Agenzia nazionale per le nuove tecnologie,
l'energia e lo sviluppo economico sostenibile

MINISTERO DELLO SVILUPPO ECONOMICO

Ricerca di Sistema elettrico

Validazione dell'accoppiamento FEM-LCORE/CATHARE attraverso la simulazione della facility TALL-3D

A. Cervone, M. Polidori, D. Cerroni, R. Da Vià, S. Manservisi

Report RdS/PAR2015/047

VALIDAZIONE DELL'ACCOPPIAMENTO FEM-LCORE/CATHARE ATTRAVERSO LA SIMULAZIONE DELLA FACILITY TALL-3D

A. Cervone, M. Polidori (ENEA)
D. Cerroni, R. Da Vià, S. Manservisi (CIRTEN – Università di Bologna)

Settembre 2016

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero dello Sviluppo Economico - ENEA
Piano Annuale di Realizzazione 2015
Area: Generazione di Energia Elettrica con Basse Emissioni di Carbonio
Progetto: Sviluppo competenze scientifiche nel campo della sicurezza nucleare e collaborazione ai programmi internazionali per il nucleare di IV Generazione.
Linea: Collaborazione ai programmi internazionali per il nucleare di IV Generazione
Obiettivo: Termoidraulica del Refrigerante
Responsabile del Progetto: Mariano Tarantino, ENEA

Il presente documento descrive le attività di ricerca svolte all'interno dell'Accordo di collaborazione "Sviluppo competenze scientifiche nel campo della sicurezza nucleare e collaborazione ai programmi internazionali per i nucleare di IV Generazione"
Responsabile scientifico ENEA: Mariano Tarantino
Responsabile scientifico CIRTEN: Giuseppe Forasassi

Titolo

VALIDAZIONE DELL'ACCOPPIAMENTO FEM-LCORE/CATHARE ATTRAVERSO LA SIMULAZIONE DELLA FACILITY TALL-3D

Descrittori

Tipologia del documento: **Rapporto tecnico**

Collocazione contrattuale: Accordo di programma ENEA-MSE su sicurezza nucleare e reattori di IV generazione

Argomenti trattati: termoidraulica del nocciolo, Generation IV reactors

Sommario

In questo rapporto vengono riportate le simulazioni della facility TALL-3D utilizzando i codici FEMLCORE e CATHARE all'interno della piattaforma SALOME, considerando un circuito monodimensionale e una sezione di prova tridimensionale con dominio sovrapposto accoppiati tra loro. Viene simulato un incidente di ULOF (unprotected loss of flow) sia col sistema 1d disaccoppiato, sia con un accoppiamento one-way, sia con un accoppiamento completo. Le simulazioni vengono confrontate nei profili di temperatura e nelle oscillazioni che si ottengono nel transitorio da regime forzato a circolazione naturale.

Note

Il presente contributo è stato preparato con il contributo del personale ENEA e CIRTEN:
A. Cervone, M. Polidori (ENEA)
D. Cerroni, R. Da Vià, S. Manservisi (CIRTEN – Università di Bologna)
Sigla documento di rif.: CERSE-UNIBO RL1362/2016

Copia n. **In carico a:**

REV.	DESCRIZIONE	DATA		REDAZIONE	CONVALIDA	APPROVAZIONE
2			NOME			
			FIRMA			
1			NOME			
			FIRMA			
0	EMISSIONE	12/09/16	NOME	A. CERVONE	M. TARANTINO	M. TARANTINO
			FIRMA	Antonio Cervone	Marcello Tarantino	Marcello Tarantino

Validation of the FEM-LCORE/CATHARE Coupling Model of the TALL-3D Facility

D. Cerroni, A. Cervone, R. Da Vià, S. Manservisi
and M. Polidori

September 12, 2016

Abstract

In this report we simulate the TALL-3D facility by using the FEMLCORE and CATHARE code in the SALOME platform. We consider, by using a defective coupling algorithm, a one-dimensional circuit and a three-dimensional test section with overlapping meshes. We simulate the evolution of an unprotected loss of flow going from forced to natural circulation flow. In this unprotected loss of flow we have the system initially in fully working conditions, then the pump stops while the supplied power is not switched off. We have studied the one-dimensional system code standalone, the one-way coupling and the full coupling case. The system code standalone has produced temperature oscillations which were shifted in phase and smaller in term of amplitude. Also the coupling has produced shifted oscillations showing that, in the one-dimensional circuit, the pressure losses have not been balanced correctly. With the coupling the pressure losses increase and the liquid flowrate decreases. The correct shift has not been obtained since this would require a redistribution and a reduction of the pressure losses in the the one-dimensional circuit away from the three-dimensional test section. However the redistribution of the pressure losses is not a very easy task due to the complexity of the system since it needs an accurate and long revision process of the one-dimensional model. Finally in order to evaluate the behavior of the coupling system we have shown that, under different pressure losses, a shift in the correct direction of the temperature oscillations is possible.

Contents

Introduction

The study on nuclear energy began in the 1950s and 60s and lead to the construction of the first series of civil nuclear power reactors. The construction of the second generation of reactors started at the beginning of the 1970s and marked the widespread appearance of Light Water Reactors (LWRs) with water as coolant and moderator. In the year 2000, in the context of the Generation IV International Forum (GIF), nuclear experts began formulating the requirements for a fourth generation of nuclear systems that could respond to the world future energy needs. In particular these new reactors should reduce demand for electricity and emissions from fossil fuels leading to a more balanced production of energy. The main aim is to make efficient use of uranium natural resources and minimize waste production, satisfy economic competitiveness and maintain stringent standards of safety and proliferation resistance as well. Among the new reactors proposed from the Generation IV International Forum there is the Lead-cooled Fast Reactor (LFR). The LFR is a flexible fast neutron reactor which can use depleted uranium or thorium fuel matrices, and burn actinides from LWR fuel. A wide range of LFR sizes is planned, from factory-built battery with 15–20 year life for small grids or developing countries, to modular 300–400 MWe units and large single plants of 1400 MWe. Operating temperature of $550°C$ is readily achievable but higher temperature can be reached with advanced materials to provide lead corrosion resistance at high temperatures which would enable thermochemical hydrogen production. A two-stage development program leading to industrial deployment is planned: by 2025 for reactors operating with relatively low temperature and power density, and by 2040 for more advanced higher-temperature designs.

Initial development on the LF reactors was focused on two pool-type reactors: Small Secure Transportable Autonomous Reactor (SSTAR) of 20 MWe in USA and the European Lead-cooled SYstem (ELSY) of 600 MWe in Europe. The SSTAR core is one meter high and $1.2m$ in diameter. The ELSY project was led by Ansaldo Nucleare from Italy and was financed by Euratom. The 600 MWe design was nearly complete in 2008 and a small-scale demonstration facility was planned. This prototype runs on MOX fuel at $480°C$ with liquid lead pumped to eight steam generators with decay heat removal by convection. However this design was superseded by the ALFRED reactor. The leading developments in 2014 were the SVBR-100 and BREST-300 in Russia, 300 MWt ALFRED and MYRRHA in Europe with research focus on fuels and materials corrosion. In 2015 Westinghouse announced the submission of an LFR project proposal for USA investment in advanced reactor concepts.

A Lead-cooled Fast Reactor is a very complex machine and its numerical simulation is a very complex task. A platform simulation for LFR technology should have multi-physics and multiscale capability. For this reason a computational platform based on

open-source SALOME software has been developed in these years as a collaboration between the University of Bologna and ENEA with the purpose of studying issues from LFR technology [10, 12, 13, 14, 15]. This platform mimics for Lead-cooled Fast Reactors (LFR) 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 (LWR). 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. We have used this open-software platform to add new codes and develop coupling interface compatible with open and closed source codes. In particular in this platform we have implemented interfaces for FEMLCORE and CATHARE code in order to pass data between them. The interested reader can find this development in [14, 15]. During the analysis of a lead-cooled facility three-dimensional effects cannot be ignored and its features cannot be modeled by simple volumetric balances of energy, momentum and mass. However the CATHARE code and other one-dimensional system codes have been developed long time ago and during this period continuously updated and verified by many experiments [19, 21]. The coupling techniques between system and three-dimensional codes gives a good opportunity to explore more complex problems but great difficulties are added from combining multi-dimensional and overlapping meshes together. The input-output formats for mesh and solutions should be compatible in order to exchange data at each time step. The SALOME platform and in particular the MED library gives the standard needed for compatibility.

Chapter 2 is devoted to a brief description of the computational platform used in this report to study a multi-physics multi-scale system.

In Chapter 3 we introduce the one-dimensional model of the TALL-3D Facility and its computation without coupling.

Chapter 4 is dedicated to the three-dimensional test section. A first one-way coupling from the one-dimensional model to the three-dimensional test section is studied.

Full coupling studies are performed in Chapter 5 by using the defective coupling algorithm. The three-dimensional test section coupled with the primary one-dimensional loop which consists of three vertical legs is simulated by using the FEMLCORE and the CATHARE codes coupled on the SALOME platform. Details on CATHARE and FEMLCORE meshes and on the supervisor code are described.

1 FEMLCORE and CATHARE on SALOME platform

This platform mimics for Lead-cooled Fast Reactors (LFR) 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 (LWR). 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 closed 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.

1.1 SALOME Platform

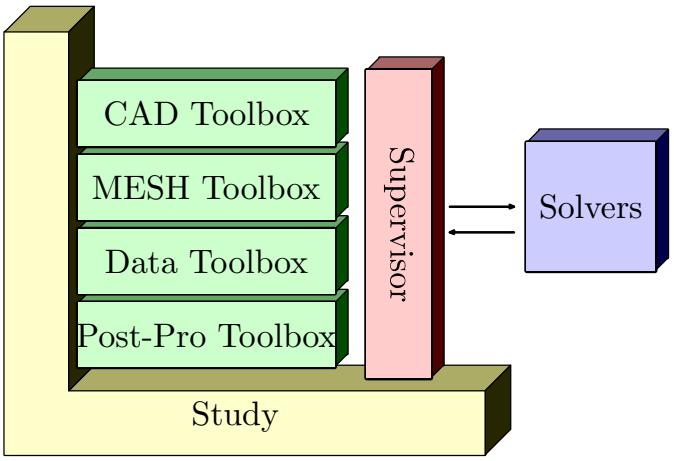


Figure 1.1: Layer structure of SALOME architecture.

In this section we briefly describe the computational platform used in this report to study the TALL-3D facility. 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.

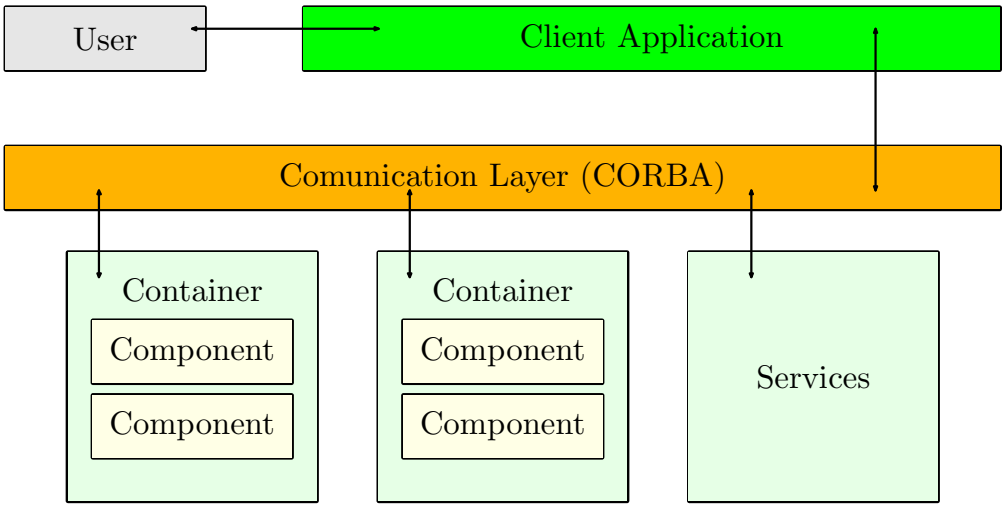


Figure 1.2: Layer structure of SALOME communication algorithm.

The collaborative project NURESAFE for the development of reliable software usable for safety nuclear reactor analysis, has been funded by the European 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 incompatible input and output formats [10]. CATHARE is a Code for the Analysis of Thermal Hydraulics during an Accident of Reactor and safety Evaluation for LWR [17]. 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 [18, 19]. TRIO_U and NEPTUNE codes solve the flow equations in multidimensional geometries with particular attention to two-phase flows [20]. 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 [19, 21]. 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 1.1. 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 1.2. 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 are 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 computational 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 [22].

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 a 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 remaining 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 FindOrStartPVServer, 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. Basically, 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, neutronics and thermo-structural analysis 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.

1.2 Code integration

In this Section we describe briefly 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. For details the interested reader can see [10, 12, 13, 14, 15]. 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 closed 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.

1.3 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.

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 content. 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 1.3, 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

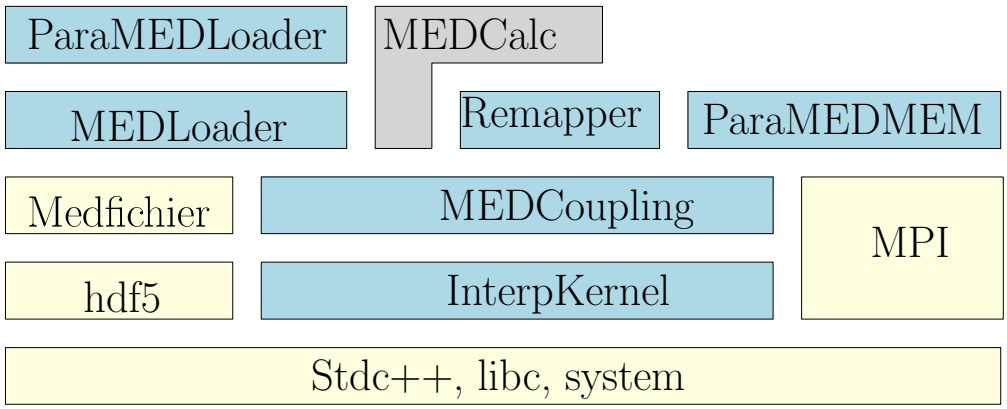


Figure 1.3: Layer structure of the packages of the library.

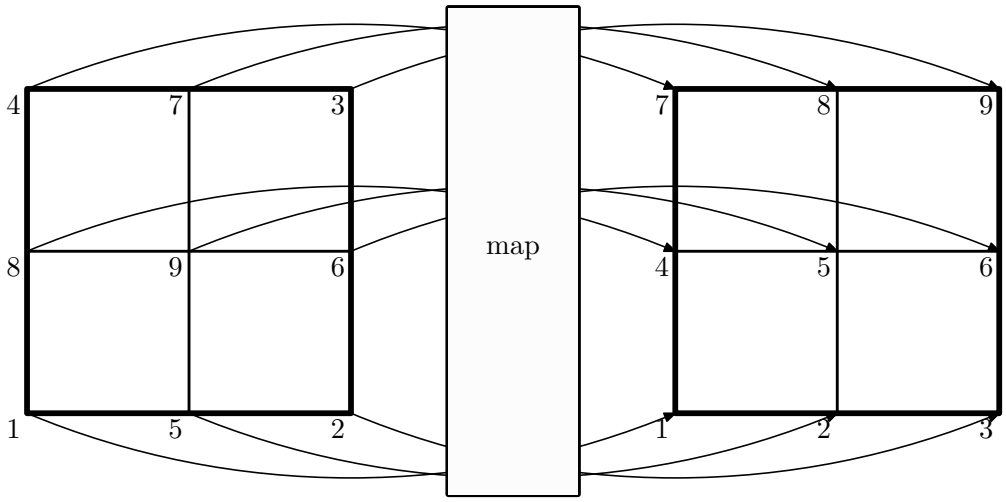


Figure 1.4: Operator that maps each node of a generic element (left) into its duplicate in med format (right).

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 1.4. The generated map can be used both for projecting a solution coming from the specific code into the MED support but also to 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.

1.4 SALOME interface

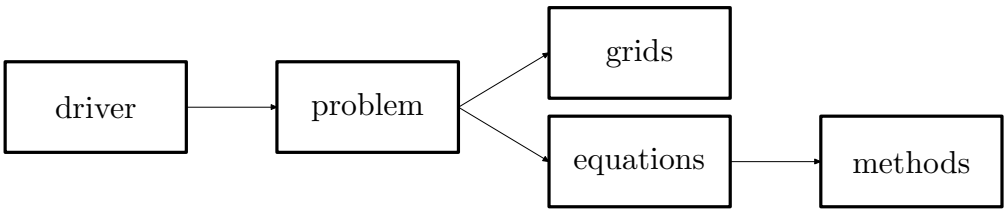


Figure 1.5: Collaboration diagram of generic CFD interface.

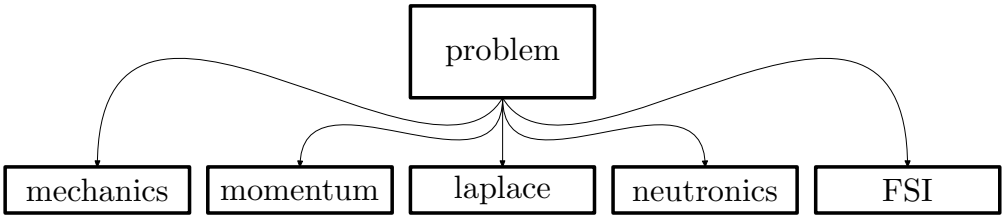


Figure 1.6: Inheritance diagram of the *problem* class.

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 1.5, 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 1.6. 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 1.5.

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. For details on CATHARE and FEMLCORE interface classes one can see [13, 14, 15].

2 TALL-3D facility on 1D system model

2.1 Mesh and code system modeling

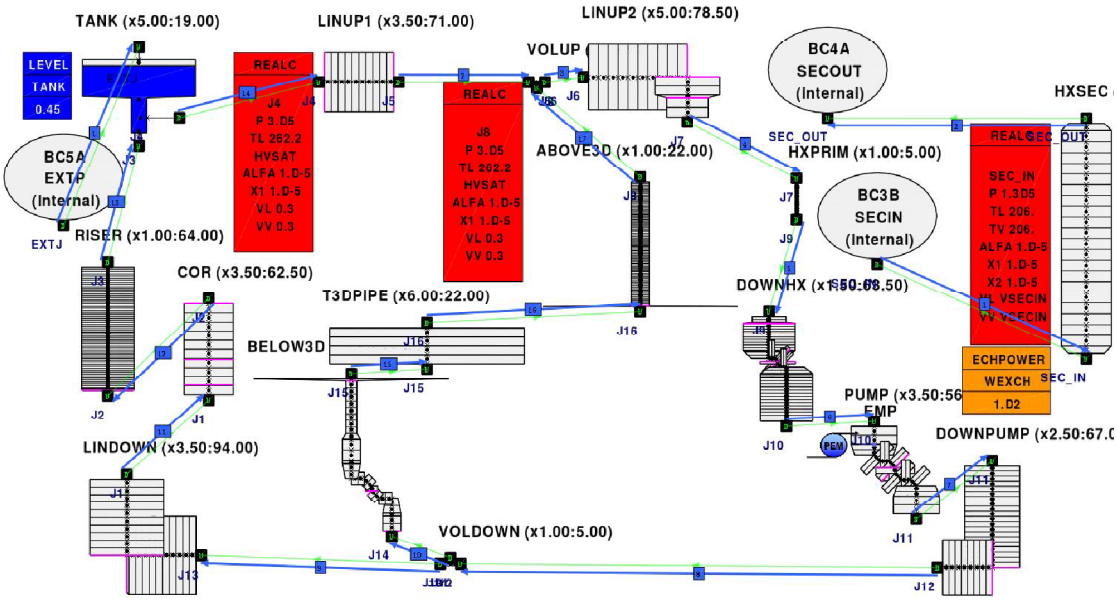


Figure 2.1: One-dimensional CATHARE model for the TALL-3D facility.

In this report we study the transient behavior of the TALL-3D facility with the FEMLCORE/CATHARE/SALOME computational platform developed in [15, 9, 10, 11, 12, 13, 14, 15, 16] and briefly discussed in the previous section. A simple model of the TALL-3D facility for a one-dimensional system code computation is shown in Figure 2.1. For this one-dimensional simulation of the whole system we propose CATHARE 2 code. Other software present in the LFR platform may be used as well.

