, d\gamma \,,$$
$$(d, \boldsymbol{\nabla} \cdot \mathbf{u}^s) = 0 \,,$$
$$\mathbf{u}^s|_{t=0} = \mathbf{u}_0^s \,, \qquad \mathbf{v}^s|_{t=0} = \mathbf{v}_0^s \,,$$

for all $\boldsymbol{\psi} \circ \mathcal{X}^s \in \mathbf{M}^0$ and $d \circ \mathcal{X}^s \in \mathbf{D}^0$. Let us introduce a global weak formulation for the fluid-structure problem. If we define the functional space

$$\mathbf{S}^t = \{(\boldsymbol{\phi}, \boldsymbol{\psi} \circ \mathcal{X}^s) \in \mathbf{V}^t \times \mathbf{M}^0 : \boldsymbol{\psi}|_{\Gamma_t^i} = \boldsymbol{\phi}|_{\Gamma_t^i}\} \,, \tag{4.20}$$

from (4.15), (4.16), (4.18) and (4.20), we can write the FSI problem in the coupled formulation as

$$\rho^f \left(\left. \frac{\partial \mathbf{v}^f}{\partial t} \right|_{\tilde{\mathcal{A}}}, \boldsymbol{\varphi} \right) + a(\mathbf{v}^f, \boldsymbol{\varphi}) - \rho^f((\boldsymbol{\nabla} \cdot \mathbf{w}^f)\mathbf{v}^f, \boldsymbol{\varphi}) + \rho^f(\left(\mathbf{v}^f - \mathbf{w}^f\right) \cdot \boldsymbol{\nabla}\mathbf{v}^f, \boldsymbol{\varphi}) -$$
$$(p^f, \boldsymbol{\nabla} \cdot \boldsymbol{\varphi}) + \rho^s \left(\frac{\partial^2}{\partial t^2} \mathbf{u}^s, \boldsymbol{\varphi} \right) + a^s(\mathbf{u}^s, \boldsymbol{\varphi}) - (p^s, \boldsymbol{\nabla} \cdot \boldsymbol{\varphi}) \tag{4.21}$$
$$- \int_{\Gamma_N^s} \mathbf{h}^s \cdot \boldsymbol{\varphi} \, d\gamma - \int_{\Gamma_N^f} \mathbf{h}^f \cdot \boldsymbol{\varphi} \, d\gamma = 0 \,, \quad \forall \boldsymbol{\varphi} \in \mathbf{S}^t$$
$$(q, \boldsymbol{\nabla} \cdot \mathbf{v}^f) = 0 \qquad (d, \boldsymbol{\nabla} \cdot \mathbf{u}^s) = 0 \,, \tag{4.22}$$
$$\mathbf{v}^f|_{t=0} = \mathbf{v}_0^f \qquad \mathbf{u}^s|_{t=0} = \mathbf{u}_0^s \qquad \mathbf{v}^s|_{t=0} = \mathbf{v}_0^s \,.$$

It is worth noting that by using the coupling conditions (4.15), (4.16) and this particular choice of the fluid-structure test functions, the boundary terms that appear in the fluid-solid interface Γ_t^i cancel out. This assures that forces at the interface are always computed in an exact way.

4.2.2 The mono-dimensional FSI model

The mono-dimensional model combines the incompressible Navier-Stokes equation system and a shell model for the vessel walls. This implies that only radial displacements are considered. Under the assumption of axial symmetry of the system we may use a mono-dimensional set of equation. We can integrate (4.4) with no ALE velocity field over the transverse surface and obtain [53, 47]

$$\begin{cases} \dfrac{\partial A}{\partial t} + \dfrac{\partial v}{\partial x} = 0\,, \\ \dfrac{\partial Q}{\partial t} + \dfrac{\partial}{\partial x}\left(\dfrac{\alpha Q^2}{A}\right) + \dfrac{A}{\rho_f}\dfrac{\partial p}{\partial x} = -2\pi\nu(\alpha+2)\dfrac{Q}{A}\,, \end{cases} \tag{4.23}$$

where A is the transverse surface of the system, Q is the flow rate and p the pressure of the system, α the momentum flux correction coefficient, ν the fluid dynamic viscosity and ρ_f is the fluid density. The system (4.23) is not closed unless we introduce a constitutive relation for the pressure [53, 47]

$$p = \beta\psi(A) + p_{ref} = \beta\frac{\sqrt{A}}{A_0} - \gamma\frac{\sqrt{A_0}}{A_0}\,, \tag{4.24}$$

where A_0 is the initial transverse surface of the system with β and γ appropriate coefficient related to the solid Young modulus E. With the area equation (4.24) the system (4.23) turns into an hyperbolic closed problem which can be solved by imposing the following appropriate boundary conditions

$$\begin{cases} A = A_0 & \text{on } \Gamma_i \cup \Gamma_o \\ u = u_0 & \text{on } \Gamma_i \\ \partial u/\partial x = 0 & \text{on } \Gamma_o \end{cases} . \tag{4.25}$$

where Γ_i and Γ_o are the inlet and outlet of the domain, respectively. The variational formulation of the mono-dimensional problem, is obtained by integrating the system (4.23) with mono-dimensional weight functions. Briefly one can set $\psi = \psi(s)$ and $\phi = \phi(s)$ assuming that the test functions are only a function of the mono-dimensional coordinate s. This module is properly used for mono-dimensional flows such as channels with single fluid. Let A be a surface perpendicular to the center-line. Over the surface A we define average density and average velocity as

$$\bar{\rho} = \frac{\int_A \rho \, dA}{A} \qquad \bar{v} = \frac{\int_A \rho \, v \, dA}{\bar{\rho} A} \,. \tag{4.26}$$

With this definition the first equation of the system (4.23) becomes

$$\int_0^L \psi(s) \frac{\partial}{\partial s}(\bar{v}) \, ds + \frac{\partial}{\partial t}(A) \, ds = \int_0^L \psi(s) \, S_s \, ds \quad \forall \psi \in P(0, L), \tag{4.27}$$

where S_s is an area deformation source from surface integral. In a similar way for the average quantities ($\bar{\rho}$, $\bar{v}$, $\bar{p}$), $\bar{Q} = \bar{\rho}\bar{v}A$ and $\phi \approx \phi(s)$ the momentum equation becomes

$$\int_0^L \left(\frac{\partial}{\partial t} \bar{Q} \right) \phi(s) \, ds + \int_0^L \left(\frac{\partial}{\partial s} \frac{\bar{Q}^2}{A} \right) \phi \, ds = \int_0^L \frac{\partial \bar{p}}{\partial s} \phi \, ds \tag{4.28}$$
$$\int_0^L A\bar{\rho}\mathbf{g} \cdot \widehat{i}_s \, \phi \, ds + \int_0^L \phi(s) \left(M_s + M_v \right) ds \qquad \forall \phi \in V(0, L) \,,$$

where M_s is from surface integral and M_v from volume contributions. Usually $M_s \approx -\, k \, \frac{\rho}{2} \, \bar{u}|\bar{u}|$ (pressure loss). The volume contribution M_v consists of several terms. It is easy to see that $M_v = M_{v,\tau}(\bar{\tau}) + M_{v,vv}(\bar{\mathbf{v}\mathbf{v}} - \bar{v}\bar{v})$ with obvious definition of the terms $M_{v,\tau}$ and $M_{v,vv}$.

4.2.3 Interfaces

As shown in Figure 4.2 the 3D/1D interfaces are the inlet and the outlet of the multidimensional regions. The interface between the outlet of the mono-dimensional system and the inlet of the more complex domain is marked with the letter A while the letter B marks the other corresponding interface. Over these interfaces the fluid must flow from one-dimensional module to a three-dimensional one. In this interface the mass and momentum must be conserved. There are many sophisticated techniques to define a numerical algorithm able to identify the values to set on the interfaces for example one can use algorithms

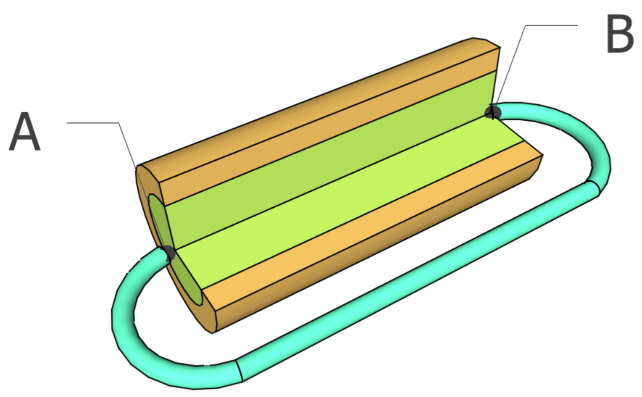


Figure 4.2: 3D/1D interfaces in a schematic diagram

based on Mortar or Lagrangian multiplier method [18, 16, 54], in this Chapter we use a direct coupling algorithm for the data exchange between the different models. Concerning the computational structure of the coupled problem and the generation of the MEDmem interface one can refer to the analogues case explained in Section 3.6. The one-dimensional module is essentially a hyperbolic differential equation and therefore it requires boundary conditions only in inflow regions. In the inlet region of the multidimensional system the average velocity must be set as boundary conditions. This imposition is a very challenging situations since from the mono-dimensional system we have the propagation of mass flux Q (or velocity in the normal direction) and cross section area A. In three-dimensional domains if the velocity vector is fully specified on boundaries, the pressure is determined so it can be specified only if the normal component of the velocity field is not imposed. Imposing time dependent pressure value on the inlet surface leads to large oscillations and velocity discontinuities. For these reasons the pressure is expressed as a function of the cross section dimension so that we avoid the coupling between the two systems pressure values and obtain the continuities of this field by imposing a fixed cross section dimension at the boundary of the system. The interface 3D-CFD and 1D that links the 3D-CFD module to the the mono-dimensional module is an outflow region for the system and therefore the pressure field, evaluated in the simplified domain is imposed in the outflow region. In this region, since the velocity profile is not imposed, the pressure can be easily set and the flow rate, evaluated with this boundary condition, is imposed as a inflow boundary condition in the mono-dimensional problem.

4.3 Solution technique of FSI

The most common solution strategy implemented in software packages for a FSI problem, is the so-called partitioned approach, which decouples the problem into two separate sub-problems and uses dedicated software for each different region [55, 56]. According to this solution strategy the coupling is achieved by enforcing continuity conditions along the fluid-solid interface. For details the interested reader can see [55, 57, 58, 59]. However in applications where large displacements may occur, one can see that explicit partitioned algorithms show instabilities due to the poor fluid-solid coupling matching where the solid-fluid power remains unbalanced at the interface. In order to overcome this limit one could implicitly enforce coupling conditions. These algorithms are called fully coupled or monolithic [60, 36]. A monolithic algorithm solves simultaneously for the fluid and structure unknowns, so that the solid and fluid regions are treated as a single continuum and the interface conditions are automatically enforced [24, 61]. Numerical experiments, with similar fluid and solid density, show that only fully coupled algorithms exhibit good stability properties [60, 62]. Monolithic fully-coupled algorithms are always more stable and robust but they are also CPU-time expensive [63, 64, 65, 66, 67]. In a monolithic fully-coupled approach for numerical simulations of incompressible fluid-structure bodies the velocity and the pressure fields are coupled. This issue is enhanced by the saddle-point character of the incompressible Lagrange multiplier formulation and by multidimensional geometries where one has to deal with a high number of degrees of freedom [43, 64, 63, 68]. The reduction of the computational cost for such problems motivate many papers published in recent years [69, 70, 50, 71]. Projection methods, which split the velocity from the pressure field, are very popular in fluids since the boundary error generated by the projection method does affect mainly the boundary layer but in the solid region the projection error propagates quickly in the interior leading to non acceptable solutions [72, 51]. In all the examples presented in this work, we use iterative penalty-projection algorithm for a generic monolithic fluid structure interaction solver based on unstructured computational grids developed in [23]. In particular our strategy is to use a one-step penalty-projection method in the fluid domain and a iterative penalty-projection method in the solid region. The most attractive feature of projection methods is that, at each time step, one only needs to solve a sequence of decoupled advective and elliptic equations for the velocity and the pressure field, making it very efficient for large scale time-dependent numerical simulations [50]. However, it is well known that standard one-step projection methods produces pressure solutions that are not exact near the domain boundary where velocity fields should be imposed and they must be integrated with iterative algorithms able to correct the projection boundary conditions and the corresponding incompressible constraint. The iterative algorithm considered here is the iterative penalty method which can be proved to be convergent under simple conditions [73]. The combination of the penalty and the projection method in the fluid and solid region leads to good performance. The solution of the system (4.21), due to its saddle-point nature, is CPU-time expensive and many authors have proposed different strategies to reduce the computational effort. Some of the most popular ones are decoupled fractional step strategies, domain decomposition methods and reduced models which try to decrease the degrees of freedom by splitting the discrete matrix or using boundary integral techniques [73]. As we exaplain, projection methods are widely used in fluid dynamics but they are not popular in incompressible structural mechanics due to the poor performance of the projection step itself over rigid media. A possible solution of this problem is to split the computation of the velocity and pressure field by introducing an iterative penalty-projection method. The standard projection method consists of two steps: a predictor step and a corrector one [74]. In the predictor step an auxiliary discrete velocity $\tilde{\mathbf{v}}_h$, which does not satisfy the free-divergence condition, is computed, while, in the corrector step, an iterative correction $\delta p^{s,n+1}_{h,proj}$ is introduced to enforce the incompressibility constraint. This projection allows us to solve the pressure and the velocity field separately but, at the same time, it does not recover the original boundary conditions on pressure which are defined implicitly in the original momentum equation [72]. This issue is particular relevant on solid boundaries when incompressible hyper-elastic materials are considered. In this case the boundary conditions involve the whole stress tensor and not only the pressure components. An error on the solid boundary, which deforms the solid surface in the wrong way, is introduced by setting to zero the pressure or its normal derivative. This is particular relevant when large displacements and moving meshes are considered. The iterative penalty procedure begins with the evaluation of an auxiliary velocity field $\tilde{\mathbf{v}}_h$ and a pressure correction term $\delta p^{s,n+1}_{h,pen}$. The quantity $\tilde{\mathbf{v}}_h = (\tilde{\mathbf{v}}^f_h, \tilde{\mathbf{v}}^s_h)$ is the solution of the following momentum balance equation

$$\begin{aligned}&\partial_t(\tilde{\mathbf{v}}^f_h, \boldsymbol{\varphi}_h) + \partial_t(\tilde{\mathbf{v}}^s_h, \boldsymbol{\varphi}_h) + c(\tilde{\mathbf{v}}^f_h, \boldsymbol{\varphi}_h) + d^s(\tilde{\mathbf{v}}^s_h, \boldsymbol{\varphi}_h) - (p^{f,k}_h, \boldsymbol{\nabla}\cdot\boldsymbol{\varphi}_h)\\ &- (\delta p^{f,k}_{h,proj}, \boldsymbol{\nabla}\cdot\boldsymbol{\varphi}_h) - (r_1 \delta p^{s,k+1}_{h,pen}, \boldsymbol{\nabla}\cdot\boldsymbol{\varphi}_h) = 0 \quad \forall \boldsymbol{\varphi}_h \in \mathbf{S}^t\,. \end{aligned} \tag{4.29}$$

where $\partial_t(\tilde{\mathbf{v}}^\cdot, \cdot)$ is the Eulerian time discretization of the velocity field defined by

$$\partial_t(\tilde{\mathbf{v}}^s_h, \boldsymbol{\psi}_h) = \frac{\rho^s}{\Delta t}(\tilde{\mathbf{v}}^{s,n+1}_h, \boldsymbol{\psi}_h) - \frac{\rho^s}{\Delta t}(\tilde{\mathbf{v}}^{s,n}_h, \boldsymbol{\psi}_h)\,. \tag{4.30}$$

The operator $c(\tilde{\mathbf{v}}^f_h, \boldsymbol{\phi}_h)$ is the fluid advection term modified by the ALE correction as

$$\begin{aligned}c(\tilde{\mathbf{v}}^f_h, \boldsymbol{\phi}_h) =& a(\tilde{\mathbf{v}}^{f,n+1}_h, \boldsymbol{\phi}_h) - \Big(\rho^f (\boldsymbol{\nabla}\cdot\mathbf{w}^{f,n}_h)\tilde{\mathbf{v}}^{f,n+1}_h, \boldsymbol{\phi}_h\Big)\\ &+ \rho^f \Big(\big(\tilde{\mathbf{v}}^{f,n}_h - \mathbf{w}^{f,n}_h\big)\cdot\boldsymbol{\nabla}\tilde{\mathbf{v}}^{f,n+1}_h, \boldsymbol{\phi}_h\Big) - \int_{\Gamma^f_N} \mathbf{h}^f\cdot\boldsymbol{\phi}_h\, d\gamma\end{aligned} \tag{4.31}$$

and $d^s(\tilde{\mathbf{v}}^s_h, \boldsymbol{\psi}_h)$ is

$$d^s(\tilde{\mathbf{v}}^s_h, \boldsymbol{\psi}_h) = \Delta t\, a^s(\tilde{\mathbf{v}}^{s,n+1}_h, \boldsymbol{\psi}_h) + a^s(\mathbf{u}^{s,n}_h, \boldsymbol{\psi}_h) - \int_{\Gamma^s_N} \mathbf{h}^s\cdot\boldsymbol{\psi}_h\, d\gamma\,. \tag{4.32}$$

The update of the penalty correction is obtained by using

$$\delta p^{k+1,n+1}_{h,pen} = \delta p^{k,n+1}_{h,pen} + r_2(\boldsymbol{\nabla}\cdot\tilde{\mathbf{v}}^k_h)\,, \tag{4.33}$$

where r_1, r_2 are real values that satisfy the following constraint [73]

$$0 < r_1 < 2r_2\,. \tag{4.34}$$

It is important to remark that in case of large penalty values the numerical convergence may deteriorate quickly [73]. In order to accelerate the convergence a projection step can be introduced.
The projector algorithm computes the L^2 orthogonal projection of $\tilde{\mathbf{v}}_h^{n+1}$ onto the space of divergence free vectors fields, which reads

$$\begin{aligned} \rho \frac{\mathbf{v}^{n+1} - \tilde{\mathbf{v}}^{n+1}}{\Delta t} + \boldsymbol{\nabla} \delta \tilde{p}_{proj}^{n+1} &= 0 \qquad \text{in} \quad \Omega_t \, , \\ \boldsymbol{\nabla} \cdot \mathbf{v}^{n+1} &= 0 \qquad \text{in} \quad \Omega_t \, . \end{aligned} \tag{4.35}$$

Now we reformulate this Darcy system by taking the divergence of the first expression in order to obtain a Poisson problem for $\delta \tilde{p}_{proj}^{n+1}$. The pressure variations $\delta \tilde{p}_{h,proj}^{f,n+1}$ and $\delta \tilde{p}_{h,proj}^{s,n+1}$ are the solutions of the following weak elliptic problem

$$\begin{aligned} (\boldsymbol{\nabla} \delta \tilde{p}_{h,proj}^{f,n+1}, \boldsymbol{\nabla} \zeta)_{\Omega_n^f} + (\boldsymbol{\nabla} \delta \tilde{p}_{h,proj}^{s,n+1}, \boldsymbol{\nabla} \zeta)_{\Omega_n^s} = & -\frac{\rho^f}{\Delta t} (\boldsymbol{\nabla} \cdot \tilde{\mathbf{v}}^{f,n+1}, \zeta)_{\Omega_n^f} \\ & -\frac{\rho^s}{\Delta t} (\boldsymbol{\nabla} \cdot \tilde{\mathbf{v}}^{s,n+1}, \zeta)_{\Omega_n^s} \, , \end{aligned} \tag{4.36}$$

for all ζ in $\mathbf{H}^1_{\Gamma_0}$, where Γ_0 is the region where the pressure is imposed. In the rest of the boundary $\Gamma - \Gamma_0$, where normal velocity are imposed, homogeneous Neumann boundary conditions must be applied. After solving these two subproblems we project the predicted velocity onto the space of solenoidal vector fields as

$$\mathbf{v}_h^{f,n+1} = \tilde{\mathbf{v}}_h^{f,n+1} - \frac{\Delta t}{\rho^f} \boldsymbol{\nabla} \delta \tilde{p}_{h,proj}^{f,n+1} \qquad \text{in} \quad \Omega_t^f \, , \tag{4.37}$$

$$\mathbf{v}_h^{s,n+1} = \tilde{\mathbf{v}}_h^{s,n+1} - \frac{\Delta t}{\rho^s} \boldsymbol{\nabla} \delta \tilde{p}_{h,proj}^{s,n+1} \qquad \text{in} \quad \Omega_t^s \, , \tag{4.38}$$

and update both the pressure $p_h^{f,n+1}$ and $p_h^{s,n+1}$

$$p_h^{f,n+1} = p_h^{f,n} + \delta p_{h,proj}^{f,n+1} + \delta p_{h,pen}^{f,n+1} \qquad \text{in} \quad \Omega_t^f \, , \tag{4.39}$$

$$p_h^{s,n+1} = p_h^{s,n} + \delta p_{h,proj}^{s,n+1} + \delta p_{h,pen}^{s,n+1} \qquad \text{in} \quad \Omega_t^s \, . \tag{4.40}$$

