



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



*Ministero dello Sviluppo Economico*

## RICERCA DI SISTEMA ELETTRICO

FEM-LCORE Code: parallelization, turbulence models and code integration

*G. Bornia, D. Cerroni, S. Manservizi, M. Polidori, F. Donato*



Report RdS/2012/039

FEM-LCORE CODE: PARALLELIZATION TURBULENCE MODELS AND CODE INTEGRATION

G. Bornia, D. Cerroni, S. Manservizi – CIRTEN Università di Bologna, M. Polidori, F. Donato - ENEA

Settembre 2012

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero dello Sviluppo Economico - ENEA

Area: Governo, gestione e sviluppo del sistema elettrico nazionale

Progetto: Nuovo nucleare da fissione: collaborazioni internazionali e sviluppo competenze in materia nucleare

Responsabile del Progetto: Mariano Tarantino, ENEA

**Titolo**

**FEM-LCORE CODE: PARALLELIZATION,  
TURBULENCE MODELS AND CODE INTEGRATION**

**Descrittori**

**Tipologia del documento:** Rapporto Tecnico

**Collocazione contrattuale:** Accordo di Programma ENEA-MSE: tema di ricerca "Nuovo nucleare da fissione"

**Argomenti trattati:** Termoidraulica dei reattori nucleari  
Termoidraulica del nocciolo

**Sommario**

In questo rapporto viene presentata la nuova versione del codice FEM-LCORE per lo studio tridimensionale della termoidraulica in reattori raffreddati a metallo fuso. Nel dettaglio vengono discussi:

1. l'integrazione del codice FEM-LCORE all'interno della piattaforma SALOME;
2. i modelli fisico-matematici implementati;
3. le simulazioni operate sulla configurazione del reattore ELSY con canali liberi ed in presenza di ostruzioni;
4. le *performance* del codice in termini di calcolo parallelo.



**Note**

Questo documento è stato preparato col contributo congiunto del seguente personale di ricerca ENEA e CIRTEN

- G. Borna, D. Cerroni, S. Manservigi (CIRTEN - Università di Bologna DIENCA)  
Sigla doc. rif.: CIRTEN - Università di Bologna: CERSE RdS/2012/1351-UNIBO
- M. Polidori, F. Donato (ENEA)

**Copia n. In carico a:**

2			NOME			
			FIRMA			
1			NOME			
			FIRMA			
0	EMISSIONE	23/09/12	NOME	F. Donato	P. Meloni	M. Tarantino
			FIRMA	<i>Filippo Donato</i>	<i>P. Meloni</i>	<i>M. Tarantino</i>
REV.	DESCRIZIONE	DATA	REDAZIONE	CONVALIDA	APPROVAZIONE	

# Contents

<b>Abstract</b>	<b>4</b>
<b>Introduction</b>	<b>5</b>
<b>FEM-LCORE Graphical User Interface</b>	<b>7</b>
1.1 Starting the FEM-LCORE GUI . . . . .	8
1.2 GUI input configuration file . . . . .	15
1.2.1 Introductory and general GUI FEMuS pages . . . . .	15
1.2.2 <i>Preprocessing</i> GUI pages . . . . .	17
Thermophysical models . . . . .	17
Mesh . . . . .	18
Physical properties . . . . .	22
Source terms, fuel distribution, pressure drops . . . . .	23
Boundary conditions . . . . .	28
Initial conditions . . . . .	30
1.2.3 <i>Computing</i> GUI pages . . . . .	31
1.2.4 <i>Postprocessing</i> GUI pages . . . . .	33
1.3 Executing the FEM-LCORE code . . . . .	34
1.3.1 Coarse mesh CAD input file and initial configuration . . . . .	34
1.3.2 The <b>gencase</b> application: mesh and multigrid files . . . . .	35
1.3.3 GUI final configuration . . . . .	37
1.3.4 Running the FEM-LCORE code . . . . .	38
1.3.5 Postprocessing FEM-LCORE results . . . . .	39
<b>Thermo-hydraulic of the FEM-LCORE reactor model</b>	<b>40</b>
2.1 Introduction . . . . .	40
2.2 Plenum model . . . . .	44
2.2.1 Navier-Stokes equations . . . . .	45
2.2.2 Turbulence model equations . . . . .	46
LES turbulence mode . . . . .	48
$\kappa$ - $\epsilon$ turbulence model . . . . .	48
$\kappa$ - $\omega$ turbulence models . . . . .	49
2.2.3 Energy equation . . . . .	50

	Sigla di identificazione	Rev.	Distrib.	Pag.	di
	NNFISS-LP3-059	0	L	3	106

2.2.4	Turbulence energy equations . . . . .	50
	Constant Turbulence Prandtl number model . . . . .	51
	$\kappa - \epsilon - \kappa_t - \epsilon_t$ turbulence model . . . . .	51
2.3	Reactor core model . . . . .	51
<b>3</b>	<b>Numerical simulations</b>	<b>56</b>
3.1	Introduction . . . . .	56
3.2	Fuel distribution tests . . . . .	59
3.2.1	Test 1: uniform fuel distribution . . . . .	59
3.2.2	Test 2: non-uniform fuel distribution . . . . .	66
3.3	Pressure drop and partial blockage tests . . . . .	74
3.3.1	Test 3: influence of reactor spacer grids . . . . .	74
3.3.2	Test 4: partial assembly blockage in closed assembly geometry	78
3.3.3	Test 5: partial assembly blockage in open assembly geometry .	83
3.3.4	Test 6: partial blockage of multiple open assemblies . . . . .	87
3.3.5	Closed and open core partial blockage comparison . . . . .	90
3.4	Parallel performance tests . . . . .	92
3.4.1	Test 7: FEM-LCORE with parallel CPUs . . . . .	92
3.4.2	Test 8: FEM-LCORE with GPU . . . . .	98
	<b>Conclusions</b>	<b>103</b>
	<b>References</b>	<b>104</b>
	<b>Notes on the Working Group of the University of Bologna</b>	<b>106</b>

 <b>Ricerca Sistema Elettrico</b>	Sigla di identificazione NNFISS-LP3-059	Rev. 0	Distrib. L	Pag. 4	di 106
--	--	-----------	---------------	-----------	-----------

## ABSTRACT

In this report we discuss the new version of the FEM-LCORE code for the study of three-dimensional thermal hydraulics of liquid metal reactors. The FEM-LCORE code is now based on the FEMuS library and has been partially integrated in the SALOME platform. The code can be launched and configured inside the SALOME GUI and all the preprocessing, running and postprocessing steps can be achieved inside the platform framework. The integration of the code in this platform opens the possibility to study Generation IV reactors by using different codes in a multiphysics and multiscale fashion. The code solves the standard Navier-Stokes equations in parallel mode with turbulence models in the reactor upper and lower plenum. In the core, where all the sub-channel assembly details are summarized in parametric coefficients, a porous medium model has been used. The generation of the mesh, the fuel distribution, the pressure drop factor and other tools have been integrated inside the code and can be configured directly by using the FEM-LCORE GUI. With this new version of the code we investigate some cases with different fuel assembly distributions and pressure drops. Some tests with partial blockage of assemblies are reported. In this case a different pressure loss is applied in the partial blocked assembly reducing the velocity field and decreasing the efficiency of the heat removal operated by the lead coolant. The code solves three-dimensional differential equations and therefore it is rather slow compared to monodimensional tools. In order to increase its performance, the PETSC parallel libraries have been implemented and computations with CPUs and GPUs are discussed.

# Introduction

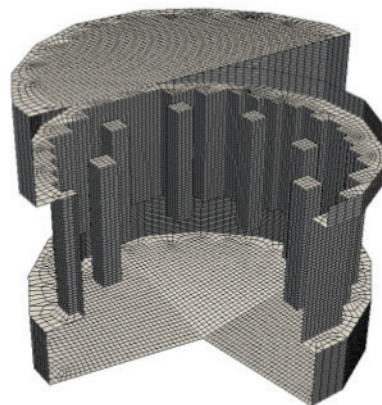
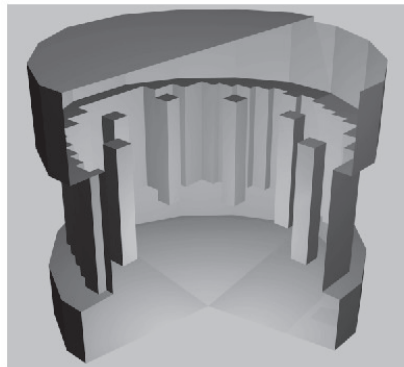


Figure 1: A computational three-dimensional reactor model (top) and mesh (bottom)

The understanding of the thermohydraulics of Generation IV reactors is the starting point to improve safety and to optimize the use of fuel resources. If the three-dimensional behavior of a reactor should be taken into account then an exhaustive simulation of a model reactor is a very hard task. In many safety and working analyses the design of a lead-cooled Gen-IV fast reactor requires an accurate study of three-dimensional core flows. Since a detailed study, which takes into account the

	Sigla di identificazione	Rev.	Distrib.	Pag.	di
	NNFISS-LP3-059	0	L	6	106

design of all the reactor channels inside all the assemblies, is not possible with the current available computational power, then it is necessary to solve the problem with a multiphysics and multiscale approach. The reactor primary circuit, the plenum and the core may be investigated at three different scale levels. The plenum can be studied with 3D-CFD tools while the core and the primary circuit may be modeled with a three-dimensional porous and monodimensional system codes, respectively. FEM-LCORE is a 3D finite element code, developed at the University of Bologna in collaboration with ENEA, for the simulation of the thermo-hydraulic behavior of the core and the plenum of liquid metal nuclear reactors. The FEM-LCORE code tries to estimate three-dimensional average temperature, pressure and velocity fields, in a reactor simplified design, as shown in Figure 1, by properly modeling the complexity of the core with a lead-cooled porous medium. This report illustrates the recent development of the new version of the FEM-LCORE code. The new version improves the modeling of the behavior of liquid metals and increases its computational performance. The code solves the three-dimensional Navier-Stokes, energy and turbulence equations by means of the finite element method. A two-level approach is considered for the thermo-hydraulic modeling of the core region, in order to describe the phenomena occurring at fine and coarse grid scales. A porous medium approach is adopted for the description of the assembly geometric details.

The validation of a code depends on the number of users. The previous developed version was written in object oriented C++ language and could be used only by people able to know how to modify functions and input files. With the purpose of easing the code use, we have developed a Graphical User Interface so that a larger number of researchers can use and run the code. Three-dimensional codes need meshes and mesh generators. Also, once the results are available they must be analyzed with proper postprocessing tools. For this purpose we integrate the FEM-LCORE GUI into the SALOME GUI where a mesh generator and the PARAVIEW application for data visualization are available. In this way FEM-LCORE is now an open-source software that can be used under an open-source platform, from mesh generation to data field analysis.

Chapter 1 is devoted to the Graphical User Interface (GUI). FEM-LCORE has been partially integrated within the SALOME platform. The code can be launched and configured inside the the SALOME GUI and all the preprocessing, running and postprocessing steps can be achieved inside the platform framework.

In Chapter 2 we briefly recall the set of equations adopted for the thermo-hydraulic reactor modeling together with their finite element approximation.

In Chapter 3 we test some examples with different fuel distributions and different pressure drops. In this chapter we test the advances obtained in the parallelization of the code with CPU and GPU architectures. Finally in this chapter some tests with partial blockage of assemblies in an open or closed core configuration are reported. The partial blockage is obtained by increasing the resistance to the flow with distributed pressure losses in the blocked volume.



# FEM-LCORE Graphical User Interface

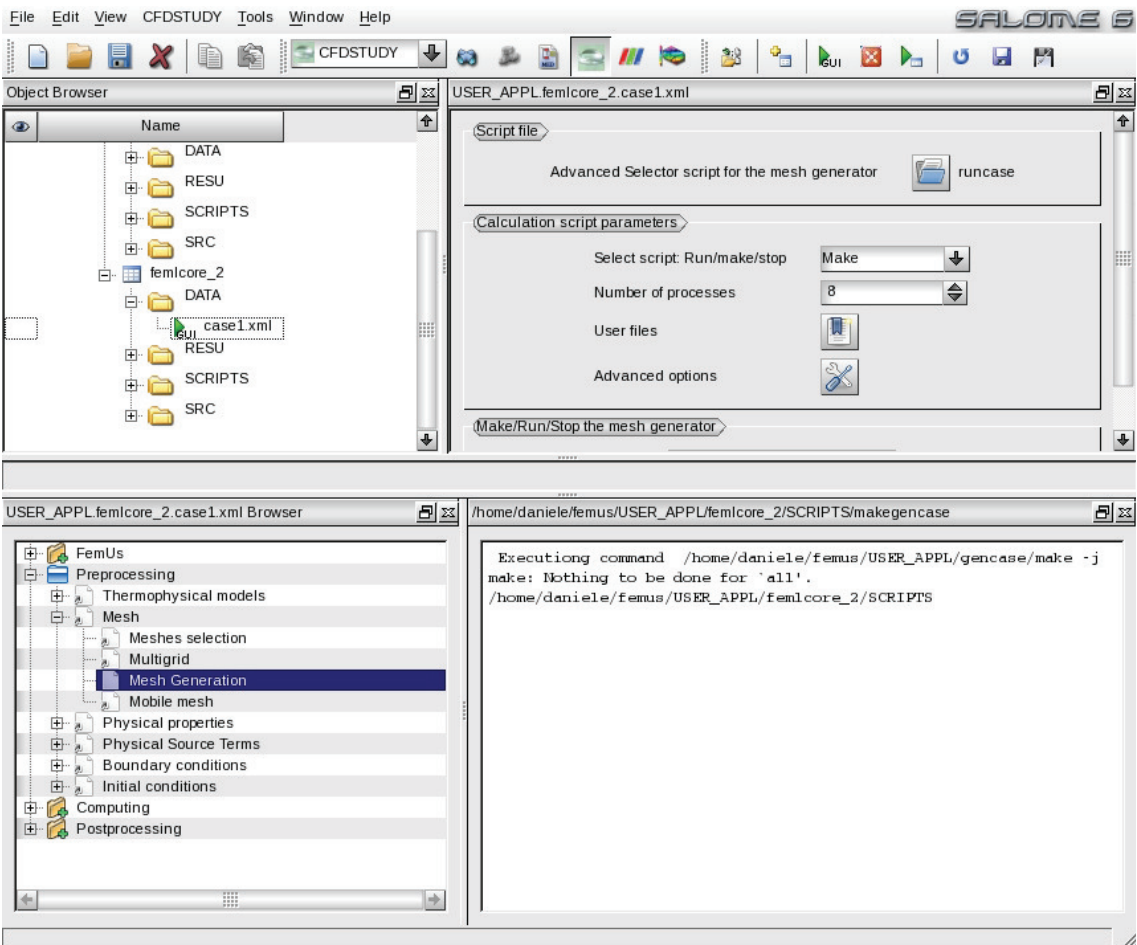


Figure 1.2: FEM-LCORE GUI integrated in the SALOME platform

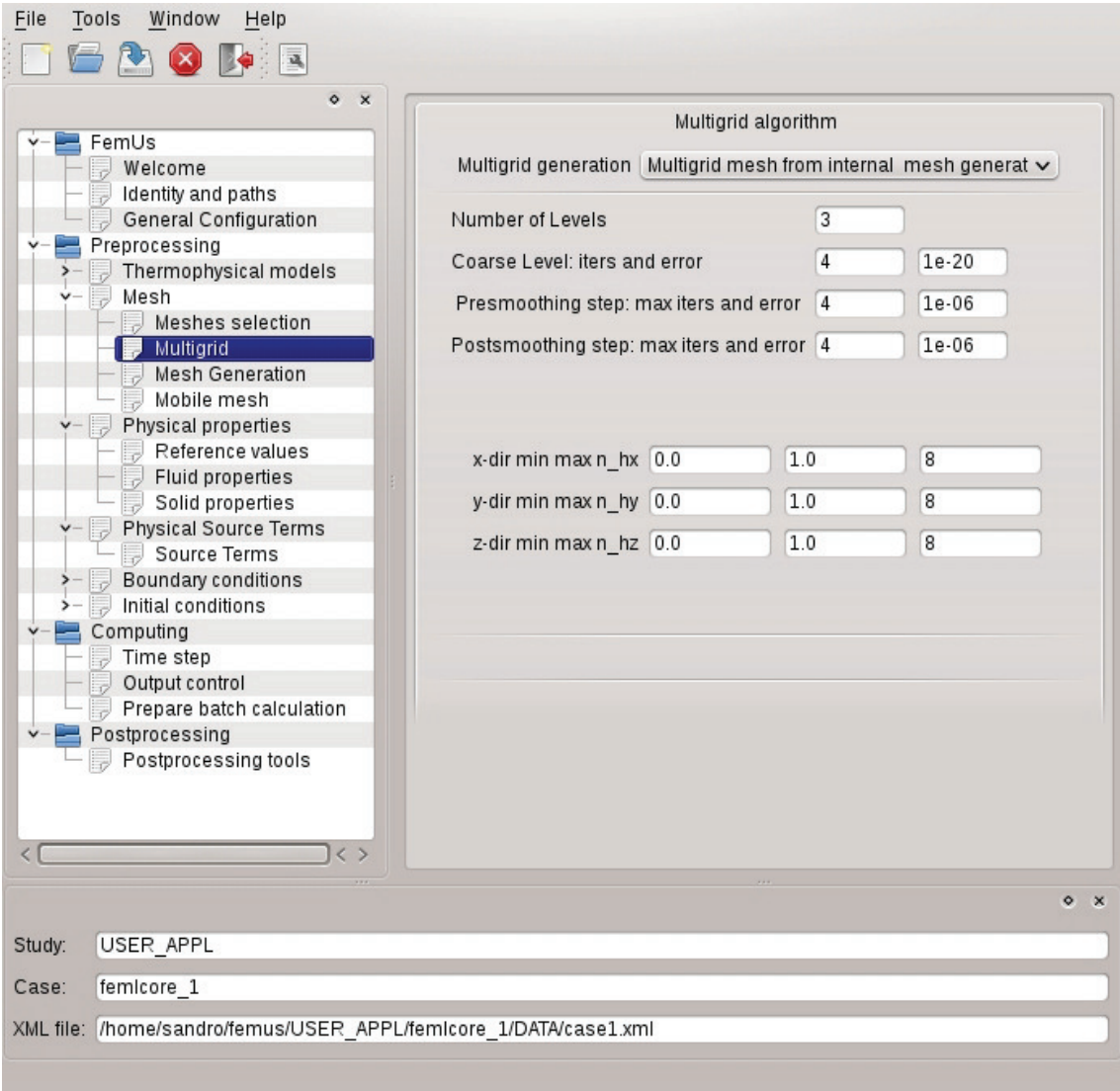


Figure 1.3: FEM-LCORE standalone GUI

## 1.1 Starting the FEM-LCORE GUI

The new version of the FEM-LCORE code is based on the FEMuS code, PETSC and LIBMESH libraries. FEMuS (Finite Element Multiphysics Solver) is a finite element code developed at the Laboratory of Montecuccolino of the University of Bologna for the solution of partial differential equations arising in various fields such as fluid dynamics, two-phase flows, thermohydraulics, neutronics and structural mechanics, with particular attention to a multiphysics approach. The PETSC software manages the parallel computing and the LIBMESH library is used to generate the multigrid mesh. The FEM-LCORE GUI is a based on the QT and Python-QT libraries. It is developed starting from the GUI interface of the open source code SATURNE (by

EDF and CEA).

The FEM-LCORE GUI is an independent QT-PythonQT application that can be used as a standalone program with the purpose of generating the input file and executing the FEM-LCORE code. Furthermore, in order to enhance its performance, a version that can work inside the SALOME platform has been developed. The SALOME platform comes with a mesh generator and the PARAVIEW visualization application that allow us to complete the preprocessing and the postprocessing steps inside the platform itself. In Figures 1.2-1.3 a snapshot of the FEM-LCORE GUI integrated in the SALOME platform and a snapshot of the standalone version are shown. In the following we describe the FEM-LCORE GUI in the SALOME framework but the same informations can be used as user guide for the standalone version.

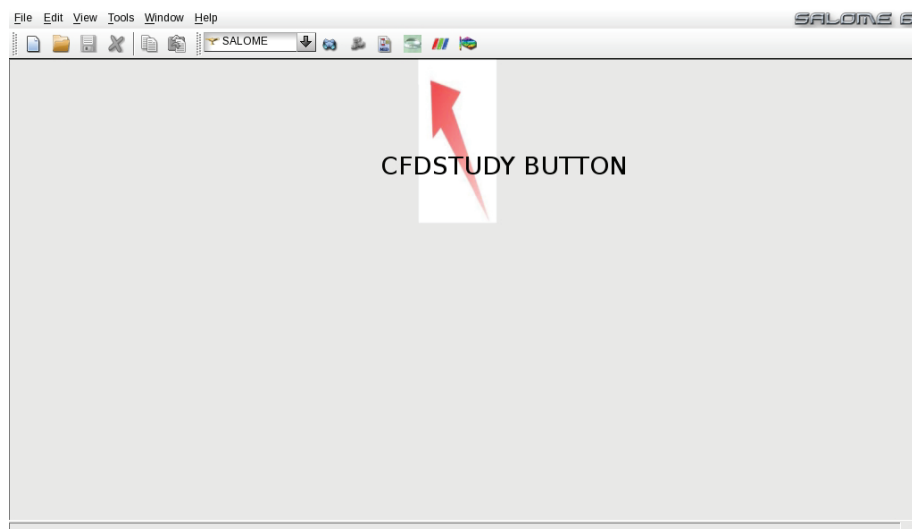


Figure 1.4: FEM-LCORE CFDSTUDY load button in salome GUI

The FEM-LCORE code is based on the FEMuS CFD code and therefore it must be started from the FEMuS main directory with the command

```
source femus.sh femlcore_1 salome
```

In case of error the application displays the following help message

```
Use: source femlcore_1.sh application_n [option]
```

```
applications: applications_n (femlcore_1; n=version)
```

```
options: None, opt, dbg, gui, nogui, gencase
         None    = no gui in optimize mode
         opt     = salome-gui in optimize mode
         dbg     = no gui in debugging mode
```

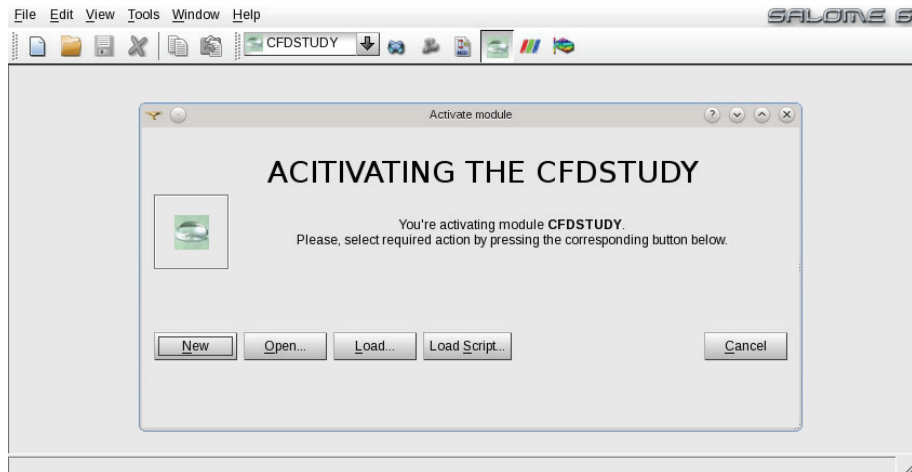


Figure 1.5: Activating a CFDSTUDY GUI

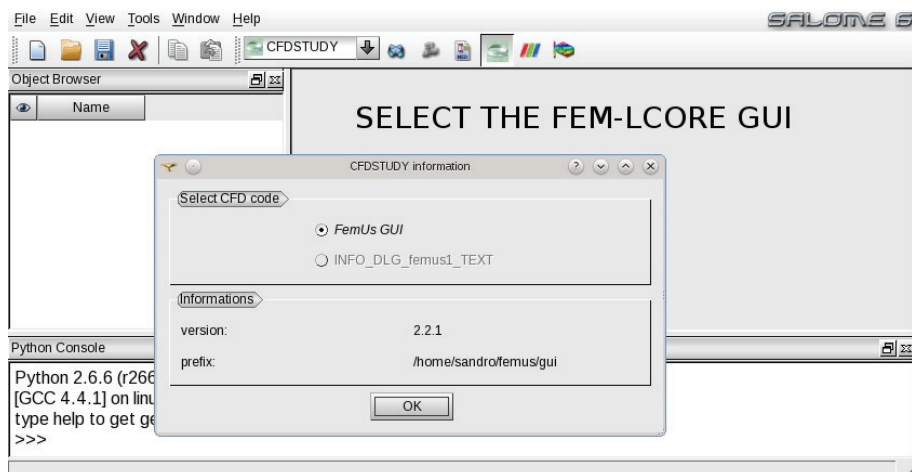


Figure 1.6: Activating the FEM-LCORE GUI in SALOME

```
nogui = no gui in optimize mode
gencase= no gui in optimize mode in gencase dir
```

The source command first sets the environment variables for the FEM-LCORE codes and the FEMuS, PETSC, MPI and LIBMESH libraries, then it starts the GUI. The `gencase` program, mentioned in the above command, is a program needed to generate the multigrid and the parallel element connectivities from the coarse mesh which is usually generated by GAMBIT and written in a file with `.neu` format. The standard running is in optimized mode and the debugging mode can be used only for detecting errors or anomalies. The `gencase` directory is the directory where the parallel multigrid mesh generator ( `gencase` program) runs. If the library environment variables are already set then from the `FEMuS/gui` directory one can simply write

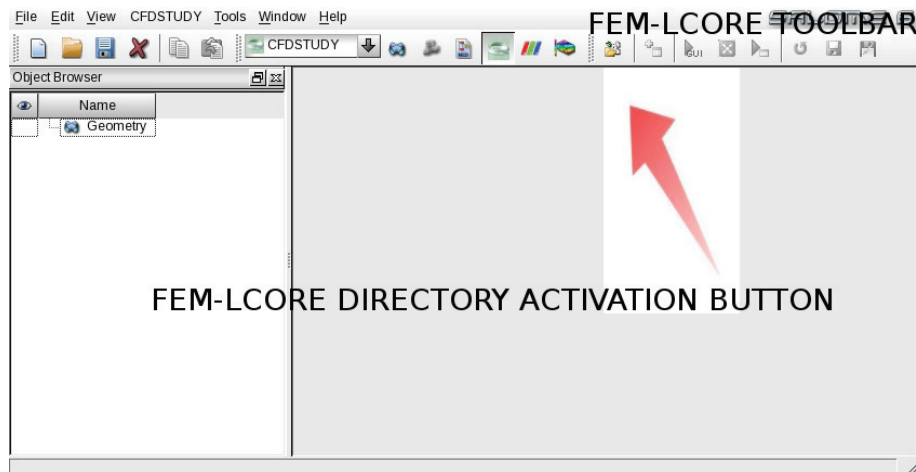


Figure 1.7: Activating the FEM-LCORE directory tree in SALOME

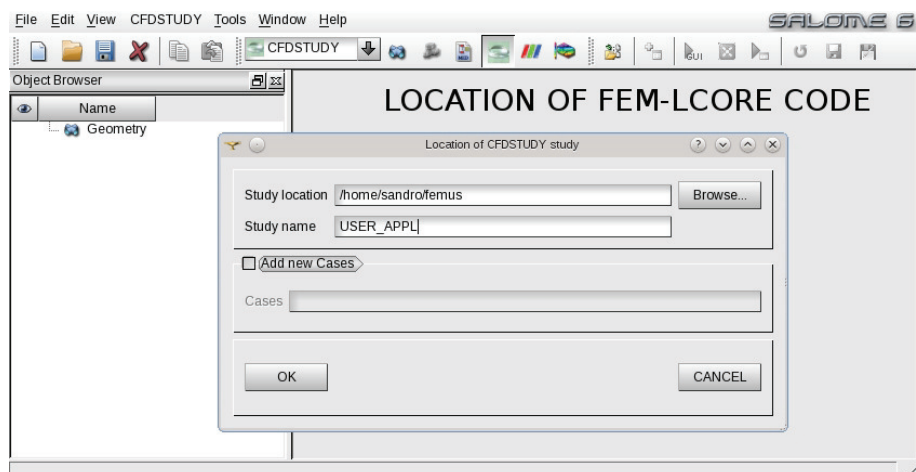


Figure 1.8: Activating the FEM-LCORE directory tree in SALOME

```
python ./bin/code_femus salome
```

The GUI interfaces for FEM-LCORE and SALOME are written in Python and therefore the Python environment must be installed.

In order to load the FEM-LCORE interface one must follow some simple steps. After the above source command is executed then the SALOME GUI starts and the screen in Figure 1.4 is shown. If the FEM-LCORE CFDDSTUDY load button is selected the screen dialogs of Figures 1.5 and 1.6 appear and the FEM-LCORE GUI is loaded in the SALOME GUI. The FEM-LCORE toolbar is loaded on the SALOME toolbar on the top right. In order to open the FEM-LCORE directory tree widget on the left of the main SALOME window one must activate a button on the FEM-LCORE toolbar on the top right as indicated in Figure 1.7.

The FEM-LCORE directory is set by filling the *Study location* and the *Study name* forms in the dialog window of Figure 1.8. The FEM-LCORE directory cannot

be changed and must be set as

```
.../FEMuS/USER_APPL/femlcore_1
```

In Figure 1.8 the study directory is defined by the directory name of the main FEMuS directory (study location= .../FEMuS/) and by the study name of the USER\_APPL directory (study name=USER\_APPL). At this point the FEM-LCORE directory tree appears in the SALOME directory window, as shown in Figure 1.9. In order

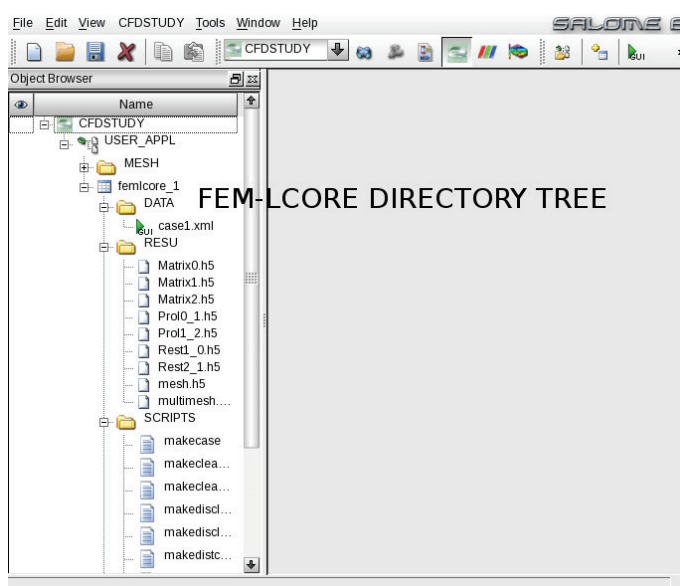


Figure 1.9: FEM-LCORE directory tree in SALOME

to open the FEM-LCORE GUI, a configuration file must be placed in the DATA directory of the FEM-LCORE application (see Figure 1.10). The configuration file in XML format is immediately detected by the SALOME platform and displayed under the DATA directory. If a XML file is not in this directory the FEM-LCORE GUI cannot be detected. In this case an empty XML file should be created in order to activate the SALOME platform. With the mouse right-click menu the *Open GUI* option can be selected and this selection immediately opens the FEM-LCORE GUI window as in Figure 1.11.

The FEM-LCORE GUI can start in different forms with different combinations of windows. The usual main FEM-LCORE graphical interface panel is divided into different zones. As one can see from Figure 1.11 we have the main SALOME window at the center of the screen. In this main window the SALOME platform can load different other applications such as CAD applications and mesh generators. It is also possible to load the PARAVIEW post-processor graphical program with the purpose of analyzing the results of the FEM-LCORE code. On the left of Figure 1.11 one can find the object browser window (FEM-LCORE directory window) with the directory tree. A large view of the SALOME browser window (FEM-LCORE directory window) can be seen in Figure 1.12 on the left. In particular one

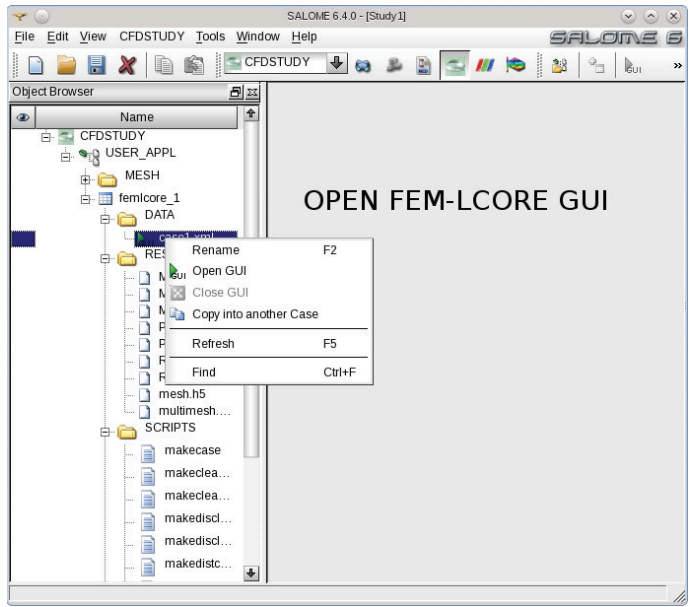


Figure 1.10: Open the FEM-LCORE GUI from the SALOME directory tree

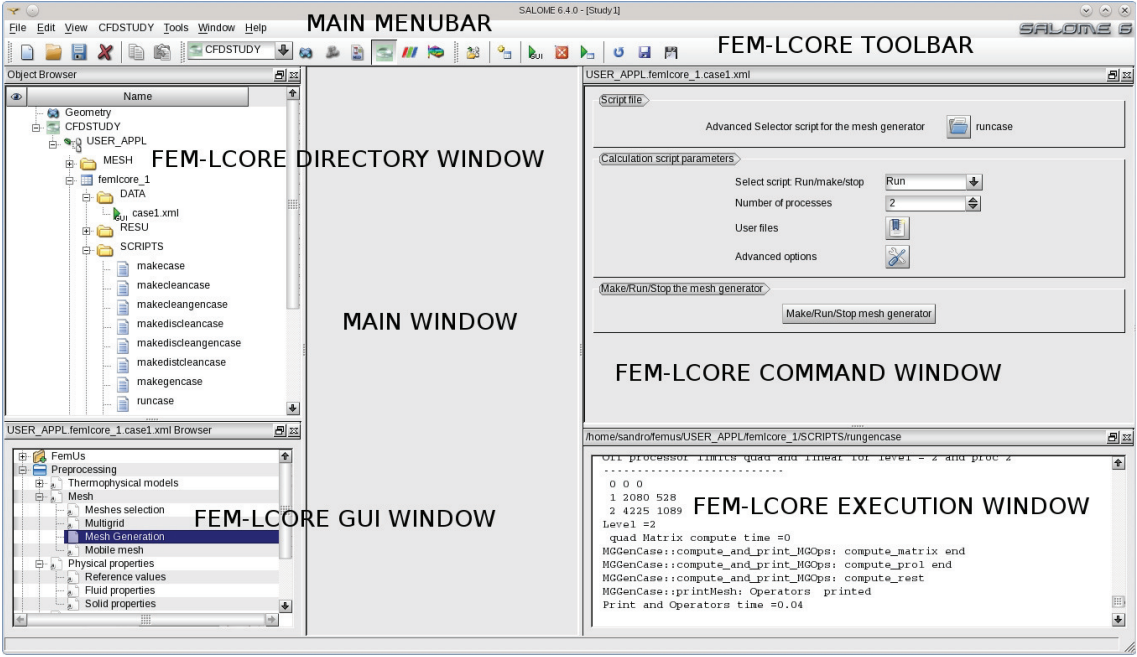


Figure 1.11: FEM-LCORE graphical interface panels: FEM-LCORE directory, execution, gui and command window

can note that the FEM-LCORE directory tree is under the CFDSTUDY and the USER\_APPL directories. Also one must remark that the MESH directory contains all the meshes for the FEM-LCORE studies and it is located immediately under

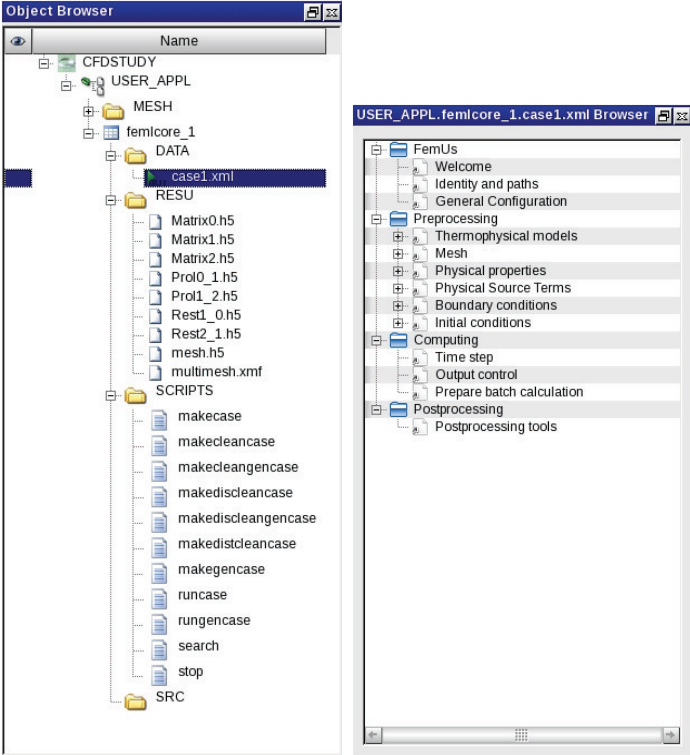


Figure 1.12: FEM-LCORE directory tree window and FEM-LCORE GUI window

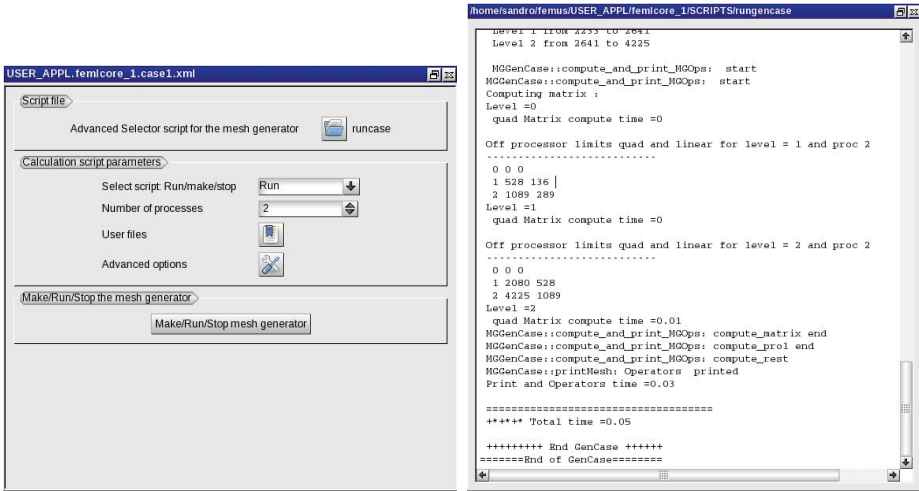


Figure 1.13: FEM-LCORE GUI command window and FEM-LCORE execution log window

USER\_APPL.