A simple schematic of the TALL-3D facility is reported in Figure 2.2. The facility consists of a LBE-cooled primary (on the left) and an oil-cooled secondary loop (on the right). As one can see in Figure 2.2 the primary loop is rather complex and consists of three legs. The total height of the facility, which operates with Lead-Bismuth Eutectic (LBE), is about $6.5m$. The LBE choice is due to the low melting point ($125°C$) which makes its application preferential in experiments with high temperature sensitive instrumentation. The secondary loop is used to control heat balance in the primary loop. The total electric power is about $80kW$, with $27kW$ in the Main Heater section (MH) and $15kW$

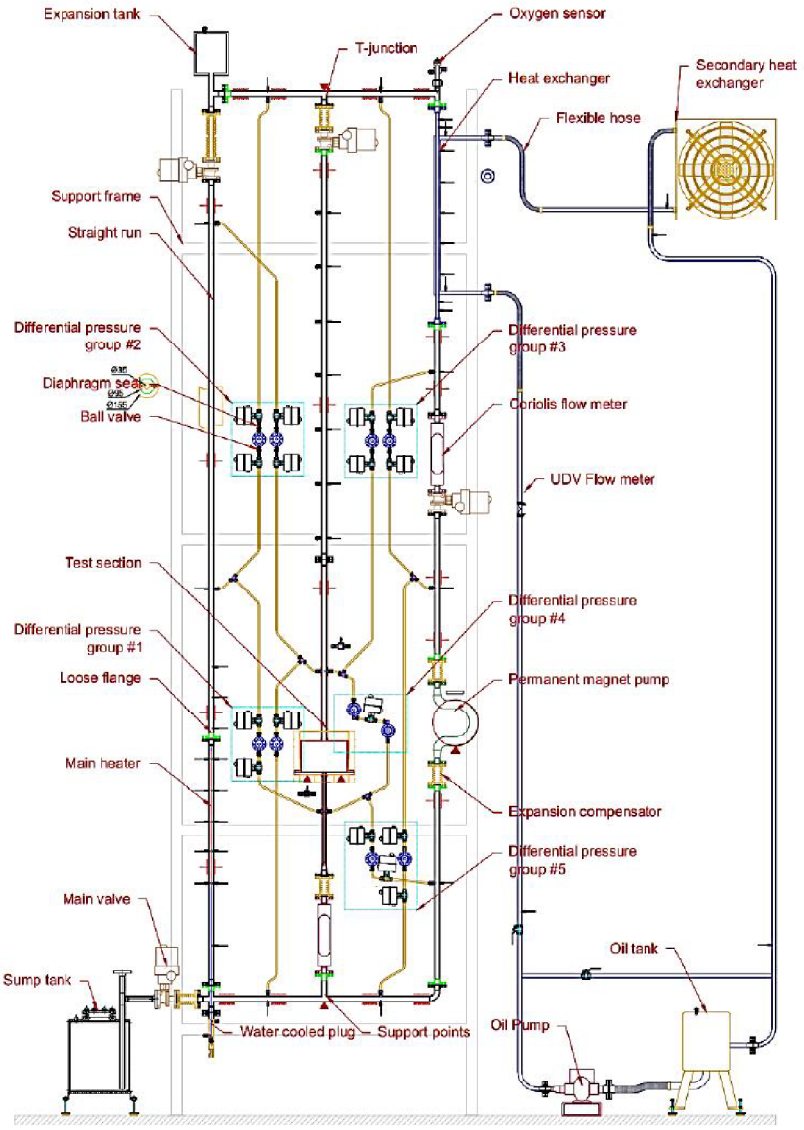


Figure 2.2: Schematic of the TALL-3D facility.

in the 3D-test section heater. The maximum measurable LBE flow velocity is $5kg/s$ in forced circulation and $0.6kg$ in natural circulation in the Heat Exchanger (HX) leg. The maximum LBE temperature is $460°C$ in the hot side and $350°C$ in the cold side. The secondary loop can be operated at temperatures in the range of $50 - 300°C$. Maximum temperature difference across the heat exchanger can be $90°C$. The static pressure at the top is $1.3bar$ while at the bottom can reach $7.8bar$. The maximum hydrostatic head from the EPM pump is therefore $2bar$. For details on TALL-3D facility the interested reader

can see [1, 2, 3, 4, 5, 6, 7]. Geometry and dimension of the primary loop are shown in

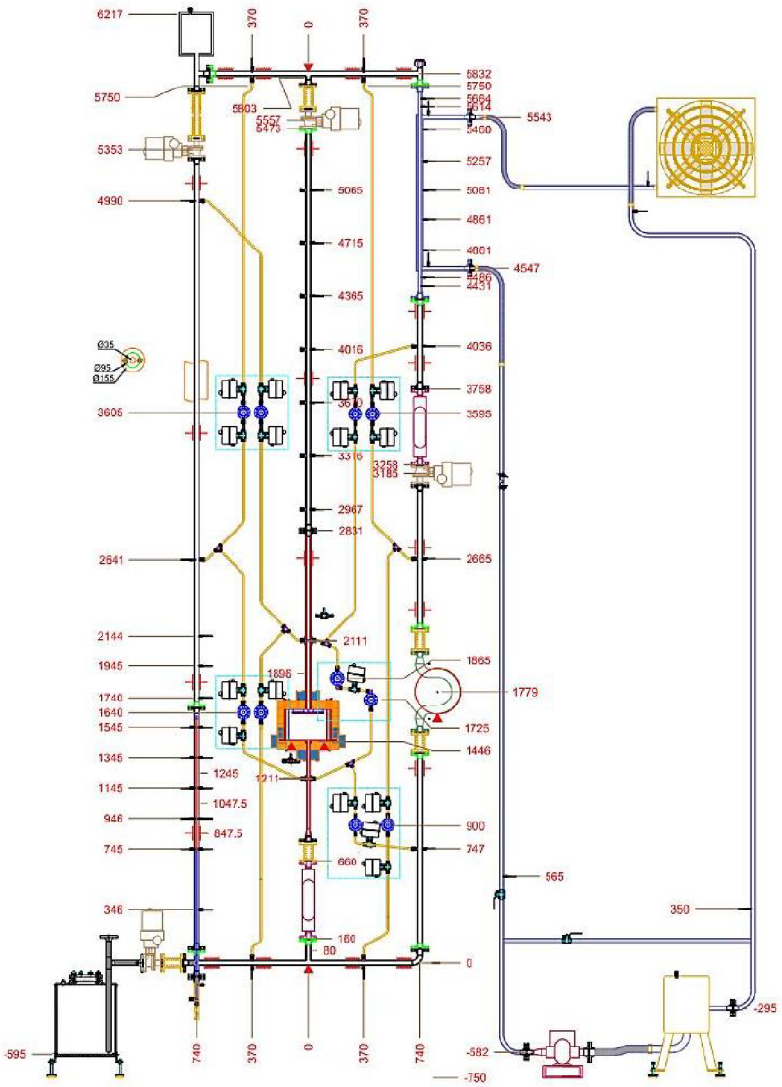


Figure 2.3: Geometry and dimension primary loop.

Figure 2.3.

Thermal hydraulics of this primary loop is the main subject of the numerical test for the 3D-1D coupling and validation of the code interfaces. The primary loop consists of the sump tank used to store, melt and supply LBE into the main loop, 3 vertical legs and 2 connecting horizontal sections. The distance between the adjacent vertical legs is $0.74m$ and the distance between the horizontal sections is $5.83m$; the nominal pipe Inner

Diameter (ID) is 27.86 mm. For further details one can see Figure 2.3. The facility is divided onto several sections, each section is equipped with different devices to measure differential pressure, temperature and mass flow rate [1, 2, 3, 4, 5, 6, 7].

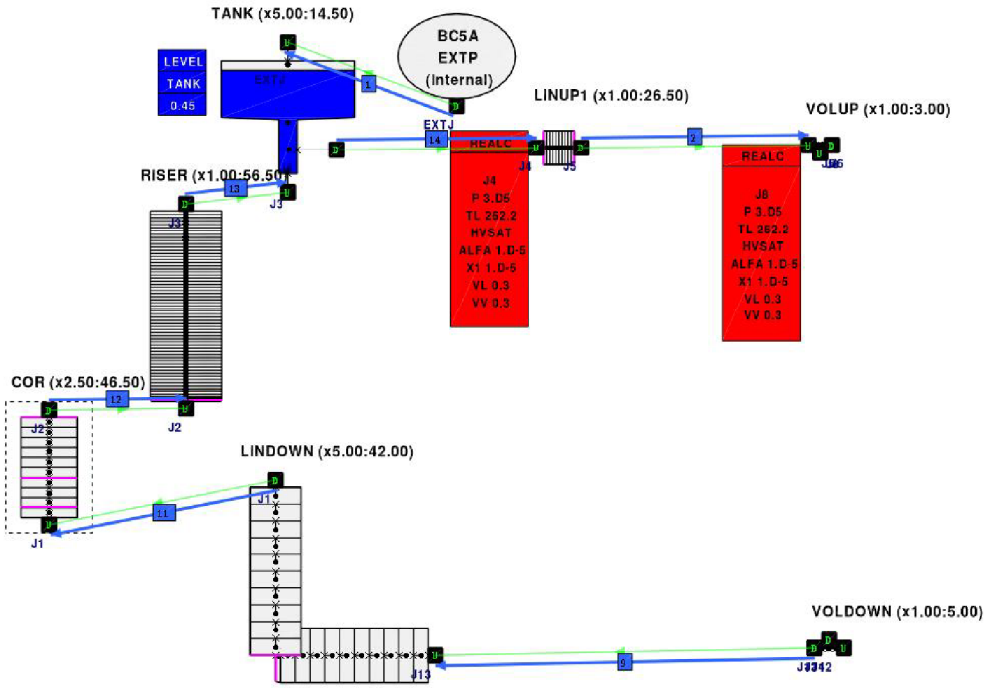


Figure 2.4: Left vertical leg. One-dimensional system model which consists of the LINDOWN, COR (Main heater) and RISER module.

The 1D model of the left vertical legs is shown in Figure 2.4. In the left vertical leg there is the main heater. The left leg is connected on the top to the main tank and on the bottom to the sump tank. The Main Heater (labeled with MH) is a rod-like electric heating element. Its maximum electric power is $27kW$. The MH geometry and the arrangement of thermocouples are provided in Figure 2.5. The heated part is indicated with red color, its length is $870mm$. Several thermocouples (TC) are located at different elevations with $200mm$ pitch. The labels can be easily read. The last point separated number in the name of the TCs indicates the distance of the thermocouple tip from the outer surface of the heater in the units of 0.1 mm. For example, the the thermocouple TC1.0745.23 is located at $1.745m$ height with $2.3mm$ away from the outer surface of the main heater. There are six thermocouples clamped to the outer surface of the MH pipe, see Figure 2.5 (on the left).

The 1D model of the central vertical leg is shown in Figure 2.4. This leg is the key part of the circuit for our test since this contains the 3D test section. It consists of three AXIALS: the ABOVE3D, BELOW3D and 3DPIPE module. TALL-3D test section is an axisymmetric cylindrical stainless steel vessel with an inlet at the bottom and an outlet at the top (see Figure 2.7). The upper part of the test section is equipped with

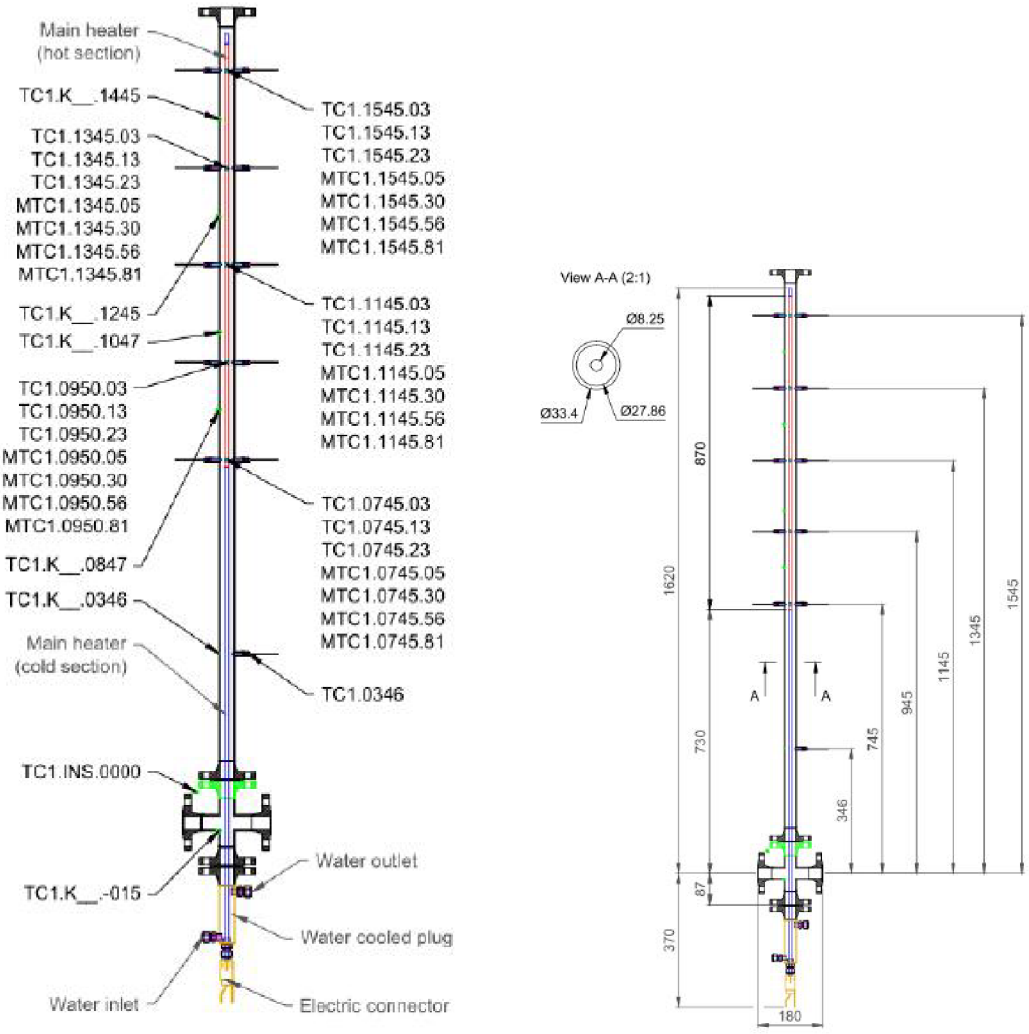


Figure 2.5: Left vertical leg. Main heater (MH) in the 1D system model.

two $7.5kW$ line heaters rolled jointly around the circumference. The heaters enhance the development of thermal stratification in the LBE pool. Inside the three-dimensional test section, a circular plate is placed orthogonally to the flow path in order to enhance pool mixing by deflecting the inlet flow. The plate is attached to the ceiling of the pool with four fin-shape separators, which are designed to inflict minimal disturbance on the flow. Two types of thermal insulation are used around the test section: ISOVER TapeLock 7300 and Nano T Ultra. They are labeled and color-marked on the left part of Figure 2.7. The simulations predict the pool to be fully mixed at $0.7kg/s$ LBE flow rate and stratified at $0.3kg/s$ LBE flow rate. The dimensions of the test section and thermal insulation are provided on the right part of Figure 2.7. The nomenclature and locations of the thermocouples are indicated in the Figure 2.7. In total 159 temperature

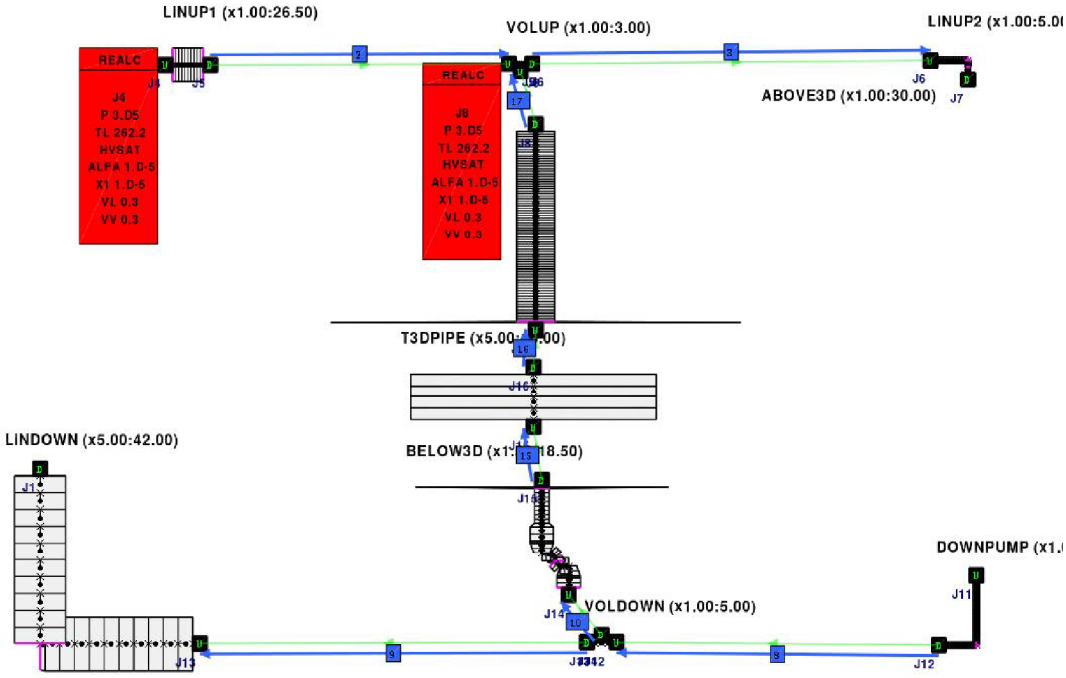


Figure 2.6: Central vertical leg in 1D system model with 3D test section.

measurements are provided including 114 inside the LBE pool, four on the outer surface of the thermal insulation and one on the outer surface of the bottom plate.

The 1D model of the right vertical leg is shown in Figure 2.8. This leg contains the heat exchanger to control the overheating in the system. It consists of five AXIALS: the LINUP2, HXPRIM, HXDOWN, PUMP and PUMPDOWN module. The axial LINUP2 and PUMPDOWN are connected with the top and bottom section of the loop. The Heat Exchanger (HX) is located at the top of the HX leg and used to maintain the heat balance in the primary loop. The geometry of the heat exchanger and the location of the measurement points (thermocouples) are provided in the Figure 2.10. A sketch of the EPM pump is given in Figure 2.9. The pump consists of the three components: a 5.5 kW AC motor, a fixed semicircular steel channel with rectangular cross-section and two magnetic discs. The inlet and outlet of the pump are connected to the main loop. The discs are rotated by the AC motor. The magnetic force induced by the discs on LBE inside the steel channel drives the flow. Rotation speed and direction are controlled by remote commands. During pump operation heat is generated in the steel channel and the flowing media. The amount of the generated heat is dependent on the rotation speed of the discs. During forced circulation, at nominal $4.5kg/s$ mass flow rate, the LBE flow receives about $500W$ of thermal energy. During natural circulation the discs do not rotate and therefore do not produce heat. The pump is the only loop element that could not be thermally insulated. A thermocouple is flashed to the outer surface of

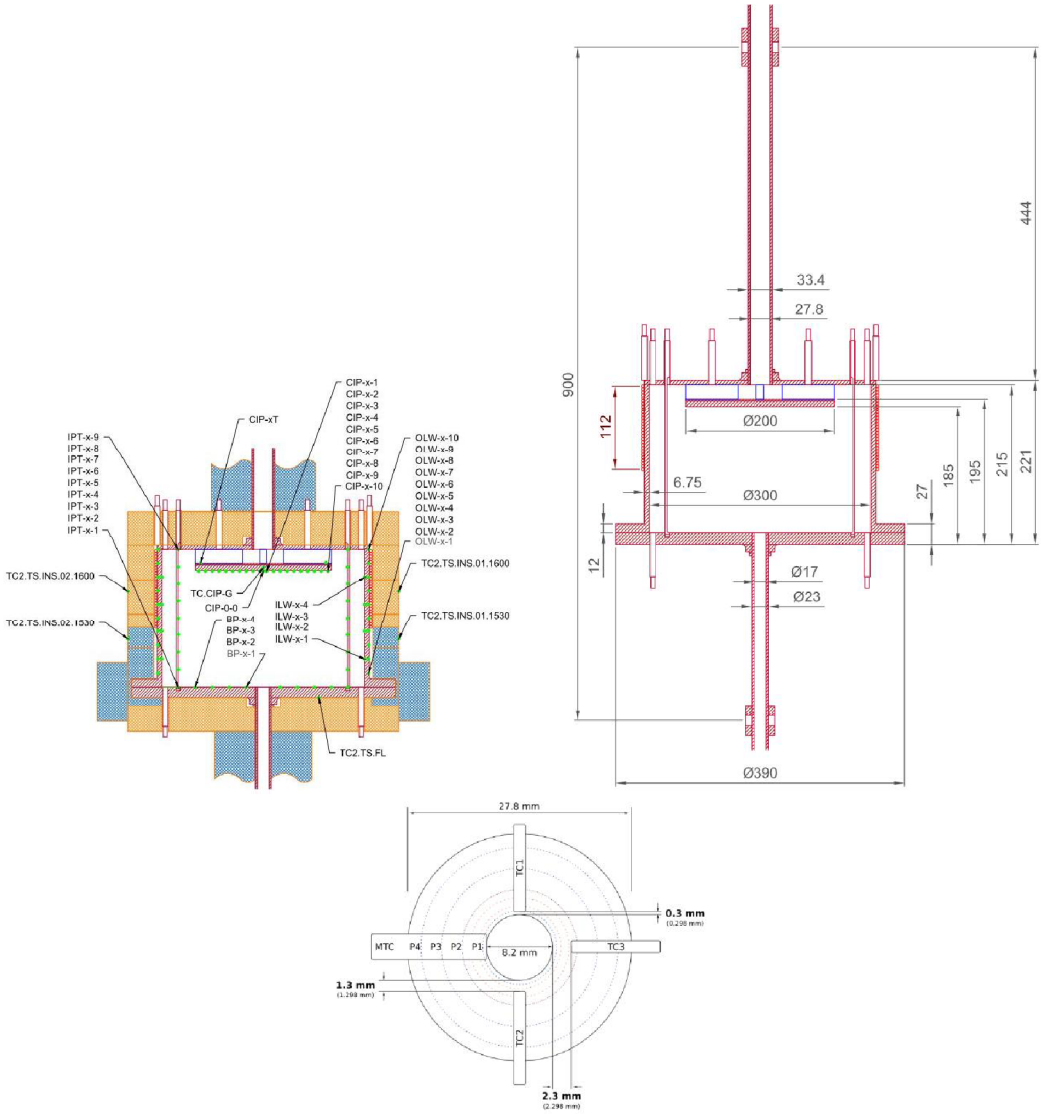


Figure 2.7: The three-dimensional test section: material, dimension and geometry. In one-dimensional simulations the test section is modeled by the axial 3DPIPE module.

the steel channel to monitor the local temperature.