We remark that the projection pressure field, on the boundary where the velocity field is imposed, has zero normal pressure derivative instead of the normal component of the Cauchy stress. This leads to a wrong representation of the pressure and stress components for an incompressible solid material which tends to reduce the bending displacement field. The iterative penalty correction is meant to reduce this boundary error and provide a stable behavior. In the rest of this Section we use the following FSI penalty-projection algorithm:

1. *set the initial domain* $\Omega_{t=0} = \Omega_0 = \Omega_0^s \cup \Omega_0^f$ *and an initial velocity-pressure-displacement state* $(\mathbf{v}^0, p^0, \mathbf{u}^0)$ *for* $n = 0$*. Also set* $\tilde{\mathbf{v}}^0 = \mathbf{v}^0$*,* $\delta p^0_{h,proj} = 0.$*;*
2. *compute the auxiliary velocity field* $\tilde{\mathbf{v}}_h^{n+1} = (\tilde{\mathbf{v}_h^{f,n+1}}, \tilde{\mathbf{v}}_h^{s,n+1})$ *and the update for the pressure penalty term* $\delta p^{k,n+1}_{h,pen}$ *at time* $n+1$*:*
   (a) *set the value of the penalty iteration* n_k*, initialize the counter* $k = 0$ *and* $\delta p^{0,n+1}_{h,proj} = 0$*;*
   (b) *solve the following iterative equation for* $k = 1, 2, \cdots, n_k$

$$\begin{aligned}
&\frac{\rho}{\Delta t}(\tilde{\mathbf{v}}_h^{k,n+1}, \boldsymbol{\phi}_h)_{\Omega_n} + c(\tilde{\mathbf{v}}^{k,n+1}, \boldsymbol{\phi}_h)_{\Omega_n^f} \\
&+ d(\tilde{\mathbf{v}}^{k,n+1}, \boldsymbol{\phi}_h)_{\Omega_n^s} - r_1(\boldsymbol{\nabla}\cdot\tilde{\mathbf{v}}^{k,n+1})_\pi, \boldsymbol{\nabla}\cdot\boldsymbol{\phi}_h)_{\Omega_n} = \\
&\frac{\rho}{\Delta t}(\tilde{\mathbf{v}}_h^{k,n}, \boldsymbol{\phi}_h)_{\Omega_n} + (p_h^n + \delta p^{k,n+1}_{h,pen} + \delta p^n_{h,proj}, \boldsymbol{\nabla}\cdot\boldsymbol{\phi}_h)_{\Omega_n} \\
&\forall \boldsymbol{\phi}_h \in \mathbf{S}_h(\Omega_n)\,; \\
&\delta p^{k,n+1}_{h,pen} = \delta p^{k-1,n+1}_{h,pen} + r_2(\boldsymbol{\nabla}\cdot\tilde{\mathbf{v}}^{k-1,n+1})\,;
\end{aligned} \tag{4.41}$$

   (c) *if* $\|\tilde{\mathbf{v}}_h^{k+1,n+1} - \tilde{\mathbf{v}}_h^{k,n+1}\| > tol$ *and* $k < n_k$ *return to b);*
3. *compute the projection term* $\delta\tilde{p}^{n+1}_{h,proj}$ *at time* $n+1$ *from the auxiliary velocity field*

$$(\boldsymbol{\nabla}\delta\tilde{p}^{n+1}_{h,proj}, \boldsymbol{\nabla}\boldsymbol{\zeta}) = -\frac{\rho}{\Delta t}(\boldsymbol{\nabla}\cdot\tilde{\mathbf{v}}^{n+1}, \boldsymbol{\zeta}) \quad \forall \boldsymbol{\zeta}_h \in \mathbf{S}_h(\Omega_n)\,; \tag{4.42}$$

4. *compute the velocity and pressure as*

$$\begin{aligned}
\mathbf{v}_h^{n+1} &= \tilde{\mathbf{v}}_h^{n+1} - \frac{\Delta t}{\rho}\boldsymbol{\nabla}\delta\tilde{p}^{n+1}_{h,proj}\,, \\
p_h^{n+1} &= p_h^n + \delta p^{n+1}_{h,proj} + \delta p^{n+1}_{h,pen} \qquad \text{in} \quad \Omega_n\,;
\end{aligned} \tag{4.43}$$

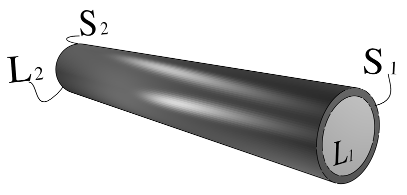
5. *solve for the displacement on the solid domain with*

$$\mathbf{u}_h^{n+1} = \mathbf{u}_h^n + \Delta t\,\mathbf{v}_h^{n+1}\,; \tag{4.44}$$

6. *update the domain of the solid* $\Omega_t^f = \Omega_{n+1}^f$ *by using the solid displacement field;*
7. *return to step 2 and compute the next time step until the final time step is reached.*

In the next Sections after a validation of the proposed model, we use this solution strategy in an arterial like geometries and we couple this solution with the simplified scheme of the cardiovascular system model.

4.4 Test 1. Mass conservation test.



γ_{S_2} , γ_{S_1}	$\mathbf{v} = 0$; $\mathbf{u} = 0$
Γ_w	$\boldsymbol{\sigma} \cdot \mathbf{n} = 0$; $\boldsymbol{\nabla}\mathbf{u} \cdot \mathbf{n} = 0$
Γ_{L_2}	$\mathbf{v} \times \mathbf{n} = 0$; $p_{in} = 1500t$
Γ_{L_1}	$\mathbf{v} \times \mathbf{n} = 0$; $p_{out} = 1300t$
$\rho_f = 1Mg/m^3$	$\mu_f = 1.Kg/ms$
$\rho_s = 1Mg/m^3$	$E = 1.57 \cdot 10^6 Pa$
$k_s = 10^9 Pa$	$t = 0.1, 0.4, 1.0.1.6$
$C_1 = 2 \times 10^5 Pa$	$C_2 = 10^5 Pa$

Figure 4.3: Test 1. Domain overview (left), boundary conditions and physical properties (right).

In the second test we reproduce the benchmark mass conservation test proposed in [25]. The domain overview of the simulation is shown in Figure 4.3 where the fluid flows inside a cylindrical thin structural domain which deforms due to the pressure fluid field. According to the labels shown in Figure 4.3 we impose an homogeneous Dirichlet boundary condition on Γ_{S_2} and Γ_{S_1} while a stress free boundary condition is set in the outer part of the solid domain Γ_w. On the liquid surfaces Γ_{L_2} and Γ_{L_1} the pressure p_{out} and p_{in} are imposed, respectively. In this test we consider incompressible material by using

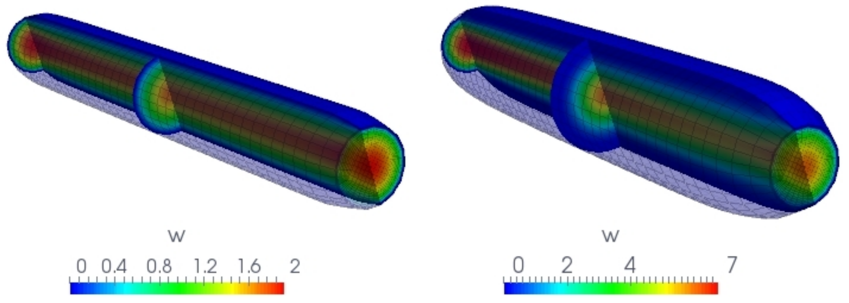


Figure 4.4: Test 1. Domain deformations and vertical component of the velocity field w for $t = 1$ and $t = 1.6$.

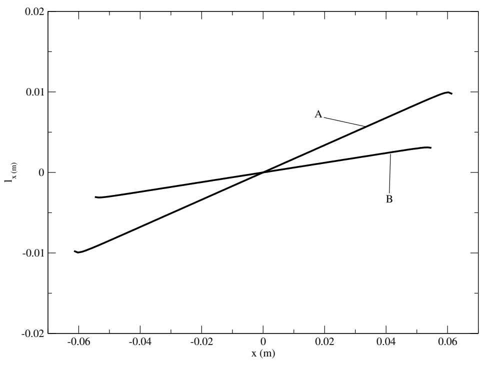


Figure 4.5: Test 1. Transverse displacement l_x for $t = 0.4(A)$ and $1(B)$ with $r_1 = 1000$ ($r_2 = r_1/2$).

pressure			MR model/div-0 model			
s	$P_{in}(Pa)$	$P_{out}(Pa)$	Re	Flow rate		
				$z = 0$	0.5	1
0.1	150	130				
0.4	600	520	31.2/30.1	2.452/2.305	2.452/2.305	2.452/2.305
1.0	1500	1300	121.1/120.8	9.512/9.25	9.512/9.25	9.512/9.25
1.6	24000	20800	361.7/343.9	28.41/26.32	28.41/26.32	28.41/26.32

Table 4.1: Test 1. Mass conservation test.

a divergence free vector field algorithm while the original test was performed with Mooney-Rivlin hyper-elastic material model. Further information about the geometrical details and the material constitutive model of the simulation can be found in [25]. At the inlet and outlet we assume constant inflow and outflow boundary conditions as

$$\mathbf{v} \times n = 0 \qquad p_{in} = 1500 \times t \qquad p_{out} = 1300 \times t \,, \tag{4.45}$$

where the parameter t is set to 0.1, 0.4, 1. and 1.6. During the inflating process the fluid must fill the domain defined by the solid walls and therefore more liquid should enter through the inlet section. In order to have exact mass conservation through different sections one needs steady or quasi steady solutions. We compute solutions by using standard time dependent algorithm and reach the steady solution with fixed pressure defined by the parameter t. We set the penalty parameter r_1 to $1/dt$, where dt is the constant time step. This choice is suggested by the considerations reported in [73]. In Figure 4.4 for $t = 1$ (on the left) and $t = 1.6$ (on the right) we show the deformed domain and the vertical component of the velocity field w through the inlet, middle and outlet section. The transverse displacement along the x-direction l_x in the central section are reported in Figure 4.5 for $r_1 = 1000$ ($r_2 = 2r_1$). The parameter t is set $0.4(A)$ and $1.0(B)$. We remark that the coupled and the penalty results are very close already for $r_1 = 100$. The numerical results in Table 4.1 are reported for the Mooney-Rivlin hyper-elastic material model and for the zero-divergence algorithm with penalty parameter set to $r_1 = 1/dt = 1000$.

4.5 Test 2. Blood vessel

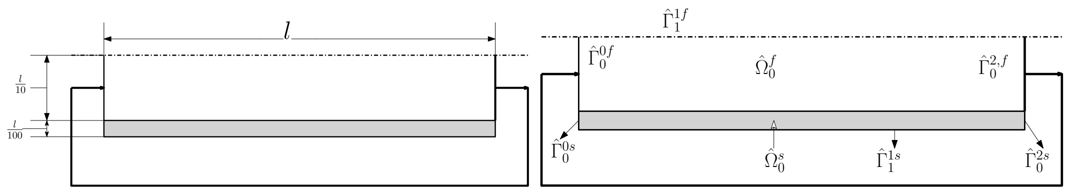


Figure 4.6: Test 2. Geometry (left) and boundary condition (right).

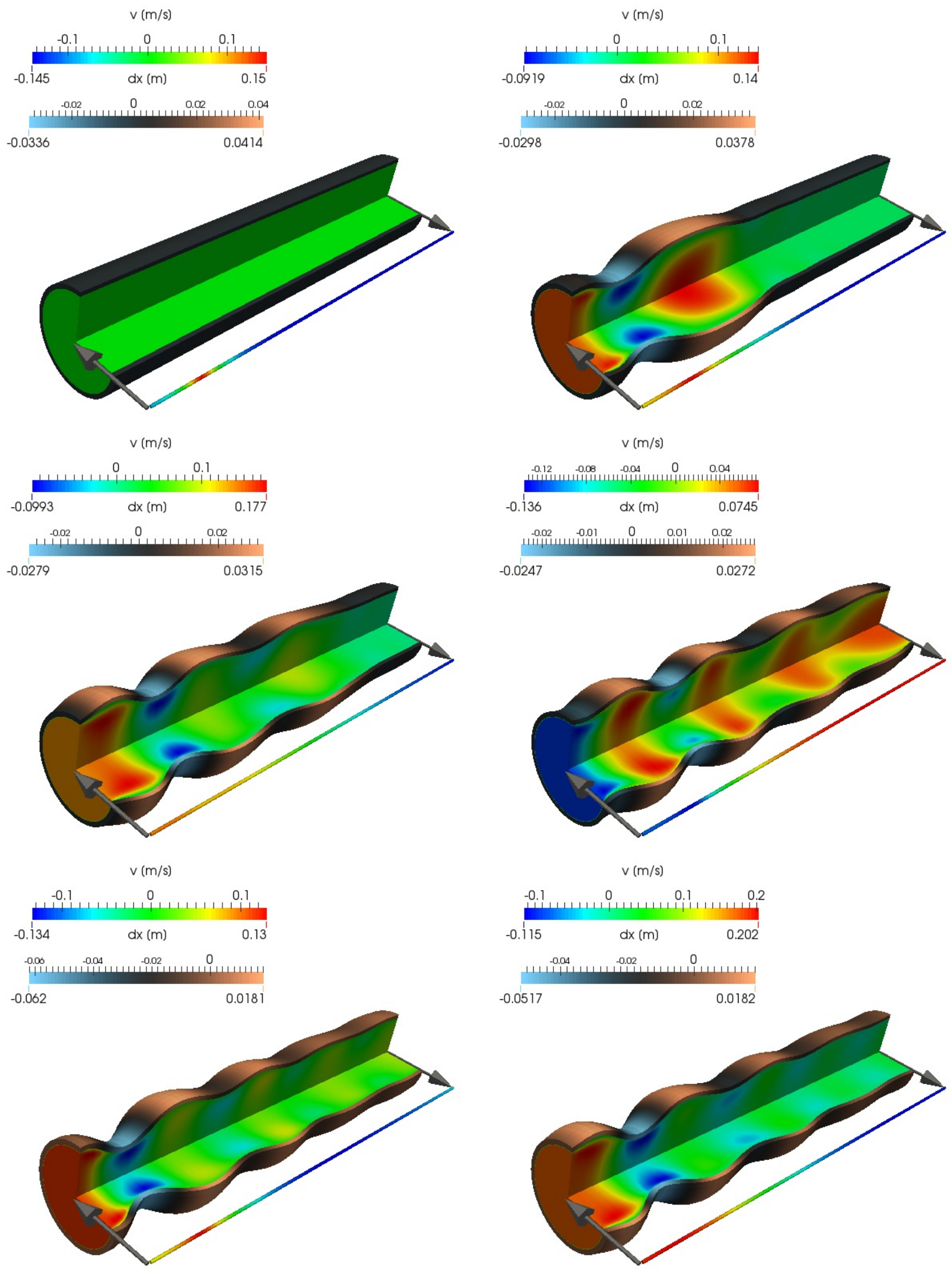


Figure 4.7: Test 2. Overview displacement and velocity field at $t = 0$, 2, 4, 6, 8 and 10s.

Physical parameters	**Fluid**	**solid**	Units
Poisson module	–	0.5	–
Young module	–	400000	Pa
Viscosity	1	–	Pa · s
Density	1000	1000	kg m^{-3}

Table 4.2: Test 2. Physical parameters of the numerical simulation.

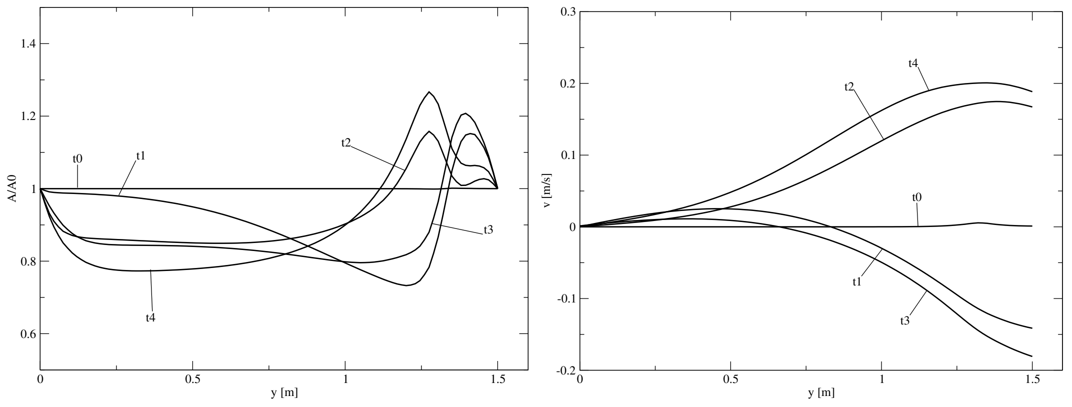


Figure 4.8: Test 2. Cross section dimension (left) and velocity field (right) in the mono-dimensional system at $t = 0$ (t_0), 2 (t_1), 4 (t_2), 6 (t_3) and $8s$ (t_4).

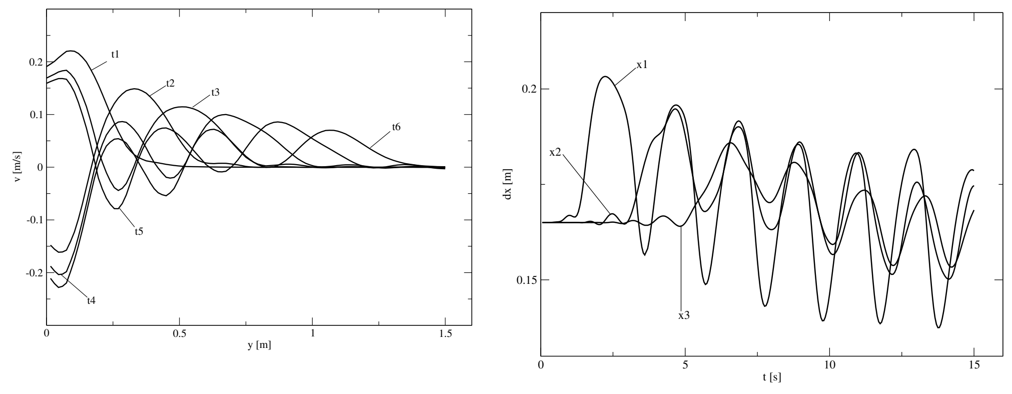


Figure 4.9: Test 2. Axial velocity in fluid domain at $t = 2$ (t_1), 4 (t_2), 6 (t_3), 8 (t_4), 10 (t_5) and $12s$ (t_6) (left) and displacement field over time in solid domain in $x_1 = (0.165, 0.375)[m]$, $x_2 = (0.165, 0.75)[m]$ and $x_3 = (0.165, 1.125)[m]$ (right).