The FEM-LCORE directory (`femlcore_1`) consists of four subdirectories: `DATA` (configuration directory), `RESU` (result directory), `SCRIPTS` (script directory) and `SRC` (source file directory). The main directory (`femlcore_1`) contains only the Makefile



which helps in the compilation of the FEM-LCORE code. The FEM-LCORE GUI browser window, which is located on the bottom left side of the SALOME GUI, is shown in Figure 1.12 on the right. The tree in the FEM-LCORE GUI selects the pages with the options of the configuration file. The RESU directory contains all the input files generated by the mesh generator and all the output files that can be analyzed through the use of the PARAVIEW application in the XDMF format using HDF5 format files. The HDF5 libraries are the same libraries that are used by the SALOME applications to store and read files and meshes. The XDMF format is used in FEM-LCORE while the MED format is used in the SALOME applications. All the files that are needed for restart and visualization are contained in this directory. In the SCRIPTS directory the main scripts for the compilation and execution of the multigrid parallel mesh generator (gencase) and of the FEM-LCORE code can be found. The DATA directory contains the configuration files. A configuration file in XML format must be contained in this directory. Informations from this XML file are then copied in the FEM-LCORE run-time and compile-time configuration files, such as `parameters.in`, `param_files.in` and `Equation_conf.h`. On the right side of the SALOME GUI there are the FEM-LCORE GUI command and FEM-LCORE execution log windows. The FEM-LCORE GUI command window is shown on the right of Figure 1.13 and contains the details of pages in the FEM-LCORE GUI. On the right of Figure 1.13 there is the FEM-LCORE execution log window which is located on the bottom right side of the SALOME GUI in Figure 1.11 when execution starts. This window reports the log of the programs executed with different GUI commands.

## 1.2 GUI input configuration file

One of the main purposes of the FEM-LCORE GUI is to define the configuration file. The configuration file can be defined by following the tree-menu in the FEM-LCORE GUI window from the top to the bottom. The configuration file is written in a single XML file. The tree in the FEM-LCORE GUI window consists of three main blocks of pages divided in agreement with the working steps for running a CFD code: preprocessing, computing and postprocessing. On the top of the GUI tree-menu there are pages that summarize the code behavior. Expert users can use these pages to open and modify all the input files without going through the rest of the tree-menu items.

### 1.2.1 Introductory and general GUI FEMuS pages

On the top of the preprocessing directory of the GUI tree one has the FEMuS directory. The FEMuS directory summarizes the working directories of the FEM-LCORE project and all the relevant files. When one selects the FEMuS block, three sub-nodes are shown: the *Welcome* page, the *Identity and path* page and the *Gen-*

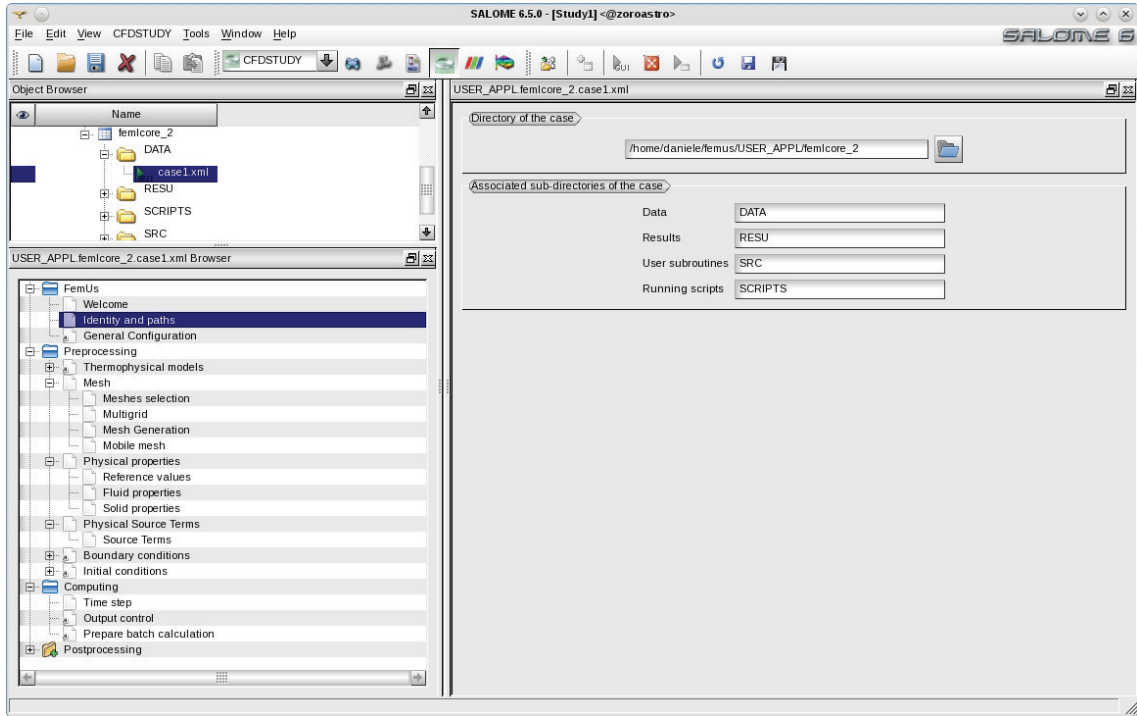


Figure 1.14: FEMuS GUI menu: working directory page

eral configuration page. If one selects the *Identity and path* node the corresponding page is opened on the right on the GUI command window. As shown in Figure 1.14 the *Identity and path* page shows the working (or case) directory of the code. On the top of the page the directory of the case is shown. The button on the top right may be used to browse the filesystem and find the case directory. The working directories, or associated subdirectories of the case, are four: SRC, RESU, DATA and SCRIPTS. The associated subdirectories are placed below the case directory. The MESH directory is placed at the same level as the case directory and it is not shown inside this summary page. The general configuration page, shown in Figure 1.15, gives the possibility to display all the files of the FEM-LCORE project. In particular this page contains four combo-menus that can be used to edit and display all the required files. The first menu is related to the configuration files stored in the DATA directory. The menu can open the following configuration files: `case1.xml`, `param_files.in`, `parameters.in`, `Equations_conf.h`, `MGFE_conf.h`, `Solverlib_conf.h`, `Domain_conf.h`, `MGSclass_conf.h` and `Printinfo_conf.h`. The `case1.xml` file is the configuration file generated from the FEM-LCORE GUI. This file is not directly involved in the compilation and execution of the code, but the information contained in it is modified through the GUI and then copied to the other configuration files. The `param_files.in` and `parameters.in` files control the code parameters that can be set at run-time and the other files ending with `_conf.h` consists of compile-time preprocessor (`#define`)

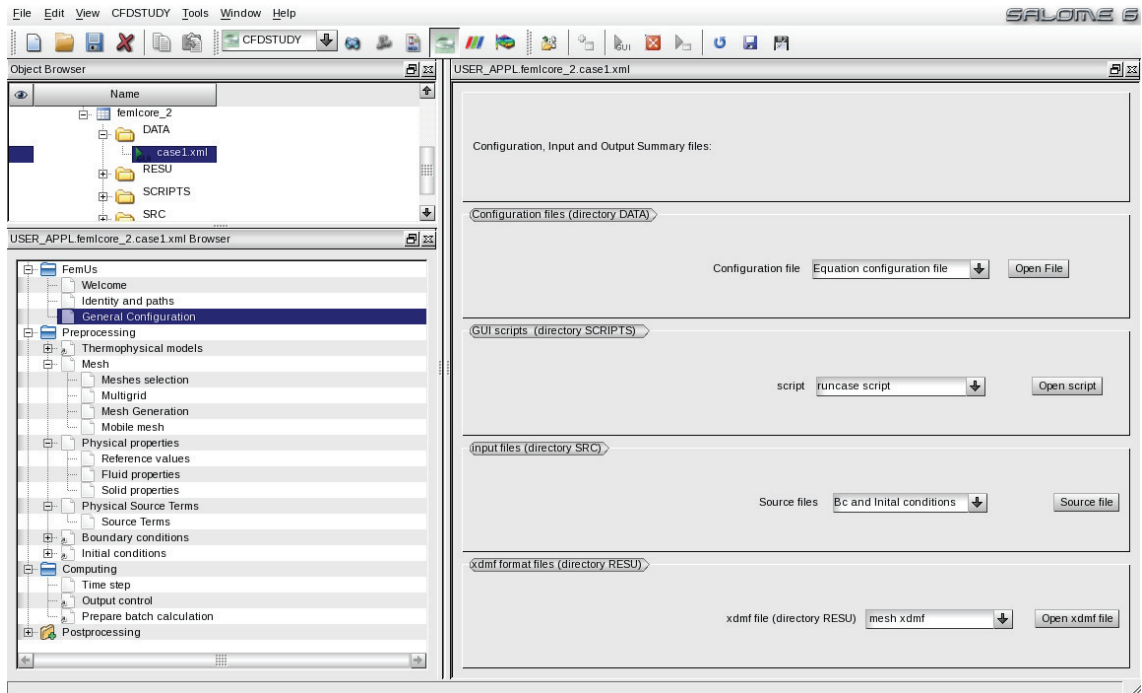


Figure 1.15: FGUI FEMuS menu: general configuration page

instructions. A modification of these data requires to recompile the code. The name of each file explains its area of interest. The other three combo-menus (SCRIPTS, DATA, RESU) control the editing and displaying of the GUI scripts, the input and the output files, respectively. The input files control boundary and initial conditions. The output files are written in XDMF format that allows PARAVIEW, or in general the VTK viewer, to display the FEM-LCORE output.

## 1.2.2 Preprocessing GUI pages

The preprocessing pages are the most important part of the GUI and consist of six directories: *Thermophysical models*, *Mesh*, *Physical properties*, *Physical source terms*, *Boundary conditions* and *Initial conditions*.

### Thermophysical models

The *Thermophysical models* item contains the list of equations that are involved in the simulation. For a detailed discussion one can read Chapter 1.3.5 of this report. In particular one can activate the Navier-Stokes model, the energy equation and different turbulence models. The *Thermophysical models* directory contains the *Equation models* and *Equation advanced setting* pages. As shown in Figure 1.16 the *Equation models* page consists of several combo-menus, each of which allows the selection of a particular algorithm for the specific equation. For instance, the combo-

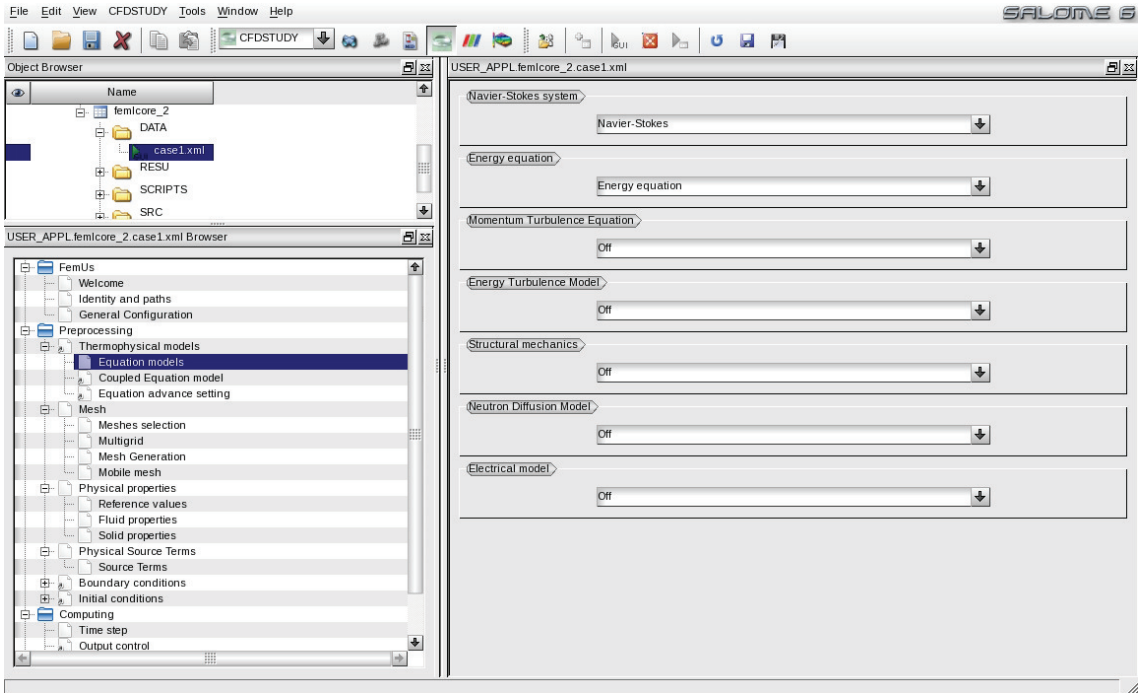


Figure 1.16: *Preprocessing* GUI menu: *Equation models* page.

menu of the Navier-Stokes equations contains three options: coupled, projection and segregated. Each option corresponds to a different solution algorithm. For example the projection method splits the velocity and pressure equations. The Navier-Stokes option is the slowest but the more precise option since it solves the system in the original velocity-pressure coupled way. Not all the physical equation models, shown in the page and used in the FEMuS code, are available for the particular FEM-LCORE application. In the *Equation advanced setting* page we can find a combobox that controls the displaying and editing of the class configuration files. This page can be seen in Figure 1.17. Each equation is defined by a C++ object-oriented class, where several parameters are defined inside specific header files.

## Mesh

The *Mesh* pages control the loading of the mesh file inside the FEM-LCORE code. This directory consists of three pages: *Mesh selection*, *Multigrid* and *Mesh generation*. The *Mesh selection* page is in Figure 1.18 and locates the mesh in the appropriate directory. The default mesh directory is the MESH subdirectory of the USER\_APPL directory. This directory is at the same level as the DATA, RESU, SCRIPTS and SRC directories. The mesh can be generated with GAMBIT or with the SALOME mesh generator. The FEM-LCORE code has an internal generator that can be used to mesh 2D and 3D box-shaped geometries. The input mesh from GAMBIT or SALOME is not a parallel or a multigrid mesh and therefore a pro-

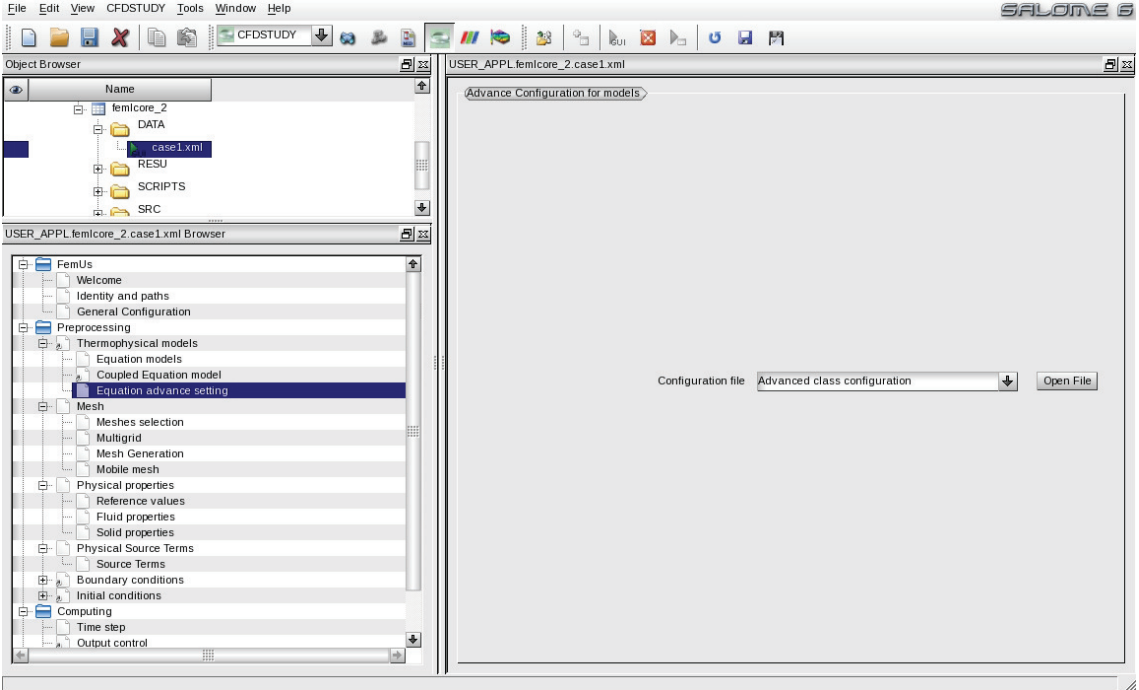


Figure 1.17: *Preprocessing* GUI menu: *Equation advanced setting* page.

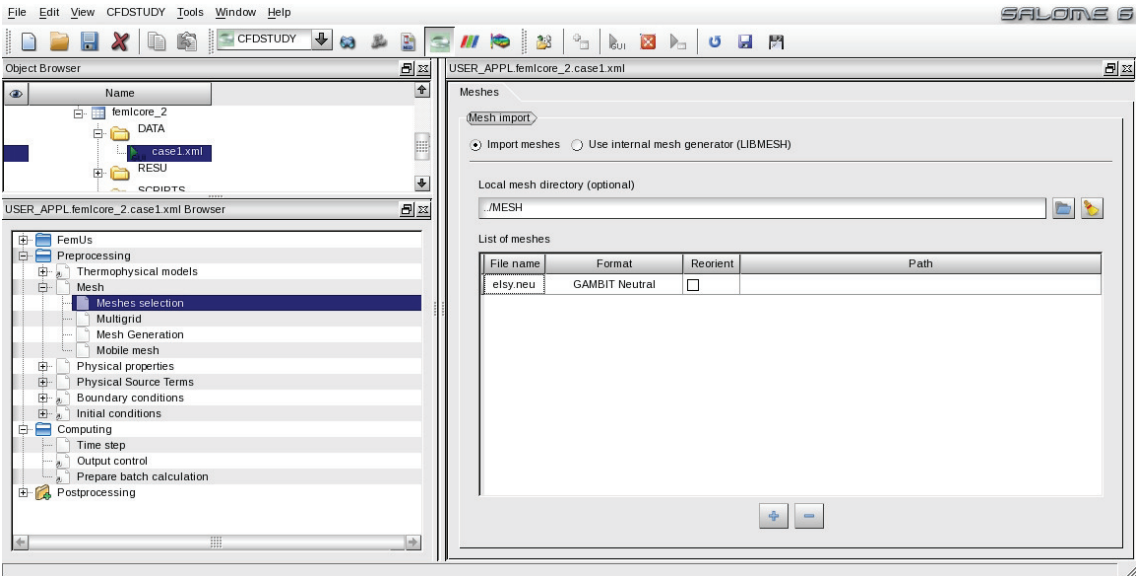


Figure 1.18: *Preprocessing* GUI menu: mesh selection page

gram must be used for parallel and multigrid computations. This program is called **gencase** and it is part of the FEMuS code. Inside the **gencase** program there is a simple internal coarse mesh generator limited to regular Cartesian geometries. To obtain a parallelepiped reactor of dimensions  $[a, b] \times [c, d] \times [e, f]$  the command,

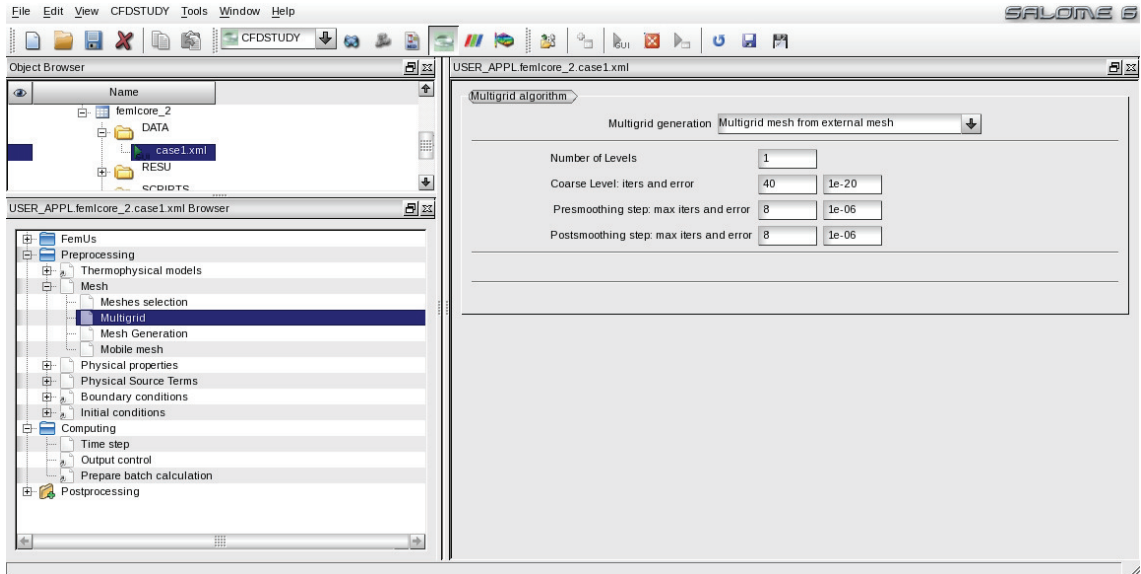


Figure 1.19: *Preprocessing* GUI menu: multigrid option page

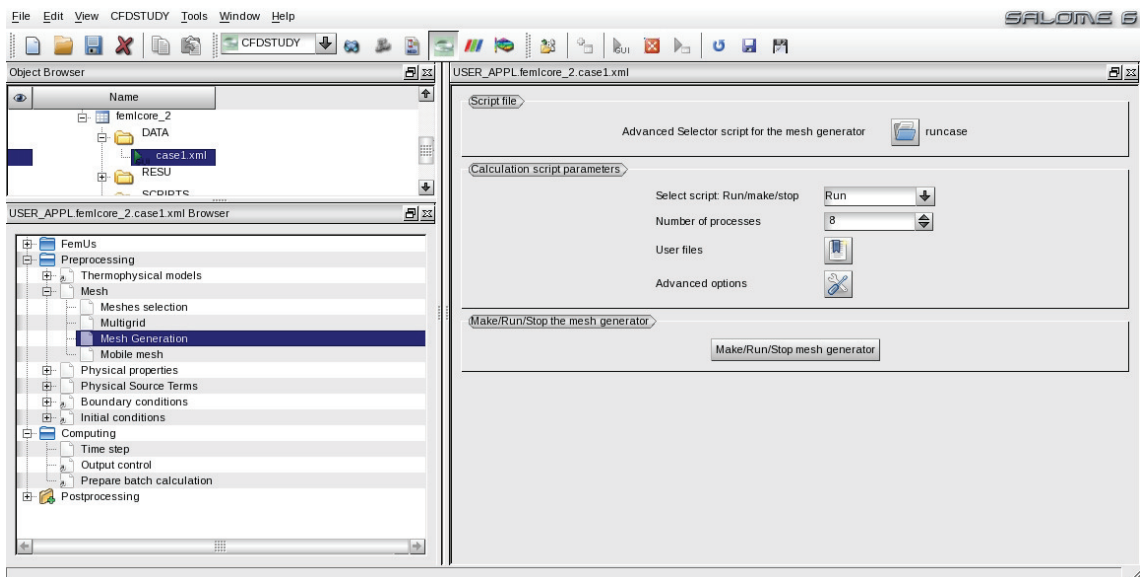


Figure 1.20: *Preprocessing* GUI menu: mesh generation page

which makes use of the LibMesh library, is written as

```
MeshTools::Generation::build_cube
(*msh[0],nintervx,nintervy,nintervz,a,b,c,d,e,f,HEX27);
```

This command can be modified, for example, if a cylindrical axisymmetric geometry is needed. The geometries obtained with such a `build_cube` function are very simple. For this reason, the need arises to interact with the CAD-based mesh software available in the SALOME platform or with commercial applications. The `gencase`

program has been extensively modified after the introduction of parallel computing feature in the FEMuS code. In particular, the parallel structure of the PETSc matrices requires that the degrees of freedom associated to each process are contiguous to each other. Therefore the vector of global degrees of freedom must be divided into as many blocks as processors. Each block is assigned exactly to one processor and therefore it is necessary to rerun the `gencase` mesh generator every time the processor number is changed. In order to facilitate the operations of projection and restriction, *children* elements are assigned to the same process as their *father* element and in this way a node that is *restricted* on a coarse grid is kept on the same processor. This increases significantly the complexity of the generated mesh files as the number of processors increases. The radio-buttons on the top of the *Mesh selection* page select the external or internal mesh generator. The FEM-LCORE code comes with some reactor meshes generated with GAMBIT. In this case one must select the external mesh generator. The mesh directory can be found through the corresponding search directory button and the path is then loaded in the appropriate line. When the directory is set the file can be selected. The plus or minus buttons allow the user to load and unload any files from the mesh directory selected above.

The *Multigrid* page, shown in Figure 1.19, controls the multigrid algorithm. The combo-menu sets the external and internal generator options. The internal generator needs the dimensions of the box-shaped domain and therefore other data are needed. The input mesh file contains the coarse mesh and this mesh can be refined multiple times. A refinement is constructed by the LIBMESH libraries with simple midpoint refinement rules. For both generators the number of levels of the multigrid algorithm is required. Information about the number of pre and post-smoothing iterations and corresponding tolerances can be set. The type of multigrid cycle can be controlled inside the multigrid class configuration file, where it can be chosen as V cycle , W cycle or full multigrid. The smoother can be picked among several PETSC linear solvers:

1. Jacobi = JacobiIter
2. SOR forward = SORForwIter
3. SOR backward = SORBackwIter
4. SOR symmetric = SSORIter
5. Chebyshev = ChebyshevIter
6. CG = CGIter
7. CGN = CGNIter
8. GMRES(10) = GMRESIter
9. BiCG = BiCGIter
10. QMR = QMRIter
11. CGS = CGSIter
12. Bi-CGSTAB = BiCGSTABIter

Among the preconditioner matrices one can choose

- 0. none = (PrecondProcType)NULL
- 1. Jacobi = JacobiPrecond
- 2. SSOR = SSORPrecond
- 3. ILU/ICH = ILUPrecond.

The CGS conjugate gradient can be chosen for symmetric matrices. By default the GMRES smoother is set with ILU preconditioner. For parallel computations, Gauss-Seidel block algorithm is used with desired smoother and preconditioner over each block.

### Physical properties

In the *Physical properties* pages the reference quantities and the physical properties are set. The *Reference values* page is shown in Figure 1.21. Usually all these

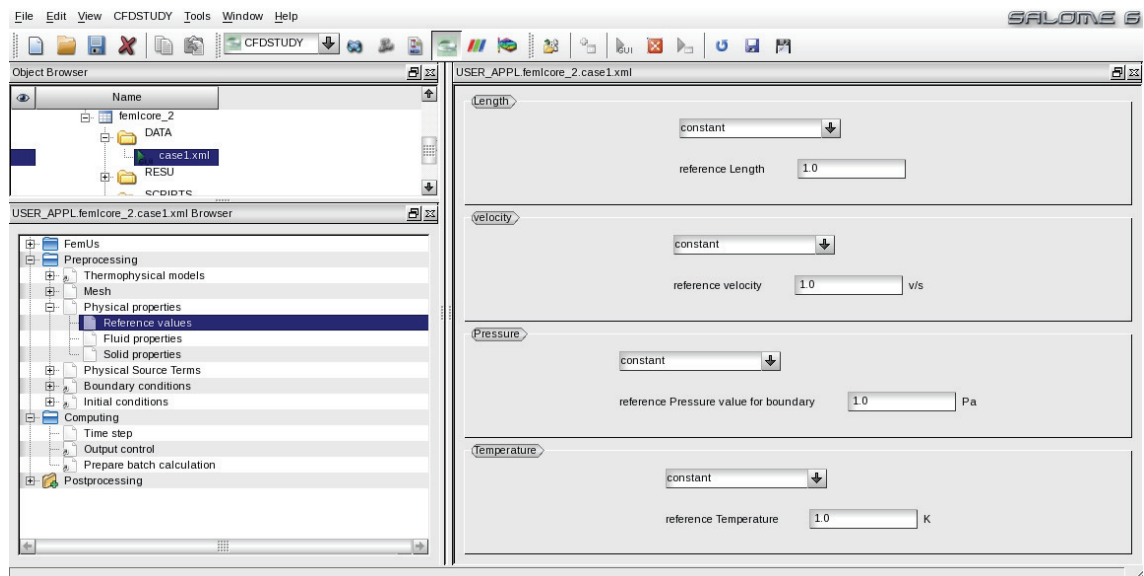


Figure 1.21: *Preprocessing* GUI menu: reference variable

quantities are set to 1. The physical properties can be inserted in the page of Figure 1.22. One can insert a single value or a parametric law. In the implementation of the FEM-LCORE code we focus only on liquid lead as coolant. For liquid lead coolant one can see [7, 10]. The lead density may be assumed to be a function of temperature as [7, 10]

$$\rho = (11367 - 1.1944 \times T) \frac{Kg}{m^3} \quad (1.1)$$

for lead in the range  $600K < T < 1700K$ . The following correlation has been used for the viscosity  $\mu$

$$\mu = 4.55 \times 10^{-04} e^{(1069/T)} Pa \cdot s, \quad (1.2)$$



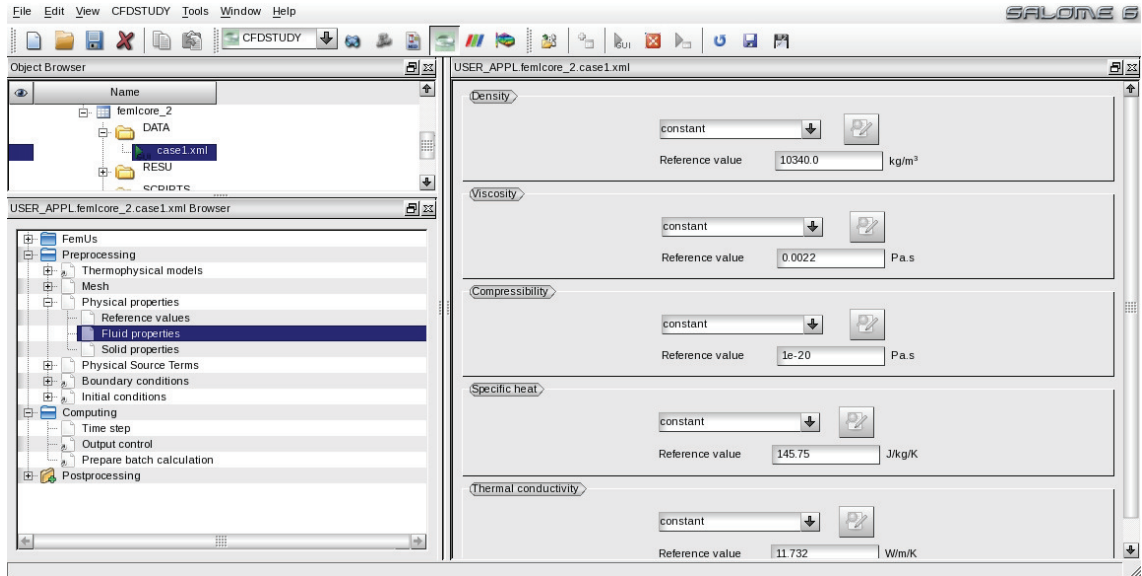


Figure 1.22: *Preprocessing* GUI menu: physical properties

for lead in the range  $600K < T < 1500K$ . For the mean coefficient of thermal expansion (AISI 316L) we assume

$$\alpha_v = 14.77 \times 10^{-6} + 12.20^{-9}(T - 273.16) + 12.18^{-12}(T - 273.16)^2 \frac{m^3}{K}. \quad (1.3)$$

The lead thermal conductivity  $\kappa$  is

$$\kappa = 15.8 + 108 \times 10^{-4} (T - 600.4) \frac{W}{m \cdot K}. \quad (1.4)$$

The constant pressure specific heat capacity for lead is assumed not to depend on temperature, with a value of

$$C_p = 147.3 \frac{J}{Kg \cdot K}. \quad (1.5)$$

Not all the options displayed in the dialog window are active and many options must be written inside the class configuration files.

### Source terms, fuel distribution, pressure drops

The sources for the momentum and energy equations can be set in the pages of the *Physical source terms* tree item shown in Figures 1.23-1.24. The volumetric momentum source, such as gravity, can be specified in the first source lines of the *Source terms* page in Figure 1.23 and the heat source in the last line. Here the heat source should be set as an average value for the reactor independently of a space-dependent configuration. The space configuration of the heat source is taken

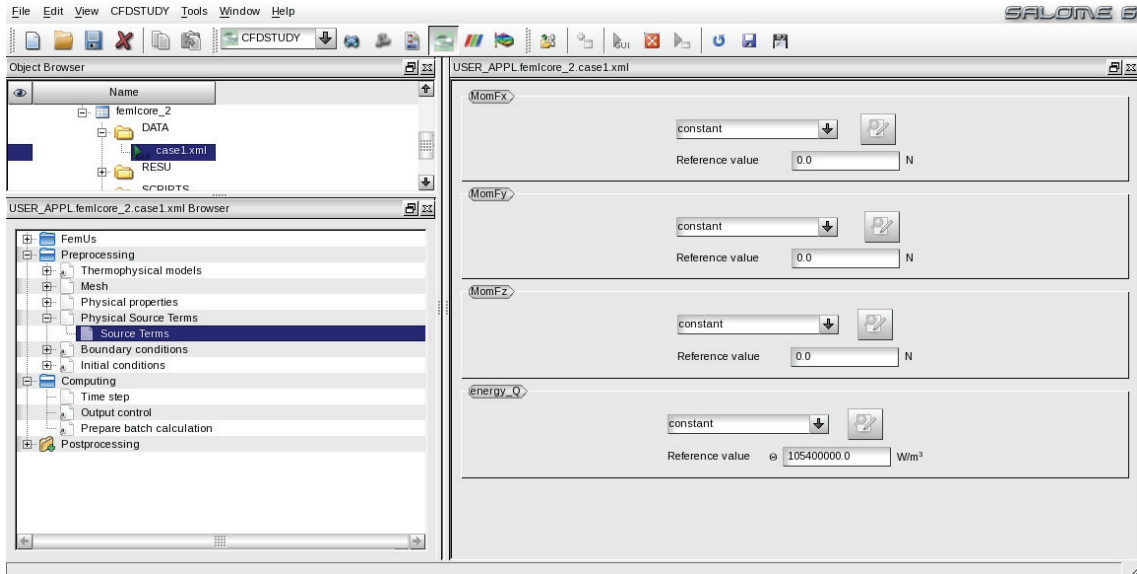


Figure 1.23: *Preprocessing* GUI menu: *Source terms* page

into account by using the peak factors which can be controlled in the *Fuel and pressure losses* page shown in Figure 1.24. The page in Figure 1.24 consists of a combo-menu where the fuel and pressure loss distribution must be defined over the core. The combo-menu has three options: *Fuel distribution file*, *Pressure loss file*

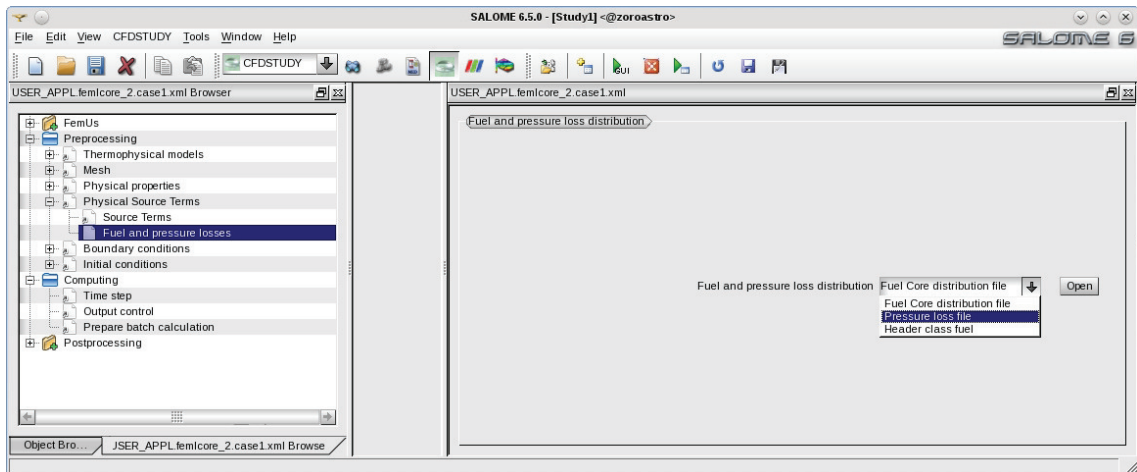


Figure 1.24: *Preprocessing* GUI menu: fuel distribution and pressure drops

and *Header fuel class file*. The fuel distribution and pressure loss options open the same file where plane and axial distributions can be modified. The power distribution and the pressure drop distribution are set in the `ReactData` class in the `SRC/ReactData.h` and `SRC/ReactData.C` files. A typical power distribution is shown in Figure 1.25.