The Heat Exchanger (HX) leg (right) consists of a counter-current double-pipe heat exchanger placed at the top and an Electric Permanent Magnet (EPM) pump used for forced circulation of LBE. The heat exchanger is the common component for primary and secondary sides. The 3D central leg connects a pool-type 3D test section to the loop. Depending on instantaneous flow characteristics, the LBE inside the test section can undergo thermal mixing or stratification. Common flow direction in the primary loop is

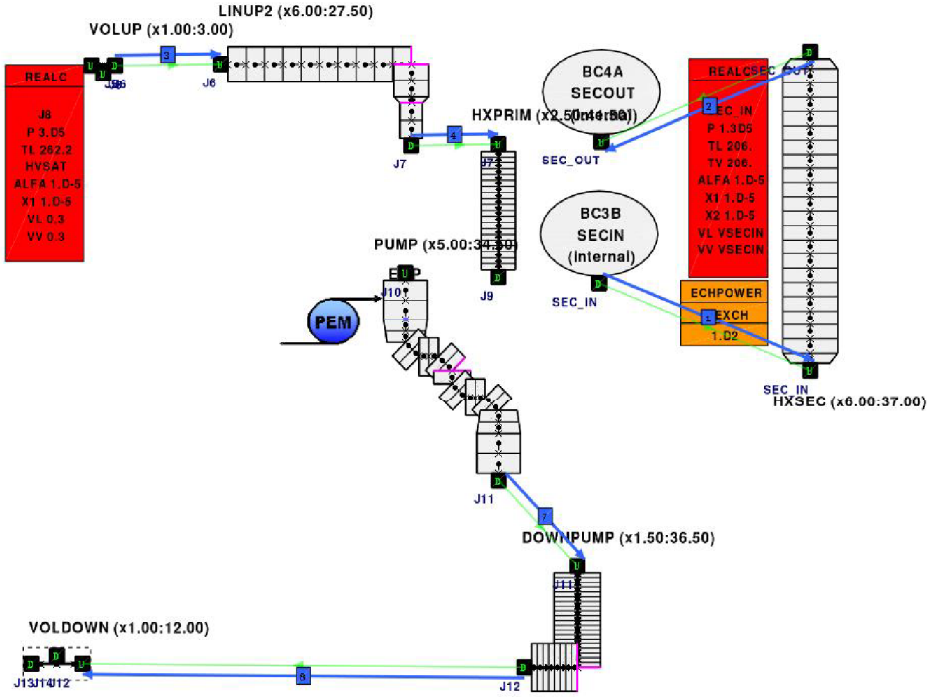


Figure 2.8: Right vertical leg in 1D system model.

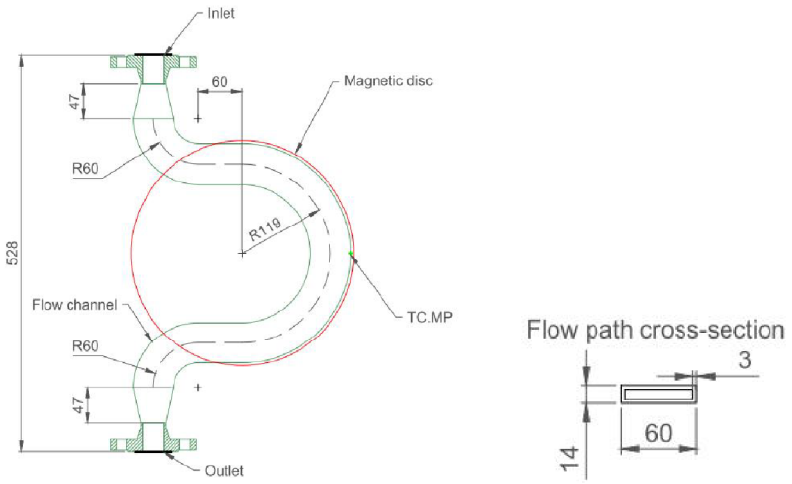


Figure 2.9: Right vertical leg (with HX). Pump geometry and steel channel with rectangular cross-section.

downwards in HX leg, upwards in the MH leg and upwards in the 3D leg. Therefore the loop section below the heat exchanger up to the 3D test section and the main heater is often referred to as the cold side of the main loop; and the part of the loop above the

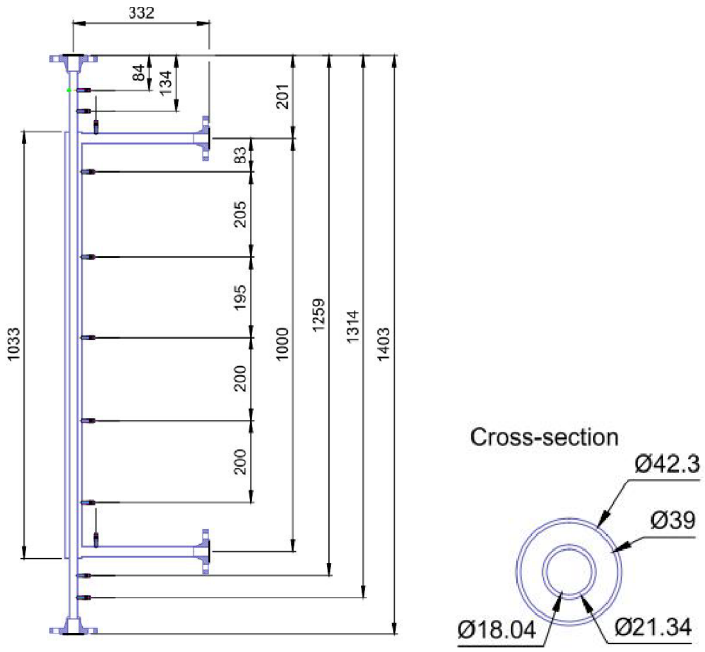


Figure 2.10: Right vertical leg. Heater exchanger: vertical (left) and section (left) layout.

3D test section and the main heater is called the hot side of the main loop. LBE mass flow meters as temperature sensitive equipment are located in the cold side. Vertical legs are equipped with ball valves for fine tuning of the hydraulic resistances and to provide possibility to cut off a specific leg. Elevations of the main loop components are provided in Figure 2.3. These elevations are used in the nomenclature for instrumentation and equipment. LBE flow temperature around the loop is measured at 23 locations. Differential pressure can be measured around any of the 12 primary loop sections. Details of the differential pressure measurement system and facility sectioning can be found in [1, 2, 3].

During loop operation temperatures up to $500°C$, pressures up to $0.7MPa$ and flow rates up to about $5kg/s$, which corresponds to a flow velocity of approx. $1.7m/s$, can be achieved in HX leg. Oxygen control system is implemented to monitor chemical potential of the LBE dissolved oxygen. It is required to stay within $1.5e-7$ and $1.85e-5\,wt\%$ to stabilize the protective oxide layer on the LBE wet components. The sensor is YSZ membrane with oxygen saturated Bi reference. The minimal operation temperature is determined by the membrane permeability which becomes active at $360°C$, though the recommended temperature interval is $400-450°C$.

The top horizontal part of the LBE circuit, modeled in one-dimensional coordinate geometry consists of the main expansion tank (labeled with TANK), two AXIALS (LINUP1, LINUP2) and a volume which links this part of the top LBE circuit to the central vertical leg with the three-dimensional test section. The TANK volume and the LINUP2 are connected with the left (with th MH) and right vertical leg (with

HX), respectively. The expansion tank is used to to monitor the LBE level in the loop and maintain loop pressure during temperature induced deformation along the loop components. The bottom horizontal part of the LBE circuit, modeled with 1D coordinate geometry, consists of 2 axial modules (LINDOWN, DOWNPUMP) and a volume module (VOLDOWN) which links the bottom to the central vertical leg with the three-dimensional test section.

2.2 Initial state of the one-dimensional system code

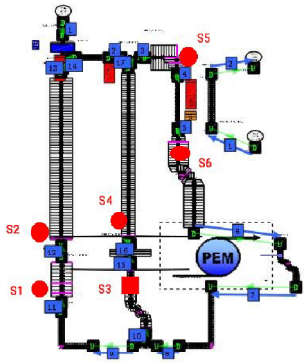


Figure 2.11: Point of interests $S1$-$S2$ of the left leg (on the left), $S3$-$S4$ of the central leg and $S5$-$S6$ of the right vertical leg.

In Figure 2.11 there are the reference points that can be used to compare the results of the state variable. We consider six points labeled by $S1$, $S2$, $S3$, $S4$, $S5$ and $S6$. The points $S1$ and $S2$ are located along the left vertical leg. The point $S1$ is in the COR module at the element mesh 3. We denote such a point with COR3 or $S1$. The point $S2$ is inside the RESERVE module at the element mesh 4. We denote such a point with RESRVE4. The points $S3$ and $S4$ are located in the central vertical leg. The point $S3$ is in the BELOW3D module at the element mesh 25. The point $S2$ is inside the ABOVE3D module at the element mesh 8. We denote such points with BELOW3D24 and ABOVE3D8, respectively. Finally the points $S5$ and $S6$ are located along the right vertical leg. The point $S5$ is in the LINUP2 module at the element mesh 14. The point $S6$ is in the PUMP module at the element mesh 0. In a similar way we denote such points with LINUP14 and PUMP, respectively. In order to set the initial conditions we need to compute an initial state which must satisfy the one-dimensional equation. It is usual to compute the steady state solution which satisfies the appropriate desired values. The test considered here is the $T01.09$ which reproduces a forced to natural circulation transient, namely an Unprotected Loss of Flow (ULOF). This is a transient where, after the pump trip, the power supplied through the rod in the left leg (MH) and 3D test section heater are not switched off and the fluid moves under buoyancy forces. In order

```
// ================================================================
// ==============  Problem c: initial state =======================
// ================================================================
double T_3=241.35;double v_3=0.6995;double flowrate=4.275;
double rho=10448;int step_count=1;bool stop=false;  bool ok=false;
double present=0.;double tmax=30000; double dt=1.0; double time= 0.;
double T2_old = T_out;
// ==========  time loop ==========================================
while(!stop) {
  ok=false;
  while(!ok && !stop) {
    present=c->presentTime(); if(present>=tmax) { stop=true; }
    if(stop) { break; }  // if stop ==true -> exit
    c->initTimeStep(dt);
    ok=c->solveTimeStep();
    // ------------------------------------------------------------
    if(!ok) { c->abortTimeStep(); } // no valid time step
    // ------------------------------------------------------------
    else {  // valid time step ------------------------------------
      time=present+dt;    c->validateTimeStep();
      T_3=c->getValue_CATHARE("LIQTEMP",point1.c_str(),ix_3, irad_3);
      T_4=c->getValue_CATHARE("LIQTEMP",point2.c_str(),ix_4,irad_4);
      flowrate=c->getValue_CATHARE("LIQFLOW","DOWNPUMP",1,1);
      tensalim = tensalim - 1.e+4*(flowrate-4.275);//pump alim
      c->setValue_CATHARE("TENSALIM","EMP",1,1,tensalim); // <-Pump
      ++step_count;if(step_count0 == 1501) {stop=true;}//update step
    }// end validateTimeStep ------------------------------------
  }
}
```

Figure 2.12: Problem C: steady state computation with EMP working.

to compute a possible initial state, the solution with the pump in working condition may be considered. A solution of this high nonlinear system is computed by using the coupling code for the Problem C shown in Figure 2.12. A steady state for the Problem C is computed. We starts with inlet temperature for the PIPE3D set to $241.35°C$. The flow rate is set to $4.275Kg/s$ with density $\rho = 10448Kg/m^3$. As one can see from Figure 2.12 the one-dimensional code has a variable time step. We start with $dt = 0.01$ and allow maximal value of $1s$. The code validates the step by using the logical variable *ok* and ends when the logical variable *stop* takes the *true* value. The PIPE3D temperatures, fluid flow and the EMP power ratio can be seen in Figure 2.14 as a function of time. In Figure 2.14 from the top left to the right bottom we can see the temperature on the left (S1-S2), central (S3-S4) right (S5-S6) vertical legs and the mass flow rate. The

	Initial Condition	Unit
T BELOW3D	241.45	$°C$
T ABOVE3D	259.81	$°C$
T LINUP2	251.76	$°C$
T PUMP	242.38	$°C$
T COR	244.16	$°C$
T RESERVE	248.33	$°C$
MH rod power	2578	W
3D vessel power	4833	W
LBE mass flowrate	4.2750	kg/s
Oil inlet temperature	61.1	$°C$
Oil outlet temperature	82.6	$°C$
Oil mass flowrate	0.1435	kg/s

Table 2.1: Initial state condition for the one-dimensional Problem C. The initial condition satisfies the steady state equation.

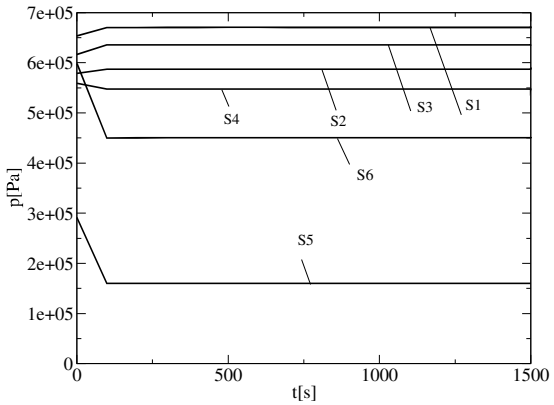


Figure 2.13: Pressure on the left (S1-S2), central (S3-S4) and right (S5-S6) vertical legs.

points on the legs are labeled by $S1$-$S2$ (left leg), $S3$-$S4$ (central leg) and $S5$-$S6$ (right leg). The liquid flow rate is shown only at $S1$, $S4$ and $S5$ each in a different leg. In the interval $[0, 1500]s$ the temperature reaches extreme values (region I). After $1500s$ the state does not change and can be considered steady. The main initial conditions in forced circulation are reported in Table 2.1.

2.3 Standalone CATHARE simulation of the TALL-3D facility

In this section we report the results of the simulation of the evolution of an Unprotected Loss of Flow going from forced to natural circulation flow. We consider the system

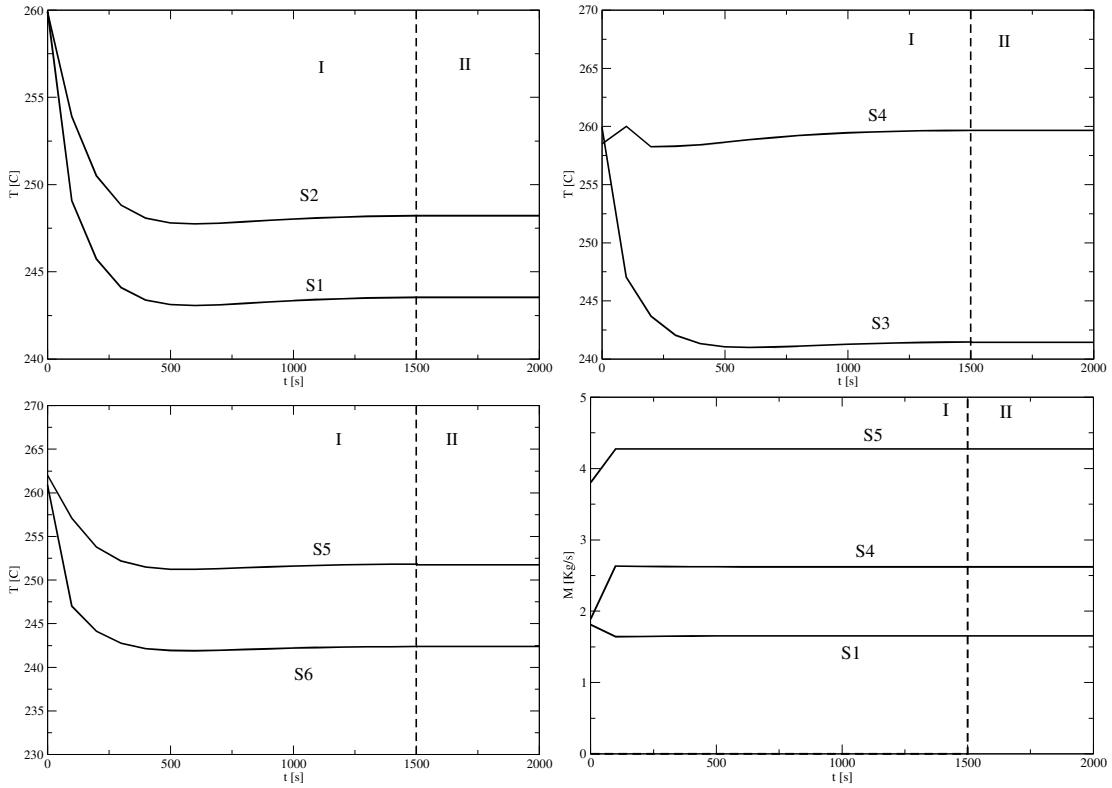
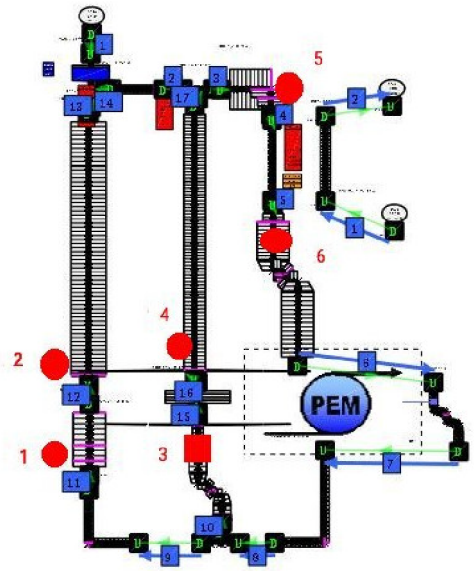


Figure 2.14: Temperature on the left (S1-S2), central (S3-S4) and right (S5-S6) vertical legs. On the right bottom the liquid flow rate during the computation of the initial state (steady state) for the one-dimensional system code.

at $t = 0^-$ in fully working conditions. At $t = 1$ the pump stops working while the power supplied through the rod in MH leg and 3D vessel heater are not switched off. As explain in Section 2.2 the main initial conditions in forced circulation are obtained as a steady state and they are shown in Table 2.1. The test is characterized by almost constant boundary conditions, also during the transient phase. To approach the correct heat transfer conditions with LBE in the secondary side, the initial LBE temperature of $206°C$ is reduced to $199°C$ in the first 10 s of transient. The mass flow rate, kept constant during the test, is set to $1.752kg/s$. Due to the lack of secondary side mass flow rate measurement, the oil flow rate has been calculated starting from the loss of enthalpy in the primary side, then with a thermal balance in the secondary side. The flow rate evolutions in the three legs are presented in Figures 2.16–2.18. The mass flow rate evolutions in the left and central vertical leg are presented on the left and on the right of Figure 2.16, respectively. Here we do not report the experimental data which can be found in [1, 2, 3, 4, 5, 6, 7]. However differences between the simulations and experimental data can be observed.

The magnitude and frequency of the external mass flow rate oscillations, in which the left and the right leg reach the reverse flow conditions, are well captured. The following



	Final Condition	Unit
T BELOW3D	218.01	$°C$
T ABOVE3D	314.93	$°C$
T LINUP2	299.41	$°C$
T PUMP	223.53	$°C$
T COR	241.89	$°C$
T RESERVE	298.58	$°C$
MH rod power	2578	W
3D vessel power	4833	W
LBE mass flowrate	0.53054	kg/s
Oil inlet temperature	61.2	$°C$
Oil outlet temperature	80.6	$°C$
Oil mass flowrate	0.1404	kg/s

Figure 2.15: Final state condition for the one-dimensional Problem C solved without coupling. The final condition satisfies the steady state equation.