In this test we consider a pipe type system coupled with a mono-dimensional circuit in which a component moves a certain mass and inject the flow periodically. This example is performed to test the stability and the robustness of the algorithm in a system in which large deformation in the structure and an highly transient velocity field are present. In Figure 4.5 the geometry of the system is shown. We consider an axial symmetric system with $l = 1.5m$, radius $r = 0.15m$ and thickness $s = 0.015m$. With reference on the labels shown in Figure 4.5 the boundary conditions imposed for the fluid ($\hat{\Omega}^f$) are

$$\begin{cases} u_1 = u_m & \text{on } \hat{\Gamma}_0^{0,f} \,, \\ u_2 = 0 & \text{on } \hat{\Gamma}_0^{0,f} \,, \\ \frac{\partial u_i}{\partial x_i} = p_i^* & \text{on } \hat{\Gamma}_0^{2,f} \quad i = \{1,2\} \,, \\ \frac{\partial u_1}{\partial x_2} = 0 & \text{on } \hat{\Gamma}_1^{1,f} \,, \\ u_2 = 0, & \text{on } \hat{\Gamma}_1^{1,f} \,, \end{cases} \tag{4.46}$$

where u_m is the velocity coming from the mono-dimensional system. For the solid domain ($\hat{\Omega}^s$) the boundary conditions becomes

$$\begin{cases} u_1 = 0 & \text{on } \hat{\Gamma}_0^{0,s} \cup \hat{\Gamma}_0^{2,s} \,, \\ \frac{\partial u_2}{\partial x_2} = 0, & \text{on } \hat{\Gamma}_0^{0,s} \cup \hat{\Gamma}_0^{2,s} \cup \hat{\Gamma}_1^{1,s} \quad i = \{1,2\} \,, \\ \frac{\partial u_1}{\partial x_2} = 0 & \text{on } \hat{\Gamma}_1^{1,s} \,. \end{cases} \tag{4.47}$$

The fluid physical parameters are $\mu^f = 2\,\mathrm{Pa \cdot s}$ and $\rho^f = 1000\,\mathrm{kg/m^3}$, whereas for the solid we have $\rho^s = 1000\,\mathrm{kg/m^3}$, $E = 400000\,\mathrm{Pa}$ and $\nu = 0.5$. These parameters are summarized in Table 4.2.
In Figures 4.7 the fluid velocity field and the solid displacement are shown at $t = 0$, 2, 4, 6, 8 and 10 sec. We can clearly see the propagation and the attenuation of the displacement wave through the three-dimensional domain. In Figure 4.8 on the left the cross section A is plotted against the spatial coordinate and the mono-dimensional velocity at different time $t = 0$, 2, 4, 6 and $8s$ is shown on the right. We can remark that the ratio A/A_0 never reaches values lower than 0.7, and the velocity field oscillates inside the interval $[-0.2, 0.2]\, m/s$. In Figure 4.9 on the left the axial fluid velocity ($x = 0.25m$) is plotted at different time $t = 2$ (t_1), 4 (t_2), 6 (t_3), 8 (t_4), 10 (t_5) and $12s$

(t_6). In a similar way on the right of Figure 4.9 the displacement field is shown over time at different points of the solid domain. The considered points are set at $(0.165, 0.375)$, $(0.165, 0.75)$ and $(0.165, 1.125)$ and labeled by x_1, x_2 and x_3, respectively. From the first plot we can see the time variation of the inlet velocity and his space propagation. In the plot on the right we can observe the delay of the deformation wave at different points and its dumping due to the viscosity of the fluid.

4.6 Test 3. Stable Aneurysm

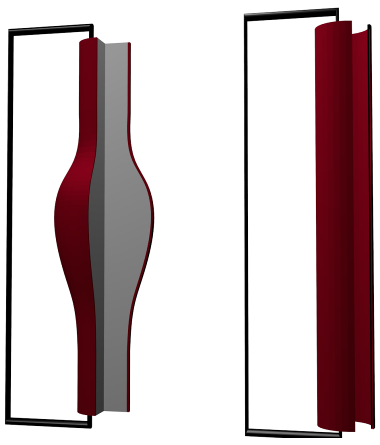


Figure 4.10: Test 3. Configurations of the aneurysm (left) and healthy aorta (right) together with the mono-dimensional domain.

A multiscale, axisymmetric, multidimensional simulation is performed to investigate the behavior of a stable aneurysm inserted in the whole cardiovascular loop. We couple the inlet and the outlet surfaces of the domain with the outlet and the inlet boundary of a mono-dimensional system as shown in Figure 4.10. The inlet velocity of the multidimensional domain is the one coming from the the simplified system and the out flow of the complex domain is imposed in the inlet region of the 1D system. This coupling represents the first step for a detailed multiscale approach that could lead to obtain realistic simulations of the whole cardiovascular loop with a reasonable computational cost. The same simulation are performed considering a non-deformed blood vessel in order to

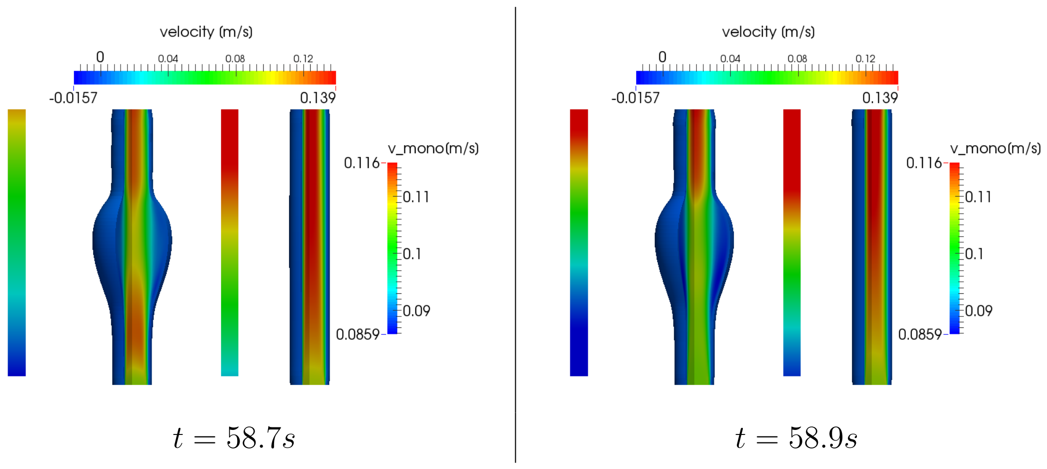


Figure 4.11: Test 3. Velocity along the y-axis. Aneurysm (on the left) and healthy aorta (on the right) at $t = 58.7$ and $58.9s$.

compare the obtained results. The Young modulus, as is common in literature for this type of problems, is set to $10^3 Pa$ and the Poisson ratio to 0.4. We also take into account a simplified effect of hearth contraction by adding a periodic momentum source in the simplified system in the form $H sin(\pi t)^4$. The multi-scale approach allows the study of the effects of the presence of the aneurysm on a simplified closed loop. When the flow becomes periodic stabilized a period of $1\,s$ is observed. In Figure 4.11 the velocity field in the axial direction, for a normal and a deformed domain, is shown at different time steps. In particular we can observe that in the deformed case the inlet velocity of the domains undergoes to large variation, this effect could not be seen without using a multidimensional approach. In Figure 4.12, the pressure field in the three-dimensional domain and the area distribution in the mono-dimensional pipe are shown at different time steps. In Figure 4.13 one can appreciate the increasing of axial stresses in the in the presence of an aneurysm at different time steps. In the healthy aorta interface stresses are negligible and in the presence of the aneurysm the stresses increase near the inlet and the outlet region of the aneurysm itself possibly causing the rupture of the blood vessel. In Figure 4.14 the pressure field along the axis at different time step. We plot in dashed line the values for the healthy aorta and in solid line the values for the aneurysm. We can notice that several differences can be observed between

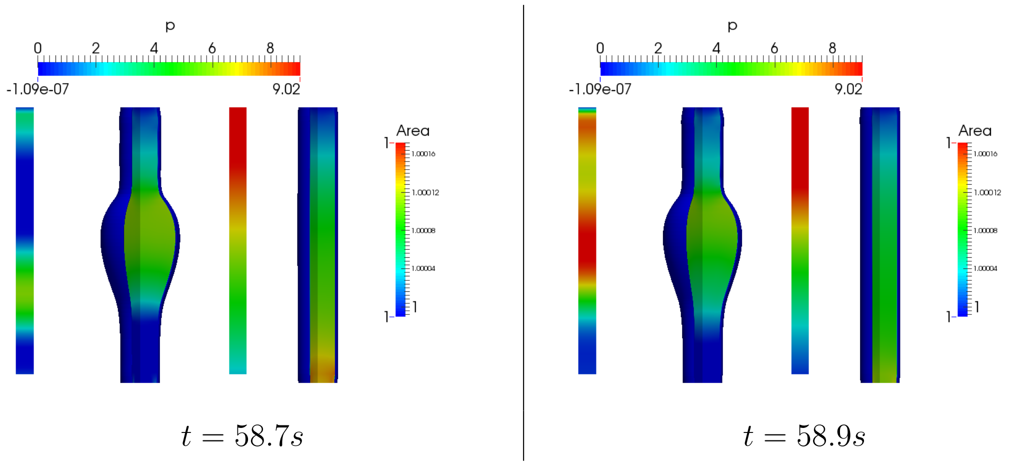


Figure 4.12: Test 3. Pressure in the 3D domain and area variation on the 1D domain. Aneurysm (on the left) and healthy aorta (on the right) at $t = 58.7$ and $58.9s$.

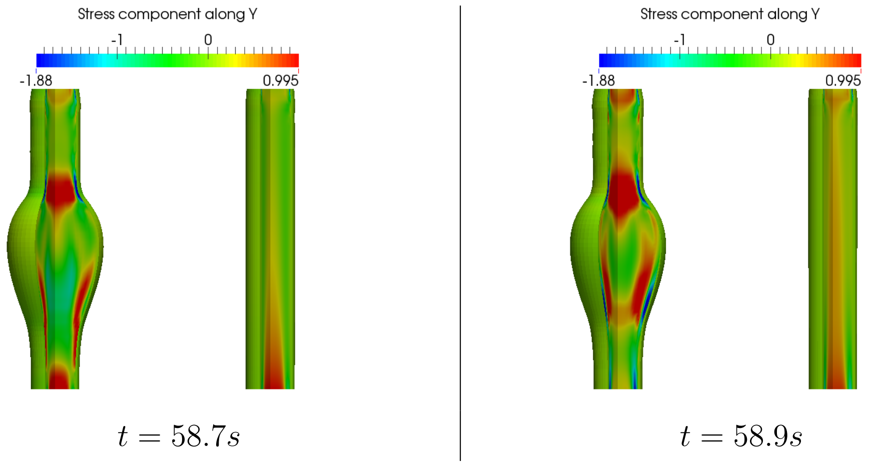


Figure 4.13: Test 3. Stress component in the 3D domain. Aneurysm (on the left) and healthy aorta (on the right) at $t = 58.7$ and $58.9s$.

the two cases. Such differences are mostly due to the recirculation phenomena that occurs in the presence of the aneurysm. In Figure 4.15 the radial displacement, at $y = 0.07\,m$ is shown together with the axial velocity. We plot in dashed style the values for the healthy aorta and in solid line the values for the aneurysm. One can notice that in the presence aneurysm displacements becomes greater that the ones evaluated in a straight blood vessel. The large deformation that occurs in the presence of the aneurysm reduces the overall

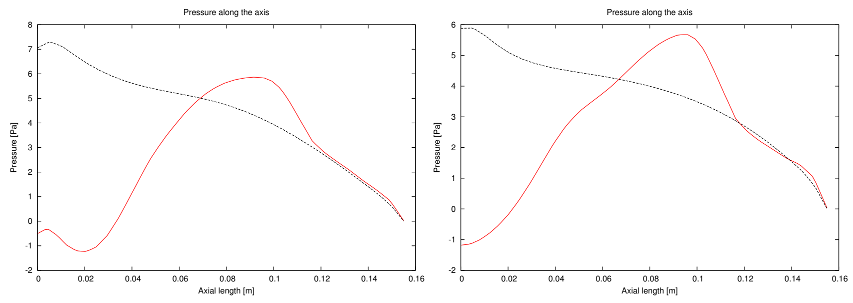


Figure 4.14: Test 3. Pressure along the axis for the 3D domain. presence of aneurysm (in solid line), and healthy aorta (in dashed line) at $t = 58.7$ and $58.9s$.

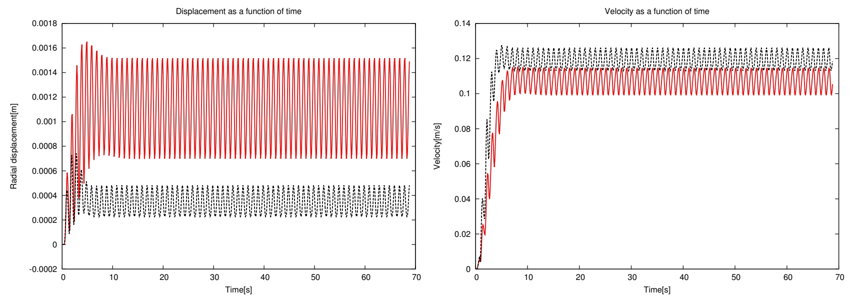


Figure 4.15: Evolution of the displacement (on the left) and of the velocity (on the right). in the presence of an aneurysm (in solid line) and of an healthy aorta (in dashed line).

magnitude of velocity field and the characteristic frequency of the structure, as we can see in the right part of Figure 4.15.

4.7 Test 4. Growing Aneurysm

In this test we simulate the growing of an aneurysm and we show the evolution of the velocity field. The formation of the aneurysm is caused by a progressive, localized loss of elasticity in the aorta wall. The mono-dimensional loop is treated with the model described previously. The boundary and the coupling conditions are the same imposed in the previous Section for the healthy aorta in order to compare the velocity profile obtained. A progressive localized loss of elasticity is simulated and the Young modulus is progressively decreased

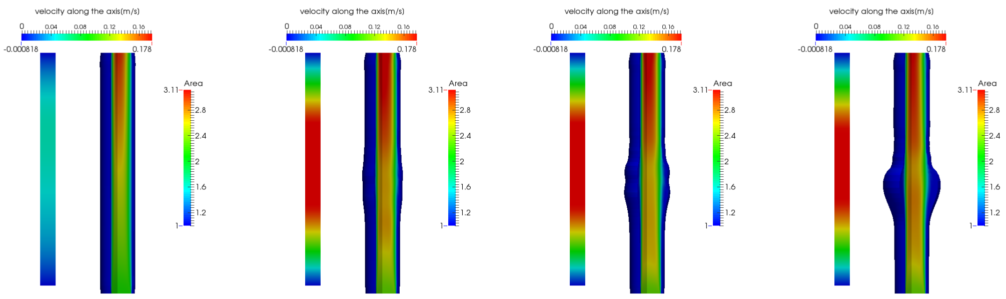


Figure 4.16: Test 4. Formation of the aneurysm. Velocity along the axis in the 3D domain and cross section variation in the 1D domain after time $t = 5\,s$ $t = 35\,s$ $t = 65\,s$ $t = 80\,s$.

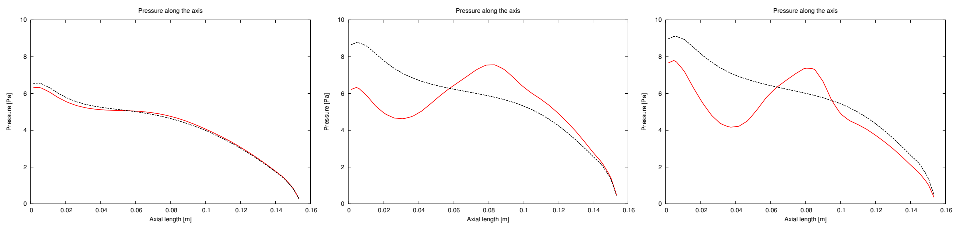


Figure 4.17: Test 4. Formation of the aneurysm. Pressure as a function of the axial coordinate for 3D domain in the case of aneurysm formation (solid style) and an healthy aorta (dashed style) at $t = 5$, 35 and $80\,s$.

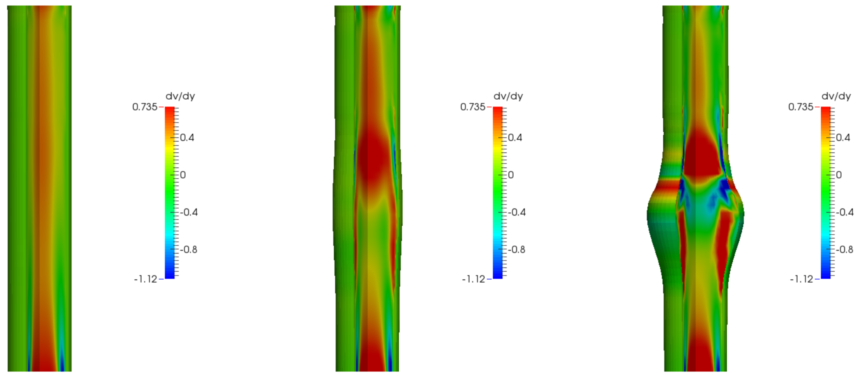


Figure 4.18: Test 4. Formation of the aneurysm. $\partial v/\partial y$ for 3D domain after $t = 5$, 35 and $80\,s$.

during time using the following relation:

$$\begin{aligned} E = E_0(1 - 80t(x - 0.04)(0.1 - x))^4 \quad & \forall x \in [0.04; 0.1]\,, \\ E = E_0 \qquad & \forall x \notin [0.04; 0.1]\,. \end{aligned} \tag{4.48}$$

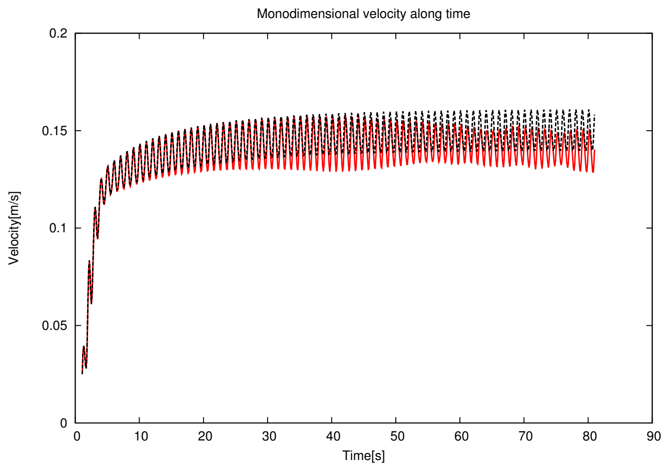


Figure 4.19: Test 4. Formation of the aneurysm. Mono dimensional velocity as a function of time at the inlet of the 1D mono-dimensional domain, in the case of aneurysm formation (solid style) and healthy aorta (dashed style).

The simulation is performed with a time range of 400 s in which the aneurysm is formed and reaches the steady state. As in the previous example a multiscale approach is used in order to study the effects of an aneurysm developing on a connected loop. This class of simulation is used to study the evolution of an aneurysm and investigate the behavior of the whole circulatory system when an aneurysm is growing in a certain blood vessel. In Figure 4.16 the velocity profile and the cross section dimension of the multidimensional and the mono-dimensional domain are shown respectively at different time steps. We can notice that the velocity field In Figure 4.17 we show the pressure distribution along the axis of the multidimensional domain at different time steps. In Figure 4.18 the axial stresses are shown at different time steps. In Figure 4.19 we can observe the variation of the inlet velocity in the mono-dimensional domain in the normal and the deformed case. We can notice that the average velocity field is not constant over time, this effect is a consequence of the huge deformation multidimensional domain. We remark that this feedback could not be taken into account without considering the simplified coupled domain.

CHAPTER 5

Multiphase Fluid Structure Interaction

In this Chapter we study the deformation of solid structures induced by a two-phase flow by coupling, through the computational platform SALOME, a FSI problem with a multiphase interface tracking problem. We use the FSI solver introduced in the previous Chapter, while the two-phase interface advection and reconstruction is computed in the framework of a Volume of Fluid (VOF) method. An unstructured computational grid and a fine Cartesian mesh are used for the FSI and the VOF problem, respectively. The interaction between the two different grids is obtained by projecting the velocity field into the Cartesian grid and the Color function into the unstructured grid. This operation is performed with the MEDmem libraries included in the Salome platform. Some numerical results are then reported in order to show the robustness and stability of this numerical approach.