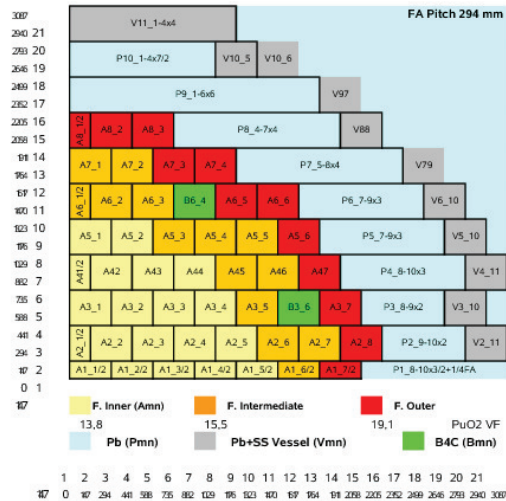


Figure 1.25: Typical core power distribution

In order to load the desired power distribution as in Figures 1.26 and 1.27, one must set the following configuration parameters in the header file, which is opened by selecting the Header fuel class file item: NLEV (number of multigrad levels), NZC (number of elements in the vertical core section), HR (half fuel assembly length), HIN (heated core inlet height), HOUT (heated core outlet height):

```
#define NLEV 1 // levels
#define NZC nc // number of core elements
#define HR hr // half fuel assembly length
#define HIN h_in // heated core start
#define HOUT h_out // heated core ends
```

The horizontal power distribution of Figure 1.25 must be reported in the double-indexed array `mat_pf` in the `ReactData` class, in which every assembly in the quarter of reactor is denoted by two indices, both ranging from 0 to 7. The fuel area is divided into different zones, classified as `INNER` (inner core), `OUTER` (outer core), `CTRLR` (control zone with no fuel) and `DUMMY` (reflector area). These are written in the `mat_zone` matrix.

```
#define INNER (0) // zone 0
#define OUTER (1) // zone 1
#define CTRLR (2) // zone 2
#define DUMMY (3) // zone 3
```

```
const int ReactData::mat_zone[8][8]={
```

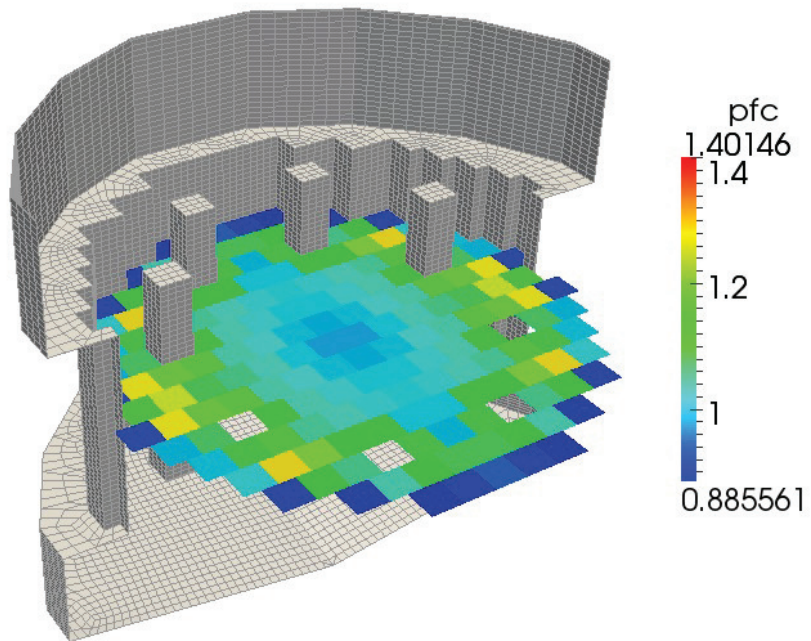


Figure 1.26: Horizontal core power distribution

```

{INNER, INNER, INNER, INNER, INNER, INNER, OUTER, DUMMY}, //A1_1-A1_7
{INNER, INNER, INNER, INNER, INNER, INNER, OUTER, OUTER}, //A2_1-A2_8
{INNER, INNER, INNER, INNER, CTRLR, INNER, OUTER, DUMMY}, //A3_1-A3_7
{INNER, INNER, INNER, INNER, INNER, INNER, OUTER, DUMMY}, //A4_1-A4_7
{INNER, INNER, INNER, INNER, OUTER, OUTER, DUMMY, DUMMY}, //A5_1-A5_6
{INNER, INNER, CTRLR, INNER, OUTER, OUTER, DUMMY, DUMMY}, //A6_1-A6_6
{INNER, OUTER, OUTER, OUTER, DUMMY, DUMMY, DUMMY, DUMMY}, //A7_1-A7_4
{OUTER, OUTER, OUTER, DUMMY, DUMMY, DUMMY, DUMMY, DUMMY} //A8_1-A8_3
};

```

The horizontal power factor is reported in the `mat_pf` matrix:

```

#define CTL_R 0.
const double ReactData::mat_pf [8] [8]={
{0.941,0.962,0.989,1.017,1.045,1.178,1.097,0.   }, //A1_1-A1_7
{0.949,0.958,0.978,0.996,1.114,1.123,1.193,0.857}, //A2_1-A2_8
{0.963,0.975,0.992,1.087,CTL_R,1.011,0.925,0.   }, //A3_1-A3_7
{0.967,0.973,0.989,1.008,1.108,1.028,0.931,0.   }, //A4_1-A4_7
{0.961,1.060,1.078,1.137,1.198,0.943,0.,   0.   }, //A5_1-A5_6

```

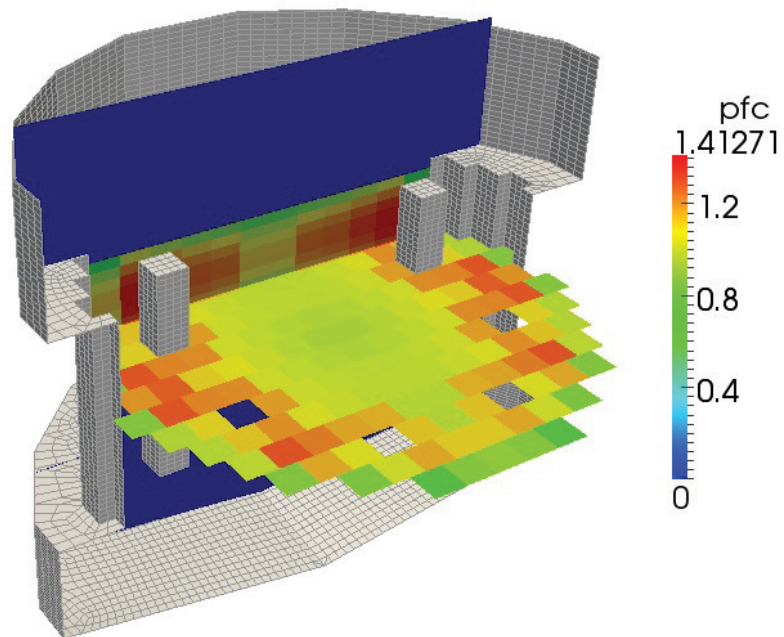


Figure 1.27: Vertical and horizontal core power distributions

```
{0.954,1.029,CTL_R,0.990,1.068,0.850,0., 0. }, //A6_1-A6_6
{1.019,1.066,0.966,0.842,0., 0., 0., 0. }, //A7_1-A7_4
{0.878,0.853,0.757,0., 0., 0., 0., 0. } //A8_1-A8_3
};
```

The vertical power factor can be assumed to have any distribution  $g(z)$ . This discrete profile is written in the `axpf` array as

```
const double ReactData::axpf [10] [2]={
    { A0 g(z_0), A1 g(z_0) },
    { A0 g(z_1), A1 g(z_1) },
    { A0 g(z_2), A1 g(z_2) },
    { A0 g(z_3), A1 g(z_3) },
    { A0 g(z_4), A1 g(z_4) },
    { A0 g(z_5), A1 g(z_5) },
    { A0 g(z_6), A1 g(z_6) }
};
```

where  $z_i$  are the discrete vertical coordinates and  $A0$  and  $A1$  are the values in the INNER and OUTER zones. In a similar way the pressure drop distribution must

be reported in the `ax1pf` matrix and in the `ax11f` vector. The power distribution defined by the grid in Figure 1.25 can be easily written by defining the modulus of the vector function  $|\bar{\beta}|$  where the discrete pressure drops are defined by  $0.5\bar{\beta}(\mathbf{x}) \cdot \mathbf{v}|\mathbf{v}|$ . In our case we assume the flow to be approximately longitudinally developed and therefore  $0.5\bar{\beta}(\mathbf{x}) \cdot \mathbf{v}|\mathbf{v}| = 0.5|\bar{\beta}||\mathbf{v}|^2$  as in the usual monodimensional approximation. Once the fuel distribution and the pressure drops are configured, one can compile and run the FEM-LCORE code in the usual way.

## Boundary conditions

The boundary conditions can be set in two ways: using an explicit routine in the source files or defining boundary zones with a graphical mesh generator. Actually only the first method works properly for the FEM-LCORE code but we plan in the future to add the boundary zone approach partially developed for the FEMuS code. In Figure 1.28 we see a combo-menu with the option to display and edit

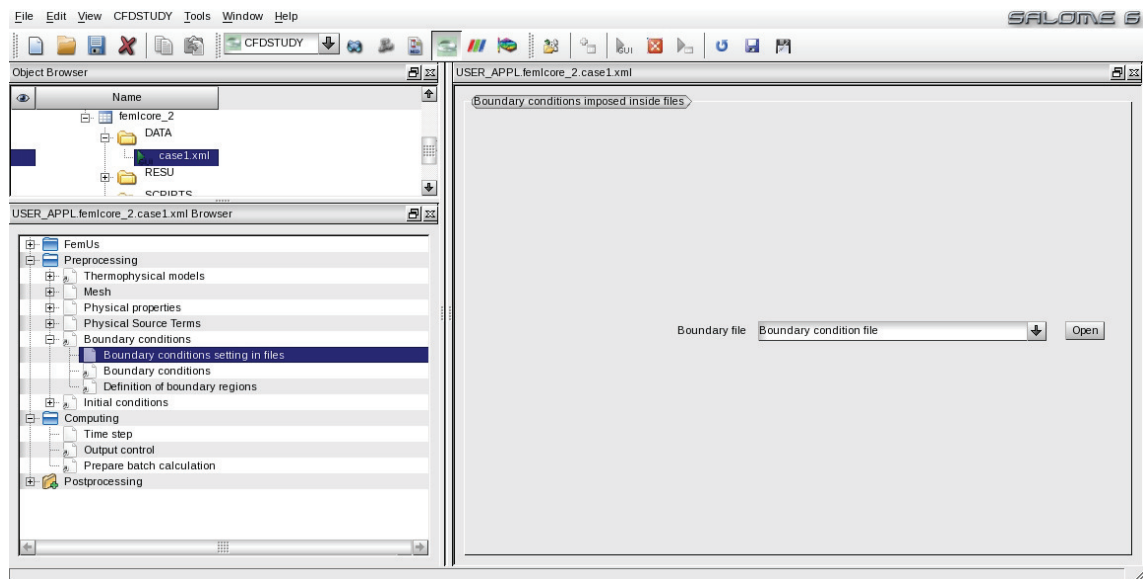


Figure 1.28: *Preprocessing* GUI menu: boundary conditions by code file

the boundary condition files. The boundary condition file is `User_BC.C` for all the boundary conditions. Sometimes this file is split into many files, one for each class. Therefore we may have `User_NS.C`, `User_T.C` and `User_TBKE.C` for Navier-Stokes, energy and turbulence equations respectively.

The boundary conditions are rather complex due to the reactor geometry. For pressure and velocity boundary conditions one must edit the function

```
void MGSolNS::bc_read(
    double xp[], // xp[] node coordinates
    int normal[], // normal
```

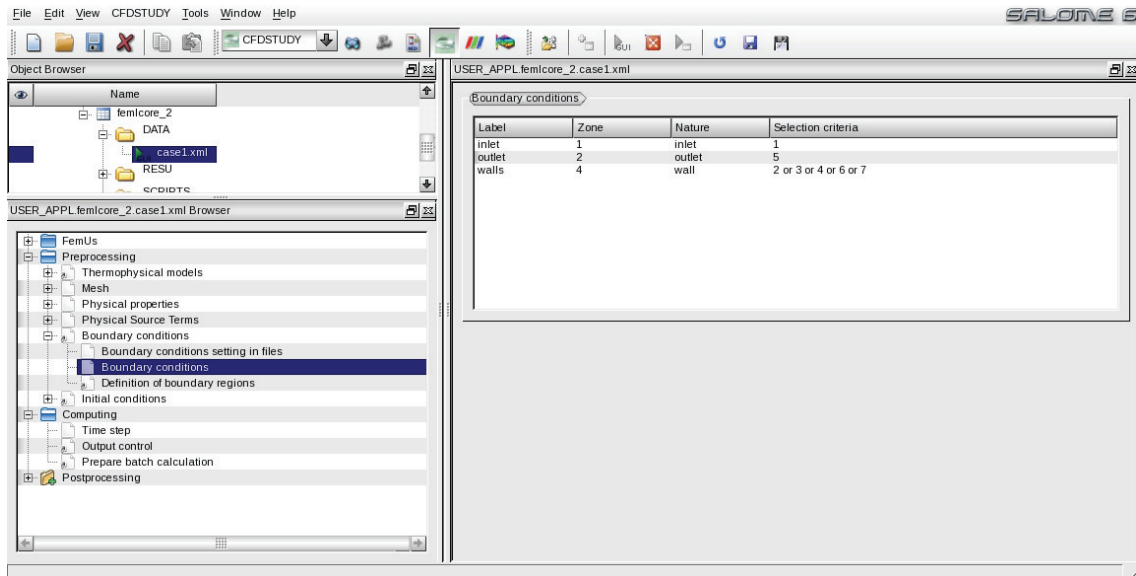


Figure 1.29: *Preprocessing* GUI menu: boundary conditions imposed by zone approach

```

    int bc_flag[] // boundary condition flag
)

```

in the class file SRC/User\_NS.C or in SRC/User\_BC.C. The  $\mathbf{xp}$  vector is the coordinate vector of each point and all the boundary conditions must be a function of the coordinates  $(x, y, z) = (x[0], x[1], x[2])$ . We remark that the points passed to the `bc_read` function are only points located on the boundary. For instance, one may set Dirichlet boundary conditions over inlet boundaries and symmetry conditions over symmetric planes. The implementation of the inlet velocity boundary conditions is quite complex since the reactor surface is not always aligned with the Cartesian reference frame.

For the boundary conditions of the energy equation one must edit the function

```

void MGSolT::bc_read(
    double xp[], // xp[] node coordinates
    int normal[], // normal
    int bc_flag[] // boundary condition flag
)

```

in the file SRC/User\_BC.C or SRC/User\_T.C. The  $\mathbf{xp}$  vector is the coordinate vector and `bc_flag[0]` is the return temperature boundary condition flag. For instance, one may set Dirichlet inlet conditions with fixed temperature or homogeneous Neumann boundary conditions by using the coordinate points of the boundary.

In Figure 1.29 the GUI page for the boundary zone approach is shown. The division in zones is detected automatically by the Gambit mesh partitioner. In this

case the user must only associate each type of boundary condition to the appropriate area.

## Initial conditions

The initial conditions can be set in two ways: using an explicit routine in the source files or with a boundary zone graphical approach. Actually only the first method works properly for the FEM-LCORE code and it is the more appropriate. We plan in the future to add the boundary zone approach now partially implemented for the FEMuS code. The combo-menu in Figure 1.30 has different options in order to

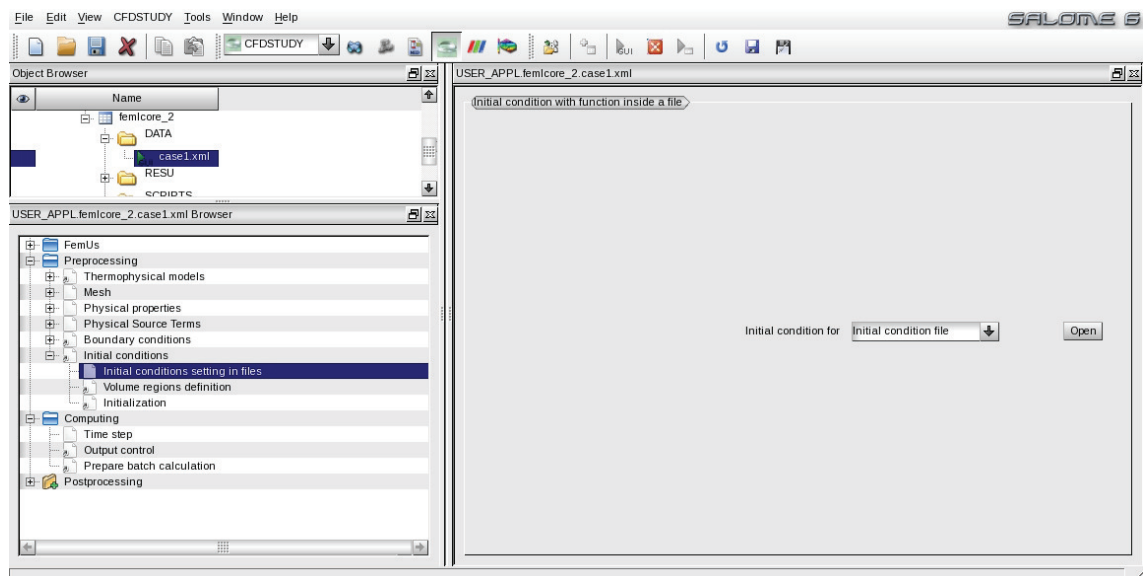


Figure 1.30: *Preprocessing* GUI menu: initial condition by code file

open the file for the initial conditions. The files where the boundary conditions and the initial conditions can be found are the same and can be edited from this page or from the boundary condition page. The initial condition file is `User_BC.C` for all the initial conditions. As already mentioned above sometimes this file is split into many files, one for each class. For example, if one wants to set the inlet velocity on the inlet surface one needs to set this velocity distribution in the function

```
void MGSolNS::ic_read(
    double xp[],
    double u_value[]
)
```

inside the `SRC/User_BC.C` file. We remark that the point passed to the `ic_read` function may be located over all the mesh volume and not only on the boundary as for the previously discussed boundary conditions. An initialization of the inlet temperature should be set inside `SRC/User_BC.C` or `User_T.C` in the function



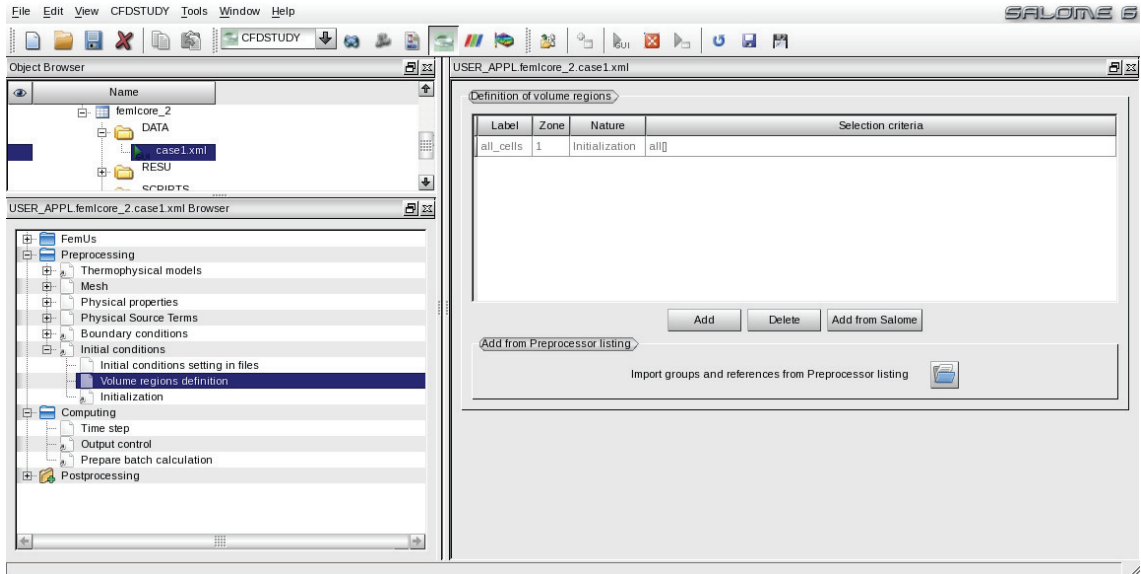


Figure 1.31: *Preprocessing* GUI menu: initial condition imposed by zone approach

```
void MGSolT::ic_read(
    double xp[],
    double u_value[]
) .
```

The `xp` vector is the coordinate vector and `u_value[0]` is the return temperature value.

The zone approach, shown in Figure 1.31, is still under development. In the zone approach a division in zones is detected automatically in the GUI since the zone partition has been defined by the Gambit mesh generator. In this case the user must only associate each type of initial condition to the appropriate area.

### 1.2.3 *Computing* GUI pages

The *Computing* menu consists of three pages: *Time step*, *Output control* and *Prepare batch calculation* page.

The *Time step* page, shown in Figure 1.33, defines the time step through a combo-menu. The FEM-LCORE code can only have a constant time step. One can also select the time step size and the number of time iterations. The steady state is obtained when the transient evolution is finished. There is no special treatment for a steady solution since the code is time-dependent. In order to restart the simulation from the output file associated to the time step `n` it is simply necessary to set to `n` the `restart number` line and then launch again the FEM-LCORE executable with the same number of processors as before. The restart initial time can be set by the user as can be seen in Figure 1.33. The new user initial time is only a label and is not involved in the computations. The initial time is important in the post-processing

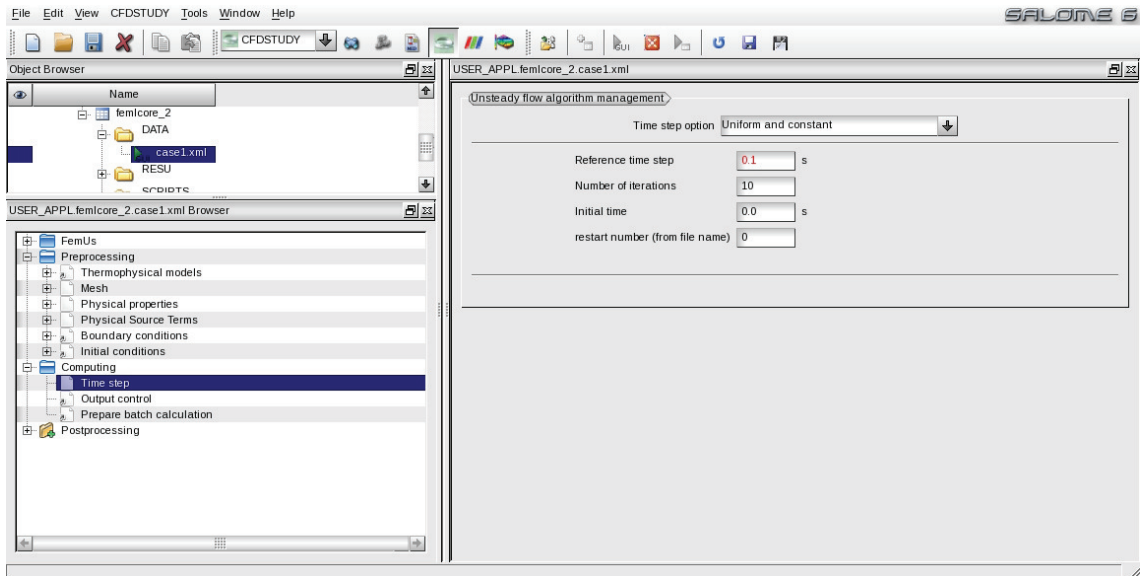


Figure 1.32: *Computing* GUI menu: time step page

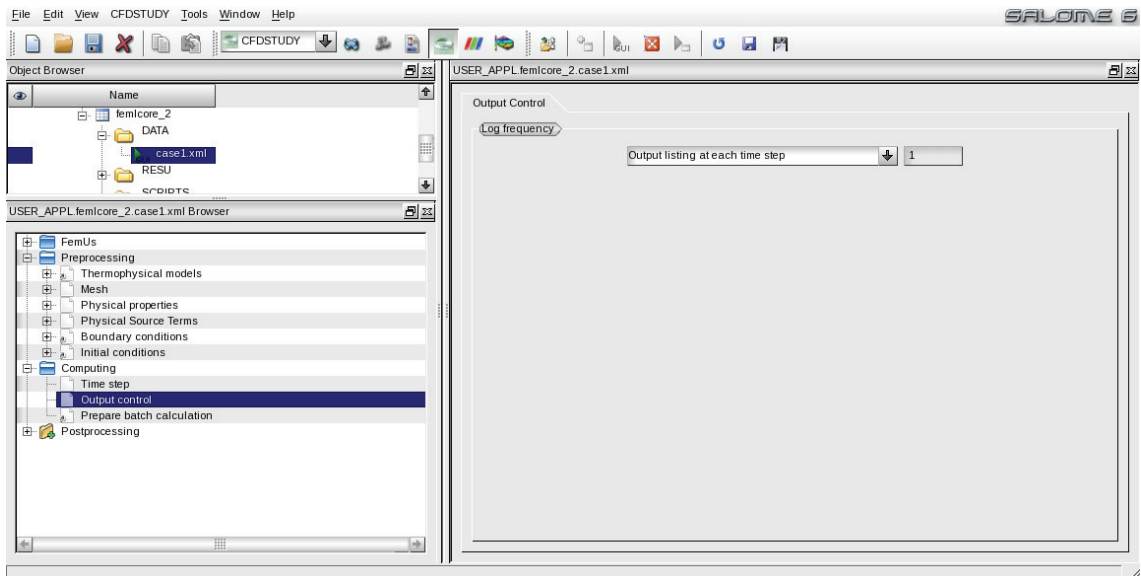


Figure 1.33: *Computing* GUI menu: time output control page

stage where some viewer, like PARAVIEW, can read this time value and adjust the time sequence.

The *Output control* page is reported in Figure 1.32. In this page there is only one combo-menu with three options: no output, one output for each time step and an output every  $n$  time steps. Only the last option needs an integer number that can be written in the related GUI field.

The final page of the *Computing* menu is the most important page. Shown in

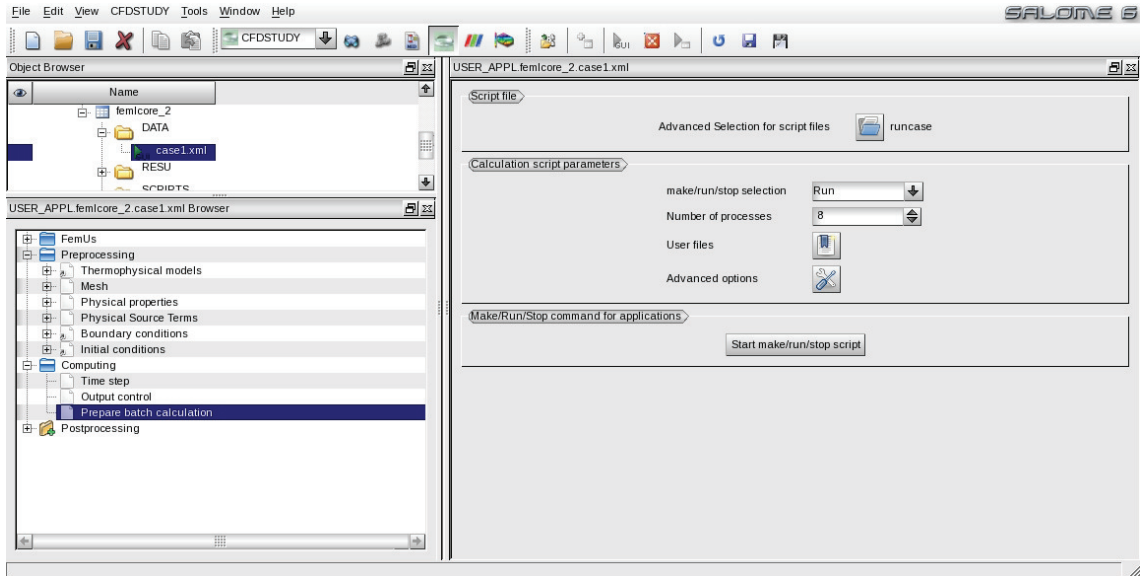


Figure 1.34: *Computing* GUI menu: run control page

Figure 1.34, this page controls the the code compilation and execution. This will be described in the following Section 1.3.

## 1.2.4 *Postprocessing* GUI pages

The *Postprocessing* menu consists of a single page shown in Figure 1.35. This page contains links to open programs that can be useful for postprocessing. The real

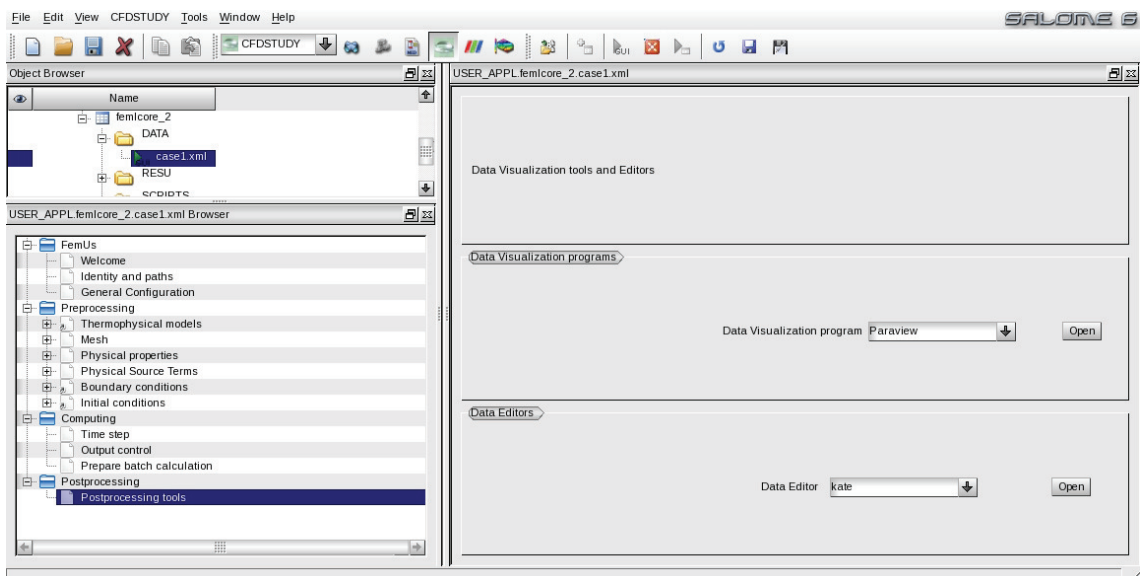


Figure 1.35: Postprocessing menu: applications

 <b>Ricerca Sistema Elettrico</b>	Sigla di identificazione	Rev.	Distrib.	Pag.	di
	NNFISS-LP3-059	0	L	34	106

postprocessing of FEM-LCORE is performed by using the PARAVIEW application which is integrated in the SALOME platform and therefore available on the top menu-bar. The postprocessing tools are divided into two combo-menus: one for launching applications and one for text editors. In order to work each tool must be available on your machine. A brief discussion of the postprocessing PARAVIEW application is postponed to the next Section 1.3 but the interested reader can consult [24].

## 1.3 Executing the FEM-LCORE code

In order to configure and execute the code from the GUI interface one must follow the steps as defined in the GUI tree (from top to bottom). We can divide this process into these steps: coarse mesh generation and initial configuration, parallel and multigrid mesh generation, configuration and execution.

### 1.3.1 Coarse mesh CAD input file and initial configuration

The geometry of the reactor is complex and therefore it can only be generated by a CAD program. With the CAD program we define the coarse mesh that contains all the geometrical details. Over this coarse grid a multigrid system is constructed to define the final fine mesh. The core consists of several assemblies with boxed shapes. In order to impose the heat source correctly, it is required that each assembly is discretized by only hexahedral cells and the assembly surfaces do not cut any mesh cell internally. This requires some care as the cross sections of the assemblies form a staggered pattern in the reactor design. The code can handle a limited number of mesh formats obtained from CAD programs: the **generic** mesh format obtained from GAMBIT (.neu extension) and the **MED** mesh format from SALOME (.med extension). The **generic** mesh GAMBIT format must be generated by using only HEX27 finite elements and the SALOME mesh MED format by using HEX20 finite elements. These restrictions are necessary for a successful execution of the **gencase** program.

If the GAMBIT software is used, the option in the **Solver** menu must be set to **generic**. This is very important since only this option produces a readable input file. The mesh must be generated using the HEX27 option on volume elements, the QUAD9 option on surface elements and the EDGE3 option on linear elements. Finally the mesh must be exported with the **Export --> Mesh** command from the **File** menu.

The second type of mesh supported by FEM-LCORE is the MED format. In this case the mesh can be generated by the SALOME GEOM-MESH modules. SALOME is a free software that provides a generic platform for pre- and post-processing numerical simulations. It is open source, released under the GNU Lesser General Public License and both its source code and executables may be downloaded from its official

website [26]. SALOME is composed of the following main modules: KERNEL (general services), GUI (graphical user interface), GEOM (editing CAD models), MESH (standard meshing) and MED (MED data file management). For mesh generation one must use the GEOM module and its drawing capabilities. The MESH module must use the Hexahedron mesh option to have a final mesh with only Hexahedral HEX8 finite elements. Then the elements must be converted to quadratic elements (from HEX8 to HEX20 elements). The HEX8 elements are not allowed as input of the program and therefore the conversion to HEX20 is necessary. After the mesh is generated one can save the mesh in MED format. The MED data model defines the data structures exchanged by codes. The modeled data concern meshes and the solution fields defined on nodes and elements. MED supports nine element shapes: point, line, triangle, quadrangle, tetrahedron, pyramid, hexahedron, polygon and polyhedron. Each element may have a different number of nodes, depending on whether linear or quadratic interpolation is used. Once the coarse mesh is available

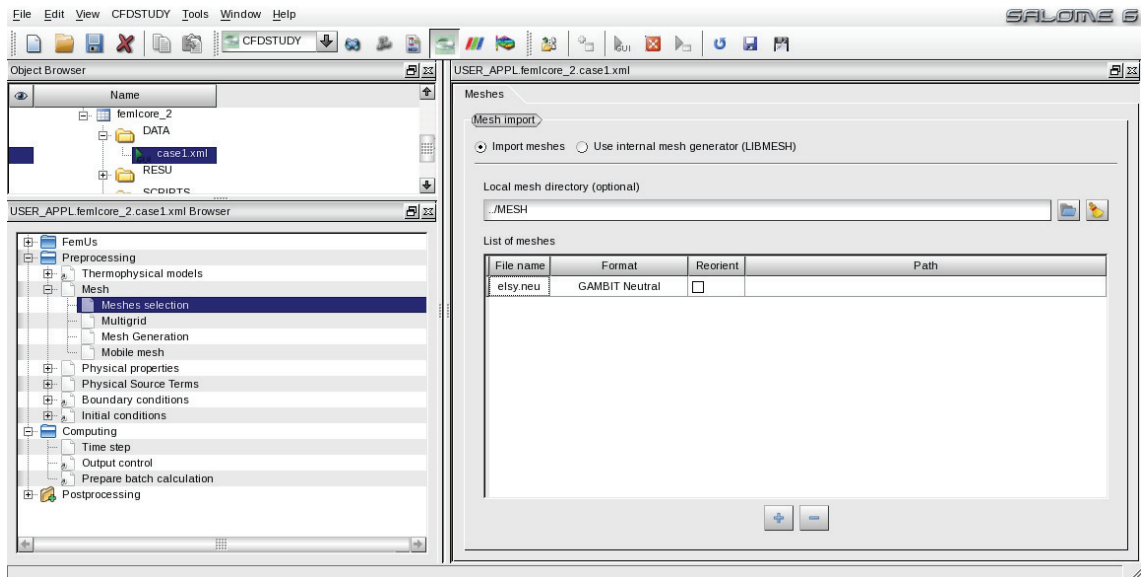


Figure 1.36: Mesh generation page for gencase configuration

we must complete the *Thermophysical models* pages in the GUI tree by introducing the model and the physical data. By using the page in Figure 1.36 and the instruction discussed above we complete the initial configuration stage.