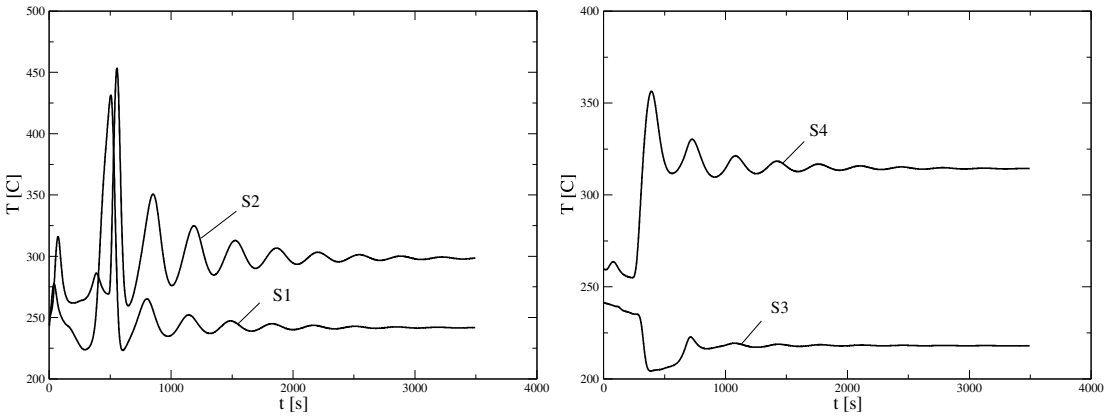


Figure 2.16: Temperature at points $S1$-$S2$ of the left leg (on the left) and at $S3$-$S4$ of the central leg (on the right) as a function of time t.

oscillations are a little bit ahead of time in frequency and underestimated in terms of amplitude. The temperatures of the secondary side cannot be directly compared, since the absolute temperatures of LBE are much higher than the oil in the experiments. During the initial transient phase, it is evident the discrepancy of the heat exchanging condition, even though the thermal balances are reliable only in steady-state conditions. The facility heat losses during the transient are almost constant at about 2.2 kW. The lower heat exchange affects the temperature evolutions as can be seen in the pictures. The experimental inlet temperature increases during the reverse flow phase due to mixing

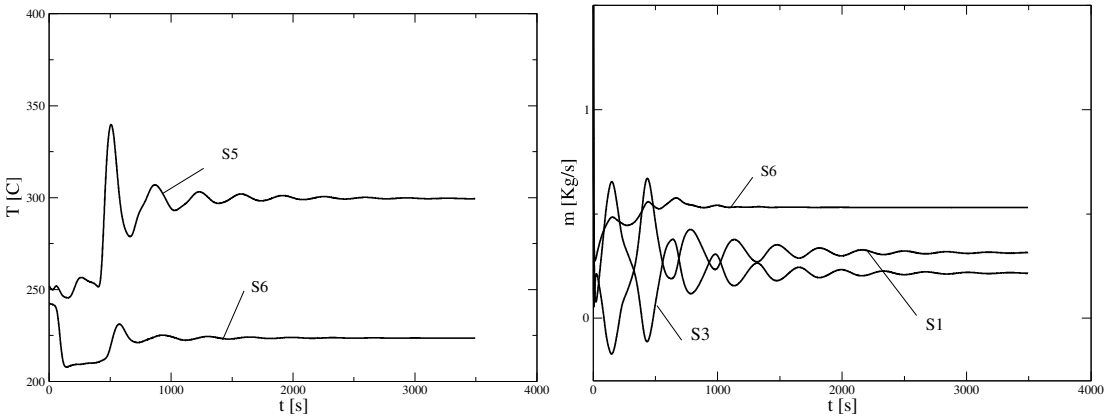


Figure 2.17: Temperature at the points $S1$-$S2$ of the right leg (on the left) and fluid flow rate at $S1$ (left), $S3$ (central) and $S4$ (right leg) (on the right) as a function of time t.

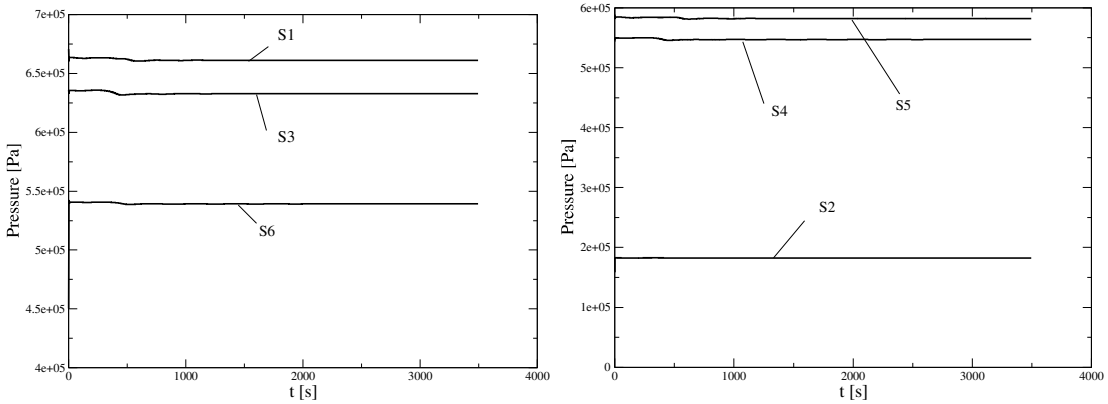


Figure 2.18: Pressure at key points $S1$-$S2$ (left leg) $S3$-$S4$ (central leg) and $S5$-$S6$ (right leg) as a function of time t.

effects in 3D vessel. This behavior cannot be caught by system codes. The differential pressure in Figure 2.18 shows a quite good agreement between the experimental values and the simulations. The final conditions in natural circulation are reported in the table on the right of Figure 2.15. Details experimental data and comparison are reported in [1, 2, 3, 4, 5, 6, 7].

In the rest of the section we report the experimental results. These are shown with a previous CATHARE model which is the same one that we use in the CATHARE standalone case. Mass flowrate from experimental data is shown in Figure 2.19 on different legs (FM1,FM2,FM3). The experimental data are label with OFFSET. The left, central and right vertical legs are labeled with FM1,FM2 and FM3, respectively. In Figure 2.20–2.21 temperature at the thermocouple points TC3_4490 and TC3_5615 along the left vertical leg (left), at TC1_0346 and TC1_1740 on the right vertical leg

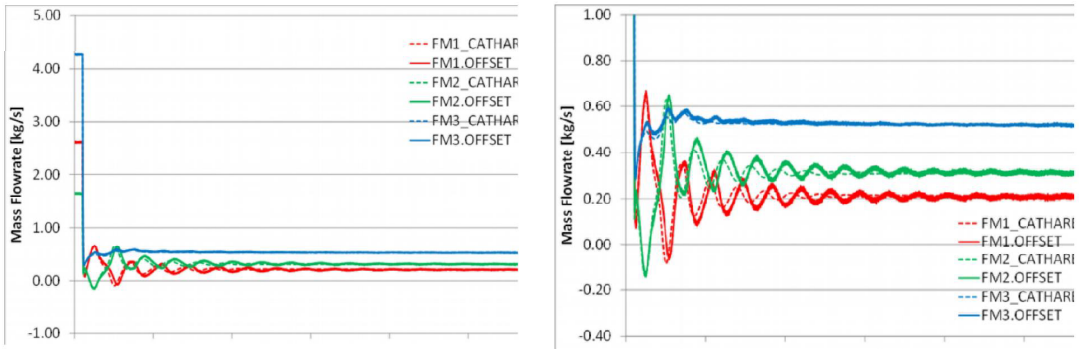


Figure 2.19: Mass flowrate on different legs (FM1,FM2,FM3) from experimental data (OFFSET).

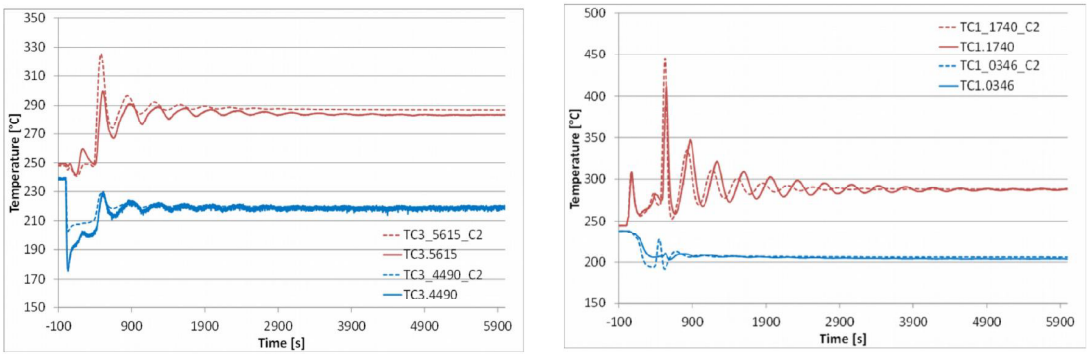


Figure 2.20: Temperature at the thermocouple points TC3_4490, TC3_5615 along the left vertical leg (left) and at TC1_0346, TC1_1740 on the right vertical leg (right) from experimental data.

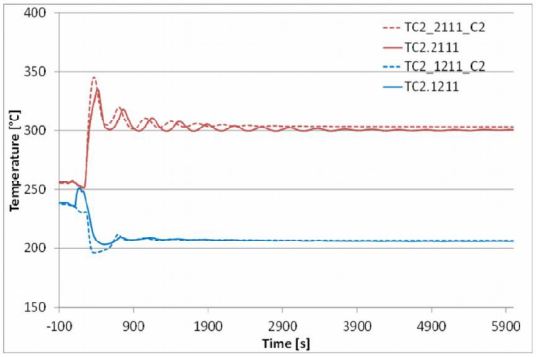


Figure 2.21: Temperature at the thermocouple points TC_1211, TC2_2111 along the central vertical leg (left) from experimental data.

(right) and at TC_1211, (TC2_2111 along the central vertical leg (left) from experimental

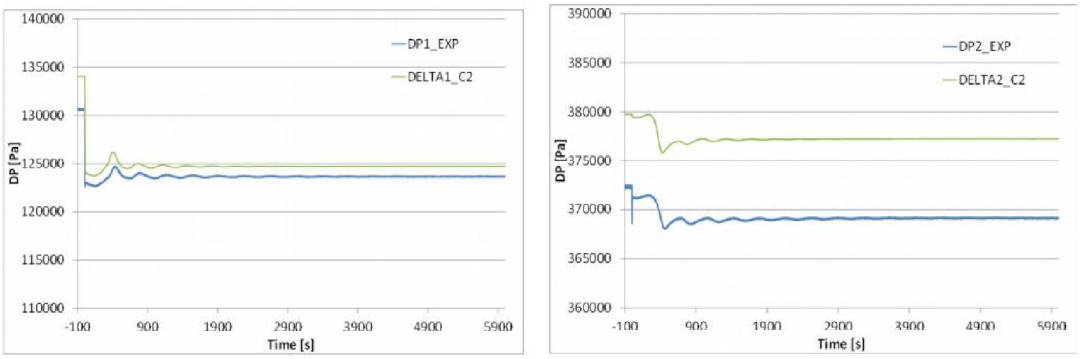


Figure 2.22: Differential pressure $DP1$ (left) $DP2$ (right) from experimental data.

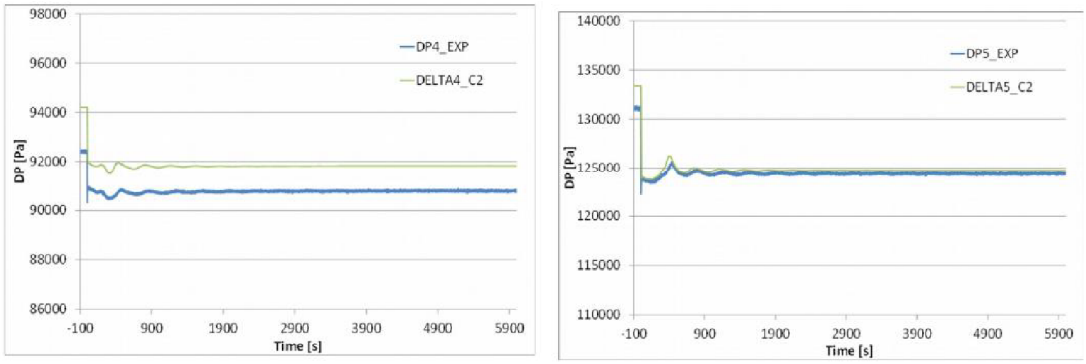


Figure 2.23: Differential pressure $DP4$ (left) $DP5$ (right) from experimental data.

data. Finally experimental differential pressure $DP1$ (left) $DP2$ (right) $DP4$ (left) $DP5$ (right) are shown in Figure 2.22-2.23.

3 TALL-3D test section in FEMLCORE simulation

3.1 Mesh and geometry

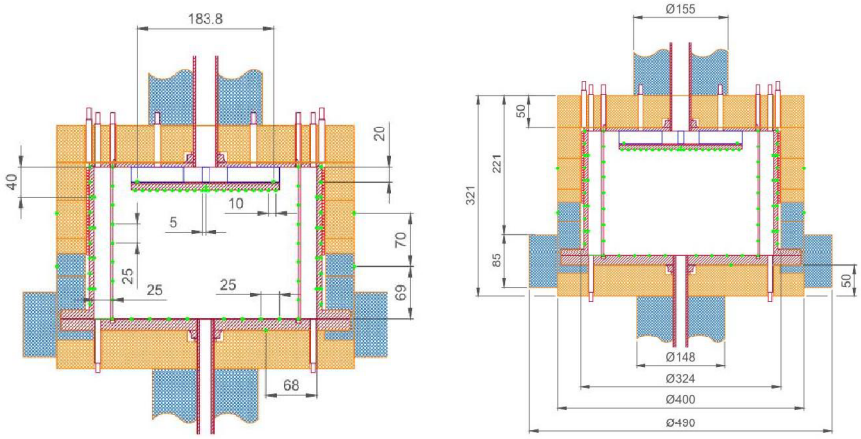


Figure 3.1: Geometry and dimension of the three-dimensional test section.

TALL-3D test section is an axisymmetric cylindrical stainless steel vessel with an inlet at the bottom and an outlet at the top. The upper part of the test section is equipped with two $7.5kW$ rope heaters rolled jointly around the circumference. The heaters promote the development of thermal stratification in the LBE pool. Inside the test section, a circular plate (inner plate) is placed orthogonally to the flow path in order to enhance pool mixing by deflecting the inlet flow to periphery. The plate is attached to the ceiling of the pool with 4 fin-shape separators, which are designed to inflict minimal disturbance on the flow.

Scaling analysis and CFD simulations were carried out in order to determine pool sizes, heater power and inlet pipe diameter. The dimensions of the test section and thermal insulation are provided in Figure 3.1. Note that internal diameter (ID) of the inlet pipe is smaller than ID of the pipes throughout the rest of the loop. This is required to ensure sufficient inlet flow velocities for pool mixing in natural circulation. CFD simulations of the test section transient pool characteristics were used to identify locations of the temperature measurements that are most sensitive to the flow characteristics, provide data on development of stratification and mixing of the pool, and could be used to assess the heat fluxes through the walls and thermal inertia of the test section structures. The

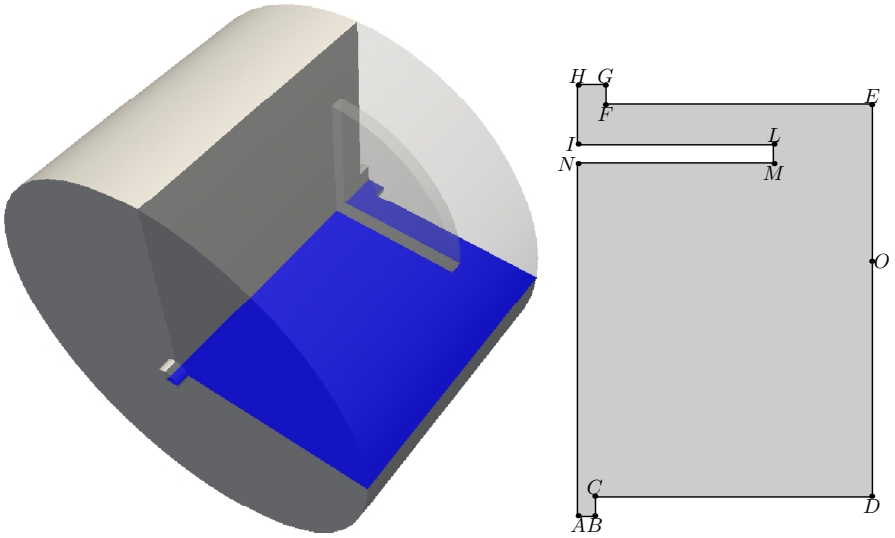


Figure 3.2: Three dimensional computational domain (on the left) and generic section (on the right).

nomenclature and locations of the thermocouples is indicated in the previous section. In total 159 temperature measurements are provided including 114 inside the LBE pool, 4 on the outer surface of the thermal insulation and 1 on the outer surface of the bottom plate.

The computational domain used for the three-dimensional description of the facility is show in the left part of Figure 3.2 and a more detailed scheme of the generic section of the axisymmetric geometry is shown on the right part of the same figure. In particular the points A, B, C, D, E, F, G, H, I, L, M, N, and O are located at $(x, y) = (0, -0.01)$, $(0.0085, -0.01)$, $(0.0085, 0.)$, $(0.15, 0.)$, $(0.15, 0.2)$, $(0.0139, 0.2)$, $(0.0139, 0.21)$, $(0., 0.21)$, $(0., 0.18)$, $(0.1, 0.18)$, $(0.1, 0.17)$, $(0., 0.17)$ and $(0.15, 0.0904118)$, respectively. According to the nomenclature shown in the right part of Figure 3.2, the segment $\overline{AB}$ is the inlet region of the domain while the segment $\overline{HG}$ is the outlet zone of the domain and finally the segment $\overline{AH}$ is the symmetry axis of the geometry. The coarse computational grid used for the discretization of the geometry is composed by quadratic elements characterized by an average edge length of $2.5mm$. Refinement tools are available to increase the number of point in selected regions.

3.2 Initial conditions for CFD in 3D-section

The coupling between the system code and the three-dimensional problem is achieved by a mutual exchange of boundary conditions between the problems. In order to produce a stable coupling between the two problems both of the systems must reach the same thermodynamical working conditions. For this reason we do not couple the system code and the three-dimensional problem from the beginning of the simulation but instead, we

```
#ifdef USE_FEMUS_STAB
  std::ostringstream a; a<< T_3;
  P.setAnalyticBoundaryValues(400,1, a.str().c_str());
  // write inside x_old the boundary values
  P.write_Boundary_value(400,"T",1);
  int step_count1 = 1+restart;
  // stabilization step nsteps1
  int nsteps1 = 5000;
  double dt1 = dt;
  for(int i=0; i < nsteps1; i++) {
    time1 += dt1;
    // velocity coupling  v_3(CATHARE) with interface 21 (bottom)
    double vlocal =  48./37.*v_3;
    std::ostringstream vel; vel<< "IVec*0.+JVec*"<<vlocal;
    // write in the interfacefunction.field
    P.setAnalyticBoundaryValues(401,2, vel.str().c_str());
    // write inside x_old the boundary values
    P.write_Boundary_value(401,"NS0",2);
    P.solve_onestep(itime_0,step_count1,print_step,time1,dt1);
    T_out=P.getAvOnBoundary_nodes(120, "T",0);
    step_count1++;
  }
#endif //FEMUS STAB
```

Figure 3.3: Problem c: steady state computation with EMP working.

firstly find the the nominal steady state working condition for the one-dimensional system and then we perform several uncoupled stabilization iterations in the 3D problem. In this stabilization process we find the steady state 3D solution of the balance equation system, considering the boundary conditions evaluated by the system code. Concerning the boundary conditions for the energy balance equation and according to the nomenclature shown in the right part of Figure 3.2, we impose the temperature evaluated by the system code on the surface $\overline{AB}$, a inhomogeneous Neumann condition on $\overline{EO}$ and a homogeneous Neumann boundary conditions in the remaining surfaces. The thermal heat flux imposed on $\overline{EO}$ is $45785 W/m^2$. Concerning boundary conditions for the axial component of the momentum balance equation, we impose the flow rate evaluated by the system code on the surface $\overline{AB}$, a vanishing Neumann in the axis $\overline{AN} \cup \overline{IH}$ and the no-slip condition in the remaining surfaces. The transverse component of the velocity field is set to zero in all the boundary zones of the domain. The initial condition for the stabilization process is a constant temperature field of $T = 260 \circ C$ and a vanishing velocity in all the domain. The block of code that performs this stabilization in our driver is shown in Figure 3.3. We can notice that the quantities `t_1` and `v_1` are the inlet temperature and velocity, respectively, evaluated by the system code in the region $\overline{AB}$ and are set in the three