5.1 Introduction

The interaction between solid and fluid may give rise to very complex phenomena driven by mutual dependence between the fluid and deformable solid parts. In this Chapter we extend the study of the interaction between a solid domain and a fluid to a multiphase flow with the inclusion of thermal effects. The study of two-phase flows is itself very challenging. In recent years this interest has inspired a great number of numerical algorithms for the solution of the two-phase problems involving efficient solvers of the incompressible Navier-Stokes equations, stable and accurate techniques for the description of the interface evolution, and surface tension models for the representation of the capillary force. These algorithms can be divided mainly into two different groups depending on the grid type. In the first case moving grids are used [75, 76]. The interface between the two phases always coincides with the cell boundaries, which are continuously advected by following the flow motion. This representation allows a precise modeling of the fluid property discontinuities, and also a correct location of the capillary force, which is a singular term in the Navier-Stokes equations. Despite these good properties, moving grid methods can be used only in simulations with small changes in topology, since severe grid stretching brings inevitably some loss of accuracy in the discretization of the Navier-Stokes equations. Furthermore, since breaking or merging of drops and bubbles are almost impossible, a continuous re-meshing of the computational domain is required. In the second group fixed grids are used. The interface does not coincide necessary with the cell boundaries, and a numerical algorithm is then required to reconstruct its motion in the computational domain. The fluid properties are evaluated within the interface cells as an average between the properties of each phase. The capillary force, which is physically located only on the interface, is redistributed inside the cells cut by the interface. With this representation, severe stretching and deformation of the interface can be tracked. For these reasons fixed grid methods are widely used. Fixed-grid methods for two-phase flows can be either Eulerian or Lagrangian. Eulerian front capturing schemes include volume-of-fluid (VOF), level set and phase field methods [77, 78, 79]. The multiphase flow motion satisfies the Navier-Stokes equations and it should be determined while taking into account the discontinuities of the fluid properties across the interface. This discontinuity, together with the high Reynolds numbers, may induce several numerical instabilities. The solution of the equations is then achieved only if the problem is well modeled, and if efficient and accurate solvers are used. In this Chapter we study the behavior of a solid structure that interacts with a multiphase flow. We consider a Fluid Structure Interaction problem in which the fluid part is composed of a multiphase flow, the interface between these two phases is moved with a VOF algorithm[61, 80, 81]. The VOF algorithm is implemented in two steps: in the first one the color function is moved with an explicit Lagrangian advection method by using the fluid velocity which has been computed by the FSI solver. In the second step the interface is reconstructed based on the color function gradient and on the conservation of volume constraint. For detailed information on this method the interested reader can consult [82, 83]. In this work we use a Piecewise Linear Interface Calculation (PLIC) reconstruction scheme together with an ELVIRA method for the computation of the interface normals, [82, 84]. The coupling between a FSI and a VOF solver allows to increase dramatically the range of problems that can be studied. In this work we use this coupling for studying the thermal stresses of a fluid-structure interaction system. In order to couple the two problems many strategies can be employed. As in the previous Chapters, we use the computational platform SALOME where two-phase flow and FSI code have been developed while a dedicated MEDmem libraries is used to exchange data between a Cartesian structered and a non structured mesh [7]. In the next Section the mathematical model for a Multiphase Fluid Structure Interaction problem is presented. In Section 5.3 we focus the attention at the interfaces that are needed to couple the FSI code, used in the previous Chapter, to a VOF module. In the last Section some numerical results are shown.

5.2 Mathematical Model

In this Section the mathematical model of the Multiphase Fluid Structure Interaction (MFSI) problem is presented. Before considering the full MFSI

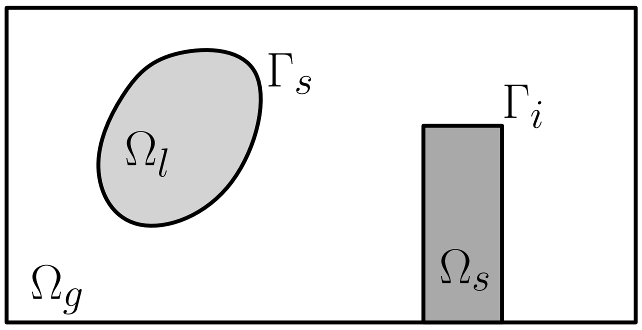


Figure 5.1: Computational domain. The fluid in Ω_g and Ω_l are the reference primary and secondary fluid phases, respectively. The boundary between the two fluid phases is denoted by Γ_s. Ω_s marks the solid region and the interface between the solid and the multiphase fluid region is labeled with Γ_i.

problem we have to introduce a two-phase system and the mathematical model that governs its evolution. Let us consider a domain show in Figure 5.1, where Ω marks the whole domain. Ω_l is the portion of the domain occupied by the secondary fluid phase, while $\Omega_g \subset \Omega$ marks the configuration of the primary phase. The boundary between the two immiscible fluids phases is denoted by Γ_s and its topology can vary during the evolution of the system since each sub-domain evolves in time. The solid domain is labeled with Ω_s and the interface between the fluid phases and the solid region is marked with Γ_i. As explained in the Section 4.2.1 the balance equations form a classic Fluid Structure Interaction problem are

$$\rho^f \frac{\partial \mathbf{v}^f}{\partial t}\bigg|_{\tilde{\mathcal{A}}} + \rho^f \left(\mathbf{v}^f - \mathbf{w}^f\right) \cdot \boldsymbol{\nabla} \mathbf{v}^f - \boldsymbol{\nabla} \cdot \boldsymbol{\sigma}^f = \mathbf{f} \quad \text{in} \quad (0,T) \times \Omega_t^f \cup \Omega_t^g \,, \tag{5.1}$$

$$\rho^s \frac{\partial \mathbf{v}^s}{\partial t} - \boldsymbol{\nabla} \cdot \boldsymbol{\sigma}^s(\mathbf{u}^s) = \mathbf{0} \quad \text{in} \quad (0,T) \times \Omega_t^s \,, \tag{5.2}$$

$$\boldsymbol{\nabla} \cdot \mathbf{v} = \mathbf{0} \quad \text{in} \quad (0,T) \times \Omega_t \,, \tag{5.3}$$

$$\mathbf{v}^f|_{t=0} = \mathbf{v}_0 \quad \text{in} \quad \hat{\Omega}_0 \,, \tag{5.4}$$

$$\mathbf{v}|_{\Gamma_{t,i}} = \mathbf{g}^f \quad \text{in} \quad (0,T) \,, \tag{5.5}$$

$$\boldsymbol{\sigma} \cdot \mathbf{n}|_{\Gamma_{t,i}} = \mathbf{h}^f \quad \text{in} \quad (0,T) \,. \tag{5.6}$$

In order to consider a multiphase fluid that interact with a structural domain, we have only to modify the fluid momentum balance equation (5.1). while the rest of the system is treated as described in the Section 4.2.1. The momentum balance equations system is formally written as in single phase formulation

$$\begin{aligned}&\rho^f \frac{\partial \mathbf{v}^f}{\partial t}\bigg|_{\tilde{\mathcal{A}}} + \rho^f \left(\mathbf{v}^f - \mathbf{w}^f\right) \cdot \boldsymbol{\nabla} \mathbf{v}^f - \nabla \cdot (\mu^f (\nabla \mathbf{v^f} + (\nabla \mathbf{v^f}^\intercal)) + \nabla p = \mathbf{f}\,, \\ &\nabla \cdot \mathbf{v^f} = 0\,, \qquad \mathbf{x} \in \Omega, t \in [0,T]\,,\end{aligned} \tag{5.7}$$

but the difference, between the single phase case, is hidden in the definition of the density ρ, the viscosity μ and the force $\mathbf{f}$. If we denote with l the properties of the reference phase and with g the values associated to the secondary phase, we define the physical properties ρ^f and μ^f as

$$\rho^f = \rho^l \chi + \rho^g (1-\chi)\,, \tag{5.8}$$

$$\mu^f = \mu^l \chi + \mu^g (1-\chi)\,, \tag{5.9}$$

where χ is the characteristic function or indicator function. This function describes the distribution of the two phases in the domain. It is equal to 1 in the secondary phase and 0 in the primary phase. We note that the function is discontinuous on the interface Γ_s. We can define χ as

$$\chi(\mathbf{x},t) = \int_{\Omega_l(t)} \delta(\mathbf{x}' - \mathbf{x})\, d\mathbf{x}' \quad \forall \mathbf{x} \in \Omega\,. \tag{5.10}$$

The indicator function is therefore a multidimensional Heaviside function that changes value on Γ_s. We can also write that

$$\nabla \chi = -\int_{\Gamma_s} \delta(\mathbf{x}' - \mathbf{x}) \mathbf{n}'\, dS' = -\mathbf{n} \int_{\Gamma_s} \delta(\mathbf{x}' - \mathbf{x})\, dS' = -\mathbf{n} \delta_s(\mathbf{x})\,, \tag{5.11}$$

where $\delta_s(\mathbf{x})$ is the Dirac delta function that is discontinuous on Γ_s. Under the hypotheses of immiscible fluids with no phase change, the characteristic function behaves like a passive scalar and is purely transported by the velocity field, following the simple advection equation

$$\frac{\partial \chi}{\partial t} + (\mathbf{v^f} \cdot \nabla)\chi = 0\,, \qquad \text{in } \Omega \times [0,T]\,. \tag{5.12}$$

The force term presents a sensible difference with respect to the single phase formulation, where it indicates only body forces such as gravity or electromagnetic fields. Here we must take into account also the surface tension, that is modeled as a force applied only on the interface

$$\mathbf{f}_s(\mathbf{x}) = \int_{\Gamma_s} \sigma \kappa\, \mathbf{n}\, \delta_s(\mathbf{x})\, dS\,, \tag{5.13}$$

where σ is a constant surface tension coefficient, as we do not consider temperature gradients or varying concentration of surfactants, κ the sum of the principal curvatures (in our convention $\kappa < 0$ for a liquid drop), $\mathbf{n}$ the unit external normal to Γ_s and $\mathbf{x}_s$ a point on Γ_s.

5.2.1 Volume-of-Fluid (VOF) method for interface capturing

The Volume-of-Fluid method is one of the most popular techniques adopted to model numerically an interface of separation between two phases. In this approach we define a color function C on each of the cells that are part of the computational domain. The value of C is taken as the integral of the characteristic function χ on the cell

$$C_i(t) = \frac{1}{\text{meas}(\Omega_i)} \int_{\Omega_i} \chi(\mathbf{x}, t)\, dV \,, \tag{5.14}$$

where Ω_i is one of the cells of the partition $\mathbb{T}_h$ of Ω, and $\text{meas}(\Omega_i) = \int_{\Omega_i} dV$. It is easy to see that

$$\begin{cases} C_i(t) = 1 & \text{if } \Omega_i \subset \Omega_l \,, \\ C_i(t) = 0 & \text{if } \Omega_i \subset \Omega_g \,, \\ 0 < C_i(t) < 1 & \text{if } \Omega_i \cap \Gamma_s \neq \emptyset \,. \end{cases}$$

The interface of separation is located on the mixed cells. Once introduced the color function, we can define the physical properties at cell level

$$\rho_i = \rho^l C_i + \rho^g (1 - C_i) \,, \tag{5.15}$$

$$\mu_i = \mu^l C_i + \mu^g (1 - C_i) \,. \tag{5.16}$$

In each mixed cell, the interface is represented by a single segment that is oriented and positioned in order to approximate as well as possible the real interface. The overall reconstruction of the interface is therefore piece-wise linear. In order to advance in time the interface reconstruction, we integrate (5.12) on the cell Ω_i to get

$$\text{meas}(\Omega_i) \frac{\partial C_i}{\partial t} + \int_{\Gamma_i} \chi \mathbf{v}^{\mathbf{f}} \cdot \mathbf{n} dS = 0 \,, \tag{5.17}$$

where the integral is extended to the boundary Γ_i of Ω_i. Since this equation is still discontinuous, it can not be integrated with standard partial differential equation methods, that tend to diffuse the interface, and a geometrical approach is usually preferred. In summary, a VOF advection algorithm will require two steps: a reconstruction and an advection one. In the first step a segment, that reproduces the interface, is placed into every mixing cell, the normal to the segment is determined from the color function distribution and the position is evaluated imposing the constraint that the underlying volume is equal to the color function value in the cell. A great number of schemes has been proposed, often based on finite differences. Once we have a reconstruction segment in each cell, the interface is advanced in time using (5.17) and the color function is updated in each cell. Since the volume in each cell is constrained to be the integral of the indicator function in that cell, the VOF method shows excellent mass conservation properties. We will now analyze in detail each step.

5.2.2 Interface reconstruction

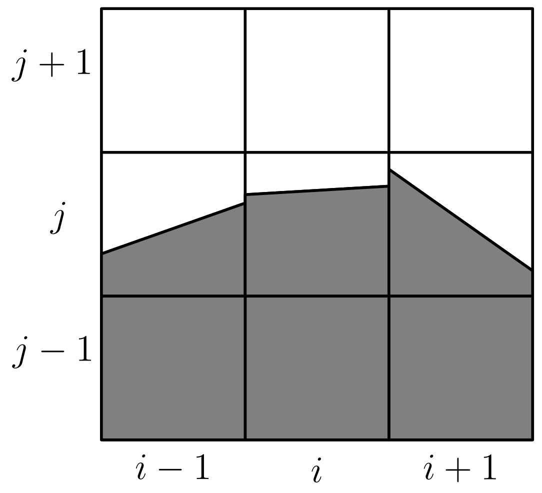


Figure 5.2: Cell stencil used for the reconstruction.

The most simple reconstruction techniques available for VOF method is called Single Line Interface Calculation (SLIC) and only provided segments parallel to one of the edges of the cell boundary. In two-dimensions, this leads

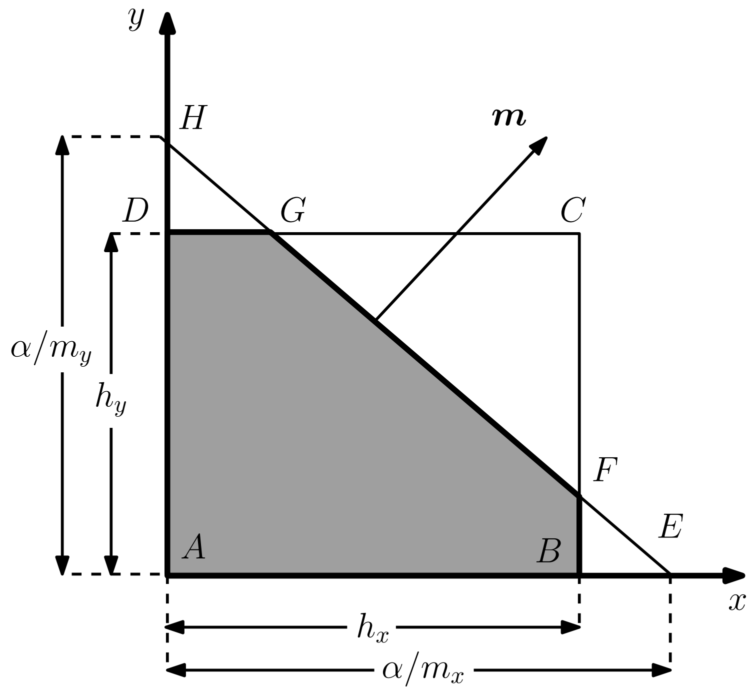


Figure 5.3: The reference phase occupies the area of the pentagon $ABFGD$.

to an equation for the segment in the form

$$x = \alpha_1 \,, \qquad \text{or} \qquad y = \alpha_2 \,, \tag{5.18}$$

where α_1 and α_2 are set by the volume fraction conservation. Recent techniques allow the segment to be freely oriented inside the cell and are therefore know as Piecewise Linear Interface Calculation (PLIC) methods. Sticking to the two-dimensional case, the segment line can be represented by

$$m_x x + m_y y = \alpha \,, \tag{5.19}$$

where $\mathbf{m} = (m_x, m_y)$ is a vector normal to the reconstruction segment. The volume conservation constraint sets the value of α. It is important to note that this imposition leads to an interface reconstruction that is not continuous across cell boundaries. Some of the most used techniques for the determination of $\mathbf{m}$ rely on a discretized derivation of the color function distribution. The Parker-Youngs method is a more sophisticated method and is here illustrate for a two dimensions domain subdivided into a Cartesian grid where each cell is identified by the couple of indices (i, j) of integer values. Let us consider a 3×3 stencil of cells around the cell $(i,\, j)$, as shown in Figure 5.2. The normal $\mathbf{m}$ is first computed on the four cell vertices. For example, in the upper right corner, identified by the indices $(i + \frac{1}{2},\, j + \frac{1}{2})$, we get

$$\begin{aligned} m_{x,\, i+\frac{1}{2}, j+\frac{1}{2}} &= -\frac{1}{2h_x}(C_{i+1,j} - C_{i,j} + C_{i+1,j+1} - C_{i,j+1})\,, \\ m_{y,\, i+\frac{1}{2}, j+\frac{1}{2}} &= -\frac{1}{2h_y}(C_{i,j+1} - C_{i,j} + C_{i+1,j+1} - C_{i+1,j})\,, \end{aligned} \tag{5.20}$$

where h_x and h_y are the grid steps in the two coordinate directions. When $h_x = h_y$ the normal at the cell center is obtained by taking the average of the four vertex values

$$\mathbf{m}_{ij} = \frac{1}{4}(\mathbf{m}_{i+\frac{1}{2}, j-\frac{1}{2}} + \mathbf{m}_{i-\frac{1}{2}, j-\frac{1}{2}} + \mathbf{m}_{i+\frac{1}{2}, j+\frac{1}{2}} + \mathbf{m}_{i-\frac{1}{2}, j+\frac{1}{2}})\,, \tag{5.21}$$

while the averaging changes slightly when $h_x \neq h_y$. This method is quite simple and has the great advantage of being easily extended to three-dimensional domains. Furthermore, when the resolution is low it shows better performances then many other methods which are more complex [85]. The ELVIRA method (Efficient Least-squares Volume-of-fluid Interface Reconstruction Algorithm [83]) uses the same stencil of cells of the previous method. We can calculate a discretized value of the height y on each of the columns of the stencil as the sum of the volume fractions on that column, $h_x y_i = \sum_{k=-1}^{1} C_{i,\, j+k} h_x h_y$. If we consider the approximation $y = m_x\, x + \alpha$ on the central cell $(i,\, j)$, we can choose for m_x between the three values obtained with a backward (m_{xb}), centered (m_{xc}) or forward (m_{xf}) finite difference method given by

$$m_{xc} = \frac{1}{2h_x}(y_{i+1} - y_{i-1}) = \frac{1}{2h_x} \sum_{k=-1}^{1} (C_{i+1,j+k} - C_{i-1,j+k})\,, \tag{5.22a}$$

$$m_{xf} = \frac{1}{h_x}(y_{i+1} - y_i) = \frac{1}{h_x} \sum_{k=-1}^{1} (C_{i+1,j+k} - C_{i,j+k})\,, \tag{5.22b}$$

$$m_{xb} = \frac{1}{h_x}(y_i - y_{i-1}) = \frac{1}{h_x} \sum_{k=-1}^{1} (C_{i,j+k} - C_{i-1,j+k})\,. \tag{5.22c}$$

We can also repeat the argument for the horizontal direction and approximate the interface with the line $x = m_y\, y + \alpha$. In this case we get for m_y

$$m_{yc} = \frac{1}{2h_y}(x_{j+1} - x_{j-1}) = \frac{1}{2h_y}\sum_{k=-1}^{1}(C_{i+k,j+1} - C_{i+k,j-1})\,, \tag{5.22d}$$

$$m_{yf} = \frac{1}{h_y}(x_{j+1} - x_{j}) = \frac{1}{h_y}\sum_{k=-1}^{1}(C_{i+k,j+1} - C_{i+k,j})\,, \tag{5.22e}$$

$$m_{yb} = \frac{1}{h_y}(x_{j} - x_{j-1}) = \frac{1}{h_y}\sum_{k=-1}^{1}(C_{i+k,j} - C_{i+k,j-1})\,. \tag{5.22f}$$

We consider each of the six cases (5.22) and use them to reconstruct the line on the whole 3×3 stencil of cells defining in this way an approximated volume fraction distribution $\widetilde{C}$. We consider the discretized error E in L_2 between the real data C and the approximated values $\widetilde{C}$

$$E(\widetilde{m}) = \left(\sum_{k=-1}^{1}\sum_{l=-1}^{1}(\widetilde{C}_{i+k,j+l}(\widetilde{m}) - C_{i+k,j+l})^2\right)^{\frac{1}{2}}\,, \tag{5.23}$$

where $\widetilde{m}$ is one of the coefficients defined in (5.22). The value of $\widetilde{m}$ that minimizes E is chosen as the normal of the segment. This technique reproduces exactly any linear interface [83] and shows better convergence rates in basic VOF tests when the resolution is not too small. The three-dimensional equivalent of this algorithm requires however a stencil of cells that extends to 5 cells in each direction, in order to reproduce exactly any planar interface. The correct positioning of the segment in the cell can be obtained by geometrical considerations, that lead to a unique relation between α and the color function value C_{ij}. With reference to Figure 5.3, we want to calculate the area of the pentagon $ABFGD$. We can suppose that m_x and m_y are both positive, even if this is not the case we can apply some mirror reflections to go back to the reference situation. The area of the triangle AEH is $\alpha^2/(2m_xm_y)$. If the points E and H are inside the cell, the reference phase occupies exactly the area of this triangle. When the points move outside the cell, we must subtract the areas of the triangles BEF and DGH. Therefore we get

$$\begin{aligned} A_1(\alpha) = \frac{\alpha^2}{2m_x m_y} \Bigg[& 1 - H(\alpha - m_x h_x) \left(\frac{\alpha - m_x h_x}{\alpha}\right)^2 \\ & - H(\alpha - m_y h_y) \left(\frac{\alpha - m_y h_y}{\alpha}\right)^2 \Bigg] , \end{aligned} \tag{5.24}$$

where $H(x)$ is the Heaviside function and $A_1 = h_x \, h_y \, C_{ij}$. The second term is different from zero when E is outside the cell, namely $\alpha > m_x h_x$, while the third appears when H is beyond D, $\alpha > m_y h_y$. The area of this two smaller triangles can be easily computed noting that they are similar to AEH. We get

$$\frac{\text{meas}(BEF)}{\text{meas}(AEH)} = \left(\frac{\alpha - m_x h_x}{\alpha}\right)^2 , \qquad \frac{\text{meas}(DGH)}{\text{meas}(AEH)} = \left(\frac{\alpha - m_y h_y}{\alpha}\right)^2 .$$

We remark that (5.24) is a strictly monotonic function, and is a polynomial of first or second order depending on the Heaviside functions into play. The properties of (5.24) guarantee that the inverse function $\alpha = \alpha(C)$ exists and can be determined easily.