### 1.3.2 The gencase application: mesh and multigrid files

Given the coarse grid, the gencase program generates the multilevel grid used by FEM-LCORE. After reading the grid configuration values contained in the configuration file DATA/parameters.in, gencase creates a mesh with the required number of levels. The data of interest in the configuration file are:

- `libmeshgen`, that indicates whether the LIBMESH internal generator should be used or not;
- `mrefine` that selects the use of the LIBMESH grid refinement function;
- `nolevels` that determines the number of levels of the multigrid solver;
- `Nx`, `Ny` and `Nz` that, in the case of internal generator, set the number of grid subdivisions along the  $x$ ,  $y$  and  $z$  directions ;

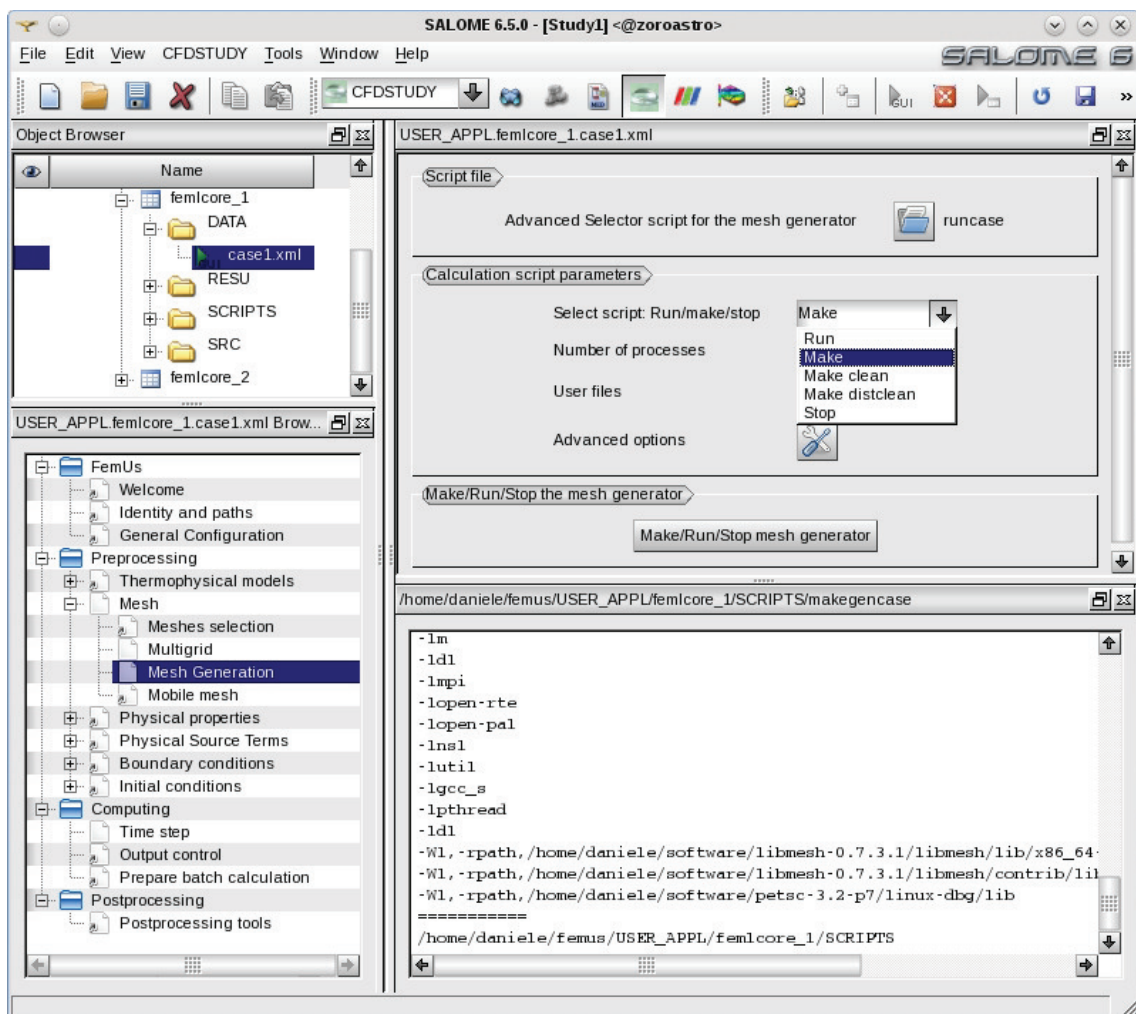


Figure 1.37: *Mesh generation page*

The `gencase` program runs from the *Mesh generation* page shown in Figure 1.37. The combo-menu selects the type of action. We have three options: run, compile (make) and clean (make clean and make distclean perform the same function). The number of processors must be set in the appropriate line. This number should not be changed during the `restart` operation since the `gencase` application does

not run if the restart flag is set. If the FEM approximation or the mesh files are modified then the code must be recompiled and run again. The code is recompiled with the `make` option. As soon as the `make` option is selected the command execution window appears and shows the results of the compilation. If the compilation does not end successfully then an error has occurred or the `make clean` command should be rerun. The `make distclean` option deletes all the output files in all the directories including the mesh refinements and the multigrid operators in the RESU directory.

### 1.3.3 GUI final configuration

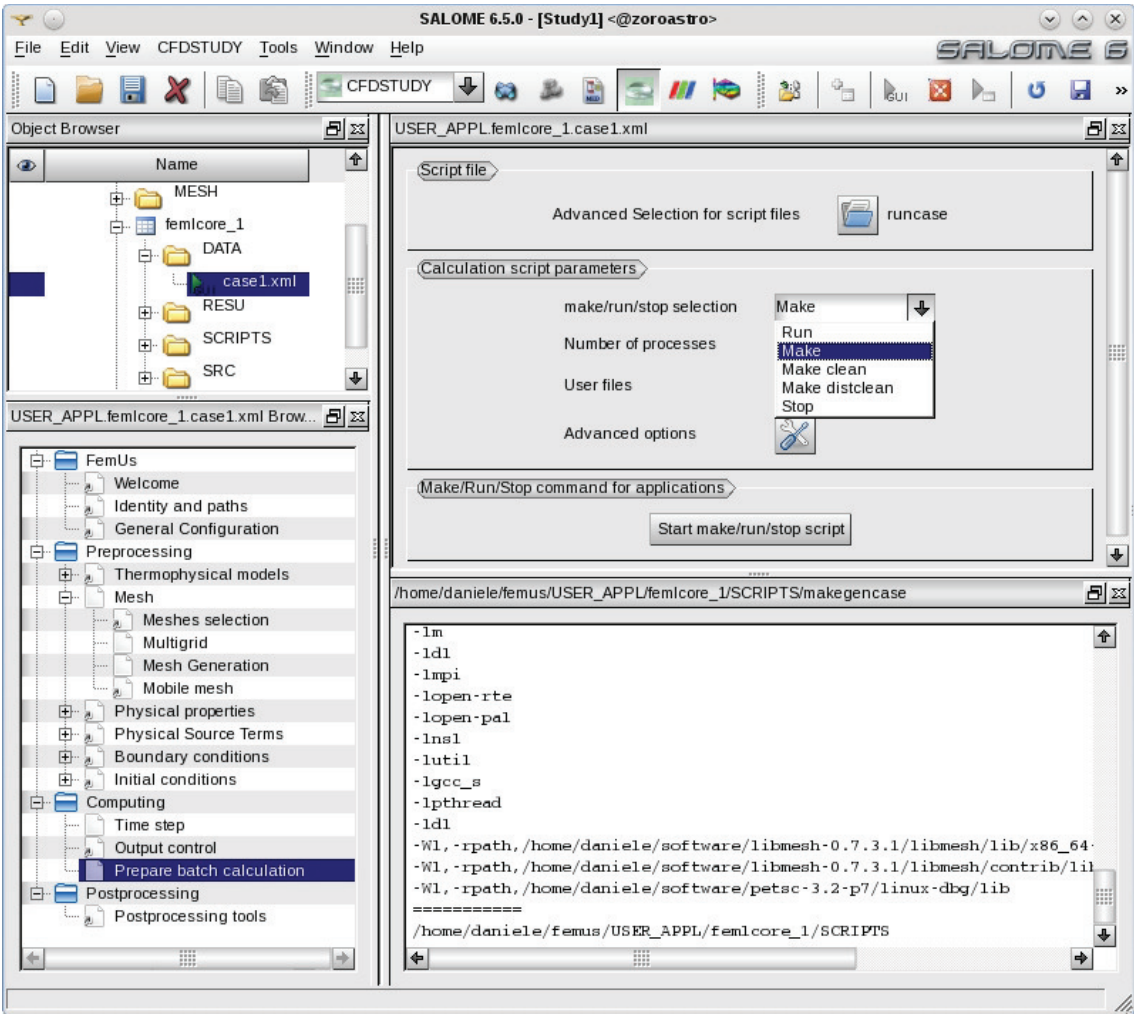


Figure 1.38: Computing GUI menu: Prepare batch calculation page

Once the multilevel and parallel mesh is available we must follow the remaining pages in the GUI tree. Fuel distribution in the core, pressure losses, boundary and initial conditions must be checked. All these files come with examples but for a

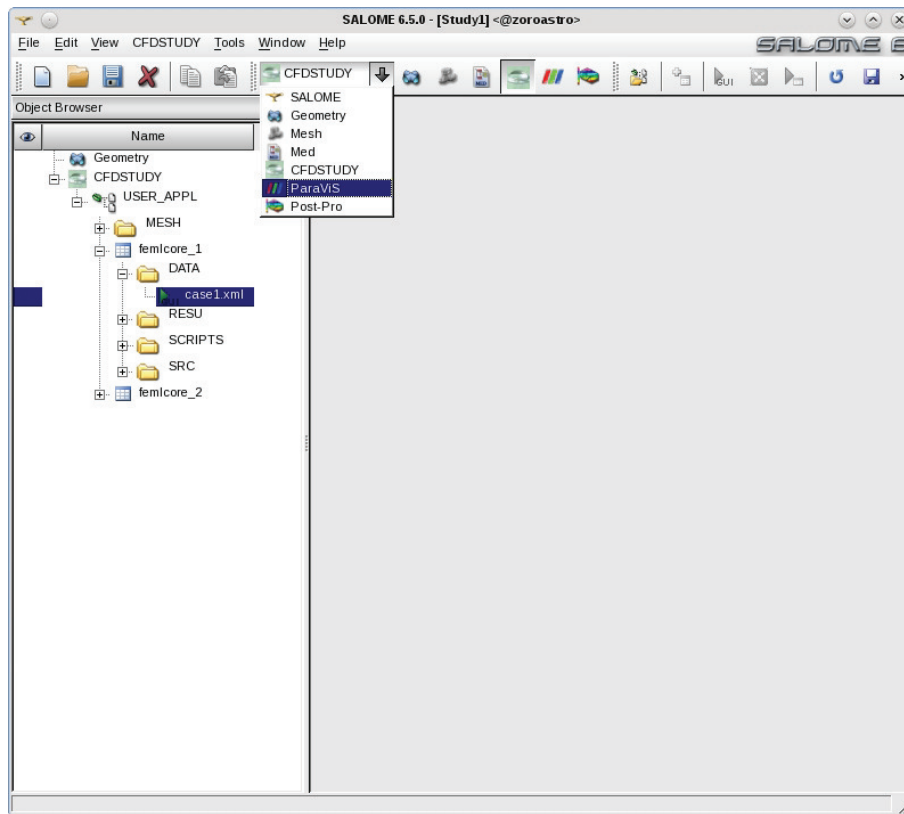


Figure 1.39: PARAVIEW selection in the FEM-LCORE GUI

completely new reactor they must be regenerated. These configuration options were discussed in the previous section and therefore one should be able to complete the final configuration stage and finally open the run page.

### 1.3.4 Running the FEM-LCORE code

In order to run the FEM-LCORE code one must complete all the pages of the GUI tree-menu window. At the bottom of the *Computing* GUI submenu one can find the *Prepare batch calculation* page as seen in Figure 1.38. The combo-menu selects the type of action.

We have three options: run, compile (`make`) and clean (`make clean` and `make distclean`). The code must be recompiled when one modifies something that is not in the `param_files.in` and `parameters.in` files. If class configuration, boundary conditions or initial conditions files are modified the code must be recompiled with the `make` option. As soon as the `make` option is selected the execution command window appears showing the results of the compilation. The `make distclean` option deletes all the files in all the directories including all the files in the `SRC` and `RESU` subdirectories.



### 1.3.5 Postprocessing FEM-LCORE results

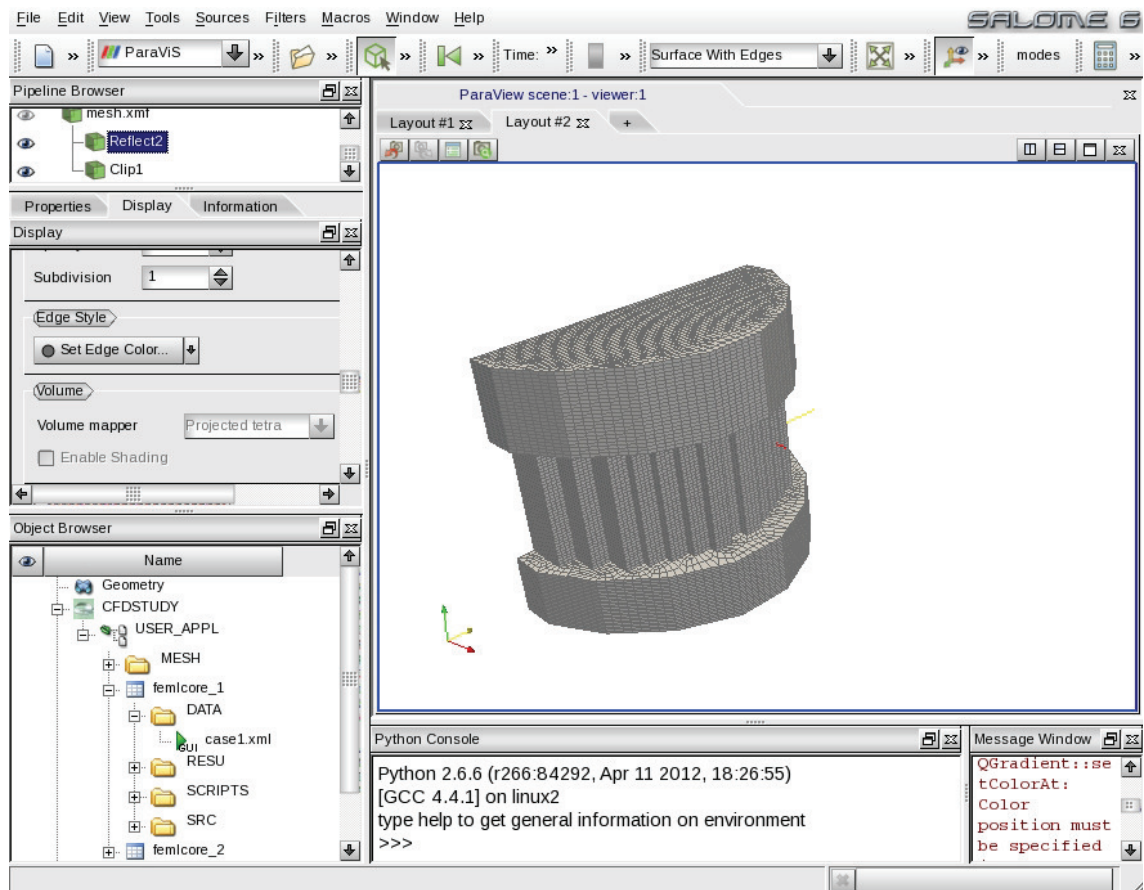


Figure 1.40: PARAVIEW integrated with FEM-LCORE GUI

The post-processing step is dedicated to display the results obtained by the code. All the output files of the program are stored in the **RESU** directory under the path `.../femus/USER_APPL/femlcore_1/`. The writing data formats are XDMF and HDF5. In the HDF5 files the field values are stored in a hierarchical structure and a compressed storage format. The XDMF files are required for reading the data contained in the HDF5 files. The XDMF format can be interpreted by some visualization software such as PARAVIEW.

This application is based on the VTK library that provides services for data visualization [27]. PARAVIEW is designed for viewing data obtained from parallel architectures and distributed or shared memory computers, but it can also be used for serial platforms. Inside the FEM-LCORE GUI under the SALOME platform we can select this postprocessor application from the general menu bar as shown in Figure 1.39. After the selection, the PARAVIEW application appears in the SALOME GUI framework. A screenshot of the PARAVIEW application in this framework can be seen in Figure 1.40.

# Thermo-hydraulic of the FEM-LCORE reactor model

## 2.1 Introduction

**Geometrical model.** The FEM-LCORE code is a 3D finite element code for the simulation of the thermo-hydraulic behavior of liquid metal nuclear reactors based on conservation equations. In this chapter we recall the basic thermo-hydraulic reactor model and its finite element discretization as proposed in [2] and [3]. Let us

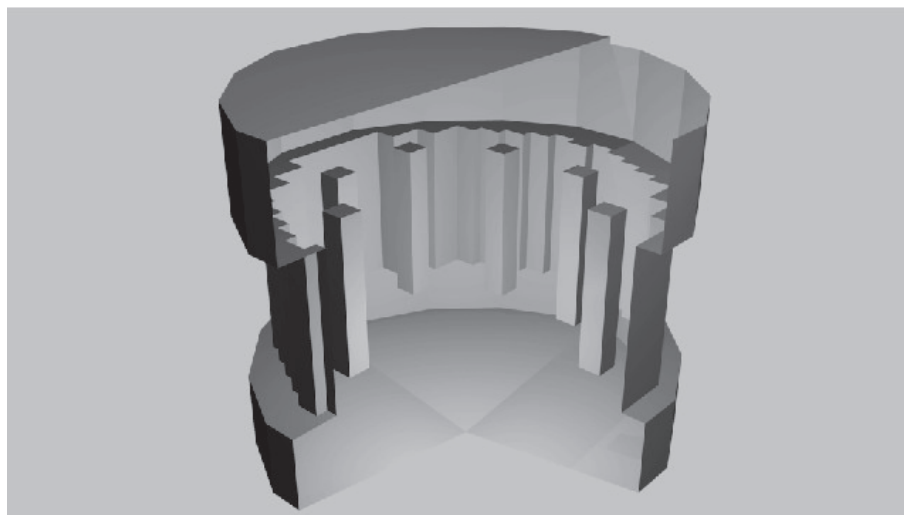


Figure 2.41: Model reactor

consider a model reactor as shown in Figures 2.41-2.42. The reactor model in Figure 2.41 is represented in a vertical more schematic section in Figure 2.42. This reactor model consists of four main volumes: the active (upper) and non-active (lower) core sections and the upper and lower plena. A vertical more schematic section is shown in Figure 2.42. If we set the zero vertical coordinate at the lower core inlet, the core region goes from 0 to  $H_{out}$ . The active core (upper core) where heat is generated ranges between  $H_{in}$  and  $H_{out}$ . Below the core we have the lower plenum with the inlet between  $-H_{lp}$  and 0. The lower plenum has an approximate hemispherical form with the lowest region at  $-H_{bot}$ . Above the core for a total height of  $H_{up}$  there

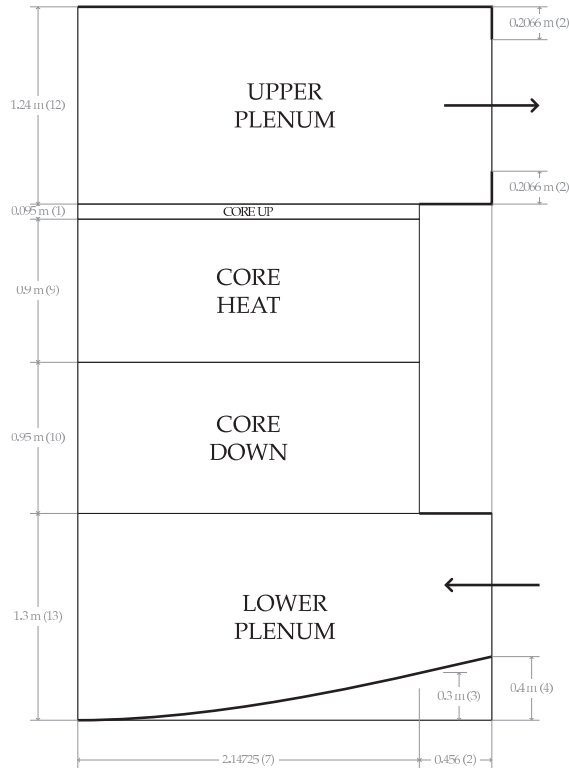


Figure 2.42: Vertical section of the reactor model.

is the upper plenum with the coolant outlet.

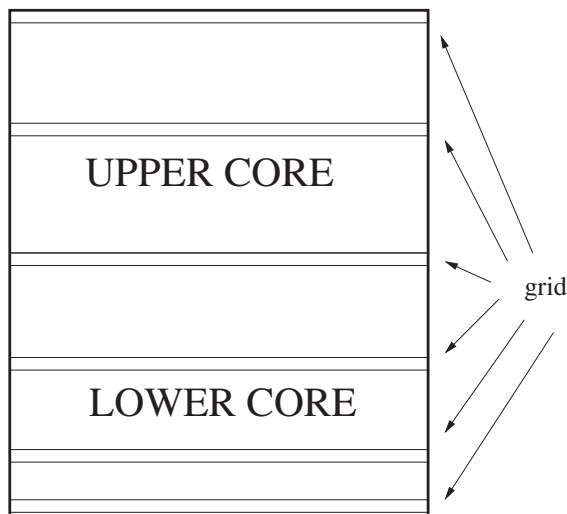


Figure 2.43: Grids on the reactor core model

**Pressure loss distribution model.** The vertical section of the core is divided by  $N_p$  spacer grids as defined in Figure 2.43. Each grid defines an accidental pressure

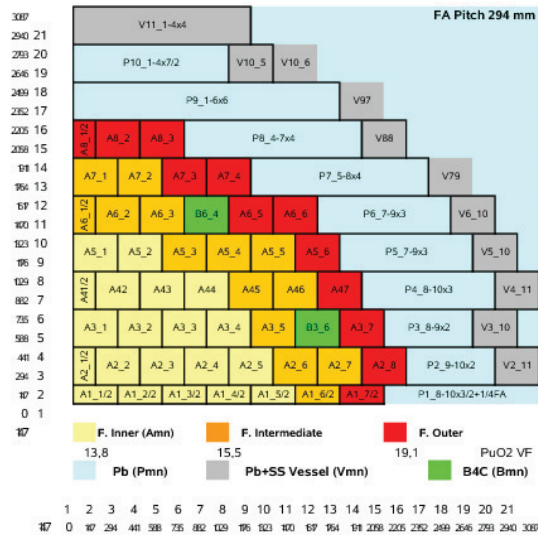


Figure 2.44: Horizontal power generation profile for the model reactor

loss defined by  $0.5\rho|\mathbf{v}|\bar{\beta}(\mathbf{v}, p, \mathbf{x}) \cdot \mathbf{v}$  where  $\bar{\beta}$  is a vector coefficient depending on many factors. This accidental pressure loss is equivalent to the standard channel pressure losses when the motion is truly vertical with no transverse components. In this case the pressure loss vector are scalar factor quantities and may be defined as  $|\bar{\beta}(z_i)| = \beta_{pi}$  for each grid  $i$ . The pressure losses for each grid reduces to  $0.5\rho\beta_{pi}\mathbf{v}^2$  as usual for monodimensional channels.

**Fuel distribution model.** The horizontal quarter section of the core is shown in Figure 2.44. Each fuel assembly consists of a  $n_p \times n_p$  pin lattice. The overall number of assembly positions in the core is  $N_a$ . Eight of these positions are dedicated to house special control rods (see [6]) and therefore the global number of fuel assemblies is  $N_a - 8$ . The transverse core area is approximately circular but not axial symmetric. Therefore, in order to predict the behavior of the reactor, a three-dimensional simulation must be performed.

The model design distributes the fuel assemblies in three radial zones:  $N_{a1}$  fuel assemblies in the inner zone,  $N_{a2}$  fuel assemblies in the intermediate zone and the remaining  $N_{a3}$  fuel assemblies in the outer one. The power distribution factors, i.e. the power of a fuel assembly over the average fuel assembly power, are mapped in Figure 2.44. The maximum power factor is  $p_{f,max}$ , while the minimum is  $p_{f,min}$ .

We label the assemblies as in Figure 2.44; the first row is labeled A1\_i for  $i=1, \dots, 8$ , the second row A2\_i for  $i=1, \dots, 7$  and so on. We remark that the fuel assembly configuration is not based on a Cartesian grid but rather on a staggered

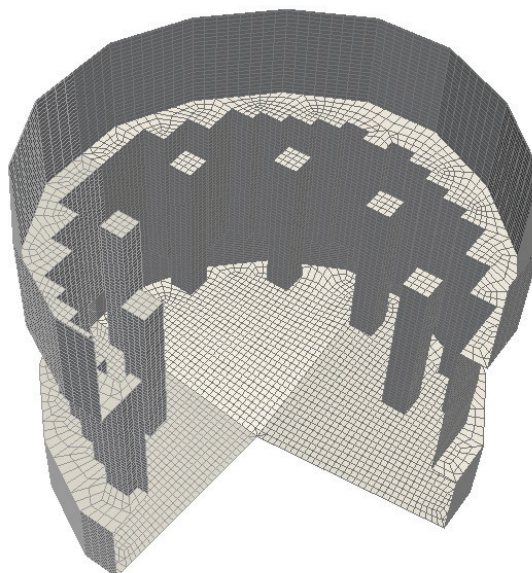


Figure 2.45: Core, lower and upper plenum reactor coarse boundary mesh

grid.

**Mesh model.** The mesh may be generated by using the GAMBIT mesh generator, as shown in Figure 2.45, with geometrical dimensions taken from Figures 2.42 and 2.44. The generation of this mesh is not trivial. In fact care must be taken in such a way that every assembly is exactly discretized by entire cells. In this way all the data attributes related to a given assembly are associated to it exclusively. For details we refer to [2].

**Heat generation model.** The reactor is cooled by lead that enters at the temperature of  $T_0$  K. Since our model describes the reactor at the assembly level the sub-assembly composition is seen as a homogeneous medium. In particular we note that the coolant/assembly ratio is  $r$ . Each assembly has a square section with a side length of  $L_a$  and this completely defines the horizontal core structure. For the vertical geometry we refer to Figure 2.42.

The heat power generation due to fission  $W_0(x, y, z)$  is assumed to be given, where  $W_0(x, y)$  is the two-dimensional distribution. At the middle section of the upper core  $z = (H_{out} - H_{in})/2$  the desired cross-sectional heat power distribution is shown in 2.44. The heat power generation due to fission is a function of the horizontal and vertical plane as one can see in Figures 1.26 and 1.27 in Chapter . We remark the presence of the eight control assemblies inside the core which are used to house special control rods (see [6]) and where no power is generated. In

 <b>Ricerca Sistema Elettrico</b>	Sigla di identificazione	Rev.	Distrib.	Pag.	di
	NNFISS-LP3-059	0	L	44	106

order to obtain the assembly averaged specific  $\dot{q}_v$  and linear  $\dot{q}_l$  heat power, the total core power is to be divided over  $N_a - 8$  assemblies instead of  $N_a$ . We obtain

$$\dot{q}_v = \frac{\dot{Q}}{A}$$

and

$$\dot{q}_l = \frac{\dot{q}_v}{H_{out} - H_{in}}.$$

**Multiscale model.** As we have shown before, the reactor is divided into four regions: the lower plenum, the lower core, the upper core and the upper plenum. Let  $\Omega_c$  be the core region and  $\Omega_{lp}$ ,  $\Omega_{up}$  be the lower and the upper plenum respectively. In the lower and upper plenum we solve the three-dimensional Navier-Stokes and energy system while in the core we use the appropriate model described in next Sections.

The reactor regions of the core and the plena must be connected with appropriate yielding conditions which can be defined by conservation of mass and momentum equations. Since the mass flow rate at the core inlet must match the mass flow rate at the top of the lower plenum and the same holds for the core outlet section, then the  $z$ -component of the velocity field cannot be continuous due to a sudden variation of the cross section, depending on the occupation factor ratio  $r$ . The occupation factor ratio is the ratio between the coolant and the total assembly cross section areas. Since the volume coolant rate is continuous in all the reactor we define a new vector field

$$\mathbf{v}^* = \begin{cases} \mathbf{v} & \text{on } \Omega_{lp} \cup \Omega_{up} \\ \mathbf{v}/r & \text{on } \Omega_c \end{cases} \quad (2.6)$$

which is continuous across the reactor.

## 2.2 Plenum model

In this section we consider the model of the upper and lower plena, as shown in Figure 2.46-2.47. The geometry and the mesh of the upper plenum reactor model that are included in this package are reported in Figure 2.46. In a similar way the boundary and the mesh of the computational three-dimensional lower plenum reactor model can be seen in Figure 2.47. In the regions below and above the core, the coolant flows in an open three-dimensional domain and the coolant state, defined by velocity, pressure and temperature, can be determined by using standard three-dimensional CFD models. In this section, we summarize the equations implemented in the code that are adopted in the modeling of the lower and upper plenum. In these regions the code solves the three-dimensional mass, momentum and energy equations coupled with turbulence models. In particular we may use turbulence models such as  $\kappa - \epsilon$ ,  $\kappa - \omega$  and LES.

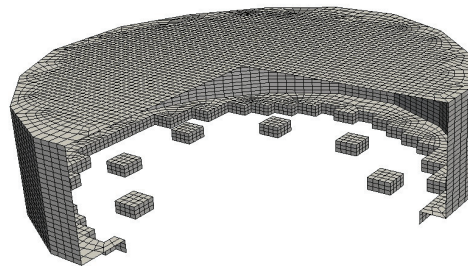
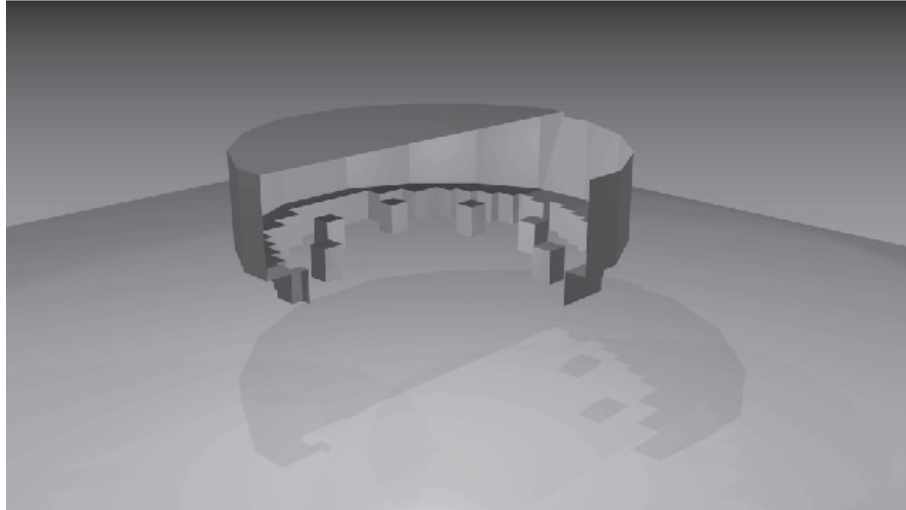


Figure 2.46: Model (top) and mesh (bottom) of the computational three-dimensional upper plenum

### 2.2.1 Navier-Stokes equations

Let  $(\mathbf{v}, p)$  be the state of the fluid flow that enters the plena defined by the velocity, pressure and  $\Omega$  be the domain with boundary  $\Gamma$ . The evolution of the system is described by the solution of the following Navier-Stokes equations

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0, \quad (2.7)$$

$$\frac{\partial \rho \mathbf{v}}{\partial t} + \nabla \cdot (\rho \mathbf{v} \mathbf{v}) = -\nabla p + \nabla \cdot \bar{\tau} + \rho \mathbf{g}. \quad (2.8)$$

The fluid can be considered incompressible, while the density may be assumed only to be slightly variable as a function of temperature, with a given law  $\rho = \rho(T)$ . The

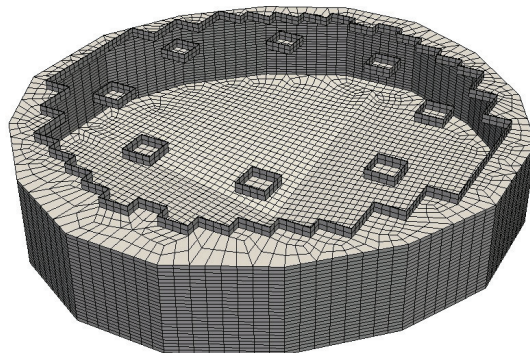
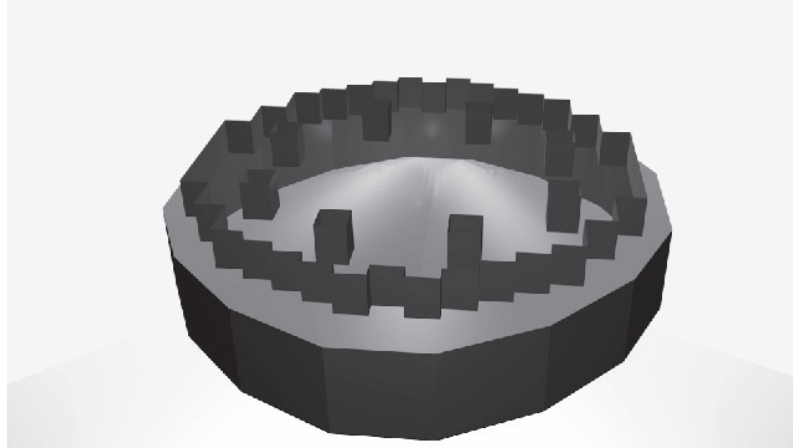


Figure 2.47: Model (top) and mesh (bottom) of the computational three-dimensional lower plenum

viscous tensor  $\bar{\tau}$  is defined by

$$\bar{\tau} = 2\mu\bar{D}(\mathbf{v}), \quad D_{ij}(\mathbf{v}) = \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) \quad (2.9)$$

with  $i, j = x, y, z$ . In a similar way the tensor  $\overline{\mathbf{v}\mathbf{v}}$  is defined as  $\overline{\mathbf{v}\mathbf{v}}_{ij} = v_i v_j$ . The Reynolds  $Re$  number is defined by  $Re = \frac{\rho u D}{\mu}$ . The quantity  $\mathbf{g}$  denotes the gravity acceleration vector. The equations are completed with these data and appropriate boundary conditions.

## 2.2.2 Turbulence model equations

For high Reynolds numbers the numerical solution of the (2.7-2.8) cannot be computed efficiently due to the necessary small grid resolution of the flow. In order



	Sigla di identificazione	Rev.	Distrib.	Pag.	di
	NNFISS-LP3-059	0	L	47	106

to keep the mesh grid sufficiently coarse we need to use approximate models. The main model approximation is to split the velocity vector into a resolved-scale field and a subgrid-scale field. The resolved part of the field represents the average flow, while the subgrid part of the velocity represents the "small scales", whose effect on the resolved field is included through the subgrid-scale model. This decomposition results in

$$\mathbf{v} = \bar{\mathbf{v}} + \bar{\mathbf{v}}', \quad (2.10)$$

where  $\bar{\mathbf{v}}$  is the resolved-scale field and  $\bar{\mathbf{v}}'$  is the subgrid-scale field.