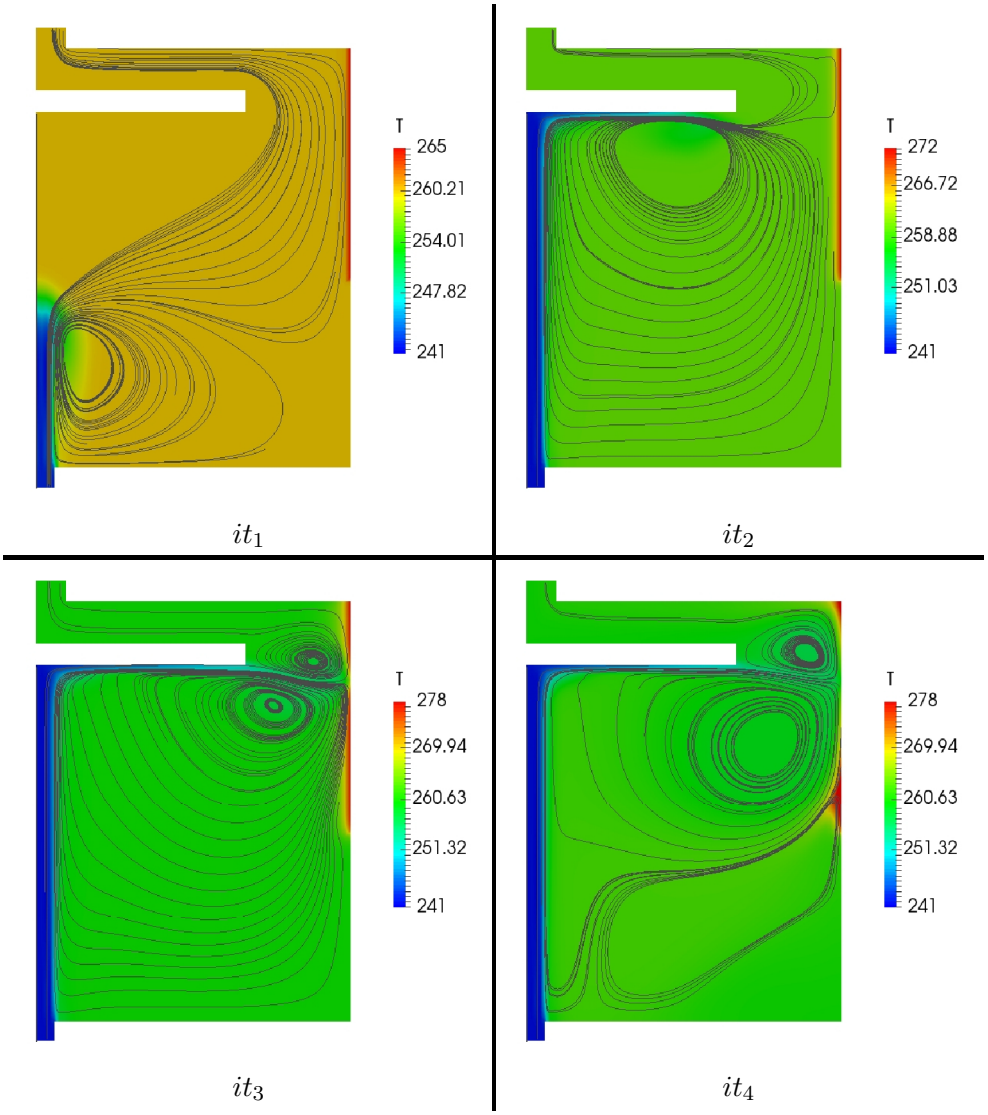


Figure 3.4: Temperature and stream line overview at different stabilization iteration number.

dimensional problem with the command *setAnalyticBoundaryValues*, a method of the 3D problem class `P`. In order to perform this projection we must associate the surface $\overline{AB}$ to a particular identification number, in this specific case the identification numbers for the temperature and the velocity field are `400` and `401`, respectively. finally we can also notice that the velocity `v_1` is multiplied by a factor of $48/37$. This factor is a correction term that ensure the imposition of the correct flow rate in the inlet zone of the three-dimensional problem and it is needed because a vanishing velocity condition is imposed in the point B. The temperature evolution and the velocity field is shown in Figure 3.4 at the stabilization iteration number $it_1 = 2$, $it_2 = 10$, $it_3 = 20$ and $it_4 = 500$.

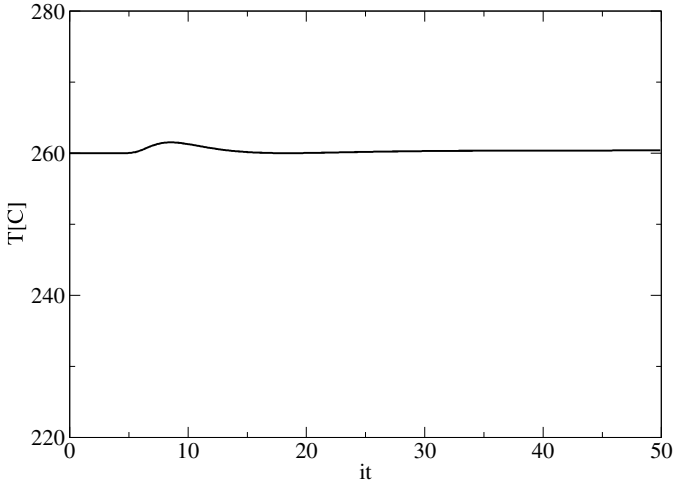


Figure 3.5: Average temperature over the the outlet region $\overline{HG}$ over the iteration numbers.

In particular in this figure the domain is colored with the temperature field while flow streamlines are marked with solid lines. The average temperature over the the outlet region $\overline{HG}$ over the iteration numbers is shown in Figure 3.5. We can notice that a steady state is reached after a little number of iterations. Finally we remark that the solution evaluated after the stabilization process is used as the initial condition for the coupling of the one- and three-dimensional problems.

3.3 One-way coupling code simulation

In this section we use the defective coupling algorithm in order to couple only the energy equation, namely the temperature variable. The pressure and the velocity fields are given as boundary conditions to the three-dimensional simulation from the one-dimensional computation. The coupling is full in energy but only one-way for pressure and velocity. In Figure 3.6 a sketch of the uncoupled and one-way coupled mesh is shown. On the left of Figure 3.6 it is shown the one-dimensional mesh where only the system code is solved. On the right the sketch shows two overlapping meshes: a one-dimensional mesh over all the domain and a three-dimensional mesh defined only in the three-dimensional test region. On the left of Figure 3.6 six reference points are shown with red circles. They are labeled with 1-2 (COR3-RESERVE4) on the left vertical leg, 3-4 (BELOW3D25-ABOVE3D3) in the central vertical leg) and 5-6 (PUMP0-LINUP14) on the right vertical leg. The point 3 (BELOW3D25) is the 1D/3D matching interface for the inlet where we impose boundary conditions The point 4 (ABOVE3D3) is the 3D/1D matching interface for the outlet of the three-dimensional domain where we impose matching condition for the temperature variable.

We report the result for the CATHARE stand-alone and one-way coupling case (with energy coupling). We label by E the CATHARE standalone computations and by F the

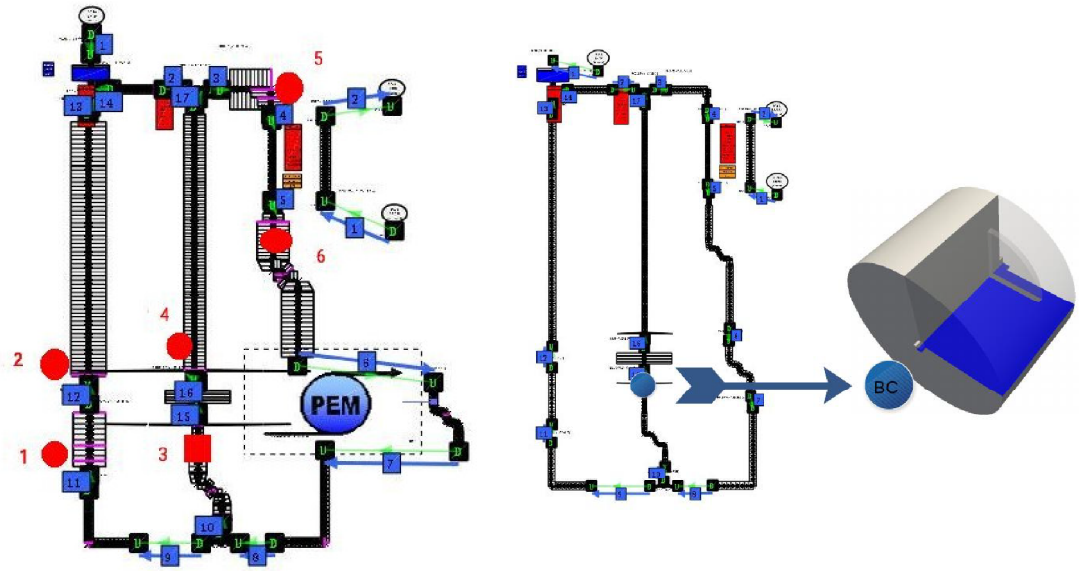


Figure 3.6: Final state condition for the one-dimensional Problem C solved without coupling. The final condition satisfies the steady state equation.

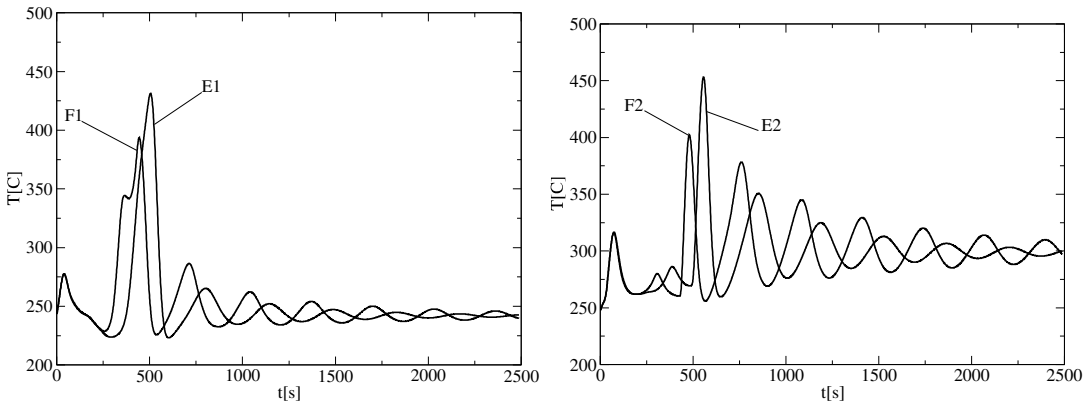


Figure 3.7: Temperature at point 1 (on the right) and point 2 (on the left) for the case E (CATHARE standalone) and F (one-way coupling).

one-way coupling case (with energy coupling). Temperature at different points is shown in Figures 3.7-3.8 for both cases. Temperature at point 1-2 and 3-4-5-6 is reported in Figure 3.7 and 3.8 respectively. In a similar way liquid flow at points 1, 3-4 and 5-5 for the case E (CATHARE standalone) and F (one-way coupling) in Figure 3.9-3.10. In order to couple the three- and one-dimensional codes we use a defective algorithm which is based on feedback control to impose the boundary conditions. The liquid flow of the one-dimensional code is imposed through the inlet boundary condition. The temperature at the point 4 in the 1D/3D interface is shown in Figure 3.10 on the left. The average

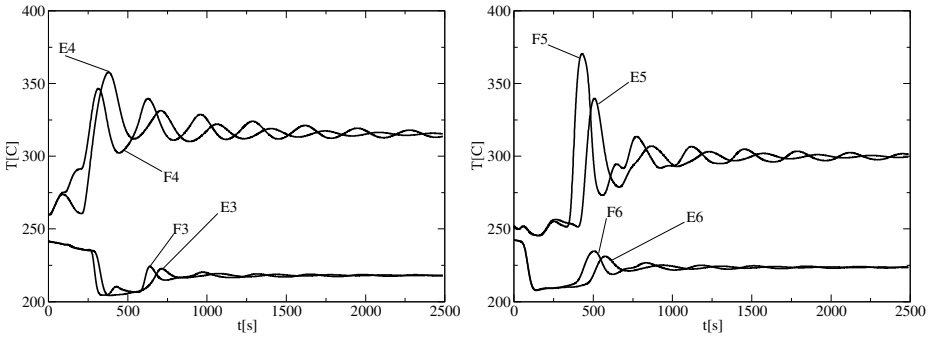


Figure 3.8: Temperature at points 3-4 (on the right) and points 5-6 (on the left) for the case E (CATHARE standalone) and F (one-way coupling).

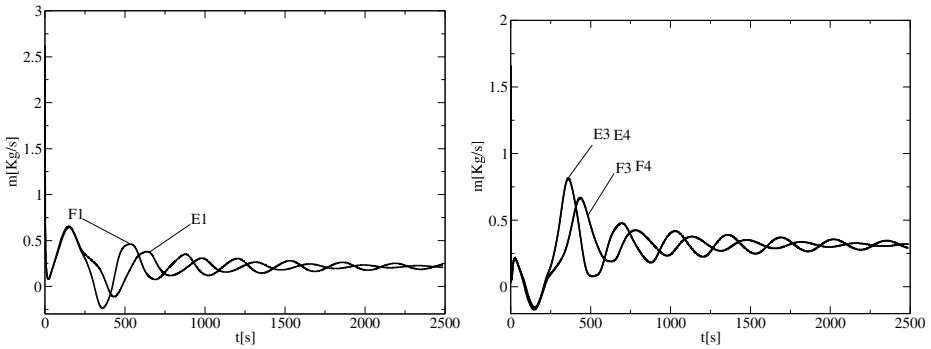


Figure 3.9: Liquid flow at point 1 (on the right) and points 3-4 (on the left) for the case E (CATHARE standalone) and F (one-way coupling).

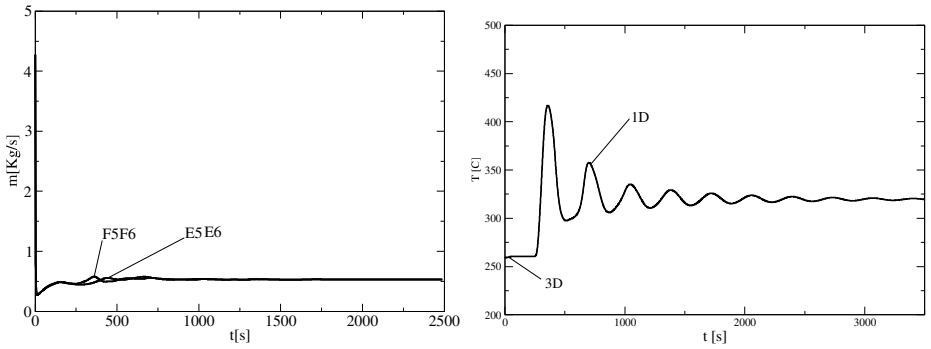


Figure 3.10: Liquid flow at the points 5-6 (on the left) for the case E (CATHARE standalone) and F (one-way coupling). On the right temperature at the 1D/3D and 3D/1D interfaces as a function of time.

temperature in the 3D matches perfectly the one-dimensional one at the same point. In Figures 3.11–3.13 temperature field and isosurfaces for different t are shown. Figure 3.11 shows the temperature field for t = 1-100s, Figure 3.12 for t = 100-1000s and finally

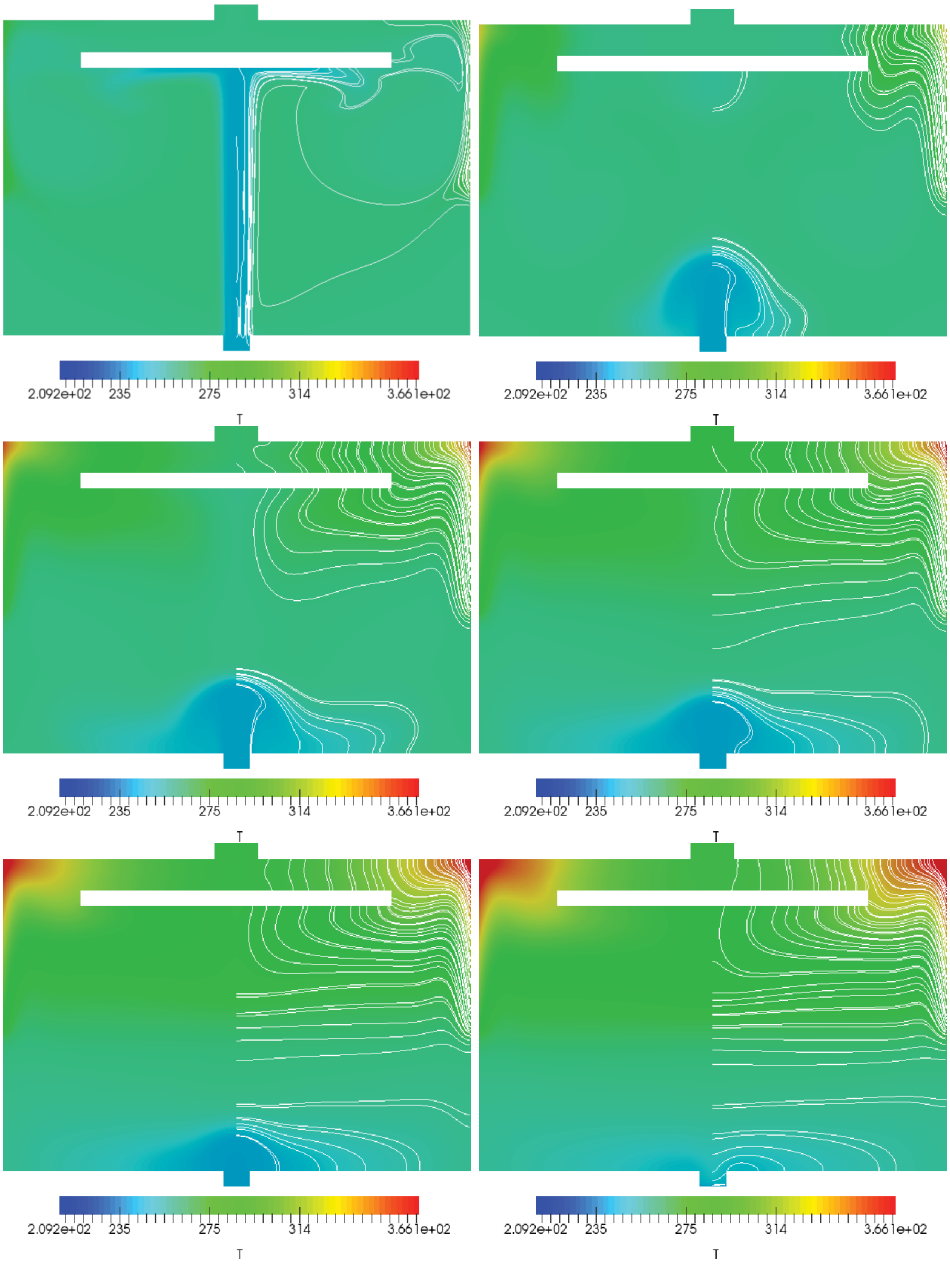


Figure 3.11: Temperature field and isosurfaces for $t = 1$, 20, 40, 60, 80 and 100s.

Figure 3.13 for $t = 2000$-$3000s$.

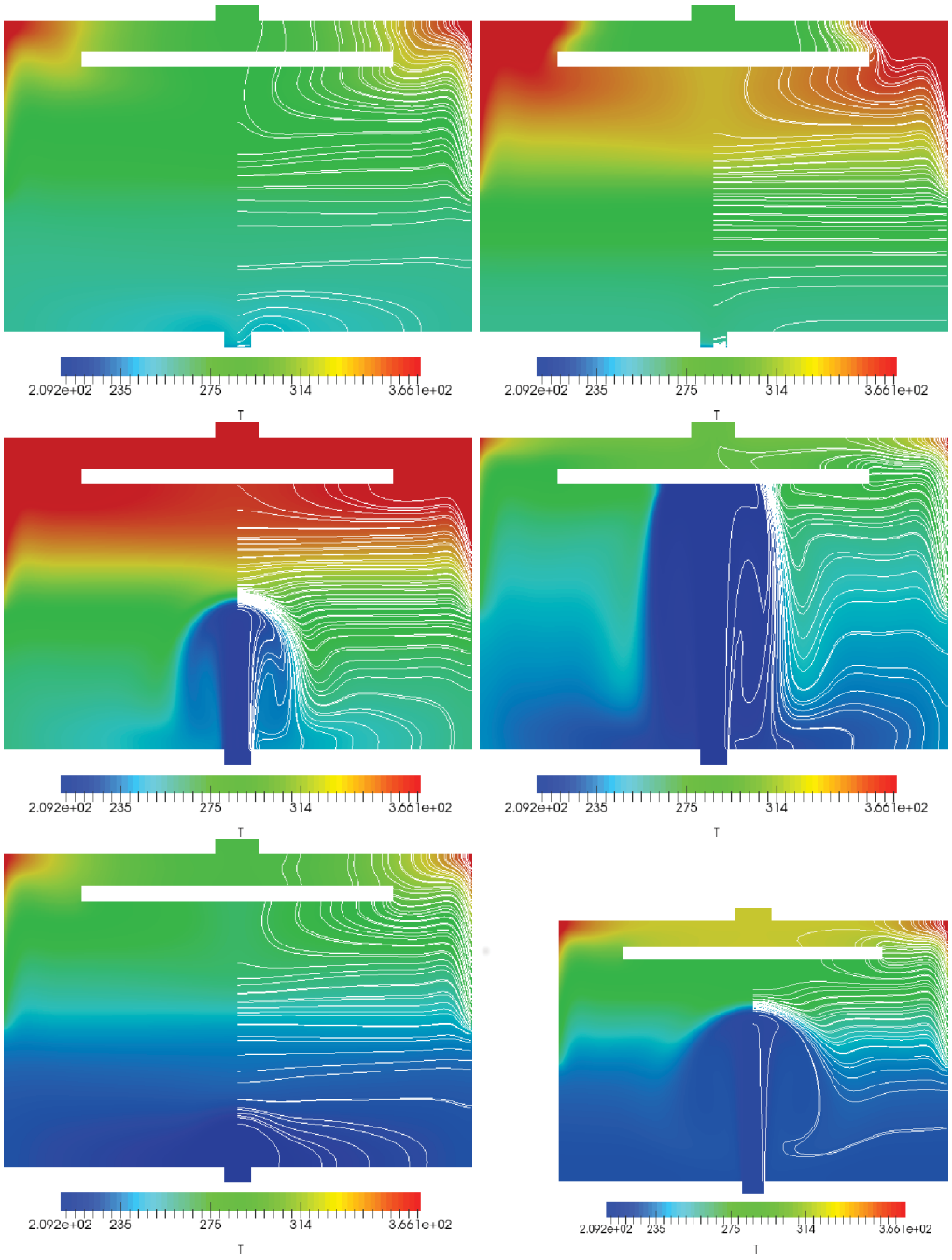


Figure 3.12: Temperature field and isosurfaces for $t = 100$, 200, 300,400,500,and 1000s.