5.2.3 Multiphase Interface advection

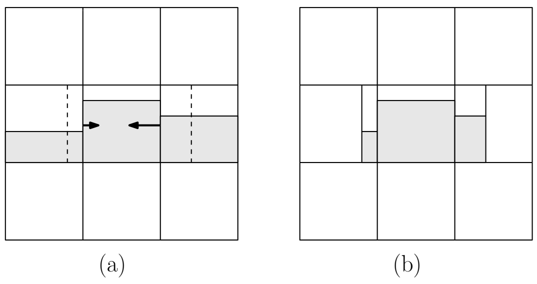


Figure 5.4: Eulerian implicit method: (a) SLIC reconstruction of the interface, (b) implicit step fluxes.

The algorithms to propagate the interface can be divided in two broad categories. Split algorithms decompose the motion along the coordinate directions and advance the color data separately, by creating an intermediate field $\widetilde{C}$ after

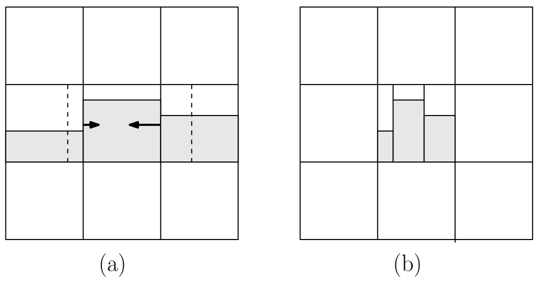


Figure 5.5: Lagrangian explicit method: (a) SLIC reconstruction of the interface, (b) final configuration.

each of the steps, while unsplit algorithms define two-dimensional fluxes and advance the volume fraction distribution in a single step. Three dimensional algorithm of this type are still too complex geometrically. We recall (5.12) and rewrite it in the conservative form

$$\frac{\partial \chi}{\partial t} + \nabla \cdot (\chi \mathbf{v^f}) = \chi \nabla \cdot \mathbf{v^f} = 0 \,. \tag{5.25}$$

For a split algorithm we can consider the mono-dimensional case in the x direction and write for the cell $(i,\, j)$

$$h_x\, h_y \, \frac{\partial C_{ij}(t)}{\partial t} + \oint_{\Gamma_{ij}} \chi(\mathbf{x}, t)\, \mathbf{v} \cdot \mathbf{n} \, d\ell = h_x\, h_y\, C_{ij}\, \frac{\partial v_x^f}{\partial x} \,, \tag{5.26}$$

that is the integration of (5.25) on the cell. The term $\partial v_x^f / \partial x$ can be assumed as a mean value of the derivative and is different from zero even when the two-dimensional field is incompressible. If we consider two temporal steps t^k e $t^{k+1} = t^k + \Delta t$, use nondimensional variables and approximate the spatial derivative with centered finite differences we get

$$C_{ij}^{k+1} = C_{ij}^k - \widetilde{\Phi}_{i+1/2,\,j} + \widetilde{\Phi}_{i-1/2,\,j} + \widetilde{C}_{ij}(u_{i+1/2,\,j} - u_{i-1/2,\,j}) \,, \tag{5.27}$$

where $\widetilde{\Phi}$ is the normalized flux and it depends on the choice of $\widetilde{C}$. We will assume $h_x = h_y$ to simplify the notation. The Eulerian implicit (EI) method sets $\widetilde{C}_{ij} = C_{ij}^{k+1}$, therefore (5.27) becomes

$$C_{ij}^{k+1} = a \left(C_{ij}^k - \widetilde{\Phi}_{i+1/2,\,j} + \widetilde{\Phi}_{i-1/2,\,j} \right) \,, \tag{5.28}$$

where $a = 1/(1 - u_{i+1/2,\,j} + u_{i-1/2,\,j})$ is the expansion/contraction coefficient of the Eulerian step. The geometrical procedure is shown in Figure 5.4. The

Lagrangian explicit (LE) method derives also from (5.27) setting $\widetilde{C}_{ij} = C^k_{ij}$ to get

$$C^{k+1}_{ij} = b\, C^k_{ij} - \widetilde{\Phi}_{i+1/2,j} + \widetilde{\Phi}_{i-1/2,j}\,, \tag{5.29}$$

where $b = (1 + u_{i+1/2,j} - u_{i-1/2,j})$ is the Lagrangian expansion/contraction coefficient. The procedure is shown in Figure 5.5. When the reconstruction is made with a SLIC method as in the Figures, the Eulerian method and the Lagrangian one produce the same result, but this is clearly not true for PLIC reconstructions that lead to a different value for C^{k+1}_{ij}. If the fluxes are not computed using these two schemes the final volume fraction distribution can be not consistent, with values of C that exceed one or are less then zero.

5.2.4 Multilevel VOF technique

VOF methods show their weaknesses when dealing with structures with characteristic length comparable to the grid spacing. In particular, thin filaments or drops that have a characteristic length of three to four cells can still be reproduced accurately, but for smaller dimensions the interface reconstruction method is not able to describe accurately the shape of the interface. This feature may lead to artificial changes of topology that are non physical. The formation of a pinch and the subsequent detachment of a drop should be driven by a physical model and not dictated by the computational grid. However, this model can be very complex and the physical process is still not fully understood. To mitigate this effect, we use an approach that tries to increase the resolution achievable with the VOF method, while the overhead induced on the FSI solver is kept to a minimum. The basic idea is to separate the grid used for the momentum balance equation system from the one used for interface evolution, in particular we use a finer grid obtained from the coarser one with repeated mid-point refinement. In this way, the number of segments in each cell is increased without the necessity to solve the velocity and pressure fields in a greater number of nodes. We will now describe the equations on these grids with detail, suppressing the h subscript that should appear on all the discretized variables. Starting from the coarse level (c) we refine up to a fine level (f) with $f = c + l$. We indicate with $\mathbf{V}^c$, S^c and $\mathbf{V}^f$, S^f the families of subspaces defined at the coarse and fine levels and with Ω_i^c and Ω_i^f the corresponding generic cell. We can also introduce some transfer operators from the fine to the coarse levels, that take into account the different resolution at which the equations are solved. If some phase structure is present only at the fine level (f), the solution $(p^c, \mathbf{u}^c)$ at the coarse level is different from $(p^f, \mathbf{u}^f)$, which satisfies the momentum balance equations with different test functions. We can start from the continuity equation, the second of (5.7), and assume that it is satisfied by the velocity field at both levels,

$$b(q^c, \mathbf{u}^c) = 0\,, \qquad\qquad b(q^f, \mathbf{u}^f) = 0\,, \tag{5.30}$$

where q^c now designates the test function on S^c. We substitute in the relation at the fine level the coarse velocity field $\mathbf{u}^c$ to get

$$b(q^f, \mathbf{u}^c) = \int_\Omega q^f\, R^{fc}(\mathbf{u}^c, \mathbf{u}^f)\, dV\,, \tag{5.31}$$

where we introduce the fine-to-coarse mass transfer operator R^{fc}, defined by

$$R^{fc}(\mathbf{u}^f, \mathbf{u}^c) = \nabla\cdot(\mathbf{u}^f - \mathbf{u}^c)\,. \tag{5.32}$$

The meaning of this operator is to quantify the residual error of the mass conservation equation when we assume that the coarse level solution $\mathbf{u}^c$ is valid at fine level. Even if the discrete solutions are divergence-free functions over the finite element mesh, the point-wise divergences $\nabla\cdot\mathbf{u}^f$ and $\nabla\cdot\mathbf{u}^c$ may be different from zero for all $\mathbf{x}\in\Omega$, because the divergence-free constraints are imposed in an integral fashion, i.e. $\int_\Omega q^f\,\nabla\cdot\mathbf{u}^f\,dV = 0$ and $\int_\Omega q^c\,\nabla\cdot\mathbf{u}^c\,dV = 0$, but $\int_\Omega q^f\,\nabla\cdot\mathbf{u}^c\,dV = 0$ is not imposed. Since the fine mesh is obtained by mid-point refinement, $S^c(\Omega)\subset S^f(\Omega)$ and therefore any test function q^c can be written as a linear combination of the test functions q^f at the fine level,

$$q^c(\mathbf{x}) = \sum_i a_i q_i^f(\mathbf{x})\,. \tag{5.33}$$

From this we get $b(q^c, \mathbf{u}^f) = \sum_i a_i b(q_i^f, \mathbf{u}^f) = 0$ and therefore

$$0 = b(q^c, \mathbf{u}^c) = b(q^c, \mathbf{u}^c - \mathbf{u}^f) = \int_\Omega q^c R^{fc}(\mathbf{u}^c, \mathbf{u}^f) dV\,. \tag{5.34}$$

Under the above assumptions, there is no net mass transfer from the fine to the coarse level. Therefore, if we implement a projection of the velocity that preserves the divergence-free constraint, such as the one shown in Section 5.2.6, the transfer operator is identically equal to zero. We repeat the same argument for the momentum conservation equation shown in the first of (5.7). Let $(p^f, \mathbf{u}^f)$ be the solution of the Navier-Stokes equation at the fine level

$$\left(\rho\,\frac{\partial\,\mathbf{u}^f}{\partial t}, \mathbf{v}^f\right) + c(\rho, \mathbf{u}^f, \mathbf{u}^f, \mathbf{v}^f) + b(p^f, \mathbf{v}^f) + a(\mu, \mathbf{u}^f, \mathbf{v}^f) = (\mathbf{f}, \mathbf{v}^f) + (\mathbf{f}_s^f, \mathbf{v}^f)\,, \tag{5.35}$$

with $\mathbf{v}^f$ the test function in $\mathbf{V}^f$. Here the density ρ and the viscosity μ are explicitly written since they are now discontinuous across the interface. Now we substitute the solution $(p^c, \mathbf{u}^c)$ of the coarse grid in (5.35)

$$\begin{aligned}\left(\rho\,\frac{\partial\,\mathbf{u}^c}{\partial t}, \mathbf{v}^f\right) + c(\rho, \mathbf{u}^c, \mathbf{u}^c, \mathbf{v}^f) + b(p^c, \mathbf{v}^f) + a(\mu, \mathbf{u}^c, \mathbf{v}^f) = \\ = (\mathbf{f}, \mathbf{v}^f) + (\mathbf{f}_s^f, \mathbf{v}^f) + (P^{fc}(p^c, p^f, \mathbf{u}^c, \mathbf{u}^f), \mathbf{v}^f) + (T^{fc}(\mathbf{u}^c, \mathbf{u}^f), \mathbf{v}^f)\,,\end{aligned} \tag{5.36}$$

and introduce the fine-to-coarse momentum transfer operator P^{fc} defined by

$$\begin{aligned}(P^{fc}(p^c, p^f, \mathbf{u}^c, \mathbf{u}^f), \mathbf{v}^f) = \left(\rho\,\frac{\partial\,\mathbf{u}^c}{\partial t}, \mathbf{v}^f\right) + b(p^c, \mathbf{v}^f) + a(\mu, \mathbf{u}^c, \mathbf{v}^f) + \\ - c(\rho, \mathbf{u}^c - \mathbf{u}^f, \mathbf{u}^c - \mathbf{u}^f, \mathbf{v}^f) - \left(\rho\,\frac{\partial\,\mathbf{u}^f}{\partial t}, \mathbf{v}^f\right) - b(p^f, \mathbf{v}^f) - a(\mu, \mathbf{u}^f, \mathbf{v}^f)\,,\end{aligned} \tag{5.37}$$

and the fine-to-coarse turbulent transfer operator T^{fc} defined by

$$(T^{fc}(\mathbf{u}^c, \mathbf{u}^f), \mathbf{v}^f) = c(\rho, \mathbf{u}^c - \mathbf{u}^f, \mathbf{u}^c - \mathbf{u}^f, \mathbf{v}^f) + c(\rho, \mathbf{u}^c, \mathbf{u}^c, \mathbf{v}^f) - c(\rho, \mathbf{u}^f, \mathbf{u}^f, \mathbf{v}^f)\,. \tag{5.38}$$

We split the contribution from the fine grid to the coarse one in two terms to get a term that can be associated to the well-known turbulence contribution from the sub-grid velocity field. The other term summarizes the difference of virtual work between the two levels. When the spaces are embedded, $\mathbf{V}^c(\Omega) \subset \mathbf{V}^f(\Omega)$, (5.36) holds for any test function on the coarse grid and

$$\begin{aligned}\left(\rho\,\frac{\partial\,\mathbf{u}^c}{\partial t}, \mathbf{v}^c\right) + c(\rho, \mathbf{u}^c, \mathbf{u}^c, \mathbf{v}^c) + b(p^c, \mathbf{v}^c) + a(\mu, \mathbf{u}^c, \mathbf{v}^c) = \\ = (\mathbf{f}, \mathbf{v}^c) + (\mathbf{f}_s^f, \mathbf{v}^c) + (S^{fc}(p^c, p^f, \mathbf{u}^c, \mathbf{u}^f), \mathbf{v}^c)\,.\end{aligned} \tag{5.39}$$

The operator S^{fc} models the whole residual between the fine grid FSI solution and the coarse one and depends only on the variables of the coarse grid

$$\begin{aligned}&(S^{fc},\mathbf{v}^f) = (P^{fc}(p^c,p^f,\mathbf{u}^c,\mathbf{u}^f),\mathbf{v}^f) + (T^{fc}(\mathbf{u}^c,\mathbf{u}^f),\mathbf{v}^f) = \\ &= \left(\rho\,\frac{\partial\,\mathbf{u}^c}{\partial t},\mathbf{v}^f\right) + c(\rho,\mathbf{u}^c,\mathbf{u}^c,\mathbf{v}^f) + b(p^c,\mathbf{v}^f) + a(\mu,\mathbf{u}^c,\mathbf{v}^f) - (\mathbf{f}_s^f,\mathbf{v}^f) - (\mathbf{f},\mathbf{v}^f)\,.\end{aligned} \tag{5.40}$$

When S^{fc} is small it means that a further refinement of the VOF grid does not modify the velocity and pressure fields, and the capillary force calculated on the fine grid is well-resolved. When this is not true, S^{fc} can be directly calculated with (5.40) and projected on the coarse grid, or can be modeled in some way at the coarse grid level. The fine grid is used only to get a higher resolution on interface reconstruction, while the relevant physics must be resolved completely at the coarse level. In the simulations presented in the next chapter, the value of S^{fc} has been monitored and kept small at any point of the simulations.

5.2.5 Numerical implementation

We consider a few features of the multilevel in order to describe the algorithms that keep the induced overhead to a minimum. In particular, the fine grid is used to compute the surface tension force $\mathbf{f}_s^f$ that is then inserted in the coarse grid equations. Since each level of refinement multiplies by a factor of four in two dimensions (or eight in three dimensions) in the number of points and cells, the complete memorization of the VOF data would increase the memory footprint quickly. For this reason we have developed a storage scheme that compresses the VOF data and show an example of its implementation in Figure 5.6. The figure shows a coarse level with 24×16 cells, and the sparse color function matrices at the two levels $f = c+2$ and $f = c+4$. At the intermediate level of refinement each cell of the coarse grid is divided in 16 sub-cells with the interface clearly marked on the grid. At the highest level of refinement each coarse cell is subdivided into 256 smaller cells, with matrix entries about 16 times those of the coarse grid. The VOF interface at this level can be

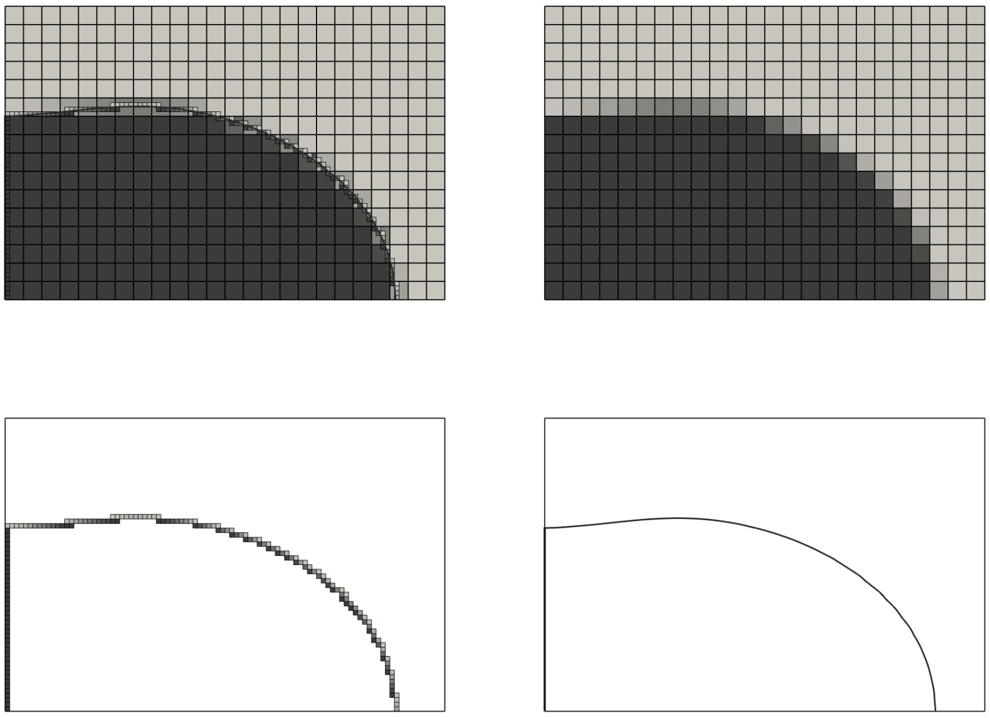
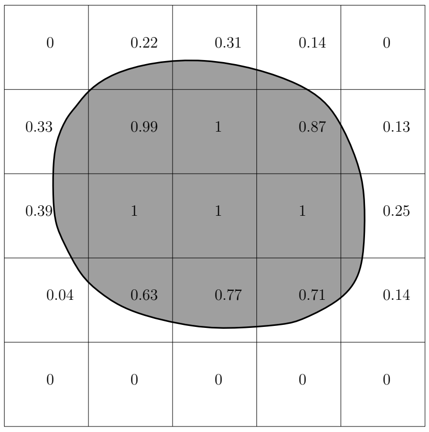


Figure 5.6: The color function distribution on different meshes (top left) and on the coarse mesh (top right). The compact data memorization with two (bottom left) and four (bottom right) levels of grid refinement.

compared in resolution to a front tracking representation with markers. The format used for data storage can be compared to the Compressed Row Storage (CRS) and we show a two-dimensional example in Figure 5.7. We consider a 5×5 stencil of cells and the associated color function data. For each row we memorize only the number of entries n_c, the C data and their column number. All empty cells are discarded, while a sequence of n consecutive full cells is stored as a single one, with its color function value equal to n in the first position, e.g. the third row of Figure 5.7 where we memorize in the second position the integer 3 to represent the sequence of full cells. With this technique we can use a large number of refinement levels while keeping the storage requirements proportional to the length of the interface divided



row	n_c	C					column				
1	3	0.22	0.31	0.14			2	3	4		
2	5	0.33	0.99	1	0.87	0.13	1	2	3	4	5
3	3	0.39	3	0.25			1	2	5		
4	5	0.04	0.63	0.77	0.71	0.14	1	2	3	4	5
5	0										

Figure 5.7: The C data distribution on a 5×5 Cartesian mesh (top) and the compressed stored data (bottom): row number, number of cells n_c, color function in the mixed and consecutive full cells and column position.

by the fine grid spacing. This representation requires an efficient numerical algorithm to extract and compress the relevant data. To further improve the performances when we consider high-resolution fine grids, we do not perform the reconstruction cell by cell, extracting the block of cells needed for every mixed cell. The implementation includes an algorithm that extracts a stripe of $3 \times n$ cells, where n is the length of one full line, and computes the new normals and the fluxes simultaneously for the whole block. After these operations, one line of cells is updated and the procedure is repeated. The normal values are stored in the same compressed way of the VOF data.