The filtered equations are developed from the incompressible Navier-Stokes equations of motion. By substituting  $\mathbf{v} = \bar{\mathbf{v}} + \bar{\mathbf{v}}'$  and  $p = \bar{p} + p'$  in the decomposition and then filtering the resulting equation we write the equations of motion for the average fields  $\bar{\mathbf{v}}$  and  $\bar{p}$  as

$$\frac{\partial \rho \bar{\mathbf{v}}}{\partial t} + \nabla \cdot (\rho \overline{\mathbf{v}\mathbf{v}}) = -\nabla \bar{p} + \nabla \cdot \bar{\boldsymbol{\tau}} + \rho \bar{\mathbf{g}}. \quad (2.11)$$

We assume that the filtering operation and the differentiation operation commute, which is not generally the case. An extra term  $\partial \tau_{ij} / \partial x_j$  is generated from the non-linear advection terms and therefore we set

$$\tau_{ij} = \overline{v_i v_j} - \overline{v_i} \overline{v_j}. \quad (2.12)$$

Similar equations can be derived for the subgrid-scale field. Subgrid-scale turbulence models usually employ the Boussinesq hypothesis, and seek to calculate the deviatoric part of stress using

$$\tau_{ij} - \frac{1}{3} \tau_{kk} \delta_{ij} = -2\mu_t \bar{S}_{ij}, \quad (2.13)$$

where  $\bar{S}_{ij}$  is the rate-of-strain tensor for the resolved scale and  $\mu_t$  is the subgrid-scale turbulent viscosity. Substituting into the filtered Navier-Stokes equations, we then have

$$\frac{\partial \rho \mathbf{v}}{\partial t} + \nabla \cdot (\rho \overline{\mathbf{v}\mathbf{v}}) = -\nabla p + \nabla \cdot [(\nu + \nu_t) \nabla \mathbf{v}] + \rho \mathbf{g}, \quad (2.14)$$

where we have used the incompressibility constraint to simplify the equation and the pressure is now modified to include the trace term  $\tau_{kk} \delta_{ij} / 3$ . In the rest of this report we drop the average notation  $\bar{\mathbf{v}}$  and  $\bar{p}$  to use the standard notation  $\mathbf{v}$  and  $p$ . With this notation these approximation models result in the Navier-Stokes equations

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0, \quad (2.15)$$

$$\frac{\partial \rho \mathbf{v}}{\partial t} + \nabla \cdot (\rho \overline{\mathbf{v}\mathbf{v}}) = -\nabla p + \nabla \cdot \bar{\boldsymbol{\tau}} + \rho \mathbf{g}. \quad (2.16)$$

identical to the (2.7-2.8) for the average fields  $(\bar{u}, \bar{p}, \bar{T})$  but a modified viscous stress tensor should be considered in the form

$$\bar{\boldsymbol{\tau}} = 2(\mu + \rho \nu_t) \bar{D}(\bar{u}), \quad (2.17)$$

	Sigla di identificazione	Rev.	Distrib.	Pag.	di
	NNFISS-LP3-059	0	L	48	106

The function  $\nu_t$  is called turbulent viscosity and must be computed by solving other transport equations, referred to as *turbulence models*. In the rest of this section we introduce different turbulence model that can be used to compute the turbulent viscosity  $\nu_t$  and therefore close the Navier-Stokes system. In particular in the FEM-LCORE code the following four models are available: LES,  $\kappa$ - $\epsilon$ ,  $\kappa$ - $\omega$ , SST- $\kappa$ - $\omega$ . Some of these model are still basic and need improvement.

### LES turbulence mode

Large eddy simulation (LES) is technique for simulating turbulent flows. A simplified popular model is based on Kolmogorov theory of self-similarity where the large eddies of the flow are dependent on the geometry but the smaller eddies are independent. This feature allows one to explicitly solve for the large eddies in a calculation and implicitly account for the small eddies by using a subgrid-scale model (SGS model).

The most simple and popular LES model is the Smagorinsky LES model. The Smagorinsky model could be summarized as

$$\tau_{ij} - \frac{1}{3}\tau_{kk}\delta_{ij} = -2(C_s\Delta)^2 |\bar{S}| S_{ij}. \quad (2.18)$$

In the Smagorinsky model, the eddy viscosity is modeled by

$$\mu_{sgs} = \rho(C_s\Delta)^2 |\bar{S}|, \quad (2.19)$$

where the filter width is usually taken as  $\Delta = (\text{Volume})^{\frac{1}{3}}$  and

$$\bar{S} = \sqrt{2S_{ij}S_{ij}}. \quad (2.20)$$

The effective viscosity is calculated from  $\mu_{eff} = \mu_{mol} + \mu_{sgs}$ . The Smagorinsky constant usually has the value  $C_s$  ranging from 0.1 to 0.2.

### $\kappa$ - $\epsilon$ turbulence model

In the  $\kappa$ - $\epsilon$  turbulence model the *turbulent viscosity*  $\mu_t$  is modeled as

$$\mu_t = \rho\nu_t\rho C_\mu \frac{\kappa^2}{\epsilon}. \quad (2.21)$$

The turbulent kinetic energy  $\kappa$  and the turbulent dissipation energy  $\epsilon$  satisfy the following equations

$$\frac{\partial\rho\kappa}{\partial t} + \nabla \cdot \rho\mathbf{u}\kappa = \nabla \cdot \left[ \left( \frac{\mu_t}{\sigma_\kappa} + \mu \right) \nabla\kappa \right] - \rho\beta_k^\epsilon\epsilon + \rho\gamma_k^\epsilon S^2 + P_b, \quad (2.22)$$

$$\frac{\partial\rho\epsilon}{\partial t} + \nabla \cdot \rho\mathbf{u}\epsilon = \nabla \cdot \left[ \left( \frac{\mu_t}{\sigma_\epsilon} + \mu \right) \nabla\epsilon \right] - \rho\beta_\epsilon\epsilon^2 + \rho\gamma_\epsilon S^2 + \frac{\epsilon}{\kappa} C_{1\epsilon} C_{3\epsilon} P_b, \quad (2.23)$$

	Sigla di identificazione	Rev.	Distrib.	Pag.	di
	NNFISS-LP3-059	0	L	49	106

where  $\gamma_k^\epsilon$  is the production coefficient of  $k$  and  $P_b$  the buoyancy term. The coefficient  $\beta_k^\epsilon$  in the standard model can be assumed unitary. The  $\gamma_e = C_\mu C_{1\epsilon} \kappa$  is the coefficient for the turbulent dissipation energy source and  $\beta_e = C_{2\epsilon} / \kappa$  the coefficient of the dissipation term for the same equation. We remark that in this formulation  $\gamma_e$  and  $\beta_e$  depend of turbulent kinetic energy  $\kappa$ . The model constants are

$$C_{1\epsilon} = 1.44, \quad C_{2\epsilon} = 1.92, \quad C_\mu = 0.09, \quad \sigma_k = 1.0, \quad \sigma_\epsilon = 1.3. \quad (2.24)$$

The production  $P_k$  of  $k$  is defined as

$$P_k = -\overline{v'_i v'_j} \frac{\partial v_j}{\partial x_i} = \nu_t S^2, \quad (2.25)$$

where  $S$  is the modulus of the mean rate-of-strain tensor, defined as

$$S \equiv \sqrt{2S_{ij}S_{ij}} = \frac{1}{2} \|\nabla \mathbf{v} + \nabla \mathbf{v}^T\|. \quad (2.26)$$

The effect of buoyancy  $P_b$  is given by

$$P_b = \alpha_t \frac{\mu_t}{Pr_t} \mathbf{g} \cdot \nabla T, \quad (2.27)$$

where  $Pr_t$  is the turbulent Prandtl number for energy and  $g_i$  is the component of the gravity vector in the  $i$ -th direction. For the standard and realizable models, the default value of  $Pr_t$  is 0.85. The coefficient  $\alpha_t$  is the thermal expansion coefficient.

### $\kappa$ - $\omega$ turbulence models

Let  $\kappa$  and  $\omega$  be the turbulent kinetic energy and the specific dissipation rate. The turbulent viscosity  $\mu_t$  is defined as  $\mu_t = \rho\kappa/\omega$ . The standard  $\kappa$ - $\omega$  system is defined by [28, 29, 11, 12]

$$\frac{\partial \rho\kappa}{\partial t} + \nabla \cdot \rho \mathbf{u} \kappa = \nabla \cdot \left[ \left( \frac{\mu_t}{\sigma_k} + \mu \right) \nabla \kappa \right] - \rho \beta_k \kappa \omega + \rho \gamma_k S^2, \quad (2.28)$$

$$\frac{\partial \rho\omega}{\partial t} + \nabla \cdot \rho \mathbf{u} \omega = \nabla \cdot \left[ \left( \frac{\mu_t}{\sigma_w} + \mu \right) \nabla \omega \right] - \rho \beta_w \omega^2 + \rho \gamma_w S^2, \quad (2.29)$$

with  $\beta_k = \beta^* = 9/100$ ,  $\gamma_k = \mu_t$ ,  $\beta_w = 5/9$ ,  $\gamma_w = \alpha = 3/40$ ,  $\sigma_k = 2$  and  $\sigma_w = 2$ .

In these equations, the term  $\gamma_k S^2$  represents the generation of turbulent kinetic energy due to mean velocity gradients. The quantity  $\gamma_w S^2$  represents the generation of  $\omega$ . The terms  $\beta_k \kappa \omega$  and  $\beta_w \omega^2$  represent the dissipation of  $\kappa$  and  $\omega$  due to turbulence.

	Sigla di identificazione	Rev.	Distrib.	Pag.	di
	NNFISS-LP3-059	0	L	50	106

### 2.2.3 Energy equation

The evolution of the system is described by the solution  $e = e(T, \mathbf{v})$  of the following equation

$$\frac{\partial \rho e}{\partial t} + \nabla \cdot (\rho \mathbf{v} e) = \Phi + \nabla \cdot \mathbf{q} + \dot{Q}. \quad (2.30)$$

The fluid can be considered incompressible with the density slightly variable as a function of temperature, with a given law  $\rho = \rho(T)$ . The heat flux,  $\mathbf{q}$ , is given by Fourier's law

$$\mathbf{q} = -\lambda \nabla T \equiv -C_p \frac{\mu}{Pr} \nabla T. \quad (2.31)$$

The laminar Prandtl  $Pr$  and the Péclet  $Pe$  numbers are defined by

$$Pr = \frac{C_p \mu}{\lambda}, \quad Pe = Re Pr. \quad (2.32)$$

In this case we set

$$\rho = a + \gamma T, \quad e \equiv C_v T + \frac{\mathbf{v}^2}{2}, \quad (2.33)$$

where  $a, \gamma$  and  $C_v$  are constant. The quantity  $C_v$  is the volume specific heat and  $\lambda$  the heat conductivity. The quantity  $\dot{Q}$  is the volume heat source and  $\Phi$  the dissipative heat term. The equation is completed with these data and appropriate boundary conditions.

### 2.2.4 Turbulence energy equations

For high Reynolds numbers the numerical solution of the (2.30) cannot be computed efficiently and therefore as in the Navier-Stokes case we need an approximate model. Again we decompose the velocity field as

$$\mathbf{v} = \bar{\mathbf{v}} + \bar{\mathbf{v}}', \quad (2.34)$$

where  $\bar{\mathbf{v}}$  is the resolved-scale field and  $\bar{\mathbf{v}}'$  is the subgrid-scale field.

With usual notation these approximation model result in the same equations as (2.30)

$$\frac{\partial \rho C_p T}{\partial t} + \nabla \cdot (\rho \mathbf{v} C_p T) = \Phi + \nabla \cdot \mathbf{q} + \dot{Q}. \quad (2.35)$$

for the average fields ( $T = \bar{T}$ ) and a modified heat flux  $\mathbf{q}$  in the following

$$\mathbf{q} = -C_p \left( \frac{\mu}{Pr} + \frac{\rho \nu_t}{Pr_t} \right) \nabla T. \quad (2.36)$$

The functions  $Pr_t$  are called turbulent viscosity and turbulent Prandtl numbers. The computation of  $Pr_t$  determines the turbulence contribution. We have implemented two models: the constant turbulence Prandtl number modeled and the  $\kappa - \epsilon - \kappa_t - \epsilon_t$  turbulence model. The second model is still in progress and it is available only for development studies [13, 14, 15, 16, 17, 18, 19, 20, 21].

	Sigla di identificazione	Rev.	Distrib.	Pag.	di
	NNFISS-LP3-059	0	L	51	106

## Constant Turbulence Prandtl number model

For many fluids  $Pr_t$  can be assumed to be constant. The values range from 0.85 to 0.95.

## $\kappa$ - $\epsilon$ - $\kappa_t$ - $\epsilon_t$ turbulence model

In this model the turbulence Prandtl number is defined as

$$Pr_t = Pr_{t0} [B(\nu, \nu_t, Pr, R) + 1] C_\alpha \frac{R + C_\gamma}{2R}, \quad (2.37)$$

where  $R = \kappa_t \epsilon / \epsilon_t \kappa$  and  $C_\alpha, C_\gamma$  constant. The function  $B(\nu, \nu_t, Pr, R)$  can take several forms and the interested reader can consult [13, 14] and their references. The equation for the *averaged temperature squared fluctuations* is defined by the following transport equation [13, 14, 15, 20, 21]

$$\frac{\partial \kappa_t}{\partial t} + u_i \frac{\partial \kappa_t}{\partial x_i} = \frac{\partial}{\partial x_i} \left( \alpha + \frac{\alpha_t}{\sigma_{\kappa_t}} \right) \frac{\partial \kappa_t}{\partial x_i} + P_t - \epsilon_t, \quad (2.38)$$

where

$$P_t := -\overline{u'_i T'} \frac{\partial T}{\partial x_i} = \frac{\nu_t}{Pr_t} \left( \frac{\partial T}{\partial x_i} \right)^2. \quad (2.39)$$

In a similar way an equation for  $\epsilon_t$  can be written as [13, 14, 15, 20, 21]

$$\begin{aligned} \frac{\partial \epsilon_t}{\partial t} + u_i \frac{\partial \epsilon_t}{\partial x_i} = \frac{\partial}{\partial x_j} \left[ \left( \alpha + \frac{\alpha_t}{\sigma_{\epsilon_t}} \right) \frac{\partial \epsilon_t}{\partial x_j} \right] + \frac{\epsilon_t}{\kappa_t} \left( C_{p1} P_t - C_{d1} \epsilon_t \right) \\ + \frac{\epsilon_t}{\kappa} \left( C_{d1} P_k - C_{d2} \epsilon \right), \end{aligned} \quad (2.40)$$

where  $P_k$  is defined by

$$P_k := -\overline{u'_i u'_j} \frac{\partial u_i}{\partial x_j} = \nu_t \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \frac{\partial u_i}{\partial x_j}. \quad (2.41)$$

The coefficients can be assumed constant with value  $C_{p1} = 1$ ,  $C_{p2} = 0.60$ ,  $C_{d1} = 1$  and  $C_{d2} = 0.9$  or as model functions. For details see [13].

## 2.3 Reactor core model

In the core region the geometry is so complex and detailed that a direct simulation with the purpose of computing the velocity, pressure and temperature distributions is not possible and an approximation is in order. This approximation is presented in [1] and in this section we briefly recall the equations.

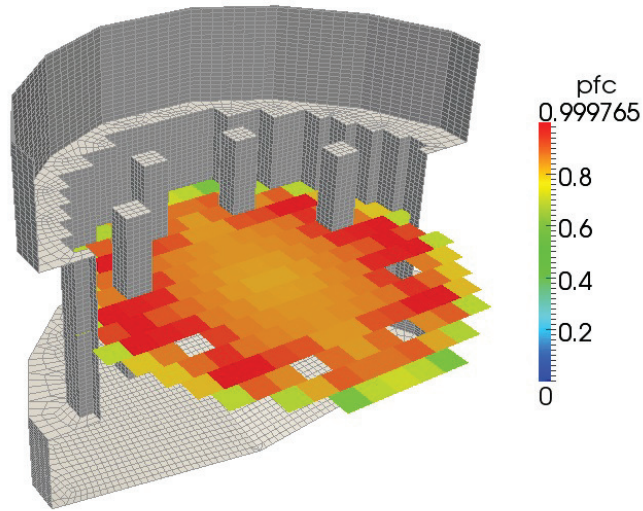


Figure 2.48: Boundary of the computational three-dimensional core reactor model and core power distribution  $W_0(\mathbf{x})$

For the reactor model, with vertical forced motion in working conditions, the state variables  $(\widehat{\mathbf{v}}, \widehat{p}, \widehat{T})$  are the solution of the following system [1]

$$\beta \frac{\partial \widehat{p}}{\partial t} + (\nabla \cdot \rho \widehat{\mathbf{v}}) = 0. \quad (2.42)$$

$$\begin{aligned} \frac{\partial \rho \widehat{\mathbf{v}}}{\partial t} + (\nabla \cdot \rho \widehat{\mathbf{v}} \widehat{\mathbf{v}}) &= -\nabla \widehat{p} + \nabla \cdot [(\widehat{\tau} + \widehat{\tau}^r + \widehat{\tau}^{eff})] + \\ \frac{2\rho \widehat{\mathbf{v}} |\widehat{\mathbf{v}}|}{D_{eq}} \lambda - \rho \mathbf{g} &= 0 \end{aligned} \quad (2.43)$$

$$\begin{aligned} \frac{\partial \rho C_p \widehat{T}}{\partial t} + \nabla \cdot (\rho C_p \widehat{\mathbf{v}} \widehat{T}) &= \\ \Phi_h + \nabla \cdot [(k + k^{eff} + \frac{\mu_t}{Pr_t}) \nabla \widehat{T}] + Q_h + W_0(\mathbf{x}) r(\mathbf{x}). \end{aligned} \quad (2.44)$$

$r(\mathbf{x}) = 1/(1 - \zeta(\mathbf{x}))$  is the coolant occupation ratio. The equations (2.42-2.44) are implemented in the code and must be completed with the appropriate boundary conditions. The functions  $W_0(\mathbf{x})$ ,  $k^{eff}(\mathbf{x})$  and  $\mu^{eff}(\mathbf{x})$  are defined in agreement to the two-level models. The exchange coefficients  $k^{eff}(\mathbf{x})$  and  $\mu^{eff}(\mathbf{x})$  can be computed numerically or experimentally and represent the effective heat exchange due to the subgrid motion inside the assemblies. See [1] for details.

The simulation of the core introduced in the previous section takes into account average quantities over the assemblies and computes average coolant temperatures. When the coolant average temperatures are known then temperature profiles inside the fuel rod and the cladding can be computed by using standard assumptions and standard heat transfer correlations. We remark that in the liquid metal case

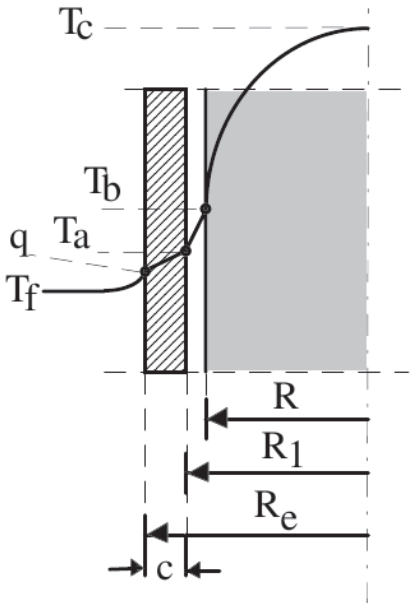


Figure 2.49: Temperature distribution in an assembly channel

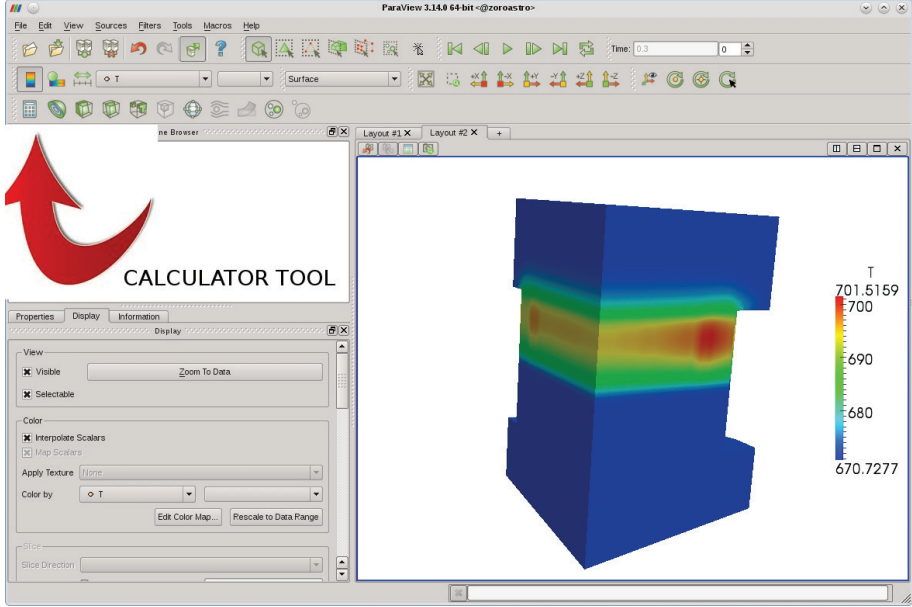


Figure 2.50: Selection of the calculator application in ParaView

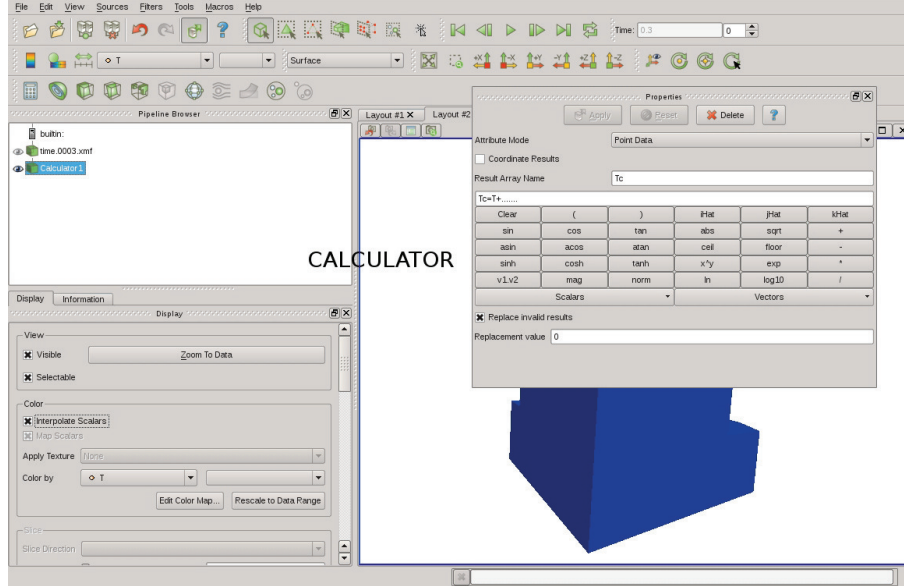


Figure 2.51: Fuel temperature computation by ParaView calculator application

and in three-dimensional configurations the heat exchange coefficient may not be constant along the vertical coordinate and standard heat exchange models cannot be appropriate. The core computations previously proposed are able to define only average assembly temperatures. For temperatures inside the fuel rod we can use the average assembly coolant temperature  $T_f$  and standard analytical formulas. Let  $T_c$  be the temperature on the fuel rod axis and  $T_d$  the cladding temperature. We assume

$$T_c = T_f + \Delta T_1 + \Delta T_2 + \Delta T_3 + \Delta T_4 = T_f + (T_c - T_b) + (T_b - T_a) + (T_a - T_d) + (T_d - T_f). \quad (2.45)$$

Let  $q_l$  be the constant linear heat flux of the fuel rod. We have that

$$\Delta T_1 = (T_c - T_b) = \frac{q_l}{4\pi \bar{K}_f} \quad (2.46)$$

where  $\bar{K}_f = \int_{T_b}^{T_c} K_f dT / (T_c - T_b)$ ,

$$\Delta T_2 = (T_b - T_a) = \frac{q_l R_c}{2\pi R_i} \quad (2.47)$$


where  $R_c$  and  $R_i$  are the fuel and the internal cladding radius,

$$\Delta T_3 = (T_a - T_d) = \frac{q_l s}{2\pi K_c} \quad (2.48)$$

with  $s$  the cladding thickness,  $K_c$  the conductivity of the cladding material and

$$\Delta T_4 = (T_d - T_f) = \frac{q_l}{2\pi R_e h_{df}} \quad (2.49)$$



	Sigla di identificazione	Rev.	Distrib.	Pag.	di
	NNFISS-LP3-059	0	L	55	106

with  $R_e$  the external cladding radius. The physical quantities  $K_f$  and  $K_c$  are well known but  $h_{df}$  must be determined. The heat transfer coefficient  $h_{df}$  is usually defined through the Nusselt number as

$$h_{df} = \frac{Nu k}{D_e q} = \frac{q''}{(T_d - T_f)}. \quad (2.50)$$

Close to the channel inlet the ratio  $h_{df}$  between the wall heat flux and  $T_d - T_f$  may not be constant and a care should be taken in the evaluation of these temperature. With this in mind we compute the temperature jump  $\Delta T_1$ ,  $\Delta T_2$ ,  $\Delta T_3$  with data from literature and use the experimental and computational values of  $h_{df}$  for  $\Delta T_4$ . The fuel temperature can be computed directly by using ParaView application. The calculator ParaView application can be chosen as shown in Figure 2.50. The fuel temperature can be obtained in any point of the fuel by using (2.45) where  $T_f$  is the temperature computed by the FEM-LCORE application and the ParaView calculator application as illustrated in Figure 2.51.

# Chapter 3

## Numerical simulations

### 3.1 Introduction

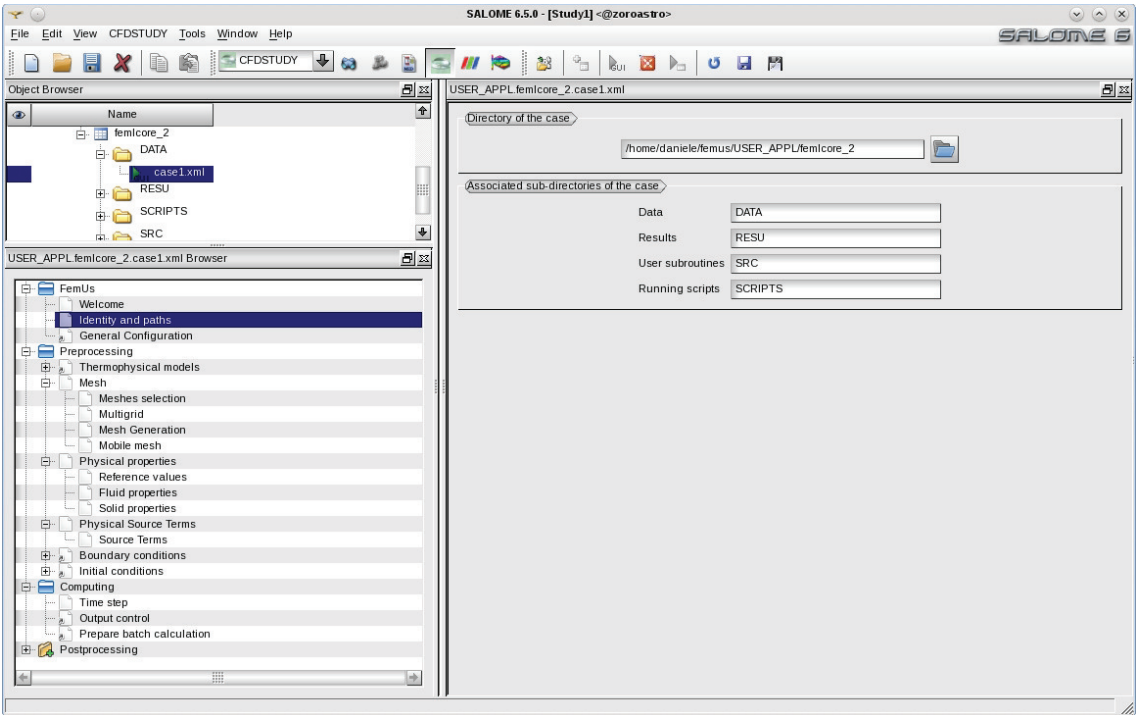


Figure 3.1: FEM-LCORE GUI menu: working directory page

In this Chapter we present some numerical results concerning the parallel version of the FEM-LCORE code with GUI integrated in the SALOME platform. By following the tree-menu from the top to the bottom of the FEM-LCORE GUI, one can configure the problem as described in Chapter 1. The FEM-LCORE tree-menu is shown on the left of Figure 3.1.

Before defining the input configuration file, one must define the geometry. Let us

consider the active (upper) and non-active (lower) core sections and the upper and lower plena, as shown in Figure 2.42. As already discussed in Chapter 2 if we set the zero vertical coordinate at the lower core inlet, the core region goes from 0 m to  $H_{out} = 1.85m$ . The active core (upper core) where heat is generated ranges between  $H_{in} = 0.95$  m and  $H_{out} = 1.85$  m. Below the core we have the lower plenum with the inlet between  $-H_{lp} = -0.9m$  and  $0m$ . The lower plenum has an approximate hemispherical form with the lowest region at  $-H_{bot} = -1.2m$ . Above the core for a total height of  $1.2 = H_{up}m$  there is the upper plenum with the coolant outlet.

The horizontal quarter section of the core is shown in Figure 2.44. Each fuel assembly consists of a  $n_p \times n_p = 21 \times 21$  pin lattice. The overall number of assembly positions in the core is 170. Eight of these positions are dedicated to house special control rods (see [6]) and therefore the global number of fuel assemblies is 162. The transverse core area is approximately circular but not axial symmetric. However, we can argue from Figure 2.44 that two symmetry planes passing through the reactor axis can be identified so that only a quarter domain has to be taken into account for the simulations.

The model design distributes the fuel assemblies in three radial zones:  $N_{a1} = 56$  fuel assemblies in the inner zone,  $N_{a2} = 62$  fuel assemblies in the intermediate zone and the remaining  $N_{a2} = 44$  fuel assemblies in the outer one. The power distribution factors, i.e. the power of a fuel assembly over the average fuel assembly power, are mapped in Figure 2.44. The maximum power factor is 1.17, while the minimum is 0.74.

In Figure 2.45 the boundary reactor mesh is shown. The mesh is generated by using the GAMBIT mesh generator with geometrical dimensions taken from Figures 2.42 and 2.44. The generation of this mesh is not trivial, for details we refer to [2]. The reactor is cooled by lead that enters at the temperature of 673.15 K. In Table

properties	value
Density $\rho$	$(11367 - 1.1944 \times 673.15) = 10562$
Viscosity $\mu$	0.0022
Thermal conductivity $\kappa$	$15.8 + 108 \times 10^{-4} (673.15 - 600.4) = 16.58$
Heat capacity $C_p$	147.3

Table 3.1: Lead properties at T=400° C

3.1 we report the lead physical properties at the inlet temperature. Since our model describes the reactor at the assembly level the sub-assembly composition is seen as a homogeneous medium. Data about the assembly geometry are reported in Table 3.2. In particular we note that the coolant/assembly ratio is 0.548. Each assembly has a square section with a side length of  $L = 0.294$  m, as shown in Table 3.3 and this completely defines the horizontal core structure. For the vertical geometry we refer to Figure 2.42. The heat power generation due to fission  $W_0(x, y, z)$  is assumed to be given, where  $W_0(x, y)$  is the two-dimensional distribution. At the middle section of the upper core  $z = (H_{out} - H_{in})/2$  the desired cross-sectional heat

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

Table 3.2: Coolant assembly area ratio data

	value
Mass flow rate $\dot{m}$	$124539 K g/s$
Heat Power $\dot{Q}_{tot}$	$1482.235 MW$
Number of Assemblies	170
Assembly length L	0.294m
Channel Equivalent Diameter $D_{eq}$	0.0129

Table 3.3: Core characteristic values at working temperature

power distribution is shown in 2.44. The heat power generation due to fission is a function of the horizontal and vertical plane as one can see in Figures 1.26 and 1.27 in Chapter 2. We remark the presence of the eight control assemblies inside the core which are used to house special control rods (see [6]) and where no power is generated. In order to obtain the assembly averaged specific  $\dot{q}_v$  and linear  $\dot{q}_l$  heat power, the total core power is to be divided over 162 assemblies instead of 170. We obtain

$$\dot{q}_v = \frac{\dot{Q}}{A} = \frac{1.482 \times 10^9}{\times 0.294 \times 0.294 \times 162} = 1.0584 \times 10^8 \frac{W}{m^2}$$

and

$$\dot{q}_l = \frac{\dot{q}_v}{H_{out} - H_{in}} = \frac{1.0086 \times 10^8}{0.9} = 1.176 \times 10^8 \frac{W}{m}$$

In this chapter some simple examples are proposed to test the new version which examines different physical cases concerning fuel distribution, pressure losses and assembly blockage. Moreover, we consider computer-related issues by analyzing the parallel performance with CPU and GPU.