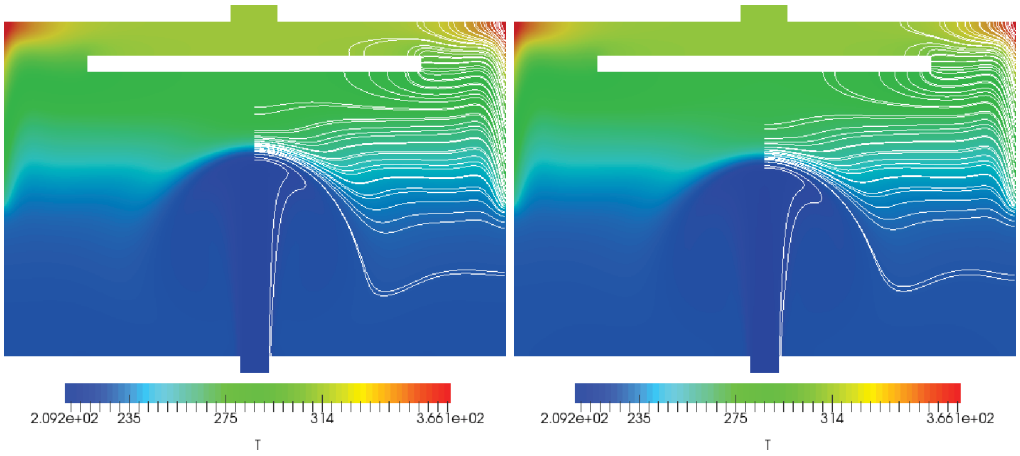


Figure 3.13: Temperature field and isosurfaces for $t = 2000$ and $3000s$.

4 Full coupling 3D-1D code simulation

4.1 Problem coupling on FEMLCORE-SALOME-CATHARE platform

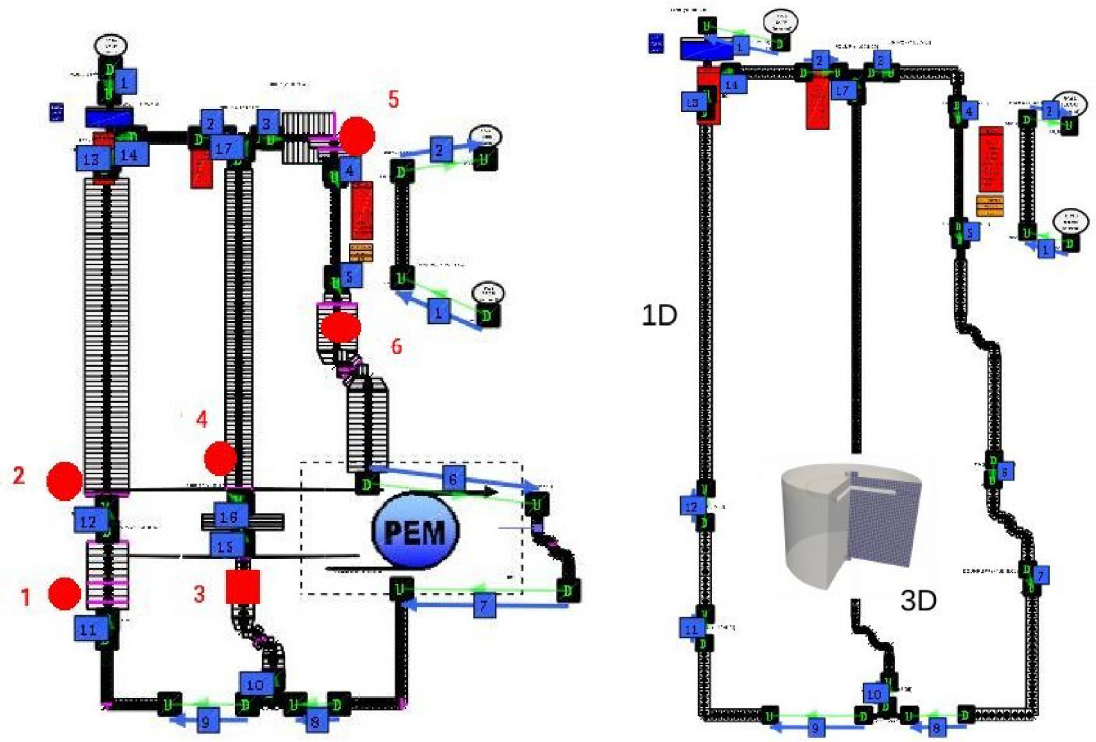


Figure 4.1: TALL-3D facility with reference points 1–2 (left leg), 3–4 (central leg), 5–6 (right leg) (on the left) and 3D-1D coupling diagram (on the right).

In this section we use the fully defective coupling algorithm. In Figure 4.1 a sketch of the uncoupled and coupled mesh is shown. On the left of Figure 4.1 it is shown the one-dimensional mesh where only the system code is solved. On the right the sketch shows two overlapping meshes: a one-dimensional mesh over all the domain and a three-dimensional mesh defined only in the three-dimensional test region. By using the defective coupling algorithm we solve at the same time the three-dimensional code and the one-dimensional system code over the overlapping domain. On the left of Figure 4.1 six

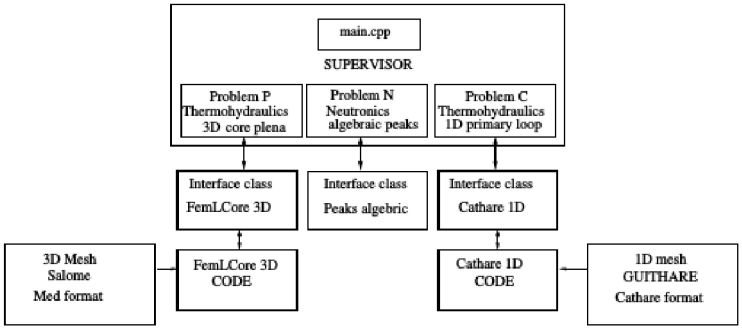


Figure 4.2: TALL-3D coupling. Problem P (3D) is coupled with problem C (1D) by using coupling defective algorithm.

reference points are shown with red circles. They are labeled with 1-2 (COR3-RESERVE4) on the left vertical leg, 3-4 (BELOW3D25-ABOVE3D3) in the central vertical leg and 5-6 (PUMP0-LINUP14) on the right vertical leg. The point 3 (BELOW3D25) is the 1D/3D matching interface for the inlet of the three-dimensional domain. The point 4 (ABOVE3D3) is the 3D/1D matching interface for the outlet of the three-dimensional domain. In Figure 4.2 one can see the computational platform FEMLCORE-CATHARE SALOME where the one-dimensional thermohydraulics problem is labeled as Problem C and the three-dimensional thermohydraulics problem as Problem P. The data exchange over the interfaces (point 4 and point 3) is driven by the interface class FEMLCORE based on the MED library implemented on the SALOME platform. The meshes are generated with GUITHARE for the one-dimensional case and with SALOME GEOM and MESH modules for the three-dimensional one. The coupling algorithm can be schematized as follows:

a. Problem P Initialization (FEMLCORE);

b. Problem C Initialization (CATHARE);

c. Set up coupled transient;

d. For each time step:

   d1. one step for the one-dimensional code;

   d2. one step for the three-dimensional code;

e. Repeat to end time.

The part a initializes the three-dimensional Problem P and the 1D/3D and 3D/1D interfaces, the corresponding code looks like this:

```
//     ============== ==============  ==============  ================
// P-> Problem P Initialization (FEMLCORE)
//     ============== ==============  ==============  ================
```

```
std::vector<FIELDS> myproblemP; myproblemP.resize(2);
myproblemP[0]=NS_F;  myproblemP[1]=T_F;
FEMUS P;                                          // constructor
P.init_param(*mgutils[0]);                        // init parameter
P.init_fem(*mggeomel,*mgfemap);                   // init fem
// setting mesh ------------------------------------------------------
P.setMesh();                                      // set MGmesh
// setting system ----------------------------------------------------
P.setSystem(myproblemP);                          // set system 2cell vector
// init iterfaces ----------------------------------------------------
P.init_interface(120,22, 2,filenameP[0].str().c_str());//top Temp
P.init_interface(400,21, 2,filenameP[0].str().c_str());//bottom Temp
P.init_interface(401,21, 2,filenameP[0].str().c_str());//bottom velocity
P.init_interface(402,21,1,filenameP[0].str().c_str()); //bottom pressure

// get parameter from  FEMUS P
int n_steps    = mgutils[0]->get_par("nsteps");
double dt      = mgutils[0]->get_par("dt");
int print_step = mgutils[0]->get_par("printstep");
int itime_0    = mgutils[0]->get_par("itime");
int restart    = mgutils[0]->get_par("restart");
double fem_rho  = 10448.0;//mgutils[0]->get_par("rho0");
double time     = 1500.;  double tmax=10000.;
int step_count=1;         int iproc=P.get_proc();
double T_out=260.; // outlet 3D

double averageP[1];averageP[0]=0.;double averageP0[1];averageP0[0]=0.;

P.solve_setup(itime_0,time1);  // initial time loop (t=0)
averageP[0] = P.getAvOnBoundary_nodes(402,"NS0",DIMENSION);averageP0[0]=0.;
```

After the initialization of the Problem class we initialize the 1D/3D and 3D/1D interfaces with the command *P.init_interface*. We create four interfaces from the three-dimensional mesh through the definition of sub-meshes (called groups):

- the interface 120 links the temperature variable on the group 22 of the three-dimensional mesh,
- the interface 400 links the temperature variable on the group 21,
- the interface 401 links the velocity variable on the group 21,
- the interface 402 links the pressure variable on the sub-mesh labeled group 21.

The group 21 defines the inlet sub-mesh of the 3D test section while the group 22 defines the sub-mesh of the outlet region. For each sub-mesh we use two different representations and the corresponding mapping: the MED and code format. Both the system and CFD codes can read and map the MED format into its own format and viceversa. It is necessary to create one interface function for each state variable.

We impose the boundary conditions in velocity and temperature at the inlet from the system code and extract the value at the outlet to impose on the system code. We set the initial condition to the Problem P with the function *P.solve_setup* and compute the pressure at the inlet and outlet to determine the pressure losses. The quantities on surface must be average by the command *P.getAvOnBoundary_nodes* in order to be passed to the one-dimensional code. In this code *averageP[0]* and *averageP0[0]* contain the inlet and outlet average pressure. Since the pressure is defined up to a constant we can only compute differential pressure or the so called pressure losses.

The part b of the algorithm corresponds to the following section of code

```
//     ============== ==============  ==============  =================
// C -> Problem C Initialization (CATHARE)  ----------------------------
//     ============== ==============  ==============  =================
// points 1D-3D interfaces
std::string point1("BELOW3D"); int ix_3=27; int irad_3=0;// Point 3
std::string point2("ABOVE3D"); int ix_4=3; int irad_4=0; // Point 4
// c constructor
Problem_Cathare* c = new Problem_Cathare();
c->initialize();// c initilization

// set up pressure 0- transient
double time1;  v_3 = flowrate / (rho * 0.0085 * 0.0085 * M_PI);
double dp_stab = 0.; double press_p3,press_p4,dp_io,dp_ioo;
press_p2=0.; press_p1=0.; dp_io=0.; dp_ioo=0.;
press_p2=c->getValue_CATHARE("PRESSURE",point2.c_str(),ix_4,irad_4);//at P4
press_p1=c->getValue_CATHARE("PRESSURE",point1.c_str(),ix_3,irad_3);//at P3
dp_ioo=c->getValue_CATHARE("DPLEXT",point2.c_str(),2,0);
double rho_3=0.;
rho_3=c->getValue_CATHARE("LIQDENS",point1.c_str(),ix_3, irad_3);
```

We define *point1* the element 27 located in the BELOW3D module (key point 3 in Figure 4.1) and *point2* the element 3 located in the ABOVE3D module (key point 4 in Figure 4.1). The Problem class for the one-dimensional code is set and initialized. The pressure from the CATHARE code is extracted with the *getValue_CATHARE* function at the point 1 and 2. In CATHARE, the source term in the momentum equation can be changed by modifying the variable DPLEXT. The momentum equation source DPLEXT controls the pressure variable. In the defective mode algorithm the pressure at the specified location is controlled by a control feedback technique. Finally, also the density, which is a function of temperature, is extracted at the point 1.

After the initialization of the Problems P and C the evolution transient should be solved. The one-dimensional code time step is computed as follows

```
// ======================================================================
//                        SET UP COUPLED TRANSIENT
// ======================================================================
// setup nonlinear control
bool stop=false; bool ok=false; // setup transient loop control
double present=0.; step_count=step_count0+1;// setup transient time control
```

```
// ================== TRANSIENT ========================================
while(!stop) { // stop if stop=true
 ok=false;
 while(!ok && !stop) { // nonlinear iteration loop (ok=false iterate)

   // ============== CATHARE STEP=======================================
   // adjusting last time step
   present=c->presentTime();  if(present>=tmax) { stop=true; }
   if(stop) { break; }   // if stop ==true -> exit
   c->initTimeStep(dt);

   // Pump off
   if(present >=1501) {c->setValue_CATHARE("TENSALIM","EMP",1,1,0.0);}

   // BELOW3D inlet evaluation at P1 ------------------------------->
   // rho_3(density) f(flowrate) h(entalpy) T(temperature) v(velocity)
   double rho_3=c->getValue_CATHARE("LIQDENS",point1.c_str(),ix_3,irad_3);
   double f_3=c->getValue_CATHARE("LIQFLOW",point1.c_str(),ix_3,irad_3);
   double P_3=c->getValue_CATHARE("PRESSURE",point1.c_str(),ix_3,irad_3);
   double h_3=c->getValue_CATHARE("LIQH",point1.c_str(),ix_3,irad_3);
   double T_3=c->getValue_CATHARE("LIQTEMP",point1.c_str(),ix_3,irad_3);
   double v_3=c->getValue_CATHARE("LIQV",point1.c_str(),ix_3,irad_3);

   // ABOVE3D outlet evaluaiton at P2 ------------------------------->
   // rho_3 (density) f(flowrate) h(entalpy) T(temperature) v(velocity)
   double t_43=c->getValue_CATHARE("LIQTEMP",point2.c_str(),ix_4,irad_4);
   double h_43=c->getValue_CATHARE("LIQH",point2.c_str(),ix_4,irad_4);
   double rho_43=c->getValue_CATHARE("LIQDENS",point2.c_str(),ix_4,irad_4);
   double h_34 = h_43-.1*(146.5*(t_43-T_out));   //evaluate h correction

   c->setValue_CATHARE("ENTLEXT",point2.c_str(),ix_4,irad_4,h_32);//<-ENTLEXT

   //  Pressure coupling  dp(3D)=fem_rho*(averageP[0]-averageP0[0])
   //  with dp_c=press_p4-press_p3 ---------------------------------->
   press_p4=c->getValue_CATHARE("PRESSURE",point2.c_str(),ix_4,irad_4);
   press_p3=c->getValue_CATHARE("PRESSURE",point1.c_str(),ix_3,irad_3);
   dp_ioo=c->getValue_CATHARE("DPLEXT",point2.c_str(),2,0);

   // Pressure Evaluation P4
   double dp3d=fem_rho*(averageP[0]-averageP0[0])-rho_3*0.22*9.8061;
   double dp1d=(press_p3-press_p4) - 9.8061*(rho_3*(0.146275)
              + 0.5*(rho_3+rho_41)*0.2 + rho_41*(0.1367 - 0.05835*1.0));
   double dp = dp3d - dp1d ;       dp_io=dp_ioo-dp;
   if(fabs(dp) > fabs(press_p3-press_p4)) { dp_io=dp_ioo-dp; }

   c->setValue_CATHARE("DPLEXT",point2.c_str(),2,0,dp_io);// <-- DPLEXT
```

```
ok=c->solveTimeStep();
if(!ok) { c->abortTimeStep(); } // no valid time step
else {  // valid time step
  time=present+dt; // time update
  // ==============END CATHARE STEP==================================
```

The transient loop is determined by a pass-fail time step for the one-dimensional code. The logical variables *stop* and *ok* are evaluated to determine the convergence of the nonlinear system. The time step ends the iterations when ok is true. The algorithm ends when stop is *true*. The time step is the same for both the codes. Since we cannot restart the one-dimensional code during the coupling the stabilization is performed before the coupling during the first $1500s$. As shown in Section 2.2, this interval of time is sufficient to reach a steady state condition that will be the initial condition for the transient phase.

The transient starts after $1500s$ when the coupling becomes active. During the transient the pump is turned off after $1s$ (total time $1501s$) with the function *setValue_CATHARE("TENSALIM","EMP",1,1,0.0).* During the transient for each time step the state value (fl, T, p), where fl is the fluid flow rate, T the temperature and p the pressure, are solved. For one-dimensional system liquid flow rate substitutes the knowledge of the velocity field ($v = \dot{m}/A\rho$). The state (fl, T, p) is extracted from the one-dimensional simulation at the point 1 and 2 to determine the source needed for correction in the mass, momentum and energy equations. The energy correction is obtained by computing

$$ENTLEXT = h_{43} - \alpha C_p(t_{43} - T_{out}); \tag{4.1}$$

where h_{43}, t_{43} are the entalpy and the temperature at the point 2. The T_{out} is the average temperature at the outlet of the 3D test section which has been computed after the FEMLCORE time step. The constant α is the feedback constant. The higher the value of α, the faster the coupling brings the two solution to match. The momentum correction is obtained by computing $DPLEXT$ as

$$\begin{aligned} dp_{3d} &= (P_{in,3D} - P_{out,3D}) - \rho_{3d} H_{3D} g \\ dp_{1d} &= (P_{in,1D} - P_{out,1D}) - g\rho_{1d} H_{1D} g \\ DPLEXT &= dp_{ioo} - \beta(dp_{3d} - dp_{1d}) \end{aligned} \tag{4.2}$$

where H_{3D} is the total height of the 3D simulation domain, g is the gravity constant. Therefore, dp_{3d} and dp_{1d} are the pressure looses when the gravity contribution is subtracted, of the three- and one-dimensional 3D test-section. The value dp_{ioo} is the old value of DPLEXT get from the one-dimensional code directly. The constant β is the feedback constant. As before, the higher the value of β, the faster the matching is achieved. If the three-dimensional pressure losses dp_{3d} are higher than the the one-dimensional one dp_{1d} the source DPLEXT increases the one-dimensional pressure losses. When $dp_{3d} = dp_{1d}$ the momentum equation source DPLEXT reaches a stationary value.

The FEMLCORE time step is computed as follows

```
// ============ FEMUS STEP ======================================
```

```
  // Temperature coupling  T_3(CATHARE) with interface 11 (bottom)-->
  std::ostringstream a; a<< T_3;
  P.setAnalyticBoundaryValues(400,1, a.str().c_str());
  P.write_Boundary_value(400,"T",1);//write bc values inside x_old

  // velocity coupling  v_3(CATHARE) with interface 21 (bottom) --->
  std::ostringstream vel; vel<< "IVec*0.+JVec*"<< v_3 * 48./37;
  P.setAnalyticBoundaryValues(401,2, vel.str().c_str());
  P.write_Boundary_value(401,"NS0",2); // bc values inside x_old

  // SOLVE  P problem
  P.solve_onestep(itime_0,step_count,print_step,time,dt);  // solv P

  T_out=P.getAvOnBoundary_nodes(120, "T",0); // <--------------------
  std::cout<< " ***T_out = "<< T_out << std::endl;
  averageP[0] =P.getAvOnBoundary_nodes(402, "NS0",DIMENSION);
  averageP0[0] =0.;// P.getAvOnBoundary_nodes(33, "NS0",DIMENSION);

  c->validateTimeStep();
  ++step_count;
// ============= END FEMUS STEP ====================================
}// =============  else validateTimeStep ============================
```

In the FEMLCORE time step we set the inlet state with boundary condition from the one-dimensional code. The temperature T_3 obtained by CATHARE is imposed to interface 11 (inlet) in analytical form with the function *setAnalyticBoundaryValues*. Also the liquid flow is imposed from the one-dimensional code to the three-dimensional inlet. After solving the FEMLCORE time step the pressure, the temperature and the velocity fields are available. We compute and average the pressures at the inlet and at the outlet to compute the pressure losses. We compute and average the temperature to impose this value to the one-dimensional code on the first element of the ABOVE3D module.