5.2.6 Velocity refinement

When dealing with a multilevel VOF method, we need to project the velocity field from the coarse grid to the fine one, since the advection of the color function needs a velocity value on each node of the fine grid. Since the divergence-free constraint is imposed on the coarse grid, we want to preserve it on the fine grid. If we consider a simple approach in which the velocity on the added points are calculated as a simple averaging of the value on the coarse grid, we can easily see that the fine velocity field is not divergence-free. We stress again that in this context the mass conservation constraint is satisfied only in an integral fashion. We introduce an optimal control problem to get a divergence-free preserving projection operator for the velocity field. We rewrite the refined velocities as a combination of all the velocities on the coarse grid, and impose the divergence-free constraint with a Lagrangian multiplier approach. The set of fine velocity values that satisfies these hypotheses is not unique, so we can also impose, as the target of the optimal approach, that the resulting velocities are the closest to the averaged values. Let us consider the

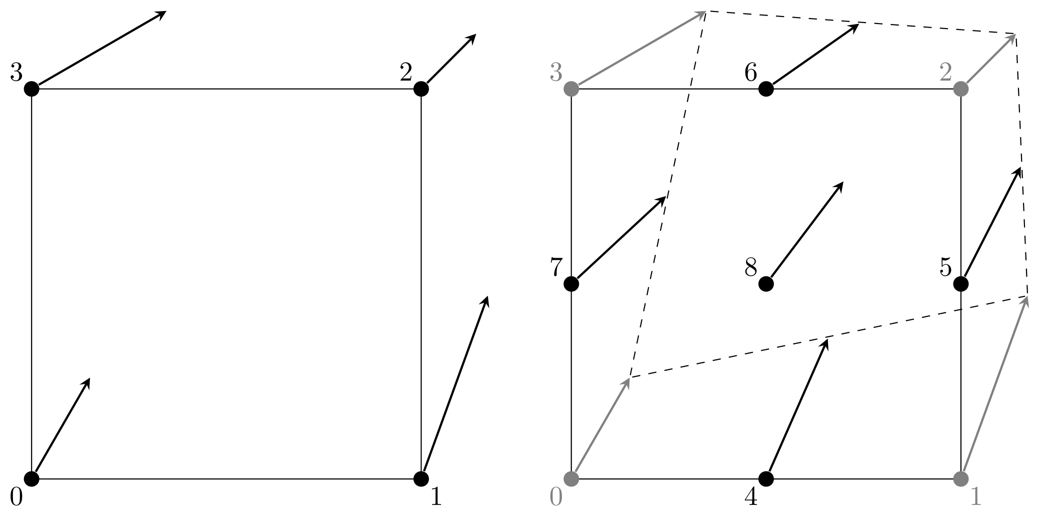


Figure 5.8: The coarse element with four nodes (left) and the refined one with nine nodes (right).

coarse cell shown in Figure 5.8, where we already know the velocities in the points 0, 1, 2 and 3. The midpoint refinement puts in the five new nodes 4, 5, 6, 7 and 8. Therefore we need to compute 10 velocity components, number that clearly outnumbers the constraints of mass conservation, given by

$$u_4 + u_8 - u_7 - u_0 + v_8 + v_7 - v_4 - v_0 = 0\,, \tag{5.41a}$$

$$u_1 + u_5 - u_8 - u_4 + v_5 + v_8 - v_1 - v_4 = 0\,, \tag{5.41b}$$

$$u_8 + u_6 - u_3 - u_7 + v_6 + v_3 - v_8 - v_7 = 0\,, \tag{5.41c}$$

$$u_5 + u_2 - u_8 - u_6 + v_2 + v_6 - v_5 - v_8 = 0\,, \tag{5.41d}$$

that represent the divergence values in each of the fine cells created from the given coarse cell. Therefore, we set some of them to the averaged value

$$\begin{aligned} v_4 &= (v_0 + v_1)/2\,, & u_5 &= (u_1 + u_2)/2\,, \\ v_6 &= (v_2 + v_3)/2\,, & u_7 &= (u_3 + u_0)/2\,, \\ u_8 &= (u_0 + u_1 + u_2 + u_3)/4\,, & v_8 &= (v_0 + v_1 + v_2 + v_3)/4\,. \end{aligned} \tag{5.42}$$

We are left with u_4, v_5, u_6 and v_7 as unknowns. We can now introduce our functional J as

$$J = \frac{1}{2}(u_4 - \tilde{u}_4)^2 + \frac{1}{2}(v_5 - \tilde{v}_5)^2 + \frac{1}{2}(u_6 - \tilde{u}_6)^2 + \frac{1}{2}(v_7 - \tilde{v}_7)^2\,, \tag{5.43}$$

where the target values are indicated by a tilde. We choose them to be the averaged values,

$$\tilde{u}_4 = (u_0 + u_1)/2\,, \qquad \tilde{v}_5 = (v_1 + v_2)/2\,, \tag{5.44a}$$

$$\tilde{u}_6 = (u_2 + u_3)/2\,, \qquad \tilde{v}_7 = (v_3 + v_0)/2\,. \tag{5.44b}$$

We now build the augmented Lagrangian functional P as

$$P = J + \sum_{i=0}^{3} \gamma_i D_i\,, \tag{5.45}$$

where γ_i are the Lagrangian multipliers associated with the four discrete divergence on the l.h.s. of (5.41) and indicated here as D_i. We can now solve our problem by determining the minimum of P. We set to zero its first variation δP

$$\begin{aligned} \delta P = (u_4 - \tilde{u}_4)\delta u_4 + (v_5 - \tilde{v}_5)\delta v_5 + (u_6 - \tilde{u}_6)\delta u_6 + (v_7 - \tilde{v}_7)\delta v_7 + \gamma_0(\delta u_4 + \delta v_7) + \\ + \gamma_1(-\delta u_4 + \delta v_5) + \gamma_2(\delta u_6 - \delta v_7) + \gamma_3(-\delta u_6 - \delta v_5) + \sum_{i=0}^{3} \delta\gamma_i D_i\,. \end{aligned} \tag{5.46}$$

All variations in (5.46) are independent from each other, so we can put to zero each of them singularly to get a minimum of P. We obtain a linear system of eight equations in the eight variables u_4, v_5, u_6, v_7 and the four Lagrangian multipliers. Since the four relations (5.41) are not linearly independent, there are an infinite number of solutions. If we take a 7 by 7 minor with full rank, we get the solution desired by leaving one of the Lagrangian multipliers as undefined. In this way, however, we would get that 3 of the (5.41) are satisfied, while the fourth is not. In general, the value of the divergence on the coarse grid is not zero, since we integrate an iterative solver. If we proceed as described, the whole error on divergence would be transferred to one fine cell, the one corresponding to the divergence constraint that we have removed. Alternatively, we can try to split beforehand the divergence error on all four sub-cells, in order to get a balanced solution on the fine cells. The previous derivation leads to

$$u_4 = \frac{2u_0 + 2u_1 + v_0 - v_1 + v_2 - v_3}{4}\,, \tag{5.47a}$$

$$v_5 = \frac{u_0 - u_1 + u_2 - u_3 + 2v_1 + 2v_2}{4}\,, \tag{5.47b}$$

$$u_6 = \frac{2u_2 + 2u_3 + v_0 - v_1 + v_2 - v_3}{4}\,, \tag{5.47c}$$

$$v_7 = \frac{u_0 - u_1 + u_2 - u_3 + 2v_3 + 2v_0}{4}\,. \tag{5.47d}$$

We remark that now the optimized velocities will depend on both u and v components on the coarse grid. The extension to the three-dimensional case is straightforward and does not present any difficulty. We note that, if in three dimensions we start from a two-dimensional coarse grid velocity field, the resulting fine field will be fully three-dimensional.

5.3 FSI$< - >$VOF Interface

The interaction between the FSI and VOF system is two-way coupling because the VOF method needs the fluid velocity on the whole fluid domain to compute the advection of the color function, while the FSI system needs the color

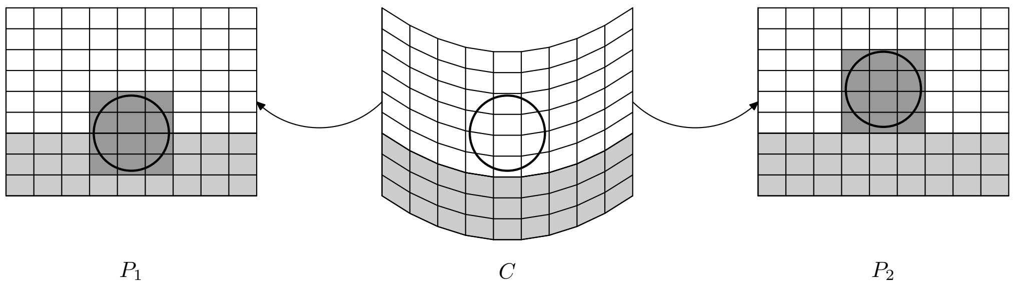


Figure 5.9: In the center a certain configuration of a multiphase FSI problem. Projection into the VOF computational grid with standard (P_1) and modified (P_2) algorythm on the left on the right, respectively.

function to compute the generic fluid property τ^f as

$$\tau^f = \tau_1 C + \tau_2(1 - C)\,, \tag{5.48}$$

where τ_1 is the parameter value of the primary phase while τ_2 is the value in the other one. Two computational meshes are used for the FSI and VOF solutions and the data exchange between these two meshes is obtained through the MEDmem libraries following the procedure explained in Section 2.4. As usual we create a duplicate of the original meshes, then the velocity field, computed by the FSI module, is projected in the med duplicate, the supervisor which can control both the FSI and the VOF problems, project the velocity field from the FSI med duplicate into the VOF duplicated computational grid. This velocity field is then projected from the VOF med mesh into the original grid so that it can be used to compute the new position of the interface. After the secondary phase has been advected, the new color function is configuration is projected back into the FSI grid with a similar procedure. This coupling procedure is accomplished every time step and needs a specific operator that maps every computational element of the FSI mesh into a computational element of the VOF grid. The MEDmem library are here used to evaluate the center of mass every computational element of the first grid and pair it with the element, in the other grid, that has the nearest center of mass. While the VOF algorithm is a geometric solver, it can only be used on structured meshes, highly reducing the computational effort in the evaluation of the nearest element of the FSI problem. In order to have a good coupling between the computational grids the characteristic dimension of the two meshes must be similar. In FSI problems where the displacement field is smaller than the characteristic length of the VOF mesh, the map operator that transfers the computational fields into the different meshes is fixed. In the case of displacement larger than the characteristic dimension of the Cartesian grid, the map operator must be updated in order to take into account the movement of the unstructured grid. Let us consider the example in Figure 5.9. In the central part of Figure 5.9 a general configuration, labeled as C, of a MFSI problem is shown. In the left and right part of this Figure two possible projections of the configuration C into the same Cartesian grid are reported: P_1 and P_2, respectively. We mark in dark gray the cells from which the advection velocity is interpolated in the two cases.. The configuration P_1 is the simple projection and as we can see, because of the large deformation, the secondary phase is advected with the velocity of the solid domain leading to wrong evaluation of the multiphase interface position. In the second case, P_2 the projection is performed considering the grid displacement. Also this correction leads to wrong evaluation of the multiphase advection due to the deformation of the computational cell that is not taken into account by the VOF algorithm. These difficulties in this Chapter are overcome by considering small displacement FSI problems where the map pairing between the computational cells is constant during the simulation. In the following Sections we report some numerical results obtained with the mathematical model just introduced. Dedicated solvers for the FSI and the VOF problems are coupled in the computational platform SALOME and the projection of the computational fields into different meshes is obtained through the MEDmem libraries in a fast and efficient way.

5.4 Test 1. Dam break

In this first test case we simulate a dam break problem in which two fluid phases and a solid region are considered. The initial configuration of the problem is shown in Figure 5.10, due to the different densities of the two fluid phases (Ω_l and Ω_g) and the gravitational field, Ω_l fall down into the solid container Ω_s which deforms to to the interaction with both the fluid phases.

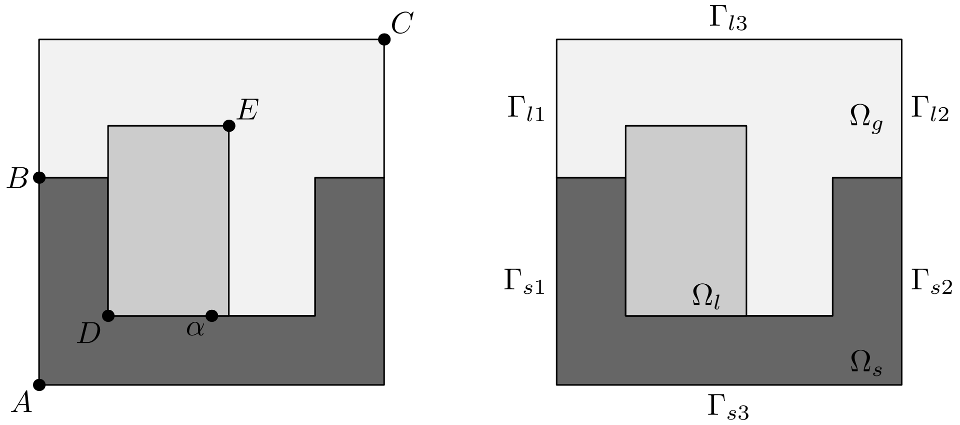
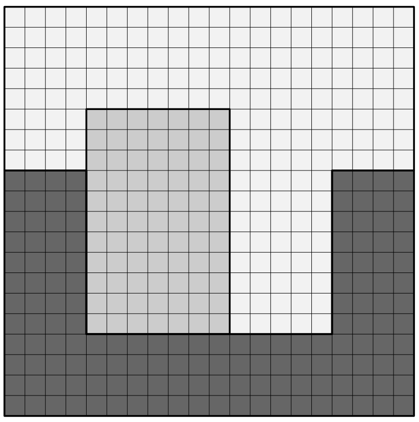


Figure 5.10: Test 1. Domain overview. On the left some reference points: $A = (0m, 0m)$, $B = (0m, 0.6)$, $C = (1m, 1m)$, $D = (0.2m, 0.2m)$, $E = (0.5m, 0.7m)$ and $\alpha = (0.2m, 0.5m)$. On the right labeling of the surfaces.

the domain overview is shown in Figure 5.10, in particular in the right part of that Figure one can see the solid,the primary and secondary fluid region marked as Ω_s, Ω_l, Ω_g, respectively. The global domain $\Omega = \Omega_s \cup \Omega_l \cup \Omega_g$ is a square with a surface of $1m^2$. The geometries of the different parts of the domain are specified by giving the coordinate of some reference points. According to the nomenclature shown in the left part of Figure 5.10, the coordinate of the reference points are: $A = (0m, 0m)$, $B = (0m, 0.6)$, $C = (1m, 1m)$, $D = (0.2m, 0.2m)$ and $E = (0.5m, 0.7m)$. The computational grid, both for the FSI and VOF module, is generated subdividing Ω into 400 cell (20 subdivision per edge) and it is shown in the left part of Figure 5.11. According to the surface labeling show in Figure 5.10, we impose a vanishing velocity field on $\Gamma_{l1} \cup \Gamma_{l2}$ and at the boundary fluid structure interface $\{\Gamma_{l1} \cap \Gamma_{s1}\} \cup \{\Gamma_{l2} \cap \Gamma_{s2}\}$. An homogeneous Neuman condition is imposed of the fluid region Γ_{l3} and on the solid boundary Γ_{s3}. The physical properties of the different phase of the problem are summarized in the Table on the right part of Figure 5.11, in particular both of the fluid phases are modeled as a Newtonian incompressible fluid while the solid is represent by a compressible linear elastic material. In Figure 5.12 the solution overview is show at different time step $t = 0.005$, 0.04, 0.08, 0.095, 0.16s and $t = 0.2s$, the axial displacement field is shown in the solid region (Ω_s) while in the fluid part we mark in red the



Parameter	Value
ρ_l, ρ_s	500 Kg/m^3
μ_l	0.005 Pa s
ρ_g/ρ_l	0.001
μ_g/μ_l	0.01
Young modulus	$4 \cdot 10^4$ Pa
Poisson coefficient	0.4

Figure 5.11: Test 1. Computational grid on the left. Physical parameters on the right.

interface between the primary and the secondary phase and in light gray the streamline of the fluid velocity field. From the initial configuration the secondary phase fall off into the solid container that is deformed due to the weight of the different phase. We can observe that the secondary phase Ω_l follows the velocity streamlines proving that the projection of the different computational fields into the different grids is consistent. The transverse and axial displacement of point α (see left part of Figure 5.10) over time is shown in the left part of Figure 5.13 in the cure A and B respectively. We can notice the the displacement field after some initial oscillation reach a steady state solution and it is always smaller than the characteristic length of the computational grid. In this condition the displacement field can be neglected in the projection of the velocity the color function in the FSI and VOF mesh, respectively. In order to check the performance of the interfaces between the two modules we can evaluate

$$\int_{\Omega(t)} C d\mathbf{x} = k. \tag{5.49}$$

We remark that the fluid is represented with and incompressible model and because of the free divergence constrain of the velocity field k must remain constant unless a wrong projection of the velocity field in the different grid. The expression (5.49) along time is shown in the right part of Figure 5.13 we can notice that, despite some smalls numerical oscillation, the values remains constant around the unit value.

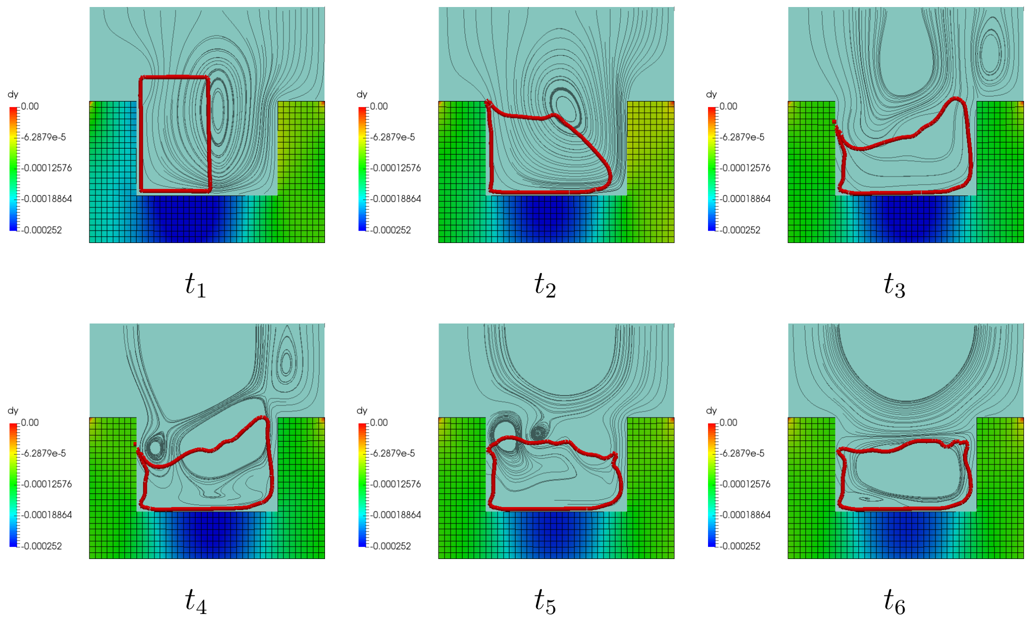


Figure 5.12: Test 1. Solution overview at different time steps: $t_1 = 0.005s$, $t_2 = 0.04s$, $t_3 = 0.08s$, $t_4 = 0.095s$, $t_5 = 0.16s$ and $t_6 = 0.2s$.