## 3.2 Fuel distribution tests

### 3.2.1 Test 1: uniform fuel distribution

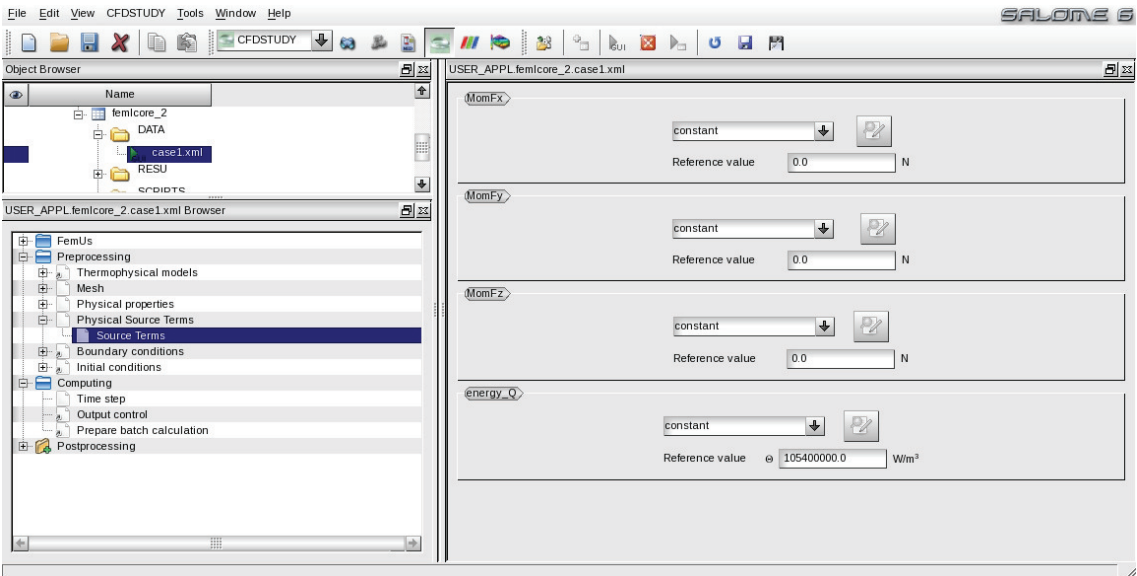


Figure 3.2: GUI Preprocessing menu: *Source terms* page

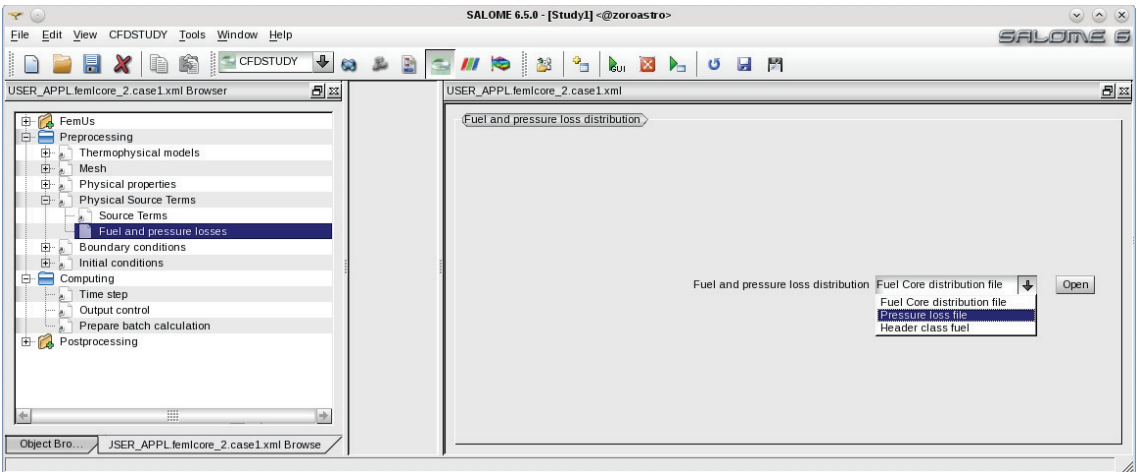


Figure 3.3: GUI Preprocessing menu: *Fuel and pressure losses* page

In order to set a uniform fuel distribution through the FEM-LCORE GUI we follow the steps as described in Chapter 1 for modifying the required configuration files. We first use the *Source terms* page and then the *Fuel and pressure losses* page. We open the page for the source of momentum  $\dot{q}_v$  and energy as in Figure 3.2. In this uniform distribution case the heat source  $\dot{q}_v$  is set to  $1.0584 \times 10^8 \text{ W/m}^2$  which is

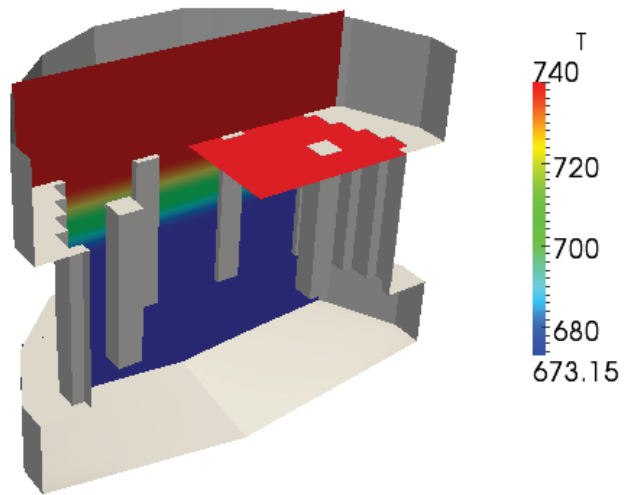


Figure 3.4: Test 1. Temperature field in the reactor

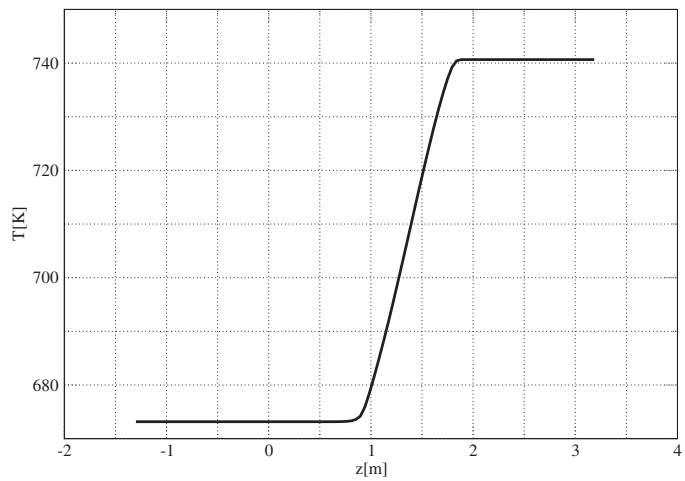


Figure 3.5: Test 1. Temperature along a  $z$ -line at  $x = 0$ . and  $y = 0$ .

the average value for the reactor. The space configuration of the heat source is taken into account by using the peak factors which can be controlled in the *Fuel and pressure losses* page shown in Figure 3.3. This page consists of a combo-menu

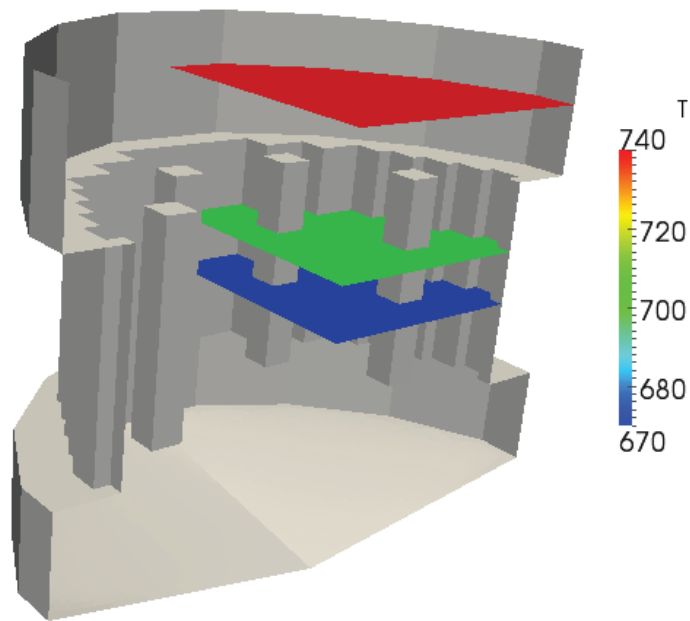


Figure 3.6: Test 1. Temperature field at different heights

where the fuel and pressure loss distributions in the core must be defined by opening and modifying the `SRC/ReacData.C` file.

First we set the following parameters: number of vertical mesh subdivisions in the core section  $NZC=9$ , half fuel assembly cross-sectional side  $HR=0.147$ , core inlet height  $HIN=0.95$ , core outlet height  $HOUT=1.85$ .

In order to specify the type of assembly for each assembly position we appropriately fill the `mat_zone` matrix, in which every assembly in the quarter of reactor is denoted by two indices, both ranging from 0 to 7. For this case we denote the fuel area as a single zone by defining the `INTER` variable, then we indicate the zone with no fuel and the reflector area with `CTRLR` and `DUMMY` respectively:

```
#define INTER (0) // zone 2
#define CTRLR (1) // zone 3
#define DUMMY (2) // zone 4
```

```
const int ReactData::mat_zone[8][8]={
  {INTER,INTER,INTER,INTER,INTER,INTER,INTER,DUMMY},//A1_1-A1_7
  {INTER,INTER,INTER,INTER,INTER,INTER,INTER,INTER},//A2_1-A2_8
  {INTER,INTER,INTER,INTER,CTRLR,INTER,INTER,DUMMY},//A3_1-A3_7
  {INTER,INTER,INTER,INTER,INTER,INTER,INTER,DUMMY},//A4_1-A4_7
  {INTER,INTER,INTER,INTER,INTER,INTER,DUMMY,DUMMY},//A5_1-A5_6
  {INTER,INTER,CTRLR,INTER,INTER,INTER,DUMMY,DUMMY},//A6_1-A6_6
```

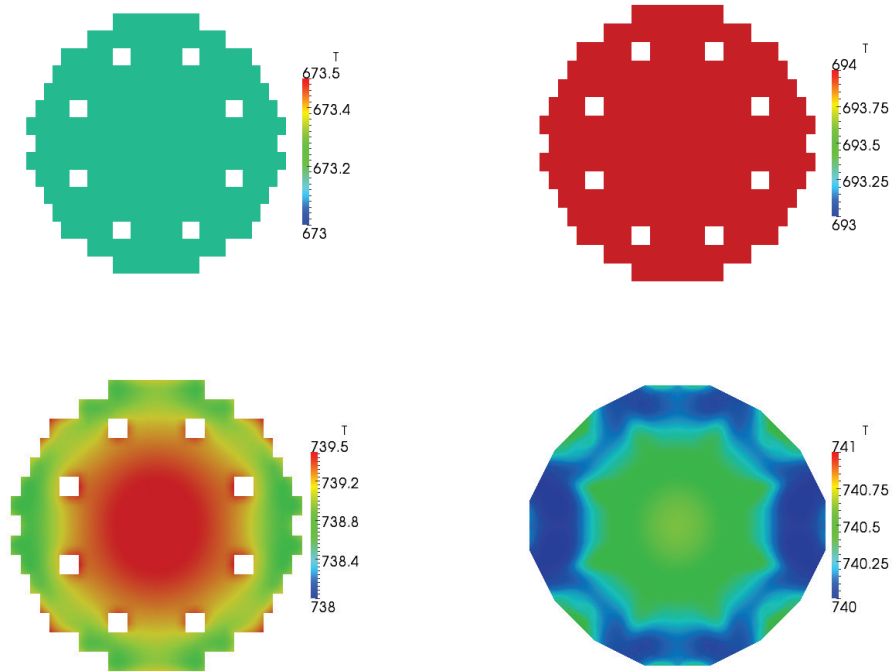


Figure 3.7: Test 1. Temperature distributions over the different plane sections  $z = 0.6$  (lower core),  $z = 1.2$  (upper core inlet),  $z = 1.8$  (upper core outlet),  $z = 2.4$  (upper plenum)

```
{INTER, INTER, INTER, INTER, DUMMY, DUMMY, DUMMY, DUMMY}, //A7_1-A7_4
{INTER, INTER, INTER, DUMMY, DUMMY, DUMMY, DUMMY, DUMMY} //A8_1-A8_3
};
```

The constant horizontal power distribution is implemented in the `mat_pf` matrix by setting to 1 all the peak factors for the fuel assemblies:

```
const double ReactData::mat_pf[8][8]={
{1.,1.,1.,1.,1.,1.,1.,0.}, //A1_1-A1_7
{1.,1.,1.,1.,1.,1.,1.,1.}, //A2_1-A2_8
{1.,1.,1.,1.,0.,1.,1.,0.}, //A3_1-A3_7
{1.,1.,1.,1.,1.,1.,1.,0.}, //A4_1-A4_7
{1.,1.,1.,1.,1.,1.,0.,0.}, //A5_1-A5_6
{1.,1.,0.,1.,1.,1.,0.,0.}, //A6_1-A6_6
{1.,1.,1.,1.,0.,0.,0.,0.}, //A7_1-A7_4
{1.,1.,1.,0.,0.,0.,0.,0.} //A8_1-A8_3
};
```

The vertical power factor is also assumed to have a uniform distribution. This profile is reported in the `xpf` array as



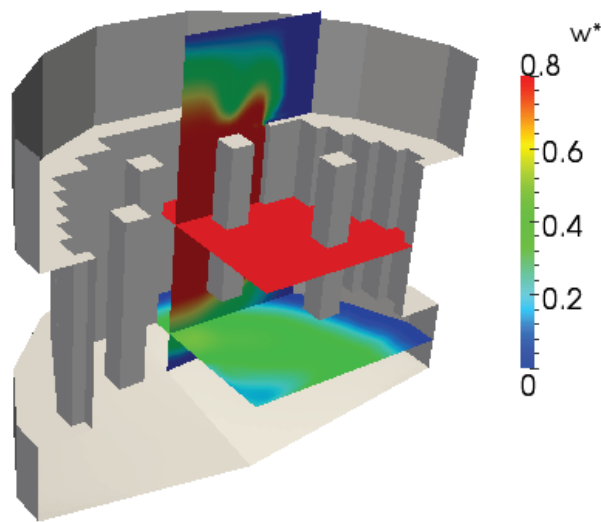


Figure 3.8: Test 1. Reduced velocity field  $w^*$  in the reactor

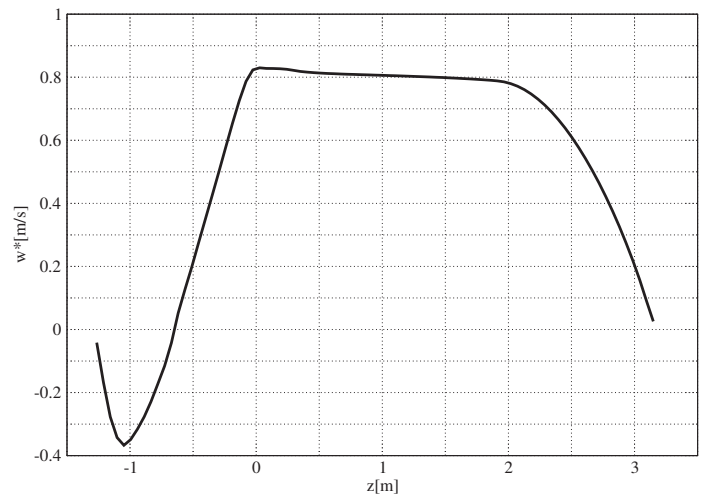


Figure 3.9: Test 1. Reduced z-component  $w^*$  of the velocity field along the core axis

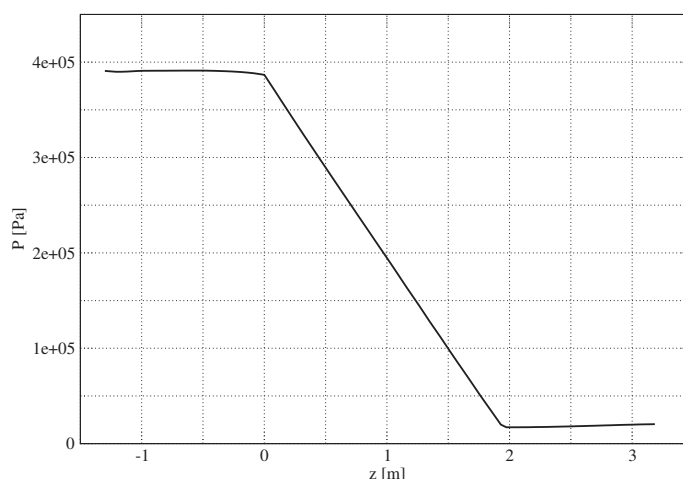


Figure 3.10: Test 1. Pressure  $p$  distribution along the reactor axis

```
const double ReactData::axpf[10][1]={
    {1.,}, {1.,}, {1. }, {1. }, {1. },
    {1. }, {1. }, {1. }, {1. }, {1. } };
```

In a similar way the pressure drop distribution must be set to zero in the `axlpf` matrix and in the `axllf` vector. Once the fuel distribution and the pressure losses are configured, one can compile and run the FEM-LCORE code in the usual way.

In Figures 3.4-3.7 the temperature distribution  $T$  over various sections and lines of the reactor is reported. In Figure 3.4 an overview of the temperature distribution  $T$  is given over a vertical section that includes the various zones of the reactor: lower plenum, lower core, upper core and upper plenum. In Figure 3.5 one can see the temperature field along a vertical line in the reactor. The temperature remains constant in the lower plenum and lower core and increases in the upper core to become again constant in the upper plenum. Considering plane horizontal sections at different heights the temperature is rather smooth due to the uniform heat source distribution. This can be seen in Figure 3.6 where the differences in temperature cannot be appreciated on each horizontal section, due to the large variations in the  $z$ -direction. This can also be observed in Figure 3.7, where we report, from top left to bottom right, the temperature distributions at the horizontal sections  $z = 0.6, 1.2, 1.8$  and  $2.4$  m, corresponding to lower core, upper core inlet, upper core outlet and upper plenum. Again we remark that the temperature differences inside each horizontal plane of the reactor are really small.

The velocity and pressure fields are shown in Figures 3.8-3.11. In Figure 3.8 one can see the overall distribution of the  $z$ -component  $w^*$  as defined in (2.6). In Figure 3.9 the velocity profile along the  $z$ -line at the center of the reactor is shown

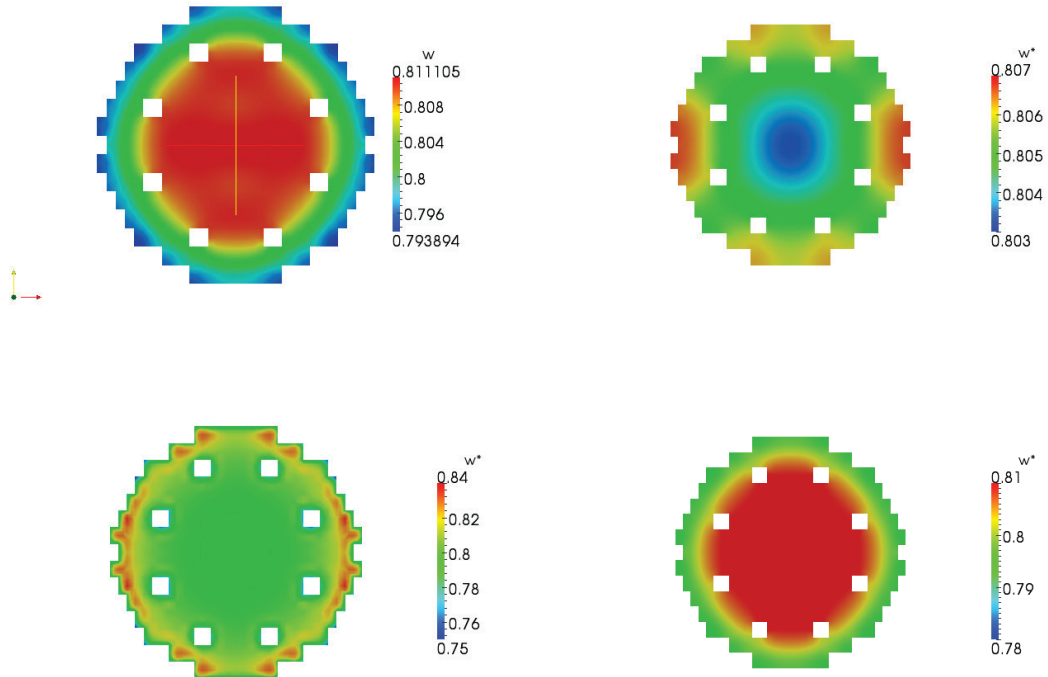


Figure 3.11: Test 1. Reduced  $z$ -component  $w^*$  of the velocity field over the plane sections  $z = 0.6$  (lower core),  $z = 1.2$  (upper core inlet),  $z = 1.8$  (upper core outlet),  $z = 2.4$  (upper plenum)

with solid line. We remark that in the core region the velocity can be considered approximately constant and the hypothesis of monodimensional motion can be applied. Still considering the reactor axis, the pressure profile is plotted in Figure 3.10. The velocity on different plane sections of the reactor is shown in Figure 3.11. Here we can see, from top left to bottom right, the velocity distributions at the plane sections  $z = 0.6, 1.2, 1.8$  and  $2.4$  m.

### 3.2.2 Test 2: non-uniform fuel distribution

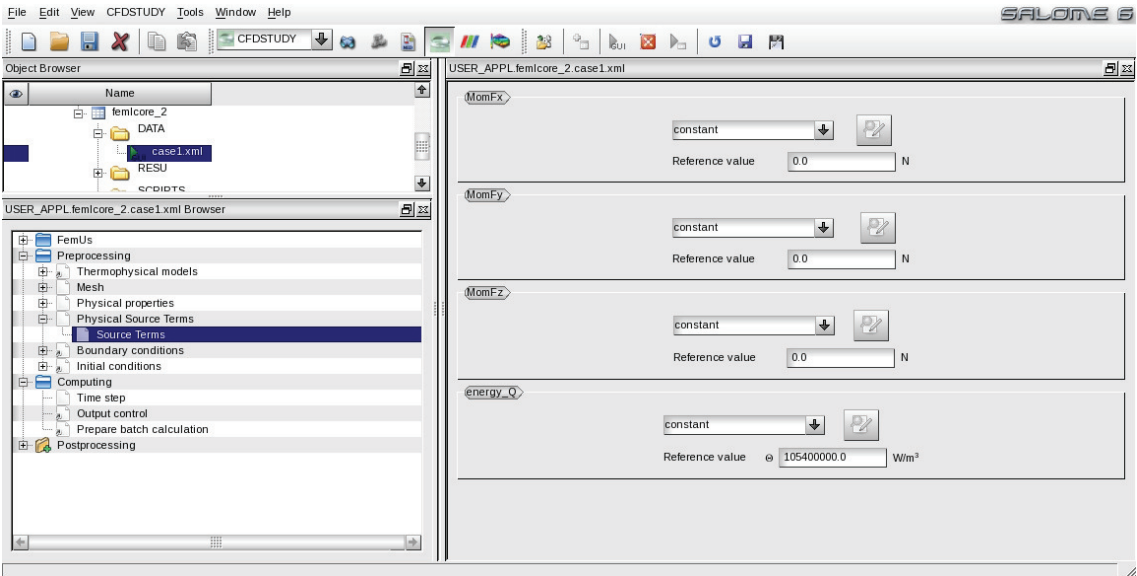


Figure 3.12: GUI Preprocessing menu: setting the average reactor power

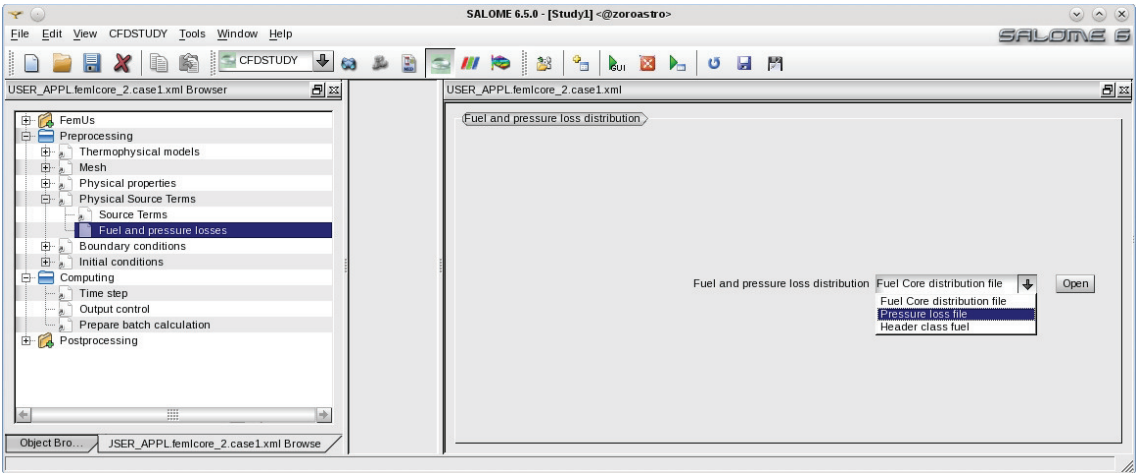


Figure 3.13: GUI Preprocessing menu: setting a non uniform fuel distribution

In order to set a non-uniform fuel distribution we proceed in a similar manner as the previous test. The only difference is that the space configuration of the heat source is implemented by using non-uniform peak factors. We select the *Fuel and pressure losses* page shown in Figure 3.13 and modify the `SRC/ReactData.h` and `SRC/ReactData.C` files. In order to load the desired power distribution as shown in Figures 3.14 and 3.15, we set the following parameters: NLEV (number of multigrid levels), NZC (number of vertical mesh subdivisions in the core section), HR

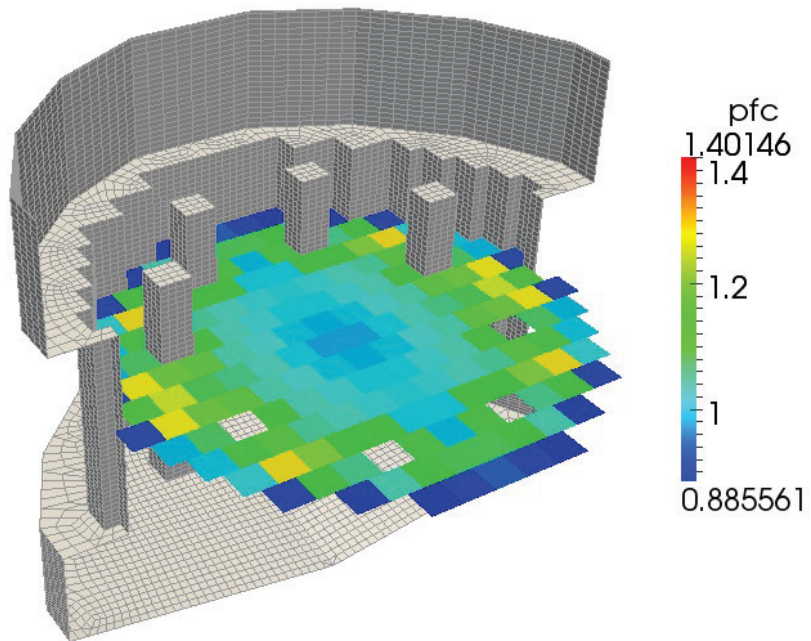


Figure 3.14: Test 2. Horizontal core power distribution

(half fuel assembly cross-sectional side), HIN (core inlet height), HOUT (core outlet height). These parameters can be modified by using the combo-menu option *Header fuel class file*, which opens the SRC/ReactData.h header file. A setting can be as follows:

```
#define NLEV 1      // number of levels
#define NZC 9      // number of vertical core subdivisions
#define HR  0.147  // half fuel assembly side
#define HIN  0.95  // heated core beginning
#define HOUT 1.85  // heated core end
```

In order to specify the type of assembly for each assembly position we appropriately fill the `mat_zone` matrix, in which every assembly in the quarter of reactor is denoted by two indices, both ranging from 0 to 7. In this case the fuel area is divided into three zones: INNER, INTER and OUTER. The three different zones correspond to different enrichment of the fuel rods. The heat power is computed by the knowledge of the neutron flux distribution. Also, we indicate the zone with no fuel and the reflector area with CTRLR and DUMMY respectively:

```
#define INNER (0)  // zone 0
```

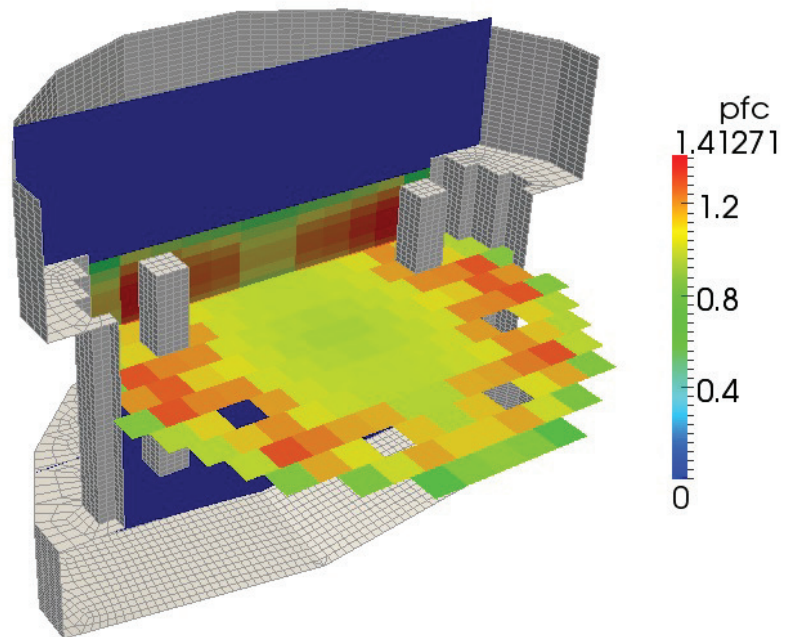


Figure 3.15: Test 2. Vertical and horizontal core power distributions

```
#define INTER (1) // zone 1
#define OUTER (2) // zone 2
#define CTRLR (3) // zone 3
#define DUMMY (4) // zone 4

const int ReactData::mat_zone[8][8]={
  {INNER,INNER,INNER,INNER,INNER,INTER,OUTER,DUMMY},//A1_1-A1_7
  {INNER,INNER,INNER,INNER,INTER,INTER,OUTER,OUTER},//A2_1-A2_8
  {INNER,INNER,INNER,INTER,CTRLR,INTER,OUTER,DUMMY},//A3_1-A3_7
  {INNER,INNER,INNER,INNER,INTER,INTER,OUTER,DUMMY},//A4_1-A4_7
  {INNER,INTER,INTER,INTER,OUTER,OUTER,DUMMY,DUMMY},//A5_1-A5_6
  {INNER,INTER,CTRLR,INTER,OUTER,OUTER,DUMMY,DUMMY},//A6_1-A6_6
  {INTER,OUTER,OUTER,OUTER,DUMMY,DUMMY,DUMMY,DUMMY},//A7_1-A7_4
  {OUTER,OUTER,OUTER,DUMMY,DUMMY,DUMMY,DUMMY,DUMMY} //A8_1-A8_3
};
```

The horizontal power factor distribution is reported in the `mat_pfc` matrix as follows:

```
#define CTL_R 0.
```

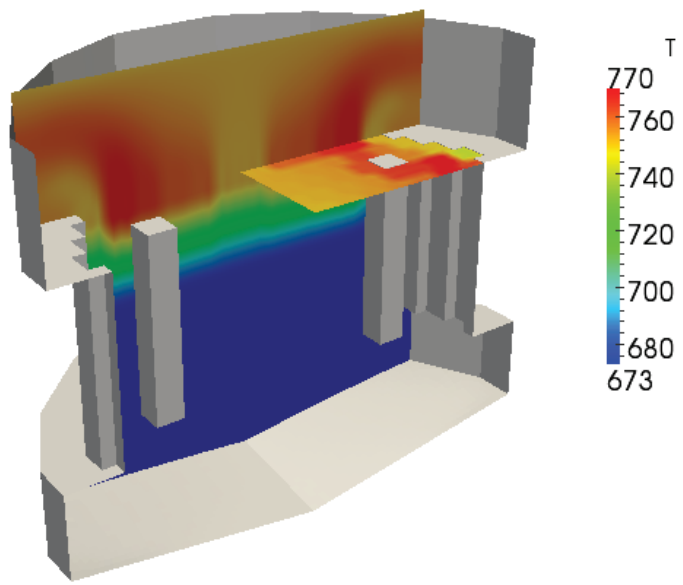


Figure 3.16: Test 2. Temperature field

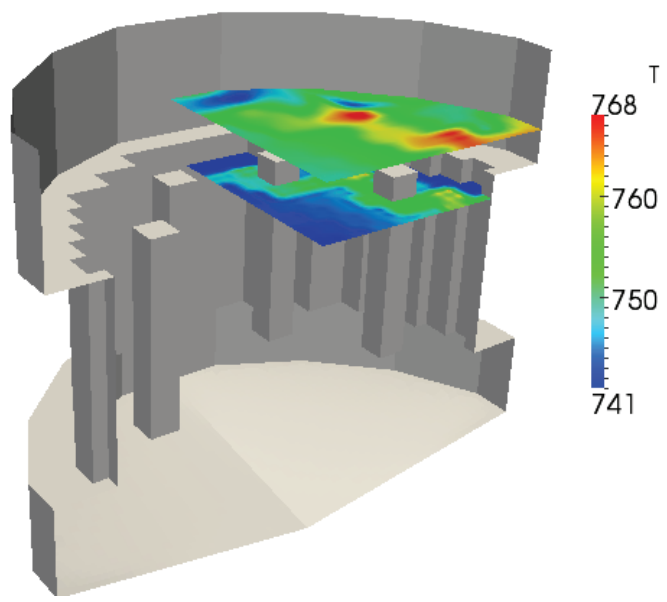


Figure 3.17: Test 2. Temperature field over different planes

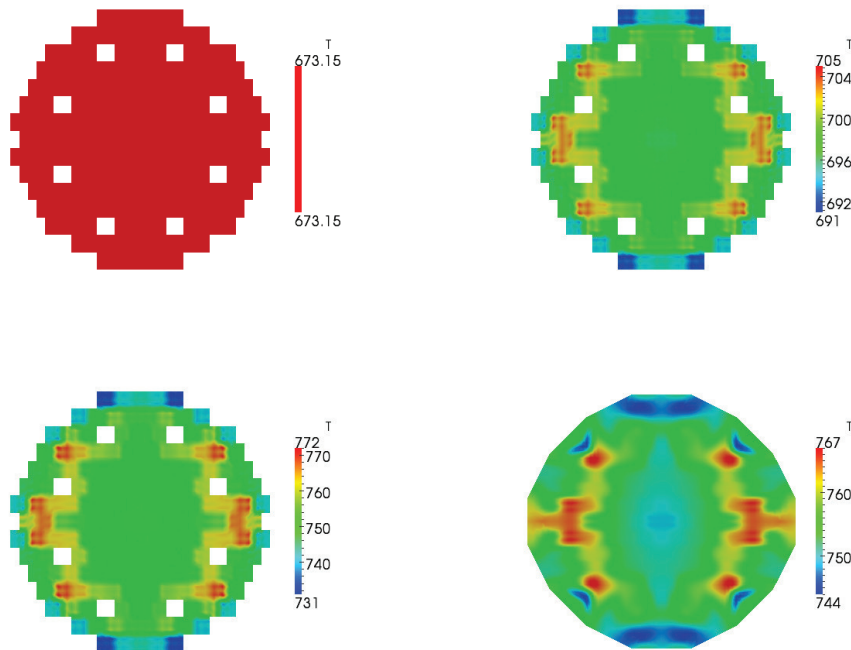


Figure 3.18: Test 2. Temperature distributions over different plane sections at  $z = 0.6$  (lower core),  $z = 1.2$  (upper core inlet),  $z = 1.8$  (upper core outlet),  $z = 2.4$  (upper plenum)

```

const double ReactData::mat_pf [8] [8] = {
  { 0.941, 0.962, 0.989, 1.017, 1.045, 1.178, 1.097, 0.    }, //A1_1-A1_7
  { 0.949, 0.958, 0.978, 0.996, 1.114, 1.123, 1.193, 0.857 }, //A2_1-A2_8
  { 0.963, 0.975, 0.992, 1.087, CTL_R, 1.011, 0.925, 0.    }, //A3_1-A3_7
  { 0.967, 0.973, 0.989, 1.008, 1.108, 1.028, 0.931, 0.    }, //A4_1-A4_7
  { 0.961, 1.060, 1.078, 1.137, 1.198, 0.943, 0.,    0.    }, //A5_1-A5_6
  { 0.954, 1.029, CTL_R, 0.990, 1.068, 0.850, 0.,    0.    }, //A6_1-A6_6
  { 1.019, 1.066, 0.966, 0.842, 0.,    0.,    0.,    0.    }, //A7_1-A7_4
  { 0.878, 0.853, 0.757, 0.,    0.,    0.,    0.,    0.    } //A8_1-A8_3
};

```

The vertical power factor can be assumed to have a cosine-like distribution. Since we have three zones we have three different axial distributions. These three profiles are reported in `axpf` as

```

const double ReactData::axpf [10] [3] = {
  { 8.60089E-01,    8.48697E-01,    8.33685E-01 },
  { 9.32998E-01,    9.63034E-01,    9.52704E-01 },
  { 1.03749E-0,    1.06399E-0,    1.06801E-0 },
};

```



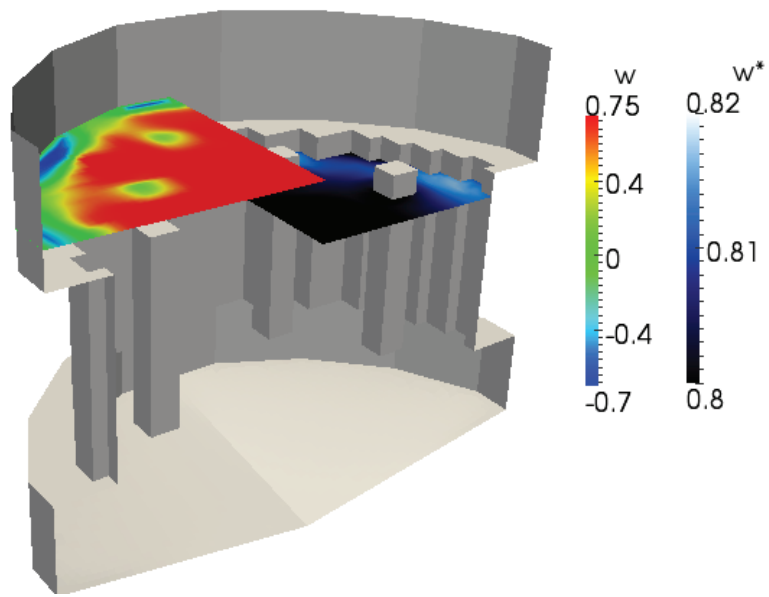


Figure 3.19: Test 2. Reduced z-component  $w^*$  of the velocity field

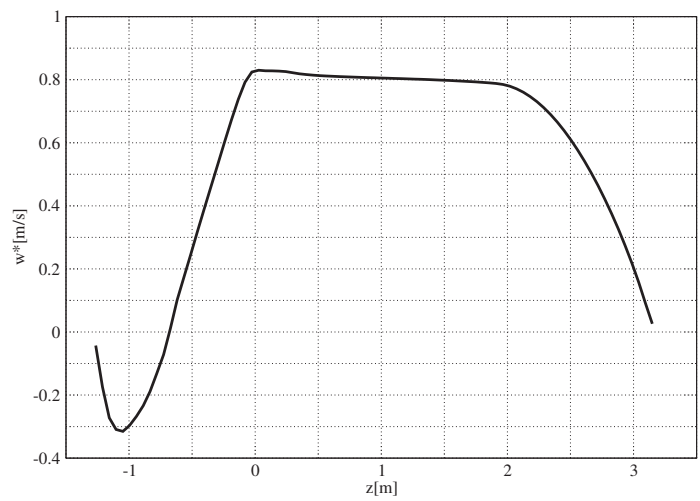


Figure 3.20: Test 2. Reduced z-component  $w^*$  of the velocity field along the z-axis

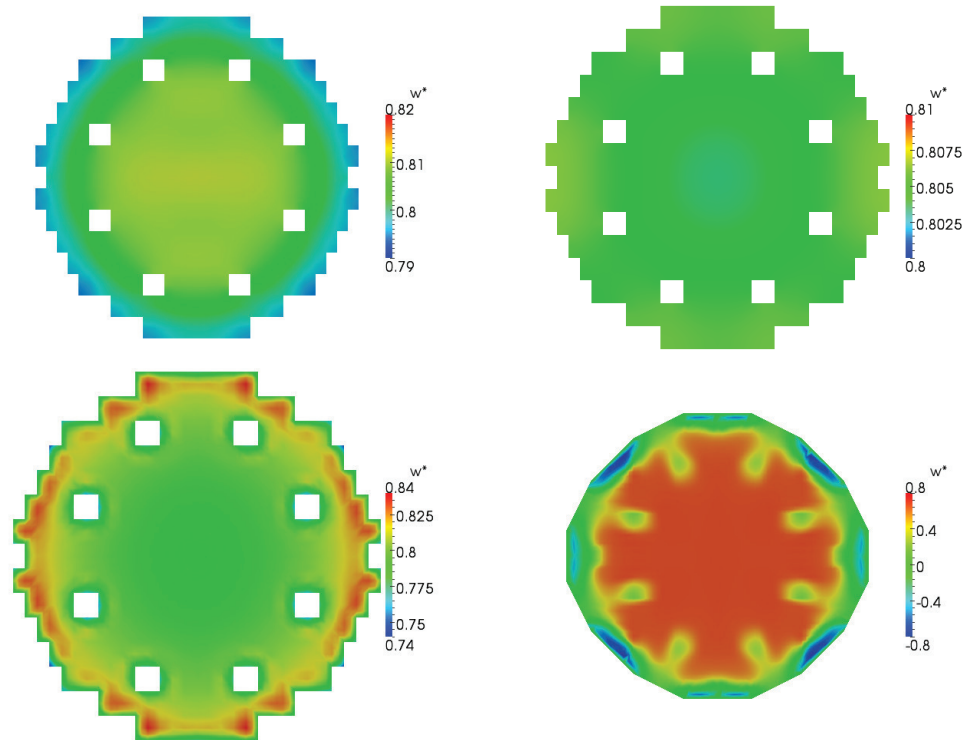


Figure 3.21: Test 2. Reduced  $z$ -component  $w^*$  of the velocity field over the plane sections at  $z = 0.6$  (lower core),  $z = 1.2$  (upper core inlet),  $z = 1.8$  (upper core outlet),  $z = 2.4$  (upper plenum)

```

{ 1.10010E-0,    1.12959E-0,    1.14484E-0 },
{ 1.14410E-0,    1.16319E-0,    1.17922E-0 },
{ 1.13892E-0,    1.15623E-0,    1.16983E-0 },
{ 1.09049E-0,    1.10313E-0,    1.10469E-0 },
{ 1.01844E-0,    1.00872E-0,    1.00793E-0 },
{ 9.05869E-01,   8.55509E-01,   8.63532E-01 },
{ 7.71503E-01,   7.07905E-01,   6.75547E-01 }
};

```

In this test the accidental pressure losses are set to zero since we neglect the spacer grids in the reactor core. The accidental pressure losses will be taken into account in the next test. Once the fuel distribution and the pressure losses are configured, one can compile and run the FEM-LCORE code in the usual way.