4.2 Simulation results

The solution of the coupled system are reported in this section for the one- and three-dimensional code. For the one-dimensional code we refer to Figure 4.1 where the reference points are reported on the left side. In Figure 4.3 the temperature on the left vertical leg at points 1-2 and on the central vertical leg at points 3-4 are shown. The point 1 (COR3) and 2 (RESERVE4) are shown on the left while the point 3 (BELO3D25), 4 (ABOVE3D3) on the right. We compare the coupled computation with the uncoupled one where only the one-dimensional code is used. For the coupled we use the label E and for CATHARE alone case the label C. Therefore $E1$ means CATHARE coupled computation at point 1 and $C2$ the computation with CATHARE alone at point 2. It

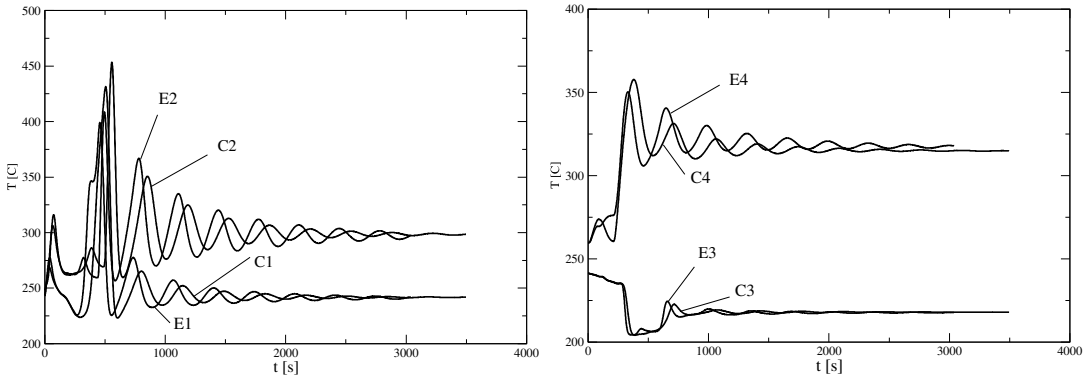


Figure 4.3: Temperature at points 1 (COR3), 2 (RESERVE4) (on the left) and 3 (BELO3D25), 4 (ABOVE3D3) (on the right) for the coupled (E) and CATHARE alone (C) case.

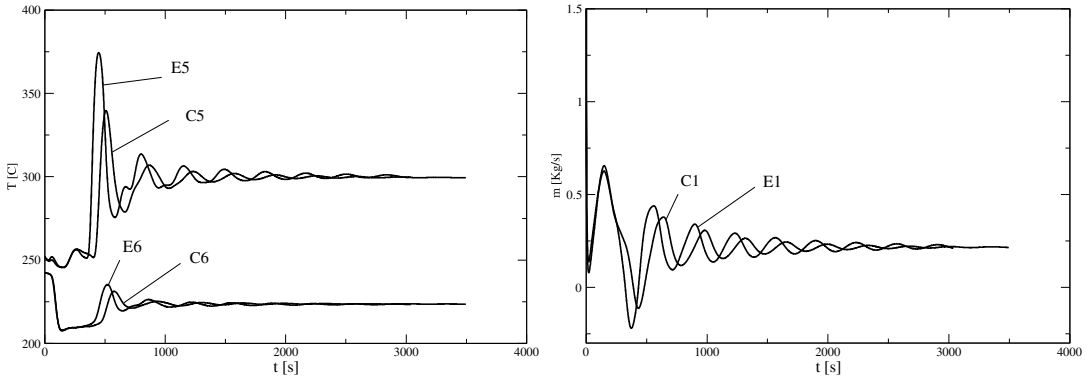


Figure 4.4: Temperature at points 5 (LINUP5), 6 (PUMP0) (on the left) and fluid flow at points 1 (COR3) (on the right) for the coupled (E) and CATHARE alone (C) case.

is easy to note that the coupling brings a delay to the dynamic of the one-dimensional system. This probably is due to the increasing of the pressure losses which decreases the velocity in the system.

In Figure 4.4 temperature at points 5 (LINUP5), 6 (PUMP0) and fluid flow at points 1 (COR3) are reported on the left and on the right, respectively. The coupled and CATHARE alone case are reported with label E and C. The liquid flow on the central leg at point 3 (BELOW3D25) is shown on the left of Figure 4.5 for the coupled (E) and CATHARE alone (C) case.

Finally we report the pressure evolution on Figure 4.5 on the right. The pressure at points 3, 4 and 6 are labeled as $C3$, $C4$ and $C6$ for the CATHARE stand-alone case and as $E3$, $E4$ and $E6$ for the coupled simulation. For the pressure variable, the coupled and uncoupled cases show similar behavior. This happens mainly because the hydrostatic pressure dominates the pressure jump between inlet and outlet. The difference between

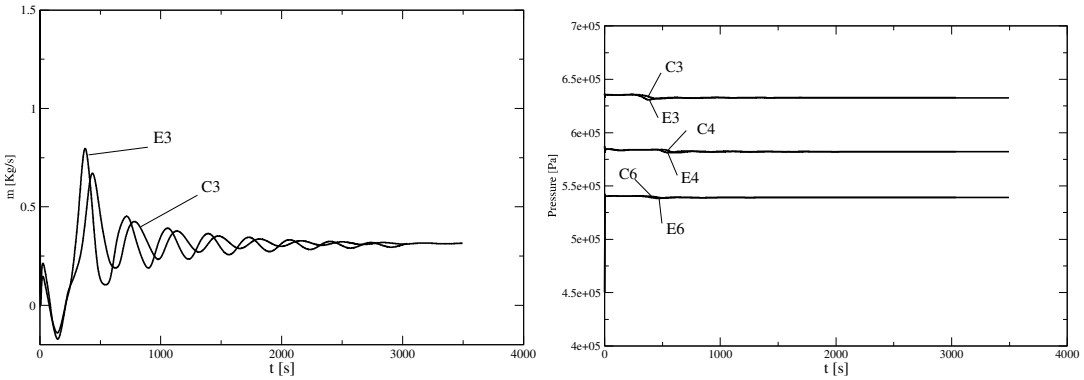


Figure 4.5: Liquid flow on the central leg at point 3 (BELO3D25) (on the left) and pressure at points 3, 4 and 6 for the coupled (E) and CATHARE alone (C) case.

the two cases lies in the pressure losses only.

One of the advantages of the coupled simulation id the availability of the three-dimensional fields on the 3D test section. The time evolution of the pressure, temperature and velocity fields are shown in Figures 4.6–4.8. On the left one can see the pressure field and the velocity streamlines, while the temperature field is shown on the right part of the domain. Figure 4.6 shows the evolution for t = 1-100s, Figure 4.7 covers t = 100-1000s and finally Figure 4.8 shows t = 1000-4000s.

4.3 Coupling 1D/3D and 3D/1D interfaces

In order to couple the three- and one-dimensional codes we use a defective algorithm which is based on feedback control to impose the boundary conditions. The liquid flow of the one-dimensional code is imposed through the inlet boundary condition. In Figure 4.9 one can see the liquid flow at the inlet (1D) and at the outlet (3D). Since the mesh are overlapping and no flow may exit from the test section, there can be no error in the mass balance equation.

The temperature at the point 4 in the 3D/1D interface is shown in Figure 4.10 on the left. The energy correction is obtained by computing

$$ENTLEXT = h_{43} - \alpha C_p(t_{43} - T_{3D}); \tag{4.3}$$

where h_{43}, t_{43} are the entalpy and the temperature at the point 2. The T_{3D} is the average temperature at the outlet of the 3D test section which has been computed after the FEMLCORE time step. The constant α is set to 0.1. As one can see the matching is almost perfect at each time step. The momentum correction is obtained by computing $DPLEXT$ as

$$DPLEXT = DPLEXT_o - \beta(\Delta p_{3D} - \Delta p_{1D}) \tag{4.4}$$

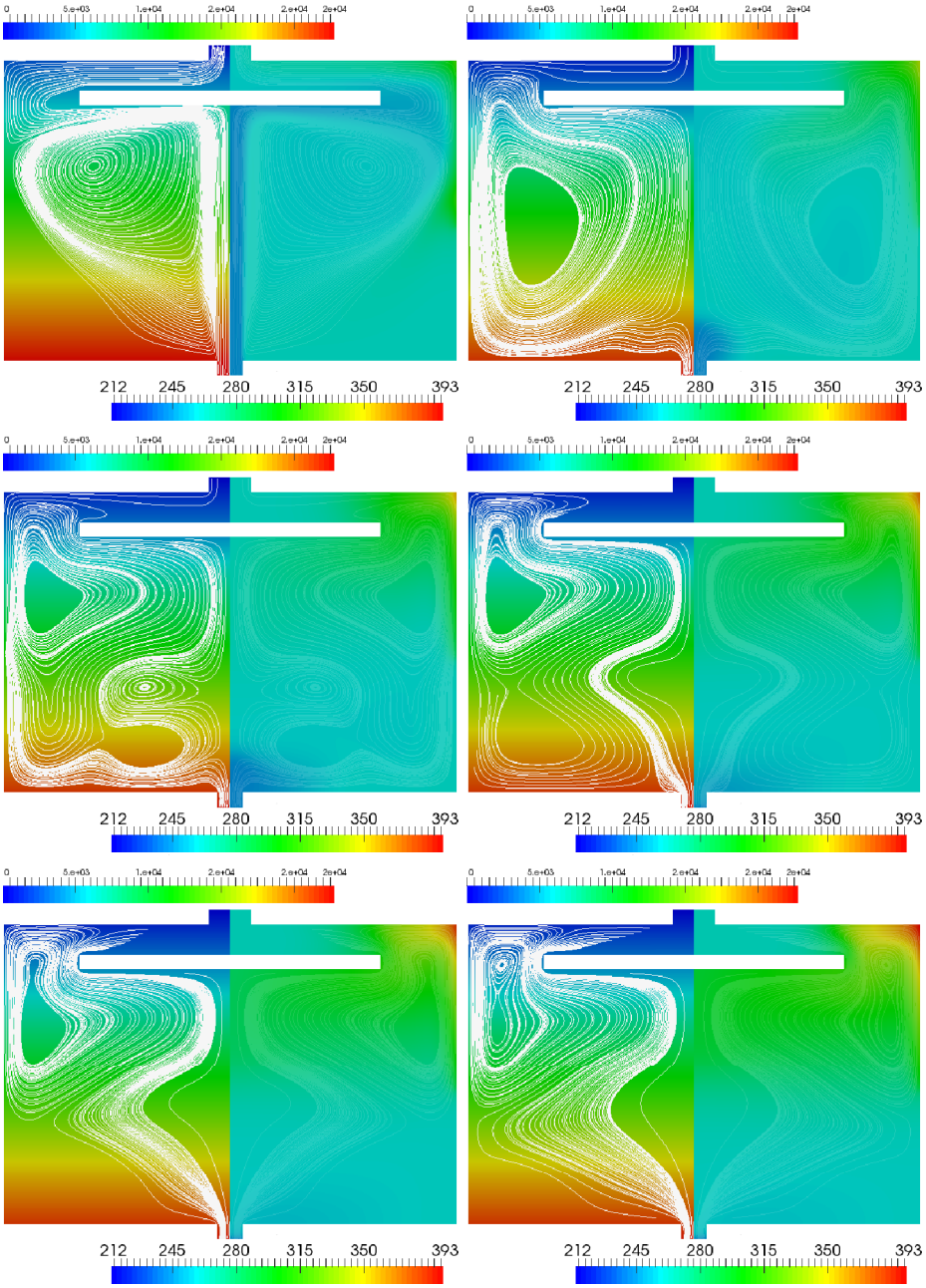


Figure 4.6: Pressure field and velocity streamlines on the left together with temperature field on the right in the 3D test section for $t = 1$, 20, 40, 60, 80 and 100s.

where Δp_{3D} and Δp_{1D} are the pressure looses, when the gravity contribution is subtracted, of the three- and one-dimensional 3D test-section. The value $DPLEXT_o$ is the old value of DPLEXT getting from the one-dimensional code directly. The constant β is set to 1.

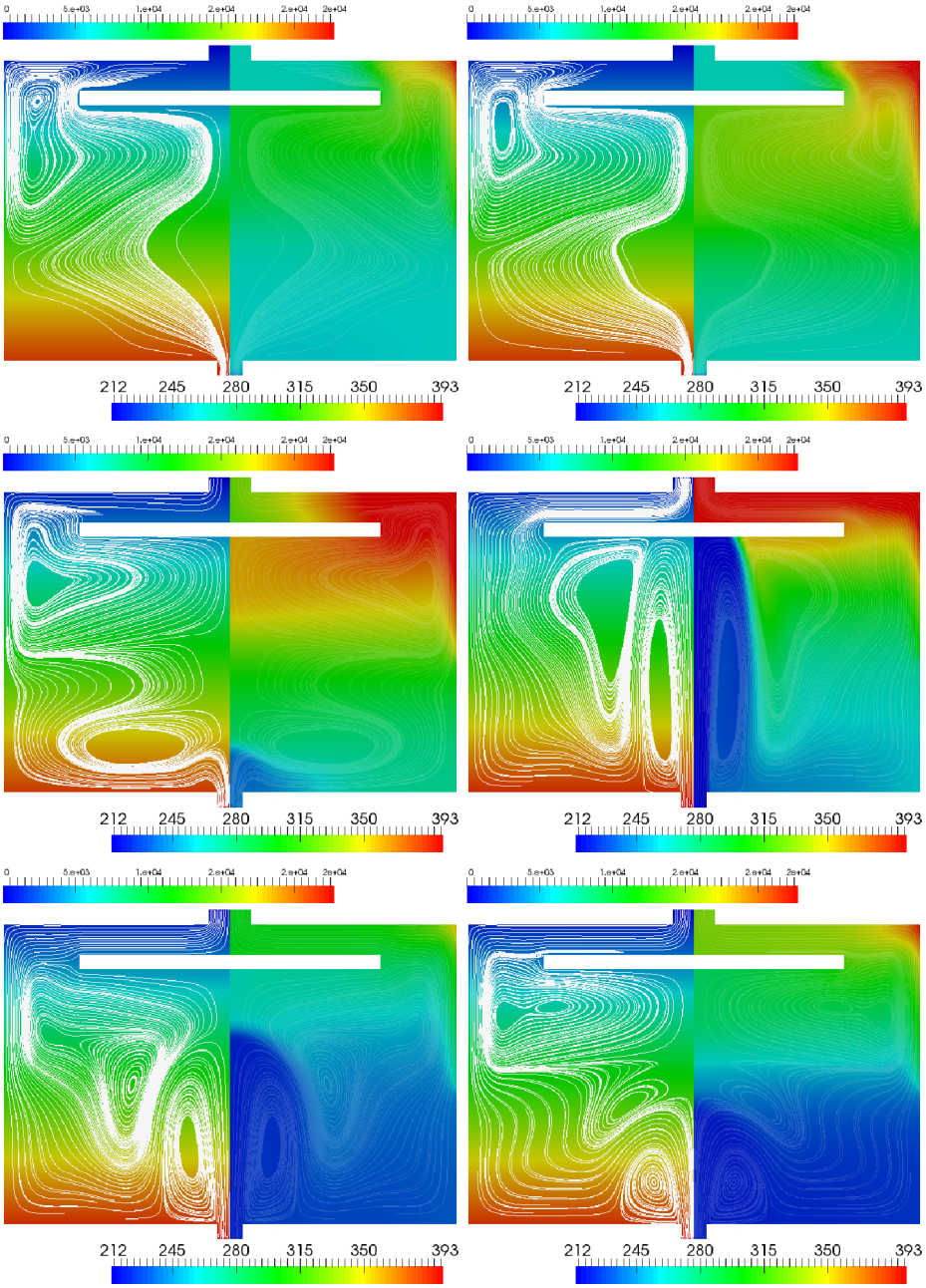


Figure 4.7: Pressure field and velocity streamlines on the left and temperature field on the right in the 3D test section for $t = 100$, 200, 300, 400, 500 and $1000s$.

The pressure at the point 4 in the 3D/1D interface is shown in Figure 4.10 on the right. Again one can see that the matching is almost perfect at each time step.

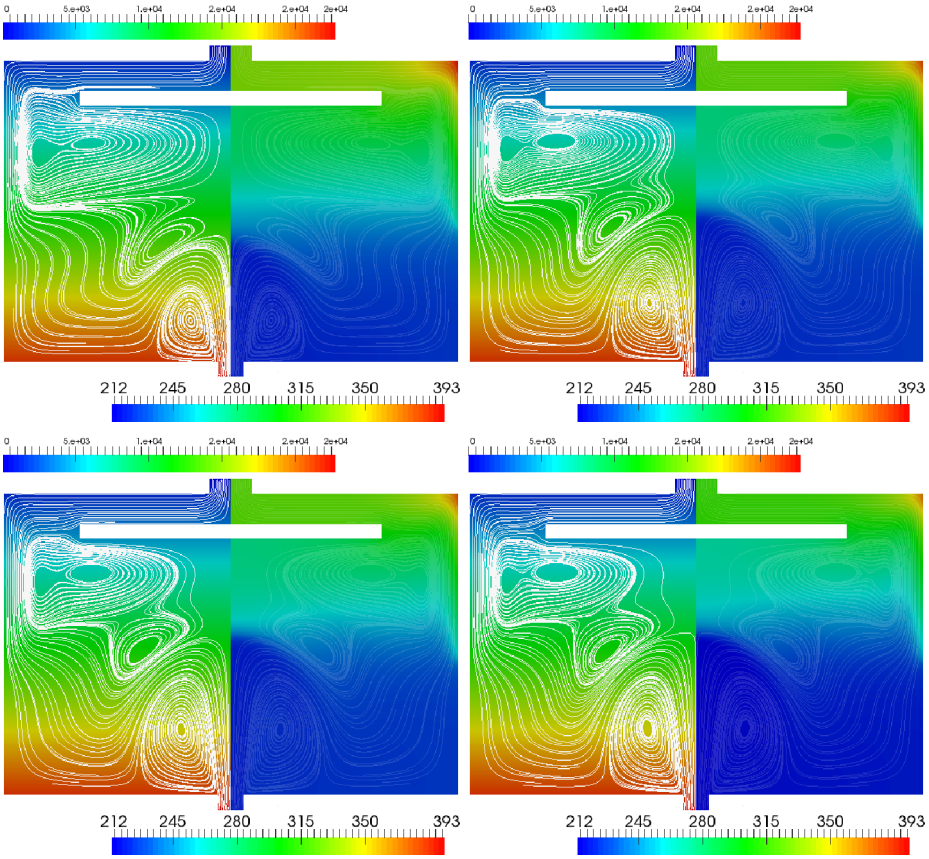


Figure 4.8: Pressure field and velocity streamlines on the left together with temperature field on the right in the 3D test section for $t = 1000$, 2000, 3000 and 4000s.

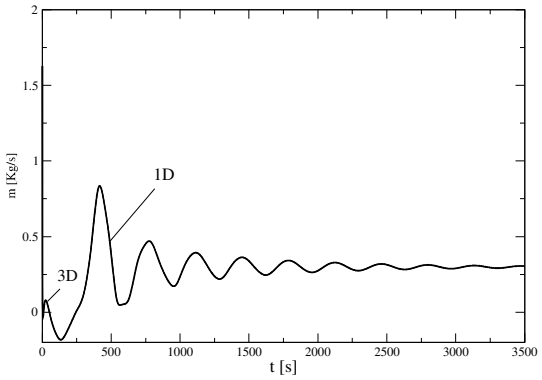


Figure 4.9: 1D and 3D liquid flow (right) at point 4 as a function of time.

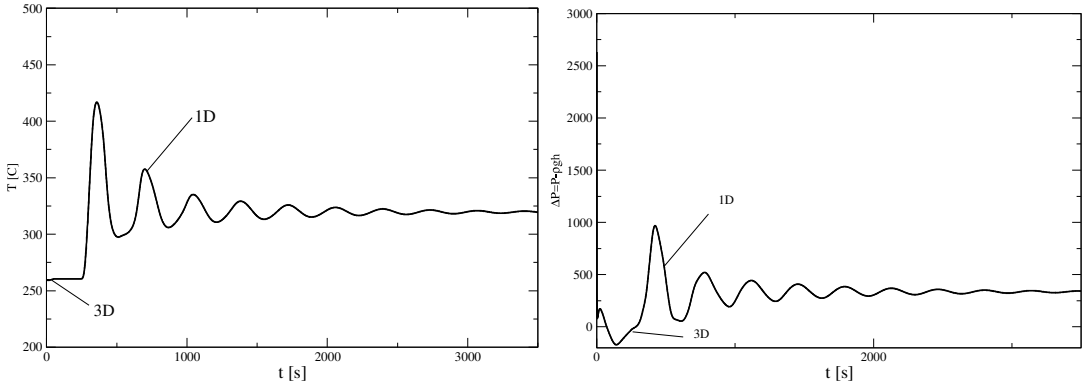


Figure 4.10: 1D and 3D temperature (left) and distributed pressure loss $\Delta P = P - \rho g h$ (right) at point 4 as a function of time.