5.5 Test 2. Tank filling

In the second test case we consider a filling problem, in which, as in the previous case, two fluid phases and a solid region are considered. The initial configuration of the problem is shown in Figure 5.14, due to the different

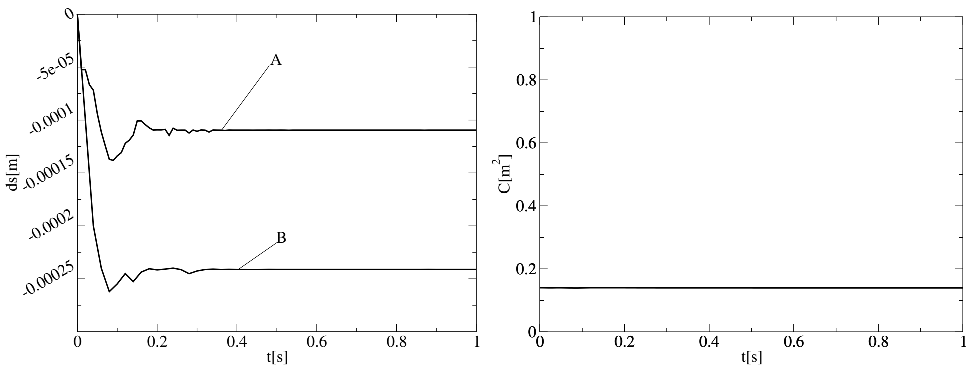


Figure 5.13: Test 1. On the left, transverse (A) and axial (B) displacement over time. On the right, color function integral over the computational domain over time.

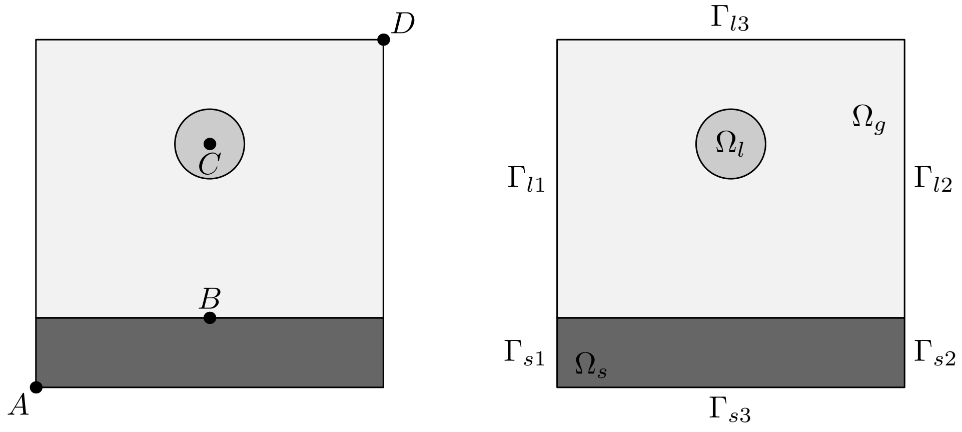


Figure 5.14: Test 2. Domain overview. On the left reference point: $A = (0m, 0m)$, $B = (0.5m, 0.15,)$, $C = (0.5m, 0.5m)$ and $D = (1m, 1m)$. On the right part labeling of the surfaces.

densities of the two fluid phases (Ω_l and Ω_g) and the gravitational field, Ω_l fall down over the solid plate Ω_s which deforms to to the interaction with both the fluid phases. The secondary phase flows inside the domain until it cover the inlet region which is the circular region Ω_l that is shown in the right part of Figure 5.14. The domain overview is shown in Figure 5.14, in particular in the right part of that Figure one can see the solid, the primary and secondary fluid region marked as Ω_s, Ω_l, Ω_g, respectively. The global domain $\Omega = \Omega_s \cup \Omega_l \cup \Omega_g$ is a square with a surface of $1m^2$. The specific geometries of the different parts of the domain are specified by giving the coordinate of some reference points. According to the nomenclature shown in the left part of Figure 5.14, the coordinate of the reference points are: $A = (0m, 0m)$, $B = (0.5m, 0.15,)$, $C = (0.5m, 0.5m)$ and $D = (1m, 1m)$. The computational grid, both for the FSI and VOF module, is generated subdividing Ω into 400 cell (20 subdivision per edge). According to the surface labeling show in Figure 5.14, we impose a vanishing velocity field on $\Gamma_{l1} \cup \Gamma_{l2} \cup \Gamma_{l3} \cup \Gamma_{s1} \cup \Gamma_{s2}$. An homogeneous Neuman condition is imposed of the solid region Γ_{s3}. Concerning the multiphase advection problem, the color function is set to 1 in the Ω_l region. The physical properties of the different materials present in the problem are the same used in the previous case and can be seen in the Table on the right part of Figure 5.11, in particular both of the fluid phases are modeled as a

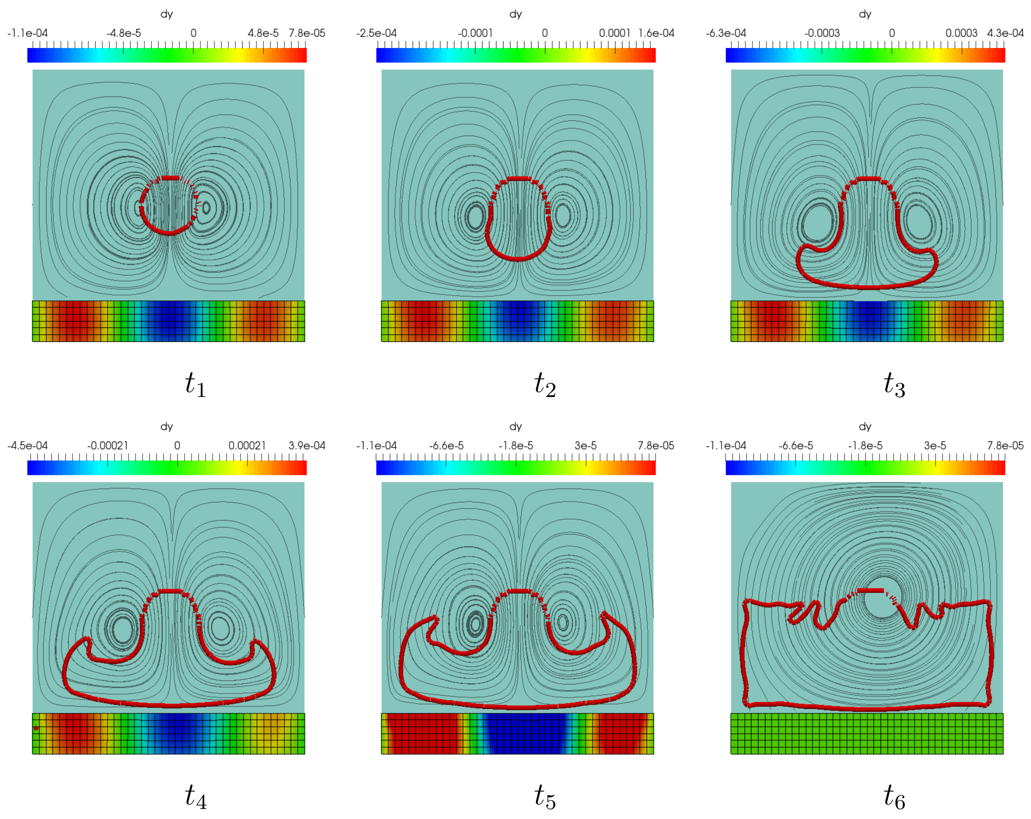


Figure 5.15: Test 2. Solution overview at different time steps: $t_1 = 0.005s$, $t_2 = 0.09s$, $t_3 = 0.2s$, $t_4 = 0.3s$, $t_5 = 0.4s$ and $t_6 = 2s$.

Newtonian incompressible fluid while the solid is represent by a compressible linear elastic material. In Figure 5.15 the solution overview is show at different time step $t = 0.005$, 0.09, 0.2, 0.3, 0.4 and $t = 2s$, the axial displacement field is shown in the solid region (Ω_s) while in the fluid part we mark in red the interface between the primary and the secondary phase and in light gray the streamline of the fluid velocity field. We can notice that as the heavy phase falls because of the gravitational field, more secondary phase is injected in the domain until it cover the inlet region. The axial displacement of the central point of the solid region over time is shown in the left part of Figure 5.16. We can notice the the displacement field shows a great oscillation due to the first contact between the secondary phase and the solid region, after this impact the average axial deformation decrease and reach a constant negative value when the injection of the secondary phase stops. Also in this case is worth to notice

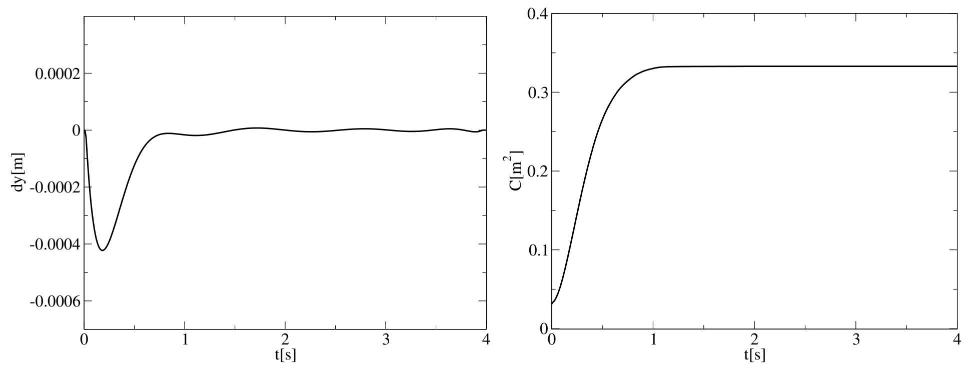
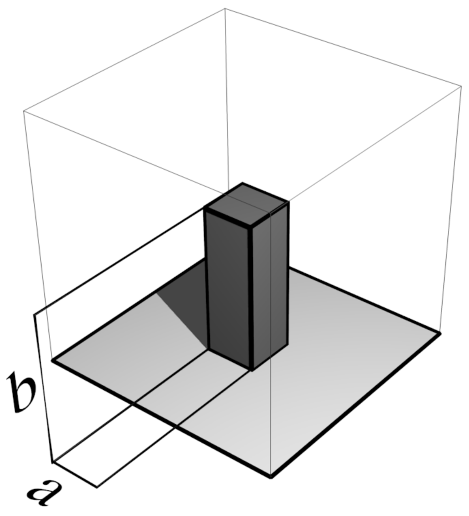


Figure 5.16: Test 2. On the left, axial displacement over time. On the right, color function integral over the computational domain over time.

the the deformations that occurs are always smaller than the characteristic length of the computational grid. In this condition the displacement field can be neglected in the projection of the velocity the color function in the FSI and VOF mesh, respectively. As in the previous case, in order to check the performance of the interfaces between the two modules, we can evaluate the expression (5.49). In this case because of the inlet of the second phase and because of the free divergence free constrain of the velocity field k must increase over time. The expression (5.49) along time is shown in the right part of Figure 5.16, we can notice that the value increases as the secondary phase is injected into the domain and remains constant as the injection is blocked.

5.6 Test 3. River flow

In this third test we consider the flows of a multiphase fluid around a solid obstacle. The domain overview is shown in the left part of Figure 5.17, as in the previous case the global domain is a cube characterized by a edge of $1m$, the obstacle is placed in the center of the cube base, it has a square transverse section a^2 of $0.01m^2$ and a height b of $0.5m$. The specific boundary condition of the problem can be seen in Figures 5.18, in particular in the colored surface of a we set an homogeneous Neuman boundary condition, this region is the outlet of the domain. In the colored surfaces in b of Figures 5.18 we impose a no slip condition while in the colored surface of c a non



Parameter	Value
ρ_l, ρ_s	500 Kg$/m^3$
μ_l	0.005 Pa s
ρ_g/ρ_l	0.001
μ_g/μ_l	0.01
Young modulus	$4 \cdot 10^4$ Pa
Poisson coefficient	1/2

Figure 5.17: Test 3. Domains overview on the left and material parameters on the right.

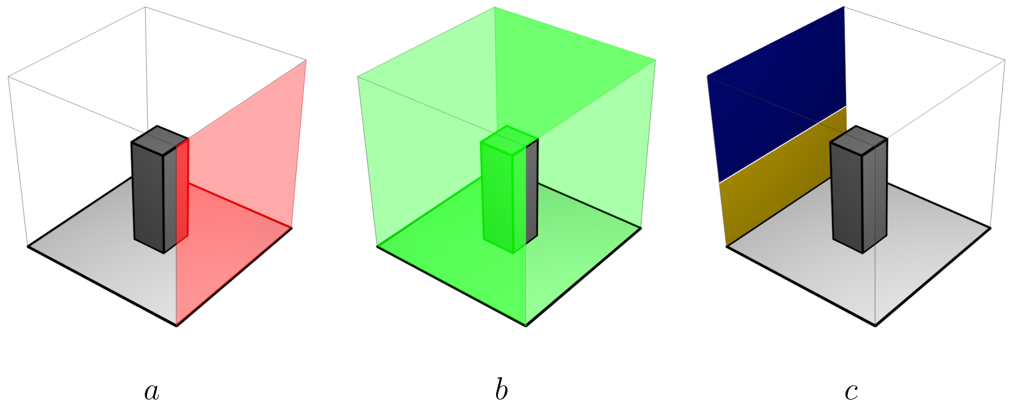


Figure 5.18: Test 3. Boundary condition overview: *a* homogeneous Neuman, *b* homogeneous Dirichlet and *c* non homogeneous Dirichlet.

homogeneous Dirichlet is set. In that surface we impose a vanishing transverse velocity and a non-vanishing axial flows. In particular the fluid flows into the domain with a constant velocity of $1m/s$. Concerning the multiphase fluid part in the inlet region of the domain (see *c* of Figures 5.18) in the upper (height of 0.7m)and lower (height of 0.3m) regions the color function is set to 0 and 1, respectively. In this test we consider the obstacle as a deformable solid rod, so the impact with the heavy phase, displaces the rod with a force based on the weight and inertia of the fluid phase. The interesting feature

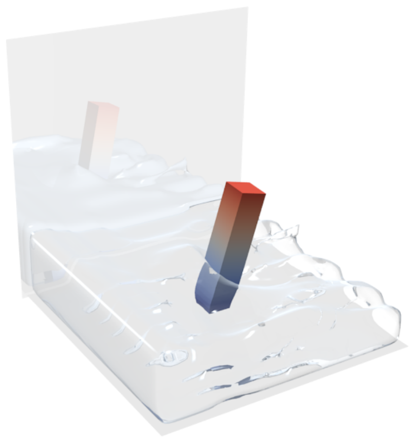


Figure 5.19: Test 3. Solution overview.

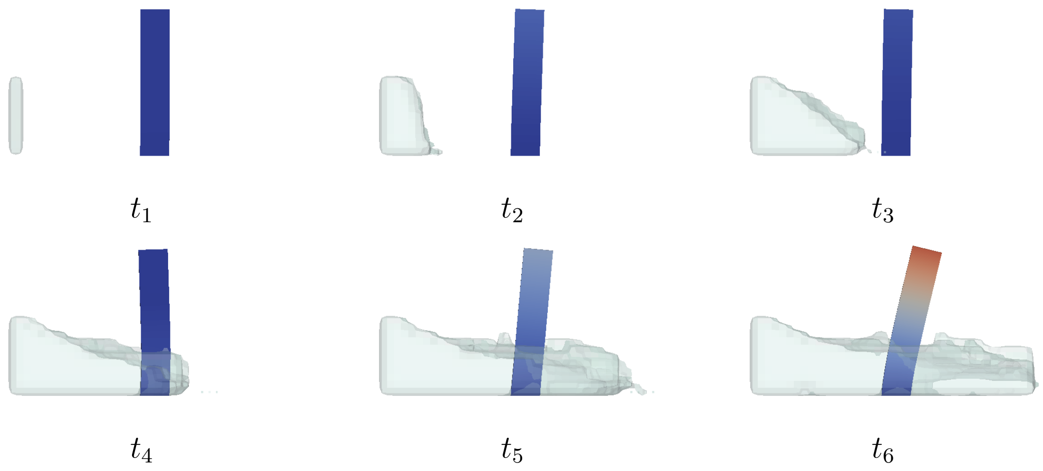


Figure 5.20: Test 3. Simulation of a dam break over a bending rod, side view. From left to right three different time steps: $t_1 = 0.01s$, $t_2 = 0.2s$, $t_3 = 0.4s$, $t_4 = 0.6s$, $t_5 = 0.8s$ and $t_6 = 1s$.

of the coupled FSI-VOF solver is that the displacements of the solid can be precisely evaluated together with the stresses inside the structure. Moreover the bending of the solid has an influence on the flow field which could not be taken in consideration without the FSI solver. The physical parameters of the material involved in the problem are reported in the Table in the right

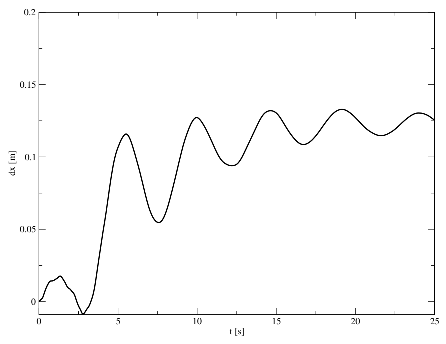


Figure 5.21: Displacement dx in the main flow direction on the most stressed point of the rod, reported as a function of time.

part of Figure 5.17. In this Table we seen that the fluid and the solid have the same density, while the density ratio between the primary and secondary phase is close to the water over air ratio. The solid has been modeled as an incompressible solid and the fluids with an incompressible Newtonian model. In Figure 5.19 an overview of the solution is shown, from that Figure we can view how the multiphase mixture enters into the computational domain and how it deforms the bending rood. In Figure 5.20 the solution overview is show at different time steps: $t = 0.01$, 0.2, $0.4\,0.6\,0.8$ and $t = 1s$. From that sequence we can appreciate the evolution of the secondary phase together with the deformation of the bending rod in particular the color of the rod represents the displacement in the main flow direction. The liquid phase is represented with a transparent surface. It can be seen that, due to the impact with the secondary phase, the rod bends and begins to oscillate. In Figure 5.21 the displacement of the central point of the top surface of the solid part, in the main flow direction is reported as a function of time. The oscillating damped behavior of the rod in the main direction (x axis) can be clearly seen together. We remark that although the deformation in the top part of the rod are large, where the solid interact with the secondary phase the deformation field is still small and smaller the characteristic length of the VOF grid. In this condition the displacement can be neglected in the projection of the computational fields into the different grids.