In Figures 3.16-3.18 the temperature distribution  $T$  over various sections of the reactor is shown. In Figure 3.16 we give an overview of the temperature distribution over a vertical section including the various zones of the reactor: lower plenum, lower core, upper core and upper plenum. Over each horizontal plane the temperature is no longer uniform like in the previous Test, due to the non-constant heat source distribution. This can be seen in Figures 3.17 and 3.18 where differences in

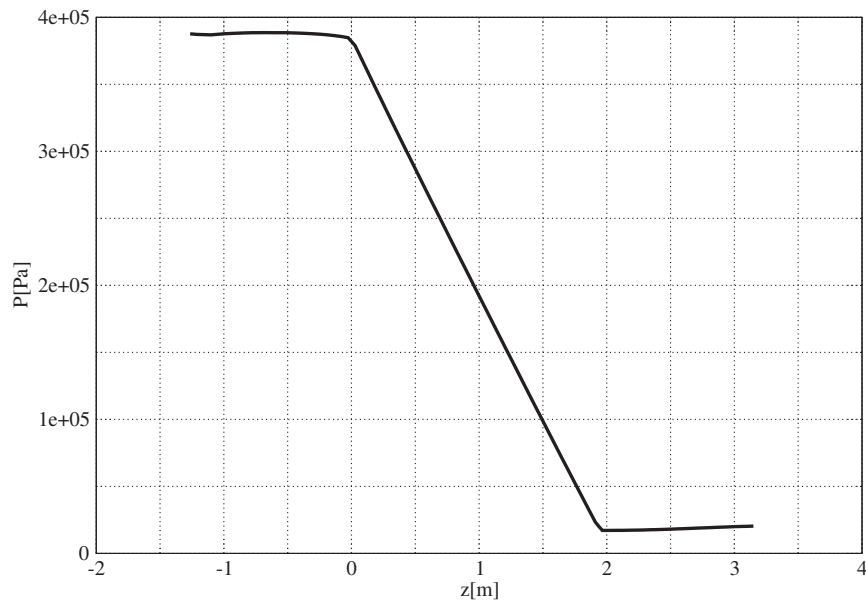


Figure 3.22: Test 2.  $P$  distribution along core axis

temperature can be noticed.

The velocity and pressure fields are shown in Figures 3.19-3.21. In Figure 3.19 one can see the overall distribution of the  $z$ -component  $w^*$  as defined in (2.6). In Figure 3.20 the velocity profile along the  $z$ -line at the center of the reactor is shown with solid line. We remark that in the core region the velocity can be considered approximately constant and the hypothesis of monodimensional motion can be applied. Still considering the reactor axis, the pressure profile is plotted in Figure 3.22. The velocity on different plane sections of the reactor is shown In Figures 3.21 we can see, from top left to bottom right, the velocity distributions at the plane sections  $z = 0.6, 1.2, 1.8$  and  $2.4$  m.

### 3.3 Pressure drop and partial blockage tests

#### 3.3.1 Test 3: influence of reactor spacer grids

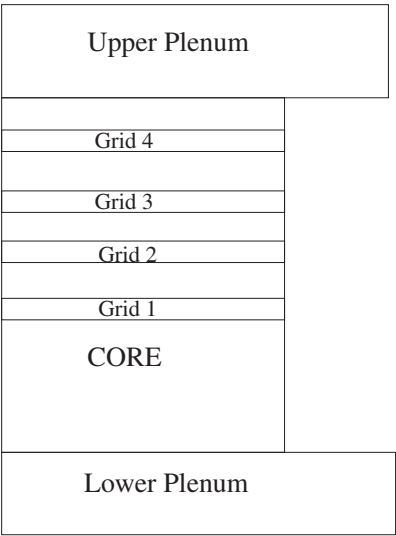


Figure 3.23: Grids on the reactor core model

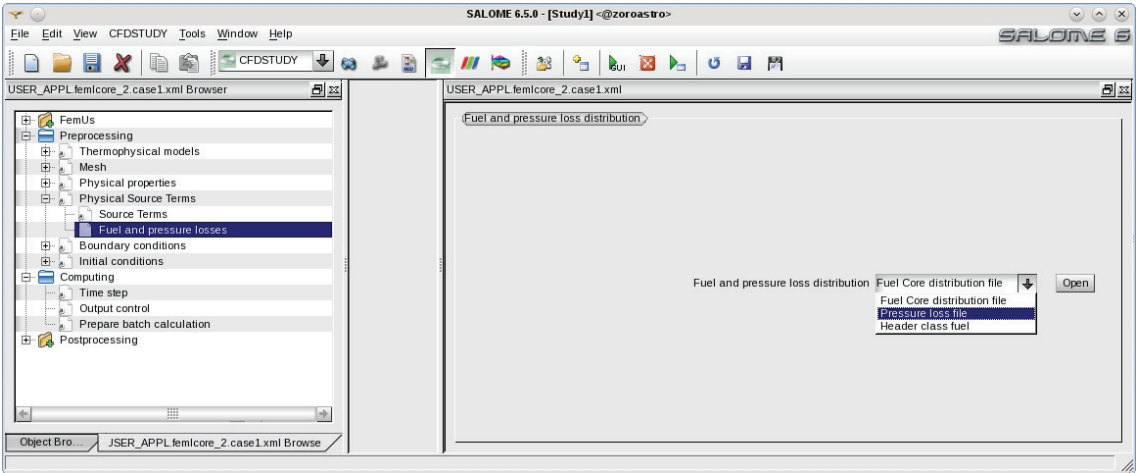


Figure 3.24: GUI Preprocessing menu: pressure loss page

In this section we take into account pressure drops due to the presence of the spacer grids. The standard grid distribution reported in Figure 3.23 can be modeled by setting the pressure peak factors in the SRC/ReactData.C file. The pressure loss can be defined in the appropriate GUI page shown in Figure 3.24. The page contains a simple combo-menu where the pressure loss file can be opened by selecting the *Pressure loss file* option.

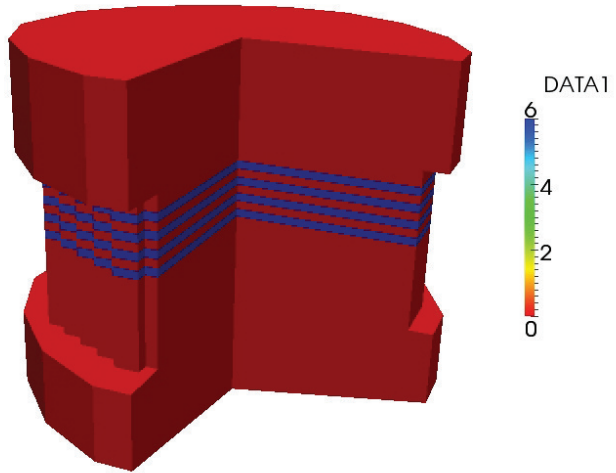


Figure 3.25: Pressure loss distribution

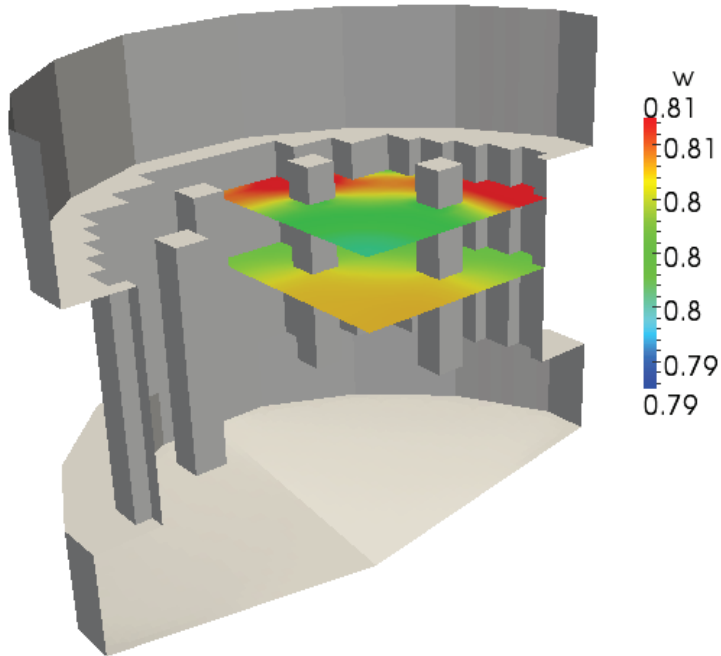


Figure 3.26: Test 3. Vertical component  $w^*$  of the reduced velocity in the reactor

In order to load the desired pressure loss factor distribution, shown in Figure 3.25, the file SRC/ReacData.C requires the following parameters: NLEV (number of multigrid levels), NZC (number of vertical subdivisions in the core section), HR (half fuel assembly cross-sectional side), HIN (heated core inlet height), HOUT (heated core outlet height). These data are already set in the header SRC/ReacDATA.h file if the

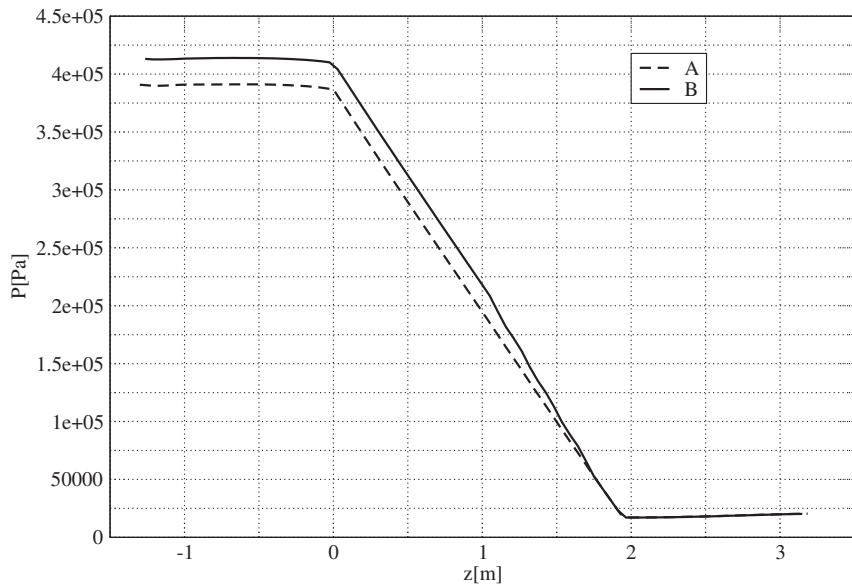


Figure 3.27: Test 3. Pressure  $P$  distribution along a  $z$ -line along the reactor axis with (B) and without (A) the spacer grids

fuel distribution has already been set. We suppose that the fuel distribution is as in Test 2 in Section 3.2.2.

The horizontal uniform pressure factor distribution must be written in the `mat_plf` double-indexed array, in which, every assembly in the quarter of reactor, is denoted by two indices, both ranging from 0 to 7.

The selection of the assembly types in the quarter reactor scheme is the same as in Test 2 in Section 3.2.2.

No pressure losses are considered in the horizontal direction, therefore the horizontal pressure loss factors are reported in the `mat_plf` matrix as

```
double mat_plf[8][8]={
{1.,1.,1.,1.,1.,1.,1.,0.}, //A1_1-A1_7
{1.,1.,1.,1.,1.,1.,1.,1.}, //A2_1-A2_8
{1.,1.,1.,1.,1.,1.,1.,0.}, //A3_1-A3_7
{1.,1.,1.,1.,1.,1.,1.,0.}, //A4_1-A4_7
{1.,1.,1.,1.,1.,1.,0.,0.}, //A5_1-A5_6
{1.,1.,1.,1.,1.,1.,0.,0.}, //A6_1-A6_6
{1.,1.,1.,1.,0.,0.,0.,0.}, //A7_1-A7_4
{1.,1.,1.,0.,0.,0.,0.,0.} //A8_1-A8_3
};
```

The vertical pressure loss factors reproduce accidental pressure drops at the heights of the grid spacers. This distribution must be written in the `axplf` array as

```
double axplf[10][3]={ 10^5 x
  { 0.00,0.00, 0.00},
  { 0.00,0.00, 0.00},
  { 0.52,0.52, 0.52},
  { 0.00,0.00, 0.00},
  { 0.52,0.52, 0.52},
  { 0.00,0.00, 0.00},
  { 0.52,0.52, 0.52},
  { 0.00,0.00, 0.00},
  { 0.52,0.52, 0.52},
};
```

Once the fuel and the pressure loss factors are configured, one can compile and run the FEM-LCORE code in the usual way.

In Figure 3.26 the reduced velocity  $w^*$  distribution over two sections corresponding to the lower and upper core regions of the reactor is reported.

In Figure 3.27 we show the pressure  $P$  distribution along the core axis with (B) and without (A) the spacer grids. We notice that the total pressure drop in the core with grids is higher in agreement with the presence of the spacers in the reactor.

### 3.3.2 Test 4: partial assembly blockage in closed assembly geometry

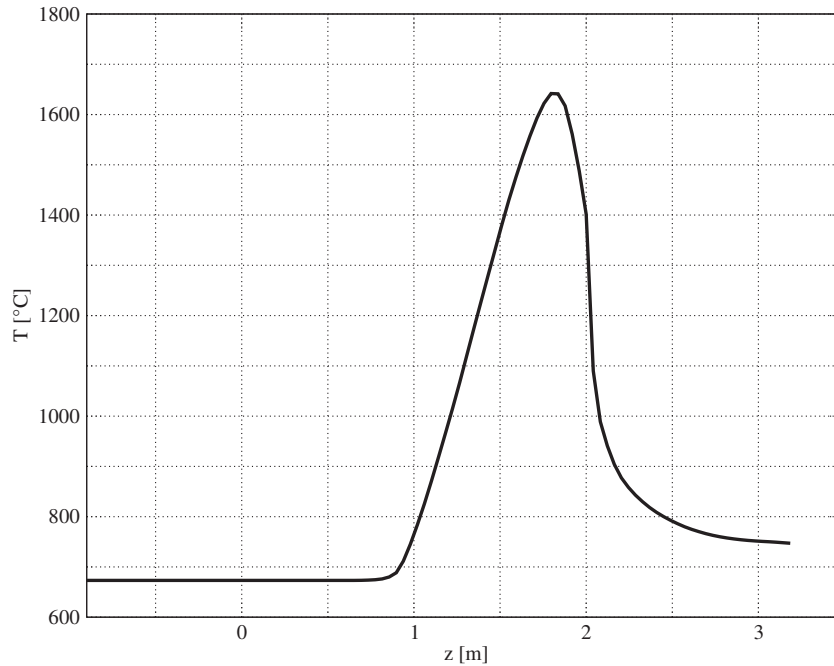


Figure 3.28: Test 4 (closed assembly). Temperature field along the  $z$ -axis at  $x = 1.323$  and  $y = 1.029$  in the partially blocked assembly with closed assembly geometry

In this section we present some results of the FEM-LCORE code concerning a partial blockage event occurring in the core with closed assembly geometry. In a partial blockage event the coolant flow is reduced in the corresponding assembly. In such case the heat removal process in the assembly is strongly decreased and a temperature increase of the coolant is expected. The blockage model used here is rather different than that reported in [3]. In this model we use pressure losses to impose the partial blockage while in [3] the velocity field is used. In [3] the blockage is total and no fluid is allowed to go through the blocked assembly which is modeled as a solid region. Here in this model the fluid goes through the assembly at low velocity with transport of heat and mass through the blocked assembly towards the upper plenum. In [3] this heat transport is not allowed and the transport of mass through the assembly is ignored.

In this case we assume the configuration of Test 2 in Section 3.2.2 and we consider the partial blockage of the assembly with cross section centered at  $x = 1.323$  and  $y = 1.029$  in the simulated quarter domain. The partial blockage is located at the beginning of the core region and it is set by adding large pressure losses in that region of space. Since the assembly is closed the mass flow rate is the same through



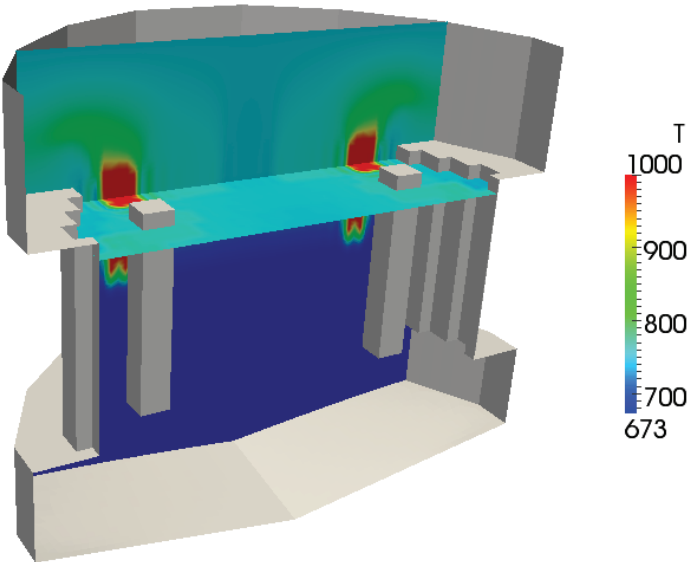


Figure 3.29: Test 4 (closed assembly). Temperature field with partial blockage over a reactor vertical section

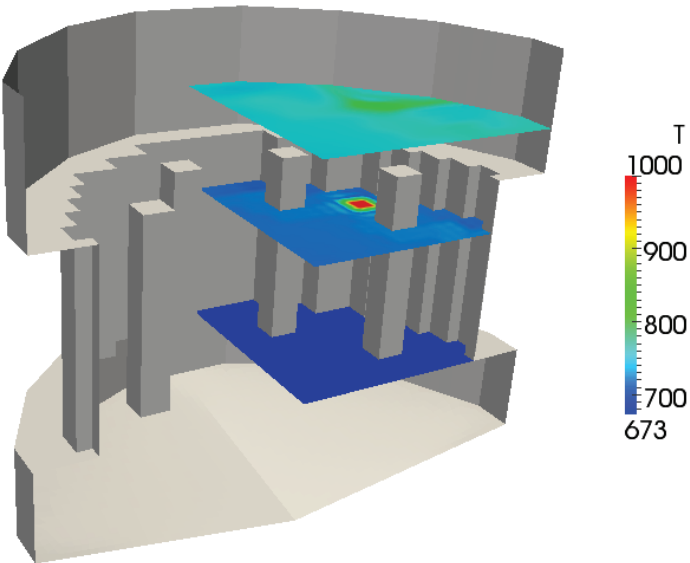


Figure 3.30: Test 4 (closed assembly). Temperature field with partial blockage over different plane sections

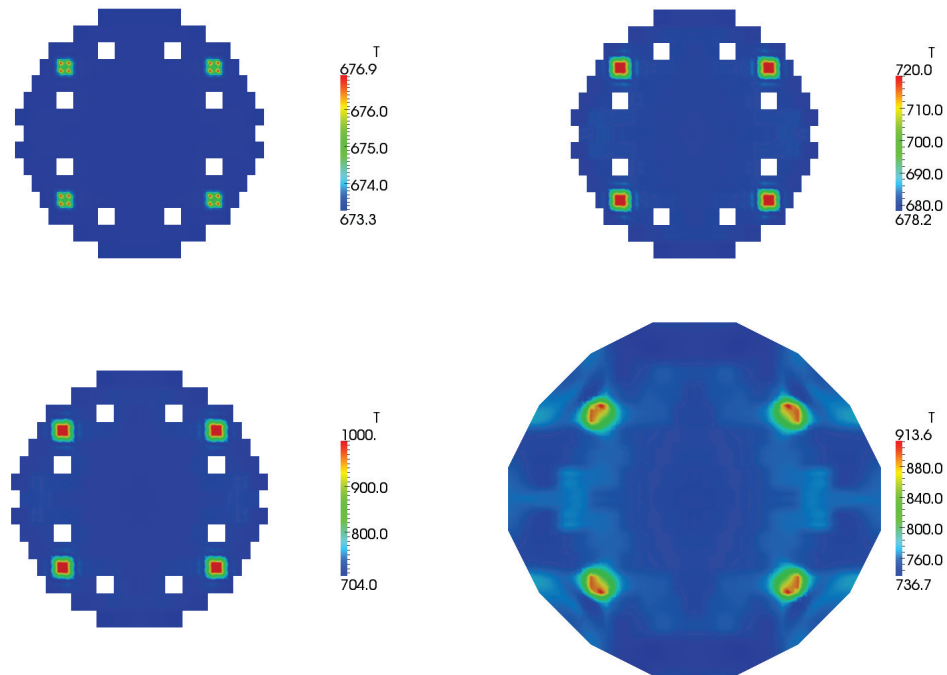


Figure 3.31: Test 4 (closed assembly). Temperature distributions over the plane sections at  $z = 0.8$  (lower core),  $z = 1.0$  (upper core inlet),  $z = 1.4$  (upper core outlet),  $z = 2.2$  (upper plenum)

all the assembly. Due to the previously discussed symmetries, the blockage of four assemblies is then simulated. We recall that the case of closed assemblies is enforced in the code by imposing the velocity fields on the lateral surface of each assembly to be parallel to the  $z$ -direction. The  $u$  and  $v$  components on those surfaces are set to zero, so that no cross flow takes place.

In Figure 3.28 the temperature profile along the  $z$ -line at the center of the partially blocked assembly is shown. The temperature of the partially blocked assembly case at the core outlet is much higher than the case of absence of blockage. In Figures 3.29-3.31 we report the temperature distribution  $T$  over various sections of the reactor. In Figure 3.29 one can see the temperature field over a vertical section of the reactor. In Figure 3.30 the overview of the temperature distribution  $T$  over the section of the reactor including the partially blocked assembly is shown. The same temperature scale is assumed in order to perform comparison. In this case the increase of temperature diffuses all around the blocked assembly creating a huge hot spot. One can remark that the increase of temperature starts at the inlet of the heated core where the blockage is located. It is very interesting to see the temperature on different plane sections of the reactor as shown in Figures 3.31, where we can see, from top left to bottom right, the temperature distributions at the horizontal sections  $z = 0.8$ , 1.0, 1.4 and 2.2 corresponding to lower core, upper core

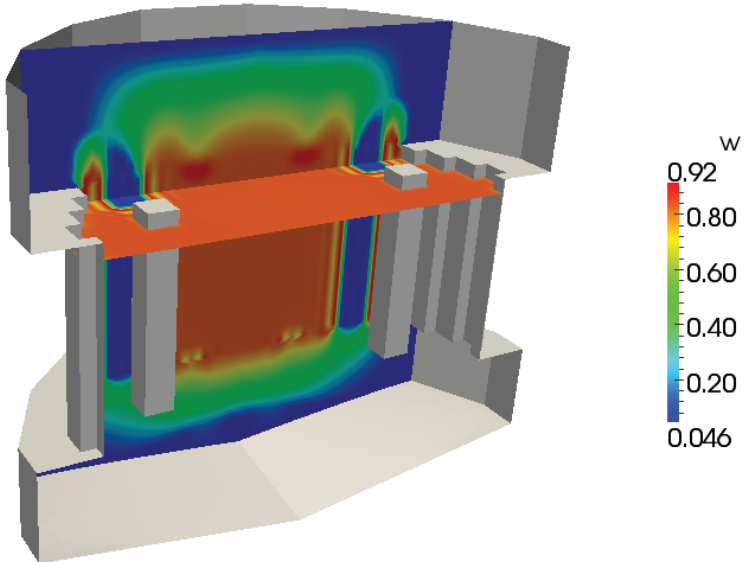


Figure 3.32: Test 4 (closed assembly). Reduced z-component  $w^*$  of the velocity field over a reactor vertical plane

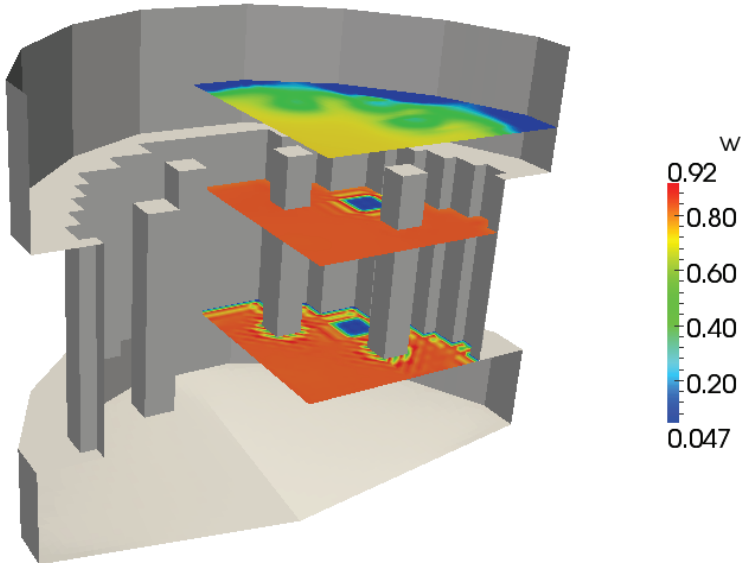


Figure 3.33: Test 4 (closed assembly). Reduced z-component  $w^*$  of the velocity field over different planes

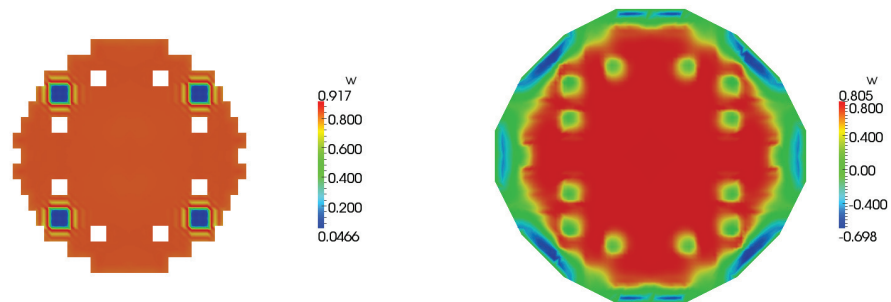


Figure 3.34: Test 4 (closed assembly). Details of the reduced  $z$ -component  $w^*$  of the velocity field over the two plane sections

inlet, upper core outlet and upper plenum, respectively. We can see the increase of temperature immediately close to the blockage height up to the upper plenum inlet.

The velocity field is shown in Figures 3.32-3.34. In Figure 3.32 one can see the overall distribution of the  $z$ -component  $w^*$  as defined in (2.6). The minimum value for the reduced velocity is  $0.046 \text{ m/s}$ . The velocity on different plane sections of the reactor is shown in Figures 3.33-3.34. In particular in Figure 3.34 we can see, from top left to bottom right, the velocity distributions over two plane sections corresponding to  $z = 0.8$  and  $z = 2.2$ . We can clearly see the minimum reduced velocity in the section where the assembly is blocked. Large peaks and oscillations of the pressure field close to the boundary of the partially blocked assembly may be considered numerical errors which disappear if one uses discontinuous elements for the pressure. We recall that we are using Taylor-Hood finite elements which consist of piecewise-continuous quadratic polynomials for velocity and piecewise-continuous linear polynomials for pressure.

### 3.3.3 Test 5: partial assembly blockage in open assembly geometry

In this section we deal with a partial blockage event occurring in the core with open assembly geometry. In a partial blockage event the coolant flow is reduced in the corresponding assembly but still takes place and lower peak temperatures are expected with respect to the closed assembly case. For this case with open

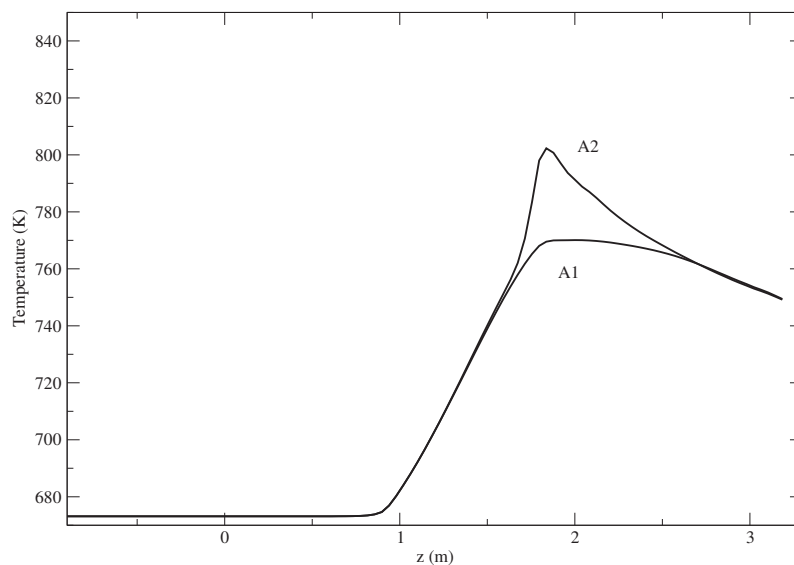


Figure 3.35: Test 5 (open assembly). Temperature field along the  $z$ -axis in the partial blocked assembly for case A1 (inlet) and A2 (outlet)

assemblies, we investigate some blockage events occurring at different core heights. In particular, we consider a blockage occurring in a small area at the inlet of the upper core, i.e. at the beginning of the heated core and a partial blockage at the core outlet, i.e. at the end of the heated core, where the LBE fluid enters the upper plenum region. Thus, we will subdivide the discussion of the results into two cases:

- Case Test 5.A1: open assemblies and blockage just before the inlet of the heated core;
- Case Test 5.A2: open assemblies and blockage near the outlet of the heated core.

In order to enforce the partial blockage condition numerically, we simply set a pressure loss factor of 100 in order to reduce the velocity components in the partial blockage volume. In all the following cases, we consider the same assembly as the previous Test. Due to the reactor symmetries, the partial blockage of four assemblies is then simulated.

In Figure 3.35 one can see the temperature field along the  $z$ -axis in the partially blocked assembly. It is very interesting to see for Cases A1 and A2 the temperature

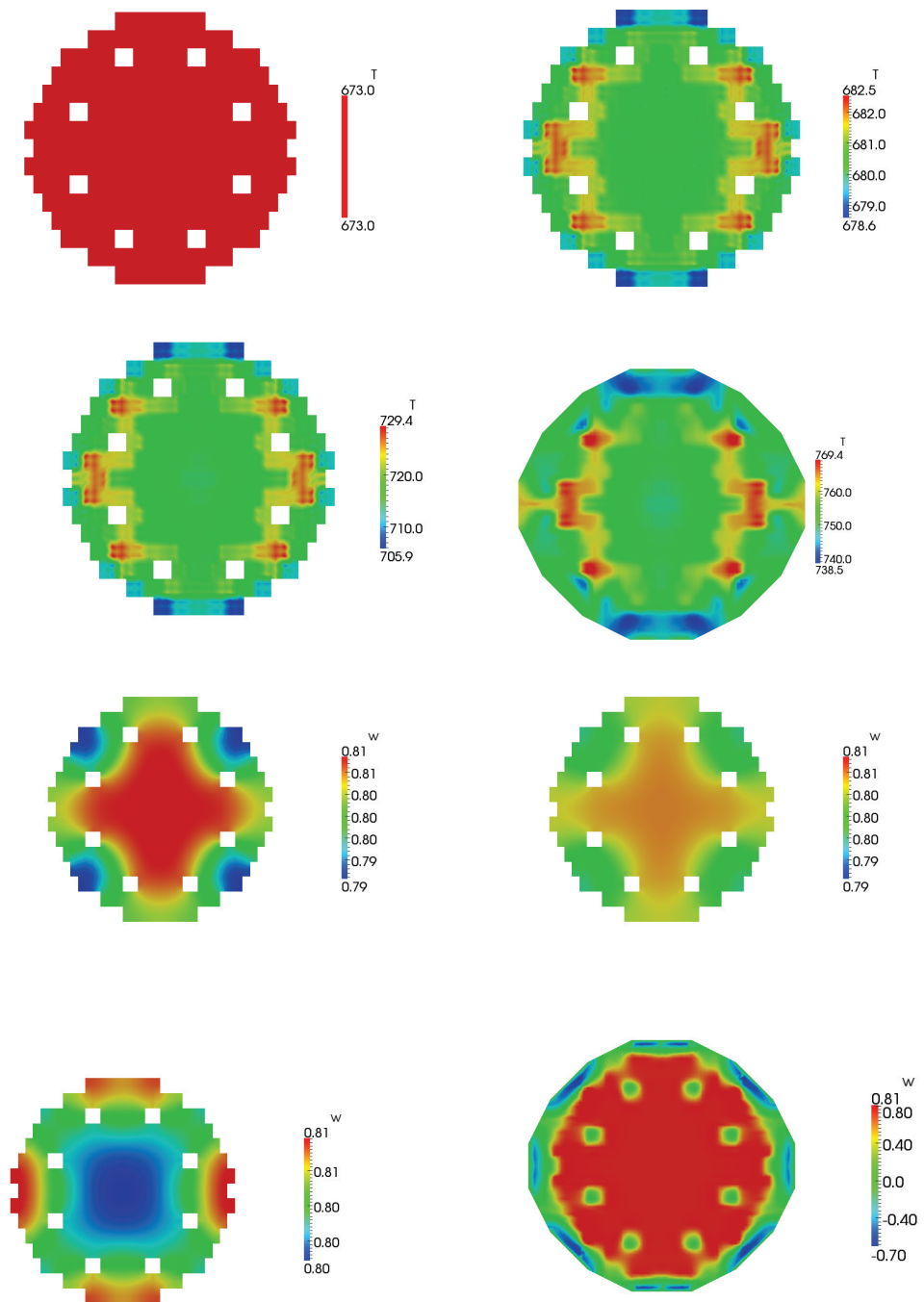


Figure 3.36: Test 5 (open assembly). Temperature and vertical velocity  $w^*$  distributions at the plane sections at  $z = 0.8$  (lower core),  $z = 1.0$  (upper core inlet),  $z = 1.4$  (upper core outlet),  $z = 2.2$  (upper plenum) for case A1 (inlet blockage)

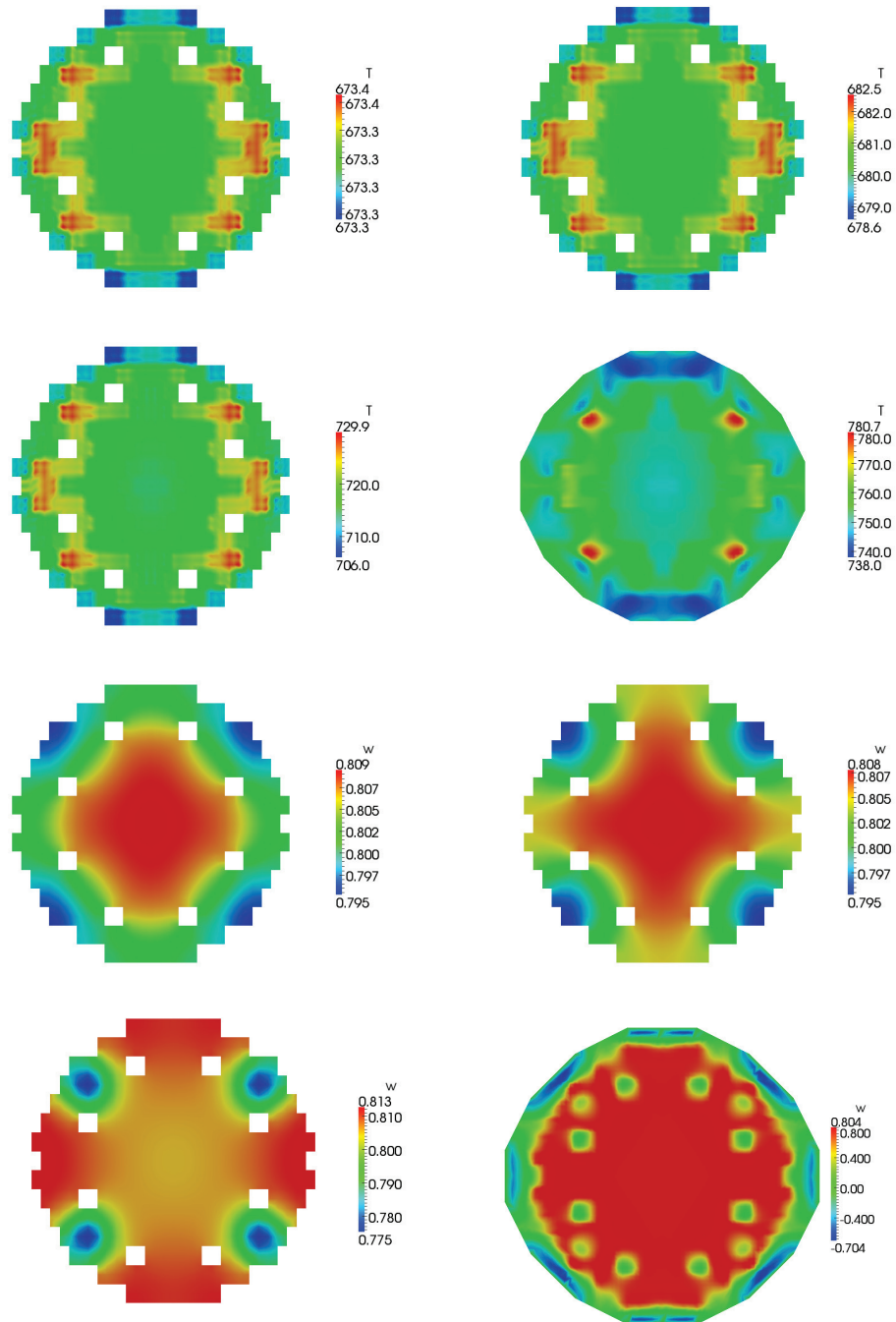


Figure 3.37: Test 5 (open assembly). Temperature and vertical velocity  $w^*$  distributions at the plane sections at  $z = 0.8$  (lower core),  $z = 1.0$  (upper core inlet),  $z = 1.4$  (upper core outlet),  $z = 2.2$  (upper plenum) for case A2 (outlet blockage)

on different plane sections of the reactor as shown in Figures 3.36-3.37. In Figures 3.36 we can see, from top left to bottom right, the temperature distributions at

	Sigla di identificazione	Rev.	Distrib.	Pag.	di
	NNFISS-LP3-059	0	L	86	106

the plane sections  $z = 0.8, 1.0, 1.4$  and  $2.2$  for Case A1. Here the partial blockage is close to the core inlet and the temperature increase with respect to the normal conditions is small. The temperature distributions over the same sections for case A2 are shown in Figure 3.37. The partial blockage between  $1.75$  and  $1.85$  m is close to the core outlet and a change in temperature is around  $50^\circ K$ , as one can see also from Figure 3.35.