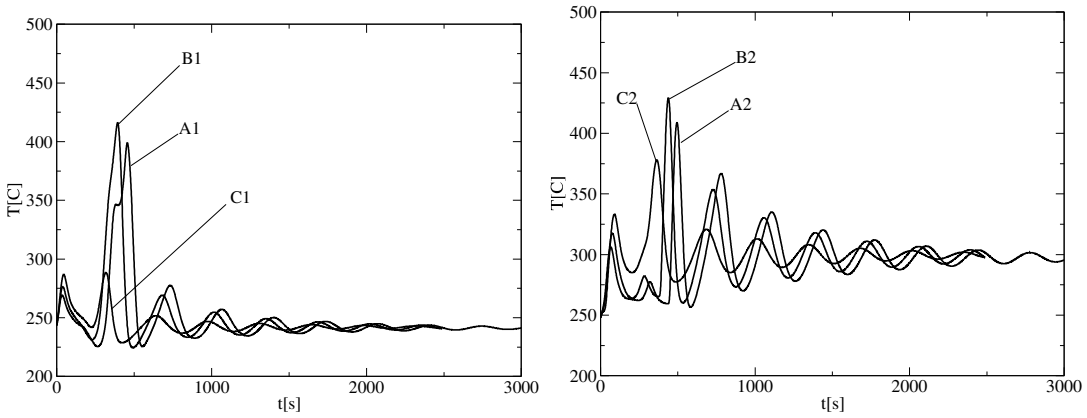


Figure 4.11: Case 1. Temperature at point 1 (left) and point 2 (right) for $\Delta P = 0$ (A), $-\Delta P_0$ (B), $-2\Delta P_0$ (C).

4.4 Pressure loss correction in PIPE3D

The one-dimensional system code produces temperature oscillations which are left shifted in phase and smaller in term of amplitude. As one can see from Figures 4.3–4.5 by introducing full coupling the oscillations are even more delayed. This shows that the pressure losses are not balanced correctly in the one-dimensional case. With the coupling the pressure losses are increased and the liquid flow decreases. In order to shift on the right the oscillations we need to increase the velocity and the pressure losses at the same time through the three-dimensional test section. This is only possible with a redistribution and a decreasing of the pressure losses in the one-dimensional circuit which does not overlap the three-dimensional test section. The redistribution of the pressure losses is not a very easy task due to the complexity of the system but we can prove that by changing the pressure losses inside the three-dimensional test section we are able to shift on the right and on the left the oscillations. As might be expected this increases or

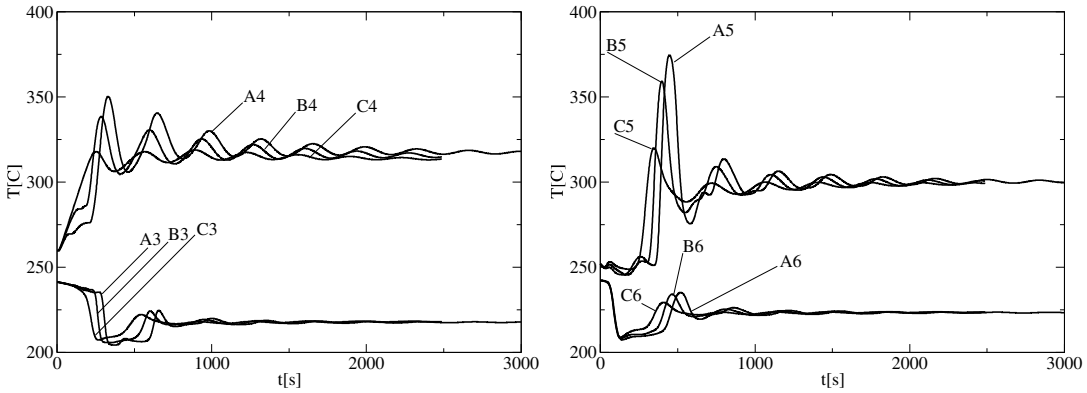


Figure 4.12: Case 1. Temperature at points 3-4 (left) and points 5-6 (right) for $\Delta P = 0$ (A), $-\Delta P_0$ (B), $-2\Delta P_0$ (C).

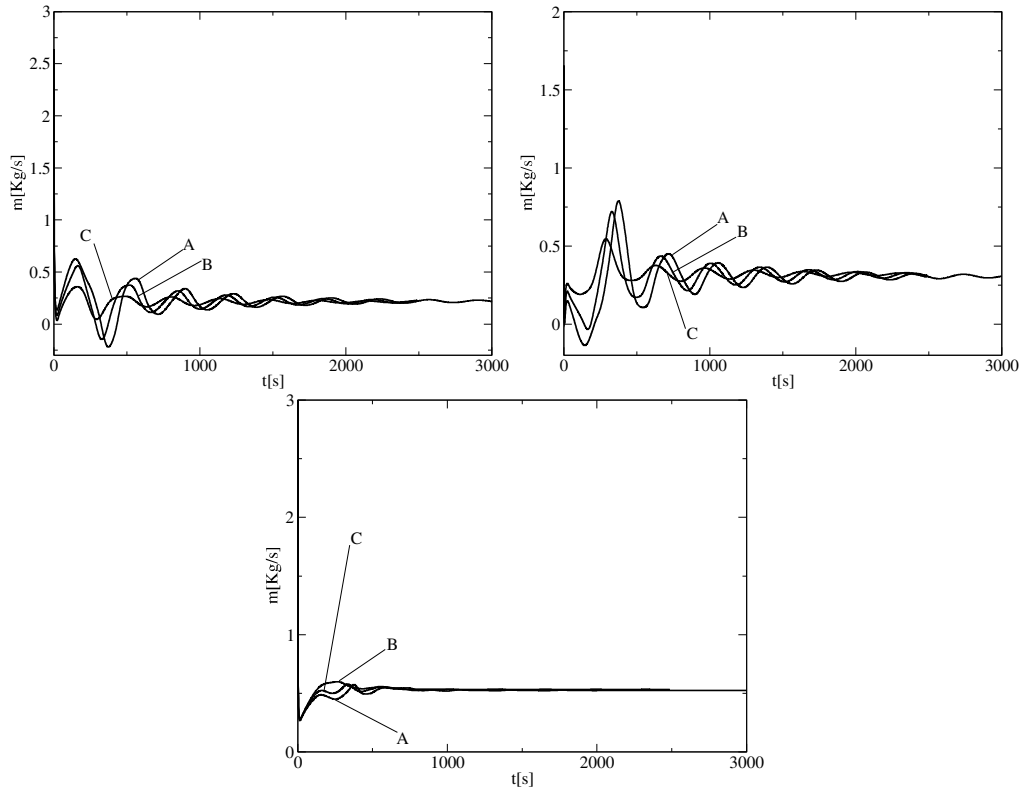


Figure 4.13: Case 1. Liquid flow at points 1-2-3 (from the left top to the bottom) for $\Delta P = 0$ (A), $-\Delta P_0$ (B), $-2\Delta P_0$ (C).

decreases the liquid flow in the opposite direction as desired.

In this section we therefore study the behavior of the system with an increase or decrease of the pressure losses in the 3DPIPE module with the purpose of showing the

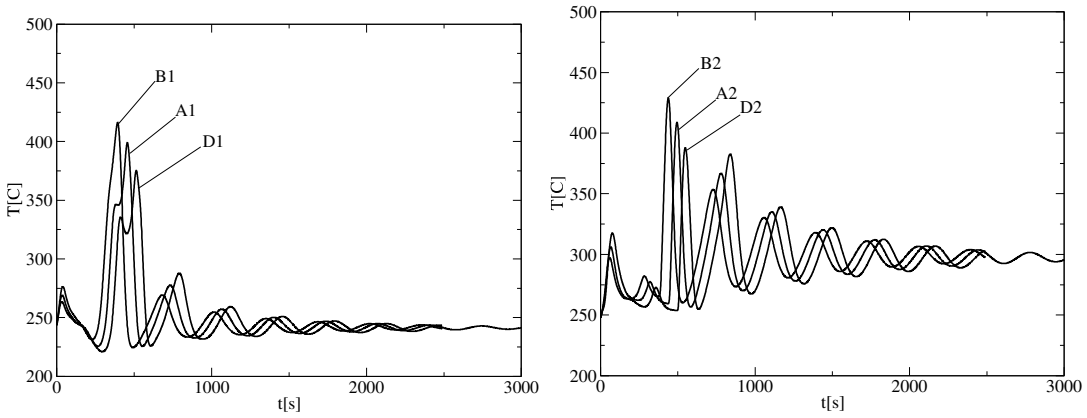


Figure 4.14: Case 2. Temperature at point 1 (left) and point 2 (right) for $\Delta P = 0$ (A), $-\Delta P_0$ (B), $-2\Delta P_0$ (C).

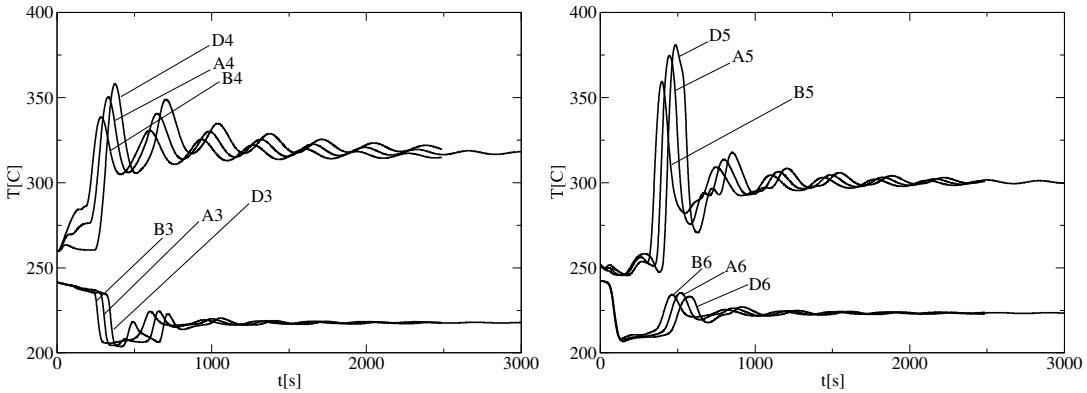


Figure 4.15: Case 2. Temperature at points 3-4 (left) and points 5-6 (right) for $\Delta P = 0$ (A), $-\Delta P_0$ (B), $-2\Delta P_0$ (C).

possibility of advance/delay the oscillations and leave to further studies the tuning of the pressure losses in the rest of the one-dimensional circuit. We study two cases: one with a decrease in pressure losses in the PIPE3D (case 1) and another with an increase (case 2). The increasing or decreasing is related only to pressure losses while hydrostatic pressure is not considered.

In case 1 we decrease the pressure losses of a quantity $\Delta P_0 = 200Pa$ and $2\Delta P_0$. In Figures 4.11–4.12 the temperature at the point 1-6 is shown. In Figure 4.11 we have the temperature at point 1 and 2 along the left vertical leg with an increase of pressure losses of $\Delta P = 0$, (A) $-\Delta P_0$ (B) and $-2\Delta P_0$ (C). As one can see the decrease of the pressure losses generate a delay in the oscillation phase. In Figure 4.12 we have the temperature at points 3-4 along the central leg and at points 5-6 along the right vertical leg on the left and on the right, respectively. Similar behavior for the liquid flow in Figure 4.13.

In Case 2 we increase the pressure losses of the same quantity ΔP_0. In Figures 4.14–4.16 the temperature profiles as a function of time at the points 1-6 are shown. In Figure 4.14

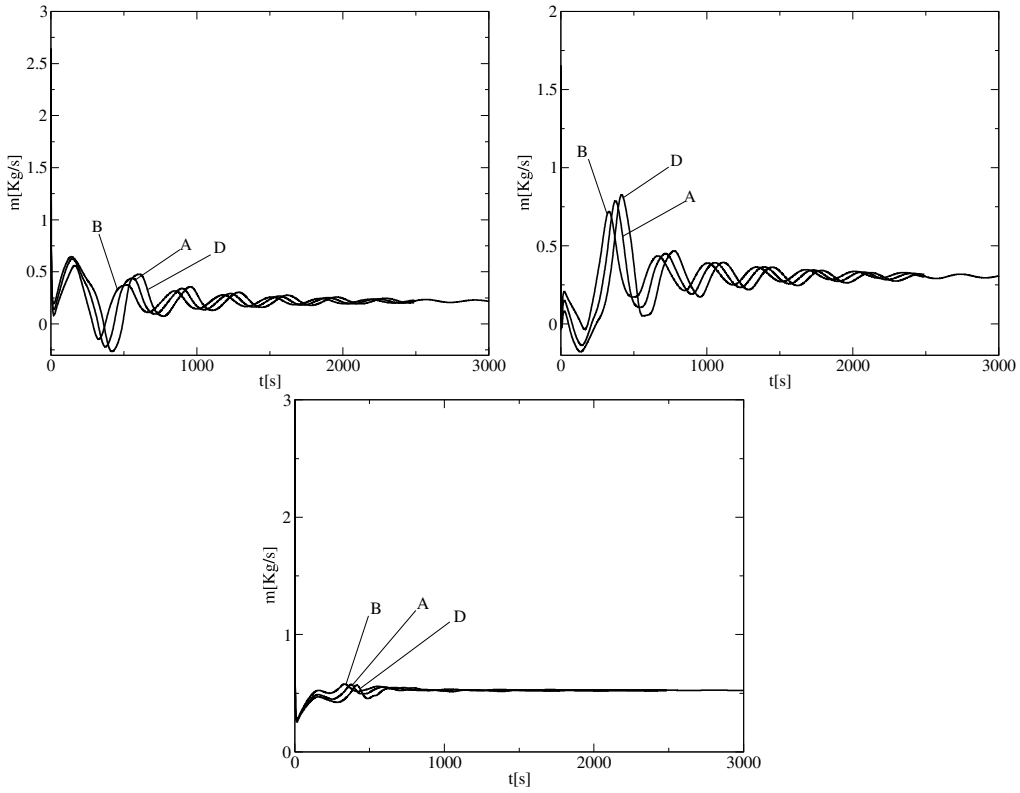


Figure 4.16: Case 2. Liquid flow at points 1-2-3 (from the left top to the bottom) for $\Delta P = 0$ (A), $-\Delta P_0$ (B), $-2\Delta P_0$ (C).

we have the temperature at point 1 and 2 along the left vertical leg for $\Delta P = 0$, (A) $-\Delta P_0$ (B) and ΔP_0 (C). As one can see the increase of the pressure losses generate a right shift in the oscillation phase as desired. In Figure 4.15 we have the temperature at points 3-4 along the central leg and at points 5-6 along the right vertical leg on the left and on the right, respectively. Finally the liquid flow in Figure 4.16 at the points 1-6 are shown.

Conclusions

In this report we have simulated the TALL-3D facility by using the FEMLCORE and CATHARE code coupled through the SALOME platform. We computed the one-dimensional circuit and the three-dimensional test section by using a defective coupling algorithm on overlapping meshes. We have reported the results of the evolution of an unprotected loss of flow going from forced to natural circulation flow.

The system is initially considered to be in fully working conditions. The one-dimensional system code standalone has produced temperature oscillations which were shifted in phase and smaller in term of amplitude. We have coupled the one-dimensional circuit and a three-dimensional test section with overlapping meshes in order to correct the one-dimensional simulation. However with the coupling the oscillations have been even more delayed showing that, in the one-dimensional circuit, the pressure losses have not been balanced correctly. By using the coupling the pressure losses have been increased and the liquid flow decreased. The correct shift has not been obtained since this requires a redistribution and a decreasing of the pressure losses in the several parts of the one-dimensional circuit. However the redistribution of the pressure losses is not a very easy task due to the complexity of the system since it needs an accurate and long revision process of the one-dimensional model. In order to evaluate the behavior of the coupling system we have simply changed the pressure losses in the three-dimensional test section to show the possibility of shifting temperature oscillations.

Bibliography

[1] Jeltsov M., K. Kööp, Grishchenko D., Karbojian A., Villanueva W., and Kudinov P., *Development of TALL-3D Facility Design for Validation of Coupled STH and CFD Codes*, Proceedings of The 9th International Topical Meeting on Nuclear Thermal-Hydraulics, Operation and Safety (NUTHOS-9), Kaohsiung, Taiwan, September 9–13, N9P0299, 2012.

[2] M. Jeltsov, K. Kööp, D. Grishchenko, A. Karbojian, W. Villanueva, P. Kudinov, *Development of TALL-3D Facility Design for Validation of Coupled STH and CFD Codes*, The 9th International Meeting on Nuclear Thermal-Hydraulics, Operation and Safety (NUTHOS-9), Kaohsiung, Taiwan, September 9–13, 2012.

[3] D. Grishchenko, M. Jeltsov, K. Kööp, A. Karbojian, W. Villanueva and P. Kudinov, *Design and Commissioning Tests of the TALL-3D Experimental Facility for Validation of Coupled STH and CFD Codes*, THINS 2014 International Workshop Modena, Italy, January 20–22, 2014.

[4] D. Grishchenko, M. Jeltsov, K. Kööp, A. Karbojian, W. Villanueva and P. Kudinov, *The TALL-3D facility design and commissioning tests for validation of coupled STH and CFD codes* , Nuclear Engineering and Design Volume 290, Pages 144–153 (2015)

[5] F. Cadinu, P. Kudinov, *Development of a "Coupling-by-Closure" approach between CFD and System Thermal-Hydraulics codes*, Proceedings of the 13th International Topical Meeting on Nuclear Reactor Thermal Hydraulics (NURETH-13), September 27–October 2, Kanazawa City, Ishikawa Prefecture, Japan (2009) Paper N13P1238

[6] M. Jeltsov, F. Cadinu, W. Villanueva, A. Karbojian, K. Kööp, P. Kudinov, *An approach to validation of coupled CFD and System Thermal-Hydraulics codes*, The 14th International Topical Meeting on Nuclear Reactor Thermal Hydraulics (NURETH-14), September 25–29, Toronto, Ontario, Canada (2011)

[7] M. Jeltsov, K. Kööp, P. Kudinov, W. Villanueva, *Development of a domain overlapping coupling methodology for STH/CFD analysis of heavy liquid metal thermal-hydraulics*, 15th International Topical Meeting on Nuclear Reactor Thermal Hydraulics (NURETH 15), May 12–17, Pisa, Italy (2013) Paper 466

[8] A. Cervone and S. Manservisi, *A three-dimensional CFD program for the simulation of the thermo-hydraulic behavior of an open core liquid metal reactor*, Technical report lin-thrg 108–2008.

[9] S. Bnà, S. Manservisi and O. Le Bot, *Simulation of the Thermal-hydraulic behavior of Liquid Metal Reactors using a Three-Dimensional Finite Element Model*, Technical Report DIENCA-UNIBO 2010.

[10] F. G. Bornia, M. Finelli, S. Manservisi, V. Mikhin, M. Polidori and K. Voukelatou, *Development and validation of FEM-LCORE code for the thermal hydraulics of open cores*, Technical Report DIENCA-UNIBO 2011.

[11] F.Bassenghi, G.Bornia, L.Deon and S. Manservisi, *Implementation and validation of the NURISP platform*, Technical report CIRTEN-UNIBO 2011.

[12] G. Bornia, D. Cerroni, S. Manservisi, M. Polidori and F. Donato, *FEM-LCORE code: parallelization, turbulence models and code integration*, Technical Report ENEA-UNIBO 2012.

[13] D. Cerroni, S. Manservisi and E. Vincenzi, *Developing multiscale transient simulations with fem-lcore code*, Technical Report ENEA-UNIBO 2013.

[14] D. Cerroni, R. Da Vià, S. Manservisi, F Menghini and G. Pozzetti *Integration of the FemLCore code in the SALOME platform*, Technical Report ENEA-UNIBO 2014.

[15] A. Attavino, D. Cerroni, A.Cervone, L. Fancellu and S. Manservisi, *FEMLCORE-CATHARE COUPLING ON SALOME PLATFORM*, Report RL 1361/CERSE-UNIBO, ENEA-UNIBO 2015.

[16] F.Bassenghi, G.Bornia, A. Cervone and S. Manservisi, *The ENEA-CRESCO platform for simulating liquid metal reactor*, Technical report LIN-THRG 210–2010.

[17] D. Bestion, "The physical closure laws in the CATHARE code", *Nuclear Engineering and Design*, vol. 124, no. 3, pp. 229–245, 1990.

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

[19] M. Robert, M. Farvacque, M. Parent, and B. Faydide, "Cathare 2 v2. 5: a fully validated CATHARE version for various applications", 2003.

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

[21] 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.

[22] 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.