CHAPTER 6

Conclusion

In this work we have investigated the potentiality of the code coupling on multiphysics and multiscale problems implemented into an in-house computational platform. In all of the studied classes of problem, different codes, that address a specific aspect of a multiphysics problem, were able to exchange computational fields in a common format over the original computational grid and a proper supervisor that controls its execution. We have focused the attention on the specific open-source numerical platform, SALOME, and we have described the structure that a code should have in order to be included in this platform. We have shown how this code coupling, can be used for the investigation of different multiscale or multiphysics problems. Concerning the first class of problems, multiscale, we have shown how a complex system such as the primary loop of LFR reactor or a simplified cardiovascular system can be studied at different resolutions. Different principal components (core, plena, aneurysm, etc.) were investigated with complex three-dimensional models while the remaining parts of the system were considered with a simplified mono-dimensional model. The two different computational modules were able to exchange data thanks to an operator, based on the MEDmem library, that handles the projection of the computational fields between the different computational grids. The coupling of a system code with a 3D module, has allowed the detailed investigation of a particular component without loosing the dynamic effect of the entire system. Two coupling schemes have been tested: direct and defective coupling. In the first case the three-dimensional and the mono-dimensional domain share common junctions where computational fields were exchanged. In the second case, defective coupling, the mono-dimensional domain consist on the whole system and the three-dimensional computational grid overlaps in some parts. In these regions the boundary conditions were imposed by the system code while the 3D solution were used to evaluate some correction terms that are introduced into the mono-dimensional model so that the two different solutions were consistent. For single equation mono-dimensional model, used in the LFR reactor, the defective coupling has shown a greater stability than the first coupling scheme, while for more complex mono-dimensional models, used in the cardiovascular system, the direct coupling has shown enough robustness. In all these examples only a single component has been represented with a full-scale model, in future works we are planning to consider multiple components with full-scale models in order to investigate more complex dynamics. Concerning the multiphysics class of problems, we have shown that the computational platform can be used for studying a specific problem considering more physical phenomena simultaneously. This approach has allowed a more sophisticate and complete description of a particular problem taking into account the different influences among the physical quantities involved. In particular we have investigated the temperature feedback in the energy production density in a PWR core. This effect has been studied coupling the solution of the neutron diffusion problem with the solution of the mass, momentum and energy balance equations system in the reactor core region. Two different computational tools have been used for the solution of this problem, a neutron code a thermal-hydraulics model. The exchange of data between those modules, has been again handled by an operator, based on the MEDmem libraries, that projects a computational field from a computational grids to the other. Finally we have coupled the solution of a Fluid Structure Interaction problem with a multiphase tracking algorithm studying the structure deformation due to the interaction with a multiphase fluid. In this case the SALOME platform has handled the projection of the velocity field between the structured and the unstructured computational grids. All the computational problem classes, presented in this work, have shown how the computational platform an the inclusion of computational tools in it, can be used to improve the simulation quality in different engineering fields. The improvement arises from the consideration of different physics phenomena of the problem simultaneously. Such improvement can be reached with a relatively small amount of time by the integration of different well developed codes into the common computational framework instead of new software developing.

List of Figures

List of Tables

Bibliography

[1] G. Bornia, M. Finelli, and S. Manservisi, “Development and validation of fem-lcore code for the thermal hydraulics of open cores,” tech. rep., Report RdS-2011-002, 2011.

[2] D. Bestion, “The physical closure laws in the cathare code,” *Nuclear Engineering and Design*, vol. 124, no. 3, pp. 229–245, 1990.

[3] F. Barre and M. Bernard, “The cathare code strategy and assessment,” *Nuclear engineering and design*, vol. 124, no. 3, pp. 257–284, 1990.

[4] M. Robert, M. Farvacque, M. Parent, and B. Faydide, “Cathare 2 v2. 5: a fully validated cathare version for various applications,” 2003.

[5] J. Lavieville, S. Quemerais, E.and Mimouni, and N. Mechitoua, *NEPTUNE CFD V1.0 theory manual.* 2006.

[6] G. Geffraye, O. Antoni, M. Farvacque, D. Kadri, G. Lavialle, B. Rameau, and A. Ruby, “Cathare 2 v2. 5_2: a single version for various applications,” *Nuclear Engineering and Design*, vol. 241, no. 11, pp. 4456–4463, 2011.

[7] A. Ribes and C. Caremoli, “Salome platform component model for numerical simulation,” in *Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International*, vol. 2, pp. 553–564, IEEE, 2007.

[8] P. G. Ciarlet, *The finite element method for elliptic problems*, vol. 40. Siam, 2002.

[9] R. Temam, *Navier-Stokes equations: theory and numerical analysis*, vol. 343. American Mathematical Soc., 2001.

[10] K. D. Kok, *Nuclear engineering handbook.* Mechanical engineering series, CRC Press, 2009.

[11] R. K. Osborn, *Foundations of Neutron Transport Theory.* 1 ed., 1966.

[12] E. E. L. Ph.D., *Fundamentals of Nuclear Reactor Physics.* Academic Press, ap ed., 2008.

[13] T. Abram and S. Ion, "Generation-IV nuclear power: A review of the state of the science," *Energy Policy*, vol. 36, no. 12, pp. 4323–4330, 2008.

[14] U. DoE, "A technology roadmap for generation iv nuclear energy systems," in *Nuclear Energy Research Advisory Committee and the Generation IV International Forum*, 2002.

[15] A. Alemberti, J. Carlsson, E. Malambu, A. Orden, D. Struwe, P. Agostini, and S. Monti, "European lead fast reactor—elsy," *Nuclear engineering and design*, vol. 241, no. 9, pp. 3470–3480, 2011.

[16] S. Bnà, S. Manservisi, and O. Le Bot, "Simulation of the thermo-hydraulic behaviour of liquid metal reactors using a three-dimensional finite element model," tech. rep., Report RdS/2010/107, 2010.

[17] G. Bornia, D. Cerroni, S. Manservisi, M. Polidori, and F. Donato, "Femlcore code: parallelization, turbulence models and code integration," 2012.

[18] A. Cervone and M. Manservisi, "A three-dimensional cfd program for the simulation of the thermo-hydraulic behaviour of an open core liquid metal reactor," tech. rep., Report RSE/2009/85, 2008.

[19] M. Sarotto, "Elsy core design static, dynamic and safety parameters with the open square fa," 2009.

[20] G. Bandini, P. Meloni, and M. Polidori, "Thermal-hydraulics analyses of elsy lead fast reactor with open square core option," *Nuclear Engineering and Design*, vol. 241, no. 4, pp. 1165–1171, 2011.

[21] G. Marleau, R. Roy, and A. Hébert, "Dragon: a collision probability transport code for cell and supercell calculations," *Report IGE-157, Institut de génie nucléaire, École Polytechnique de Montréal, Montréal, Québec*, 1994.

[22] R. Roy and A. Hébert, "The gan generalized driver," *Report IGE–158, Institut de génie nucléaire, Ecole Polytechnique de Montréal, Montréal, Québec*, 2000.

[23] D. Cerroni and S. Manservisi, "A penalty-projection algorithm for a monolithic fluid-structure interaction solver," *Journal of Computational Physics*, vol. 313, pp. 13–30, 2016.

[24] J. Hron and S. Turek, *A monolithic FEM/multigrid solver for an ALE formulation of fluid-structure interaction with applications in biomechanics.* Springer, 2006.

[25] K.-J. Bathe and G. A. Ledezma, "Benchmark problems for incompressible fluid flows with structural interactions," *Computers & structures*, vol. 85, no. 11, pp. 628–644, 2007.

[26] M. Hamamoto, Y. Ohta, K. Hara, and T. Hisada, "Application of fluid–structure interaction analysis to flapping flight of insects with deformable wings," *Advanced Robotics*, vol. 21, no. 1-2, pp. 1–21, 2007.

[27] J. Zhang, L. Guo, H. Wu, A. Zhou, D. Hu, and J. Ren, "The influence of wind shear on vibration of geometrically nonlinear wind turbine blade under fluid–structure interaction," *Ocean Engineering*, vol. 84, pp. 14–19, 2014.

[28] Y. Bazilevs, M.-C. Hsu, I. Akkerman, S. Wright, K. Takizawa, B. Henicke, T. Spielman, and T. Tezduyar, "3d simulation of wind turbine rotors at full scale. part i: Geometry modeling and aerodynamics," *International*

Journal for Numerical Methods in Fluids, vol. 65, no. 1-3, pp. 207–235, 2011.

[29] C. Graczykowski and J. Holnicki-Szulc, "Adaptive flow control based airbags for waterborne and aeronautical applications," in *Proc. of the 4th European Conference on Structural Control, September*, pp. 8–12, Citeseer.

[30] K. Stein, R. Benney, V. Kalro, T. E. Tezduyar, J. Leonard, and M. Accorsi, "Parachute fluid–structure interactions: 3-d computation," *Computer Methods in Applied Mechanics and Engineering*, vol. 190, no. 3, pp. 373–386, 2000.

[31] K. Takizawa, Y. Bazilevs, T. E. Tezduyar, M.-C. Hsu, O. Øiseth, K. M. Mathisen, N. Kostov, and S. McIntyre, "Engineering analysis and design with ale-vms and space–time methods," *Archives of Computational Methods in Engineering*, vol. 21, no. 4, pp. 481–508, 2014.

[32] Y. Cheng, H. Oertel, and T. Schenkel, "Fluid-structure coupled cfd simulation of the left ventricular flow during filling phase," *Annals of biomedical engineering*, vol. 33, no. 5, pp. 567–576, 2005.

[33] H. Liu, H. Xu, P. J. Ellison, and Z. Jin, "Application of computational fluid dynamics and fluid–structure interaction method to the lubrication study of a rotor–bearing system," *Tribology Letters*, vol. 38, no. 3, pp. 325–336, 2010.

[34] Y. Bazilevs, K. Takizawa, and T. E. Tezduyar, *Computational fluid-structure interaction: methods and applications.* John Wiley & Sons, 2013.

[35] S. Piperno, "Explicit/implicit fluid/structure staggered procedures with a structural predictor and fluid subcycling for 2d inviscid aeroelastic simulations," *International journal for numerical methods in fluids*, vol. 25, no. 10, pp. 1207–1226, 1997.

[36] C. Michler, S. Hulshoff, E. Van Brummelen, and R. De Borst, "A monolithic approach to fluid–structure interaction," *Computers & fluids*, vol. 33, no. 5, pp. 839–848, 2004.

[37] J. Degroote, K.-J. Bathe, and J. Vierendeels, "Performance of a new partitioned procedure versus a monolithic procedure in fluid–structure interaction," *Computers & Structures*, vol. 87, no. 11, pp. 793–801, 2009.

[38] J. Degroote, P. Bruggeman, R. Haelterman, and J. Vierendeels, "Stability of a coupling technique for partitioned solvers in fsi applications," *Computers & Structures*, vol. 86, no. 23, pp. 2224–2234, 2008.

[39] W. A. Wall, S. Genkinger, and E. Ramm, "A strong coupling partitioned approach for fluid–structure interaction with free surfaces," *Computers & Fluids*, vol. 36, no. 1, pp. 169–183, 2007.

[40] W. K. Liu, Y. Liu, D. Farrell, L. Zhang, X. S. Wang, Y. Fukui, N. Patankar, Y. Zhang, C. Bajaj, J. Lee, *et al.*, "Immersed finite element method and its applications to biological systems," *Computer Methods in Applied Mechanics and Engineering*, vol. 195, no. 13, pp. 1722–1749, 2006.

[41] D. Bluestein, K. Dumont, M. De Beule, J. Ricotta, P. Impellizzeri, B. Verhegghe, and P. Verdonck, "Intraluminal thrombus and risk of rupture in patient specific abdominal aortic aneurysm–fsi modelling," *Computer methods in biomechanics and biomedical engineering*, vol. 12, no. 1, pp. 73–81, 2009.

[42] C. Long, A. Marsden, and Y. Bazilevs, "Shape optimization of pulsatile ventricular assist devices using fsi to minimize thrombotic risk," *Computational Mechanics*, vol. 54, no. 4, pp. 921–932, 2014.

[43] L. Formaggia, A. Quarteroni, and A. Veneziani, *Cardiovascular Mathematics: Modeling and simulation of the circulatory system*, vol. 1. Springer Science & Business Media, 2010.

[44] R. A. Adams and J. J. Fournier, *Sobolev spaces*, vol. 140. Academic press, 2003.

[45] P. Sackinger, P. Schunk, and R. Rao, "A newton–raphson pseudo-solid domain mapping technique for free and moving boundary problems: a finite element implementation," *Journal of Computational Physics*, vol. 125, no. 1, pp. 83–103, 1996.

[46] J. Donea, S. Giuliani, and J. Halleux, "An arbitrary lagrangian-eulerian finite element method for transient dynamic fluid-structure interactions," *Computer methods in applied mechanics and engineering*, vol. 33, no. 1, pp. 689–723, 1982.

[47] P. Le Tallec and J. Mouro, "Fluid structure interaction with large structural displacements," *Computer Methods in Applied Mechanics and Engineering*, vol. 190, no. 24, pp. 3039–3067, 2001.

[48] D. Cerroni, S. Manservisi, and F. Menghini, "A new moving mesh technique for monolithic multigrid fluid-structure interaction solver," *Recent Advances in Civil Engineering and Mechanics*, vol. 1, pp. 146–152, 2014.

[49] O. C. Zienkiewicz, R. L. Taylor, O. C. Zienkiewicz, and R. L. Taylor, *The finite element method*, vol. 3. McGraw-hill London, 1977.

[50] A. Guittet, M. Theillard, and F. Gibou, "A stable projection method for the incompressible navier–stokes equations on arbitrary geometries and adaptive quad/octrees," *Journal of Computational Physics*, vol. 292, pp. 215–238, 2015.

[51] A. J. Chorin, "Numerical solution of the navier-stokes equations," *Mathematics of computation*, vol. 22, no. 104, pp. 745–762, 1968.

[52] C. Wang and J. D. Eldredge, "Strongly coupled dynamics of fluids and rigid-body systems with the immersed boundary projection method," *Journal of Computational Physics*, vol. 295, pp. 87–113, 2015.

[53] E. Onate and M. Cervera, "Derivation of thin plate bending elements with one degree of freedom per node: a simple three node triangle," *Engineering computations*, vol. 10, no. 6, pp. 543–561, 1993.

[54] E. Aulisa, S. Manservisi, and P. Seshaiyer, "A computational multilevel approach for solving 2d navier–stokes equations over non-matching grids," *Computer Methods in Applied Mechanics and Engineering*, vol. 195, no. 33, pp. 4604–4616, 2006.

[55] F.-B. Tian, H. Dai, H. Luo, J. F. Doyle, and B. Rousseau, "Fluid–structure interaction involving large deformations: 3d simulations and applications to biological systems," *Journal of computational physics*, vol. 258, pp. 451–469, 2014.

[56] J. Hron, A. Ouazzi, and S. Turek, *A computational comparison of two FEM solvers for nonlinear incompressible flow.* Springer, 2003.

[57] B. Froehle and P.-O. Persson, "A high-order discontinuous galerkin method for fluid–structure interaction with efficient implicit–explicit time stepping," *Journal of Computational Physics*, vol. 272, pp. 455–470, 2014.

[58] F. Nobile, M. Pozzoli, and C. Vergara, "Inexact accurate partitioned algorithms for fluid–structure interaction problems with finite elasticity in haemodynamics," *Journal of Computational Physics*, vol. 273, pp. 598–617, 2014.

[59] M. Bukač, S. Čanić, and B. Muha, "A partitioned scheme for fluid–composite structure interaction problems," *Journal of Computational Physics*, vol. 281, pp. 493–517, 2015.

[60] P. Causin, J.-F. Gerbeau, and F. Nobile, "Added-mass effect in the design of partitioned algorithms for fluid–structure problems," *Computer methods in applied mechanics and engineering*, vol. 194, no. 42, pp. 4506–4527, 2005.

[61] M. Heil, “An efficient solver for the fully coupled solution of large-displacement fluid–structure interaction problems,” *Computer Methods in Applied Mechanics and Engineering*, vol. 193, no. 1, pp. 1–23, 2004.

[62] C. Förster, W. A. Wall, and E. Ramm, “Artificial added mass instabilities in sequential staggered coupling of nonlinear structures and incompressible viscous flows,” *Computer methods in applied mechanics and engineering*, vol. 196, no. 7, pp. 1278–1293, 2007.

[63] C. A. Figueroa, I. E. Vignon-Clementel, K. E. Jansen, T. J. Hughes, and C. A. Taylor, “A coupled momentum method for modeling blood flow in three-dimensional deformable arteries,” *Computer methods in applied mechanics and engineering*, vol. 195, no. 41, pp. 5685–5706, 2006.

[64] Y. Bazilevs, V. Calo, T. Hughes, and Y. Zhang, “Isogeometric fluid-structure interaction: theory, algorithms, and computations,” *Computational mechanics*, vol. 43, no. 1, pp. 3–37, 2008.

[65] S. Zhao, X. Xu, and M. Collins, “The numerical analysis of fluid-solid interactions for blood flow in arterial structures part 2: development of coupled fluid-solid algorithms,” *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, vol. 212, no. 4, pp. 241–252, 1998.

[66] F. Nobile and C. Vergara, “An effective fluid-structure interaction formulation for vascular dynamics by generalized robin conditions,” *SIAM Journal on Scientific Computing*, vol. 30, no. 2, pp. 731–763, 2008.

[67] J.-F. Gerbeau and M. Vidrascu, “A quasi-newton algorithm based on a reduced model for fluid-structure interaction problems in blood flows,” *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, vol. 37, no. 4, pp. 631–647, 2003.

[68] T. E. Tezduyar, S. Sathe, T. Cragin, B. Nanna, B. S. Conklin, J. Pausewang, and M. Schwaab, “Modelling of fluid–structure interactions with the space–time finite elements: Arterial fluid mechanics,” *International*

Journal for Numerical Methods in Fluids, vol. 54, no. 6-8, pp. 901–922, 2007.

[69] M. A. Fernández, M. Landajuela, and M. Vidrascu, “Fully decoupled time-marching schemes for incompressible fluid/thin-walled structure interaction,” *Journal of Computational Physics*, vol. 297, pp. 156–181, 2015.

[70] X. I. Yang and R. Mittal, “Acceleration of the jacobi iterative method by factors exceeding 100 using scheduled relaxation,” *Journal of Computational Physics*, vol. 274, pp. 695–708, 2014.

[71] A. P. S. Bhalla, R. Bale, B. E. Griffith, and N. A. Patankar, “A unified mathematical framework and an adaptive numerical method for fluid–structure interaction with rigid, deforming, and elastic bodies,” *Journal of Computational Physics*, vol. 250, pp. 446–476, 2013.

[72] J. Guermond, P. Minev, and J. Shen, “An overview of projection methods for incompressible flows,” *Computer methods in applied mechanics and engineering*, vol. 195, no. 44, pp. 6011–6045, 2006.

[73] S. C. Brenner and R. Scott, *The mathematical theory of finite element methods*, vol. 15. Springer Science & Business Media, 2008.

[74] M. Jobelin, C. Lapuerta, J.-C. Latché, P. Angot, and B. Piar, “A finite element penalty–projection method for incompressible flows,” *Journal of Computational Physics*, vol. 217, no. 2, pp. 502–518, 2006.

[75] G. Ryskin and L. Leal, “Numerical solution of free-boundary problems in fluid mechanics. part 1. the finite-difference technique,” *Journal of Fluid Mechanics*, vol. 148, pp. 1–17, 1984.

[76] T. Ye, W. Shyy, and J. N. Chung, “A fixed-grid, sharp-interface method for bubble dynamics and phase change,” *Journal of Computational Physics*, vol. 174, no. 2, pp. 781–815, 2001.

[77] R. Scardovelli and S. Zaleski, “Interface reconstruction with least-square fit and split eulerian–lagrangian advection,” *International Journal for Numerical Methods in Fluids*, vol. 41, no. 3, pp. 251–274, 2003.

[78] W. J. Rider and D. B. Kothe, "Reconstructing volume tracking," *Journal of computational physics*, vol. 141, no. 2, pp. 112–152, 1998.

[79] D. Jamet, O. Lebaigue, N. Coutris, and J. Delhaye, "The second gradient method for the direct numerical simulation of liquid–vapor flows with phase change," *Journal of Computational Physics*, vol. 169, no. 2, pp. 624–651, 2001.

[80] D. Cerroni, S. Manservisi, and F. Menghini, "Multiscale tecniques for the coupling of 3d-1d fsi equations system in compliant vessels,"

[81] E. Aulisa, S. Manservisi, and P. Seshaiyer, "A multilevel domain decomposition approach to solving coupled applications in computational fluid dynamics," *International Journal for Numerical Methods in Fluids*, vol. 56, no. 8, pp. 1139–1145, 2008.

[82] G. Tryggvason, R. Scardovelli, and S. Zaleski, *Direct numerical simulations of gas–liquid multiphase flows.* Cambridge University Press, 2011.

[83] J. E. Pilliod and E. G. Puckett, "Second-order accurate volume-of-fluid algorithms for tracking material interfaces," *Journal of Computational Physics*, vol. 199, no. 2, pp. 465–502, 2004.

[84] A. Cervone, S. Manservisi, and R. Scardovelli, "A fem solver coupled to a multilevel vof method for simulation of axisymmetric jets and to a front-tracking method for simulation of spreading droplets," *Atomization and Sprays*, vol. 20, no. 2, 2010.

[85] R. Scardovelli and S. Zaleski, "Direct numerical simulation of free-surface and interfacial flow," *Annual review of fluid mechanics*, vol. 31, no. 1, pp. 567–603, 1999.