The velocity field is shown on the bottom of Figures 3.36-3.37. In Figure 3.36 we can see, from top left to bottom right, the reduced velocity  $w^*$  distributions for Case A1 at the plane sections  $z = 0.8, 1.0, 1.4$  and  $2.2$ . Similar maps can be seen in Figure 3.37 for Case A2.



### 3.3.4 Test 6: partial blockage of multiple open assemblies

In Test 6 we assume that the assemblies are open, so that a flow exchange between them with transverse velocities is permitted. Then we enforce a partial blockage over a group of four assemblies as shown in Figure 3.38.

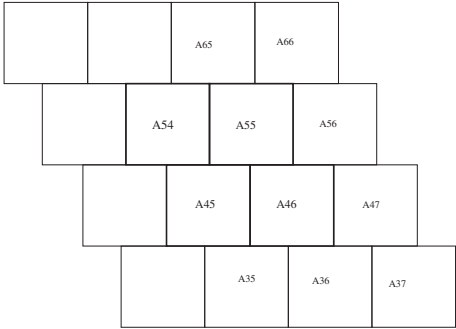


Figure 3.38: Test 6 (open assembly). The four partially blocked assemblies: A55 (1.029, 1.176), A56 (1.323, 1.176), A45 (1.176, 0.882), A46 (1.470, 0.882)

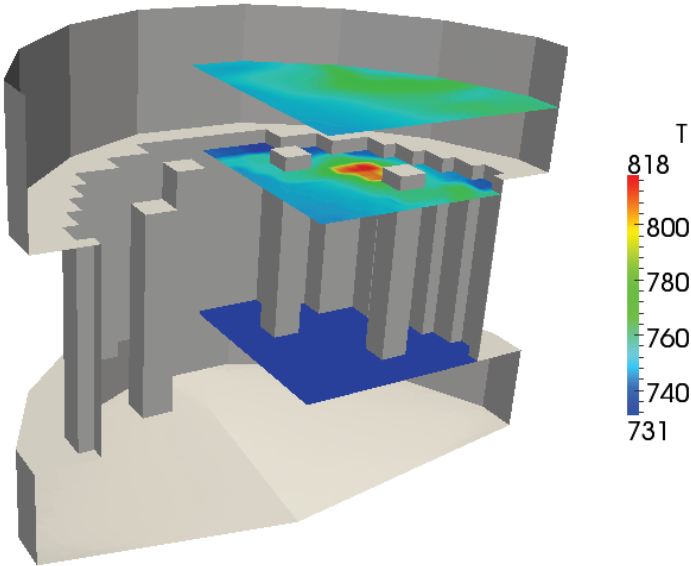


Figure 3.39: Test 6 (open assembly). Temperature field along horizontal planes in case of partial blockage of four assemblies

In order to enforce numerically the partial blockage of a group of assemblies, we simply increase the assembly pressure drop to obtain low velocity in the vertical direction. In this case, we consider the partial blockage of the following four assemblies: A55 (1.029, 1.176), A56 (1.323, 1.176), A45 (1.176, 0.882), A46 (1.470, 0.882).

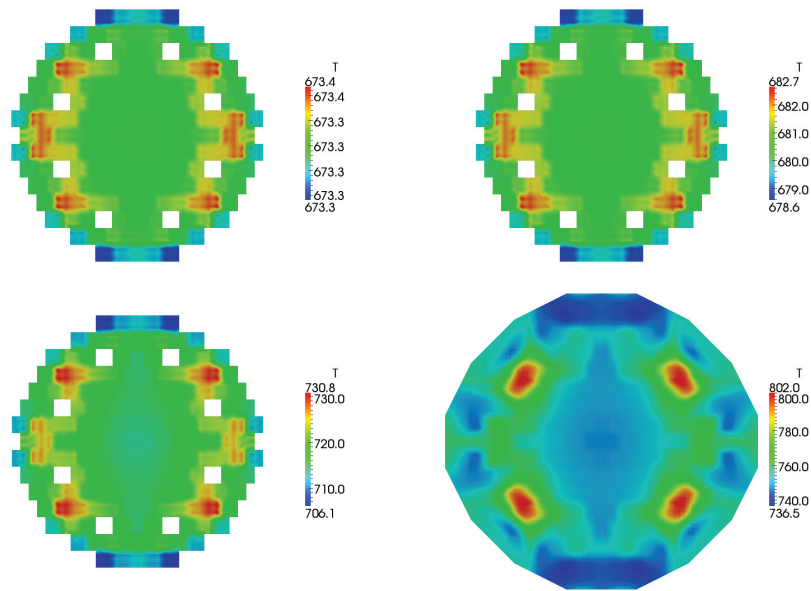


Figure 3.40: Test 6 (open assembly). Temperature distributions at the plane sections at  $z = 0.8$  (lower core),  $z = 1.0$  (upper core inlet),  $z = 1.4$  (upper core outlet),  $z = 2.2$  (upper plenum).

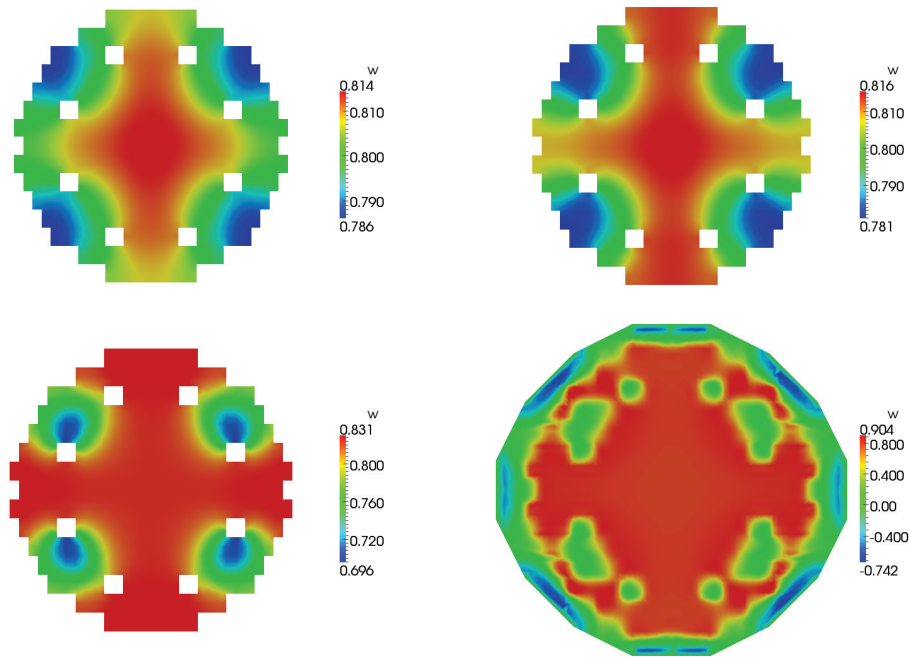


Figure 3.41: Test 6 (open assembly). Vertical component of the velocity  $w^*$  over the plane sections at  $z = 0.8$  (lower core),  $z = 1.0$  (upper core inlet),  $z = 1.4$  (upper core outlet),  $z = 2.2$  (upper plenum).

	Sigla di identificazione	Rev.	Distrib.	Pag.	di
	NNFISS-LP3-059	0	L	89	106

In the blockage region near the heated core outlet, between 1.75 and 1.85 m, the coolant cannot flow freely and the velocity reduces to  $0.046\text{ m/s}$ . In Figure 3.39 one can see the temperature field along horizontal planes and in particular one can notice the blockage region close to the core outlet where the temperature increase is visible. In Figure 3.40 we can see, from top left to bottom right, the temperature distributions at the plane sections  $z = 0.8, 1.0, 1.4$  and  $2.2$  corresponding to lower core, upper core inlet, upper core outlet and upper plenum. We can see the temperature increase at the blocked assemblies. In Figure 3.41 the distribution of the vertical component of the reduced velocity  $w^*$  can be seen in the same plane sections as before.

### 3.3.5 Closed and open core partial blockage comparison

We now report some brief considerations that can be drawn from the comparison of Test 1, Test 4.A1, Test 4.A2, Test 5 and Test 6. In Figure 3.42 the temperature profile along the  $z$ -line at the center of the partially blocked assembly ( $x = 1.323$  and  $y = 1.176$ ) for closed and open assembly is compared. The temperature

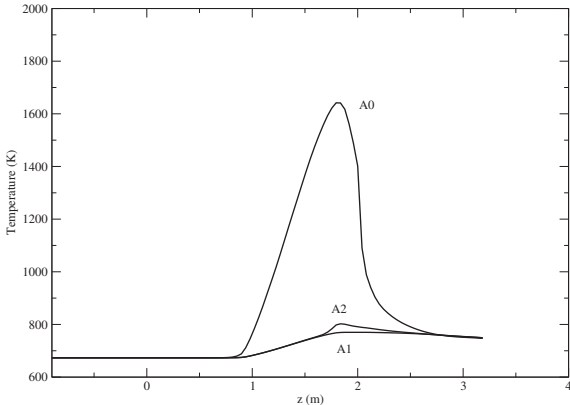


Figure 3.42: Comparison. Temperature field along the  $z$ -axis at  $x = 1.323$  and  $y = 1.176$  for Test 4 (A0), Test 5.A1 (A1) and Test 5.A2 (A2)

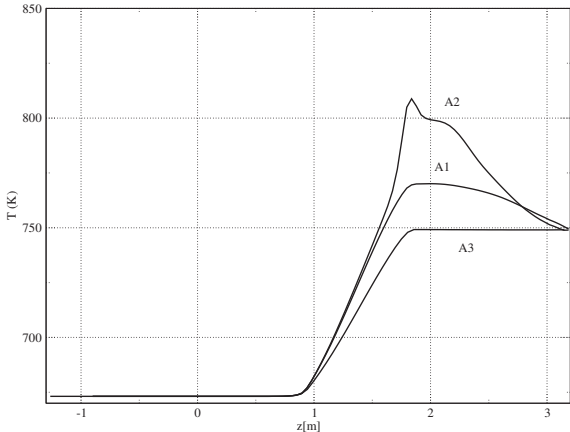


Figure 3.43: Comparison. Temperature field along the  $z$ -axis at  $x = 1.323$  and  $y = 1.176$  for Test 1 (A3), Test 5.A1 (A1) and Test 5.A2 (A2)

field along the  $z$ -axis for Test 4 is labeled A0, the result for Test 5.A1 with A1 and Test 5.A2 with A2. The temperature of the closed assembly case at the top core is much higher than the open core cases given by Test 5.A1 or Test 5.A2. This is clearly due to the fact that the fluid going along the vertical line is not allowed to cool in the transversal directions. This result is rather different than that reported in [3]. The blockage model here is completely different. In this model we use pressure losses to impose the partial blockage while in [3] the velocity field is

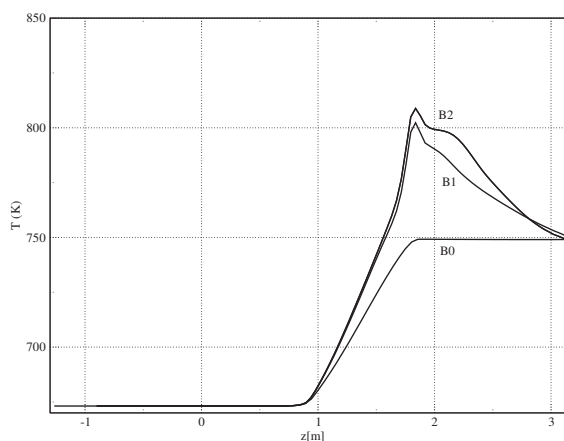


Figure 3.44: Comparison. Temperature field along the  $z$ -axis at  $x = 1.323$  and  $y = 1.176$  for the Test 1 (B0), Test 5.A2 (B1) and Test 6 (B2)

used. In [3] the blockage is total and no fluid is allowed to go through the blocked assembly which is modeled as a solid region. The temperature around the assembly is computed by using the heat flux generated inside the solid blocked region. Here in this model the fluid goes through the assembly at low velocity (about  $0.046 \text{ m/s}$ ) and its temperature increases by taking heat from the interior of the assembly. The fluid at high temperature transports heat and mass through the blocked assembly towards the upper plenum. In [3] this heat transport is not allowed and the transport of mass through the assembly is ignored.

In Figure 3.43 we have the comparison for the open core case with and without blockage. Temperature field along the  $z$ -axis for the Test 1 (A3), Test 5.A1 (A1) and Test 5.A2 (A2) at  $x = 1.323$  and  $y = 1.176$  are shown. Test 1 is the test with uniform fuel distribution and therefore a measure of core average temperature distributed through the core. We remark that, in an open core configuration, the partial blockage located before the heated-core region does not change much the temperature profile. The partial blockage near the end of the upper-core brings a temperature increase of about  $50^\circ\text{K}$ .

Finally a comparison between multiple assembly and single assembly blockage for the open core case. In Figure 3.44 temperature field along the  $z$ -axis for the Test 1 (B0) Test 5.A2 (B1) and Test 6 (B2) at  $x = 1.323$  and  $y = 1.176$  are shown. We remark, that for the open core case, an accident with partial multiple assembly blockage near the reactor outlet does not change much the temperature compared to a single blockage event. In this report we limit to these simple example configurations but for a deep investigation a large range of cases should be studied with different vertical and horizontal location of the partial assembly blockage.

### 3.4 Parallel performance tests

#### 3.4.1 Test 7: FEM-LCORE with parallel CPUs

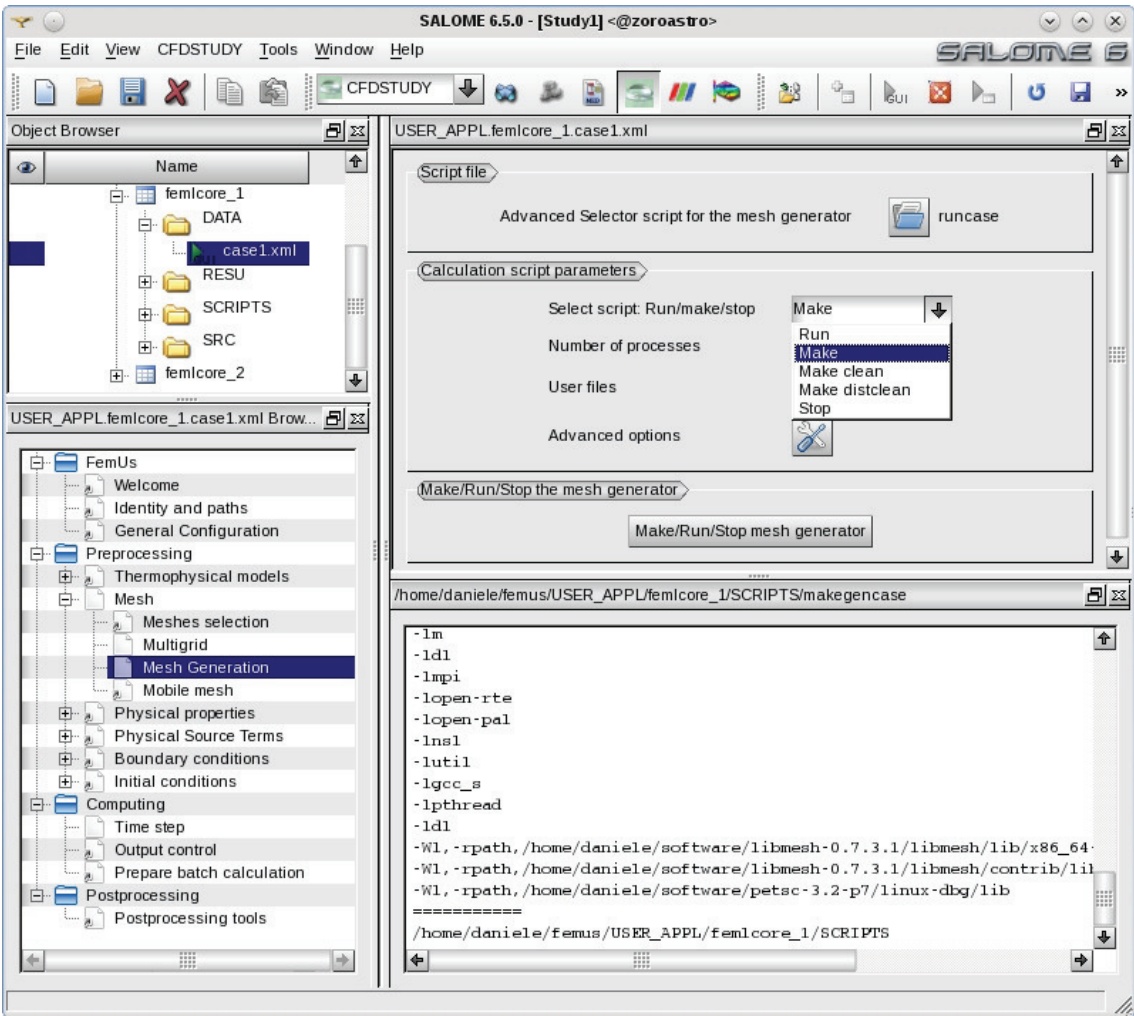


Figure 3.45: FEM-LCORE GUI menu: *Mesh generation* page

Code parallelization is nowadays very popular since it reduces CPU time and makes the use of numerical computations more attractive. It is of fundamental importance for three-dimensional time-dependent simulations for which huge computational resources and a great number of machine-hours are required. The previous version of FEM-LCORE was running only with one processor but in this new version we can take advantage of multiprocessor architectures. We plan to further improve the current parallel version to take full advantage of supercomputing architectures such as the CRESCO-ENEA grid [5, 4].

The transition of a code from a monoprocessor to a multiprocessor architecture is a difficult and long task requiring great efforts. In order to simplify this implemen-

tation the MPI libraries and various tools made available in the scientific community have been used. However the use of calls to external libraries may lead to a slowdown in the execution of the code which should affect the overall increase in performance obtained with parallel computing. In this section we test the FEM-LCORE code on a machine with a small number of CPUs in order to check the MPI implementation.

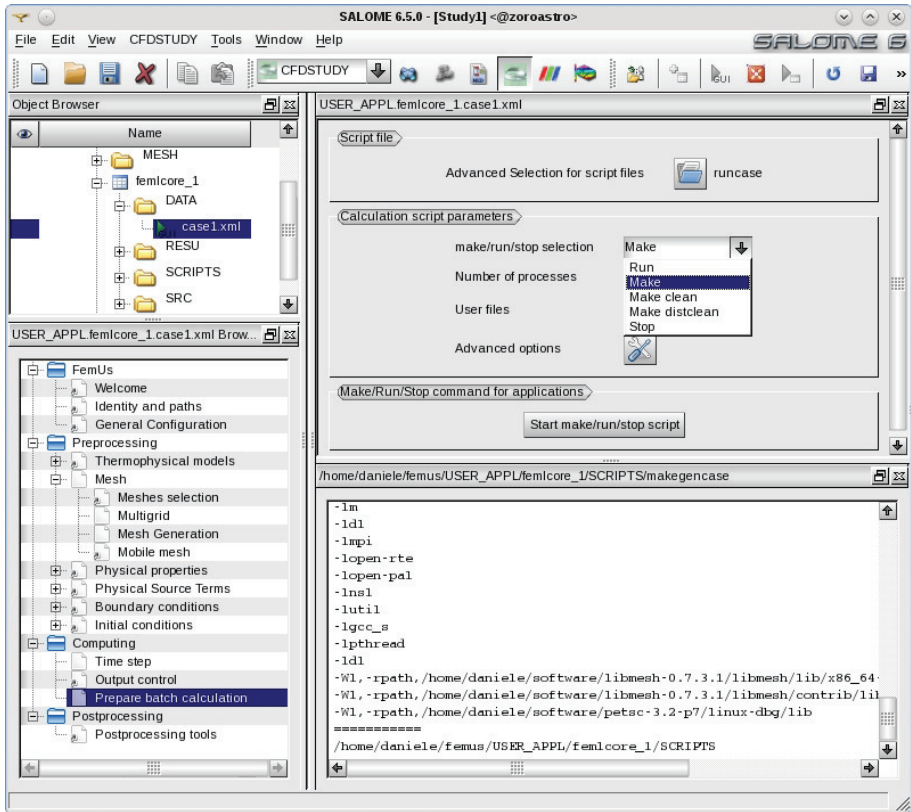


Figure 3.46: FEM-LCORE GUI menu: page for run control

In order to run the code in parallel one must simply set the number of processors during the compilation/generation of the mesh and during the compilation/running of the FEM-LCORE code. The `gencase` program runs from the *Mesh generation* page shown in Figure 3.45. The combo-menu selects the type of operation the code must execute: run, compile (`make`) and clean (`make clean` and `make distclean` perform the same action). The number of processors must be set in the appropriate line and it should not be changed during any restart operation. The `gencase` program compiles with the `make` option and starts the mesh generation with the `run` command.

Once the multilevel and parallel mesh is generated we can run the FEM-LCORE code. Core fuel distribution, pressure losses, boundary and initial conditions must be set. These configuration options were discussed in Chapter . Following these instructions we should be able to complete the final configuration stage and open the FEM-LCORE run page. The run page is shown in Figure 3.46. This command

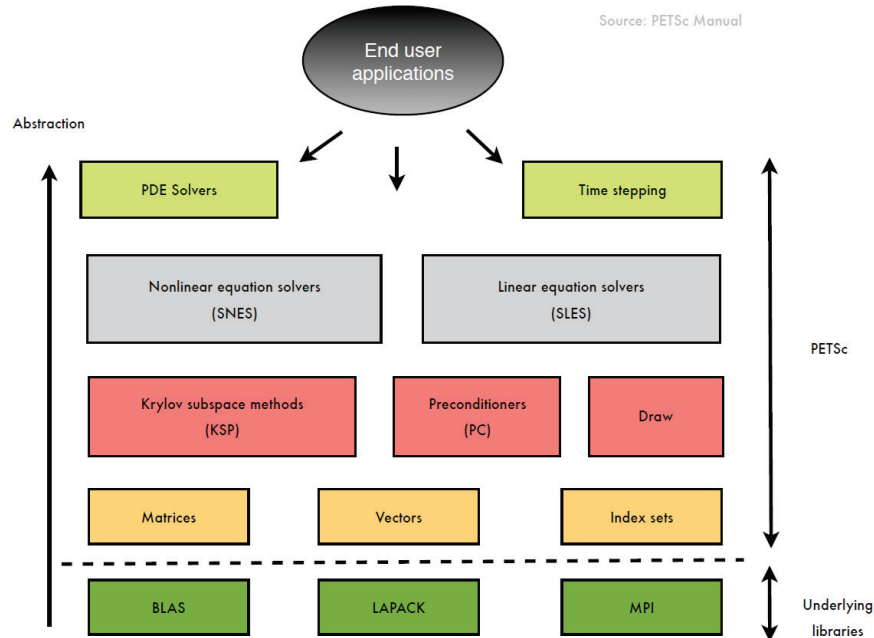


Figure 3.47: PETSc library

window is similar to the `gencase` mesh generator window. The combo-menu selects the type of operation the code must perform: run, compile and clean.

In order to calculate the execution time of a given block of instructions one can simply set, at the beginning and at the end of the block, a call to a function that returns the time interval. It is clear that the numbers are characteristic of the machine. In order to have machine-independent values, CPU times must be normalized with respect to a single processor value. In addition, in order to mitigate any disturbances due to the operating system as much as possible, the values are averaged over ten simulations for a given number of processors.

The de facto standard paradigm for parallelization is called *MPI* (Message Passing Interface). We use the implementation of this paradigm known as the OpenMPI library [22, 23]. The objectives of the OpenMPI library are high performance and high scalability. There are two notions of scalability: strong and weak. Strong scalability is defined as how the speedup varies with the number of processors for a fixed total problem size. We have perfect strong scalability if the speedup is linear. Weak scalability is defined as how the speedup varies with the number of processors for a fixed problem size per processor. A perfect weak scalability results in a time-invariant behavior with respect to a varying number of processors. The speedup is defined as

$$S = \frac{t_1}{t_n}, \quad (3.1)$$

which is the ratio between the execution time  $t_1$  on a sequential architecture and



the execution time  $t_n$  with  $n$  processors. The efficiency is defined as

$$E = \frac{S}{n} \cdot 100 [\%], \quad (3.2)$$

which is given by the ratio between the speedup  $S$  and the number of processors  $n$ , expressed in percent.

In order to avoid a direct use of MPI primitives and ease the implementation of the parallel matrix and vector algebraic operations we use the PETSc library. The PETSc (Portable Extension Toolkit for Scientific Computation) library is a set of data structures and functions for scientific applications, in particular for problems modeled by partial differential equations and solved by parallel algorithms [25]. This library can operate on vectors, matrices, preconditioners and linear/nonlinear solvers. It can be used to manage objects with standard C and FORTRAN languages through abstract interfaces (see Figure 3.47).

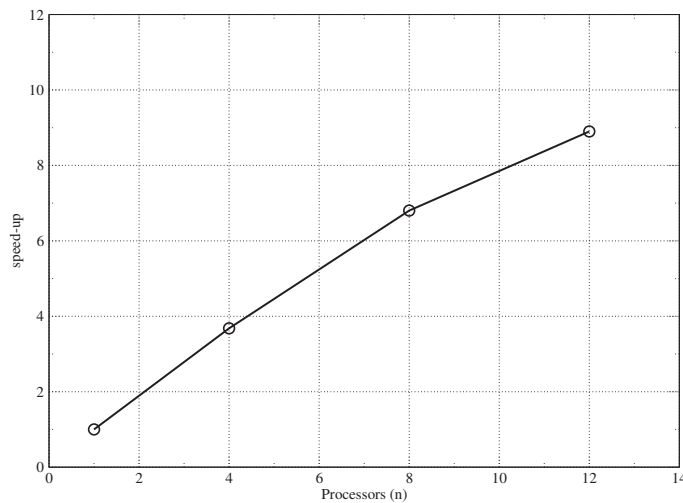


Figure 3.48: Test 7. Speedup of the FEMLCORE Navier-Stokes finite element assembly routine as a function of the number of processors

With its flexibility, the PETSc library is designed to split the effort between the tasks of parallelization and model development. The PETSc functions and variables use the MPI communication paradigm among processes. Normal operations are guaranteed also in case of sequential architecture and their use does not prevent the explicit reference to the MPI functions for special operations. It is important to emphasize that this library does not operate any load balancing and does not generate grids.

In Figures 3.48-3.49 we consider the speedup of the FEMLCORE Navier-Stokes and energy finite element assembly routine as a function of the number of processors. The configuration is the same as Test 2 in Section 3.2.2. In Figure 3.48 the performance for the Navier-Stokes assembly routine is shown while in Figure 3.49

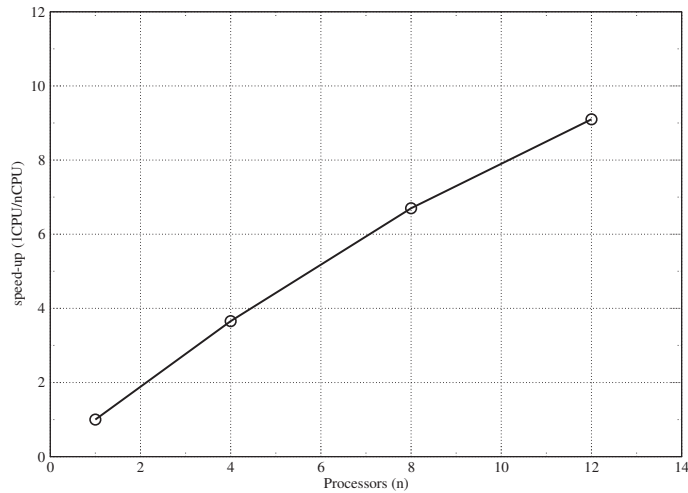


Figure 3.49: Test 7. Speedup of FEMLCORE temperature finite element assembly routine as a function of the number of processors

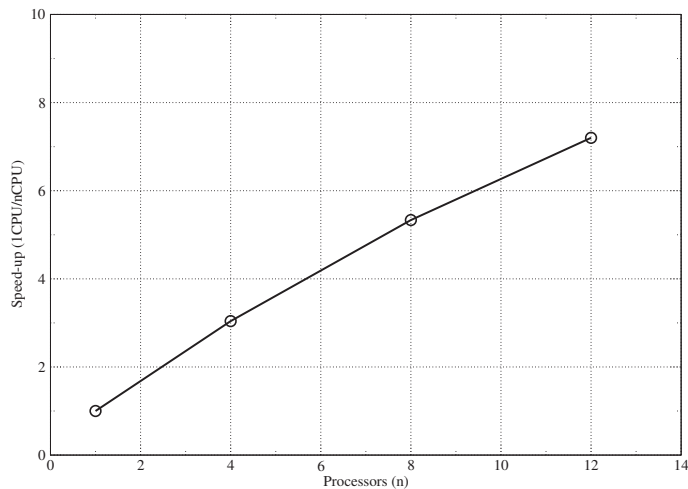


Figure 3.50: Test 7. Speedup of the FEMLCORE Navier-Stokes solver routine as a function of the number of processors

the energy equation case is reported. The computations are performed on a hexa-core machine with hyper-threading technology, so that twelve parallel processes can be instantiated. The assembly performance is rather good if one thinks that the twelve processes are not completely independent. The evaluation of the speedup of the FEMLCORE Navier-Stokes and energy solver routine as a function of the number of processors are shown in Figure 3.50 and 3.51 respectively. In this case the behavior deviates from the ideal more than in the assembly routine case, probably because of the increase in complexity of the functions involved in linear algebra operations. In this case the results are worse and some improvements should be implemented in the code. For example the code does not use the ghost cell method

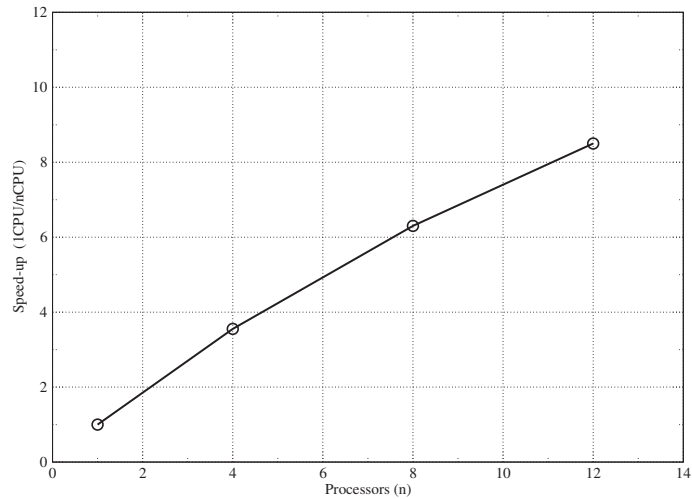


Figure 3.51: Test 7. Speedup of the FEMLCORE energy solver routine as a function of the number of processors

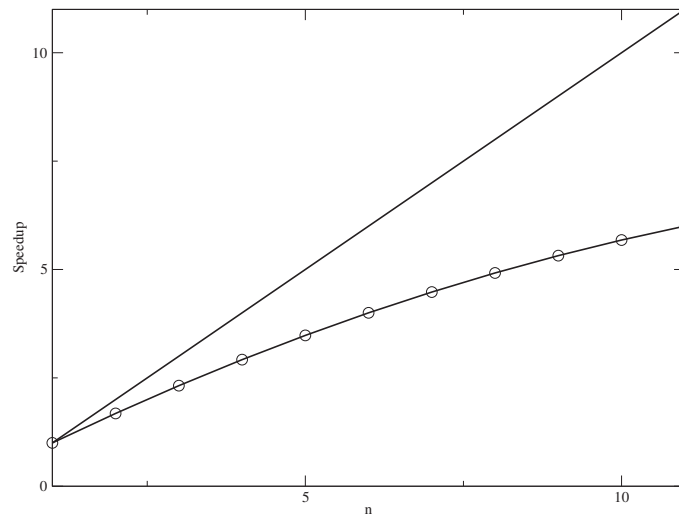


Figure 3.52: Test 7. Speedup as a function of the number of processors with the  $\kappa - \omega$  turbulence model

for exchange information between adjacent subdomains and this increases the data exchange time.

The speedup with the introduction of the  $\kappa - \omega$  turbulence model is reported in Figure 3.52. In this case we can see that the behavior detaches from the linear ideal profile as the number of processors increases and this lets room to improvements to the parallel version of the FEM-LCORE code.

### 3.4.2 Test 8: FEM-LCORE with GPU

The study of increasingly complex engineering systems has led to the development of machines with an increasing computing power. A big step in this direction is the introduction of parallel computing, that is the division of the computational load over more processors that can work simultaneously, when properly coordinated. A Graphics Processing Unit (GPU) may contain several hundreds of processors and therefore can be used to speed up fluidynamics computations. GPUs have evolved to the point that many real-world applications can be implemented on them and run significantly faster than multi-core CPU systems. Future computing architectures will be hybrid systems with parallel-core GPUs working in tandem with multi-core CPUs. The FEM-LCORE code uses the PETSC libraries that have started supporting GPU computing in their more recent versions. A preliminary study has been carried out to explore the possibility of using the FEM-LCORE application with the new GPU technology.

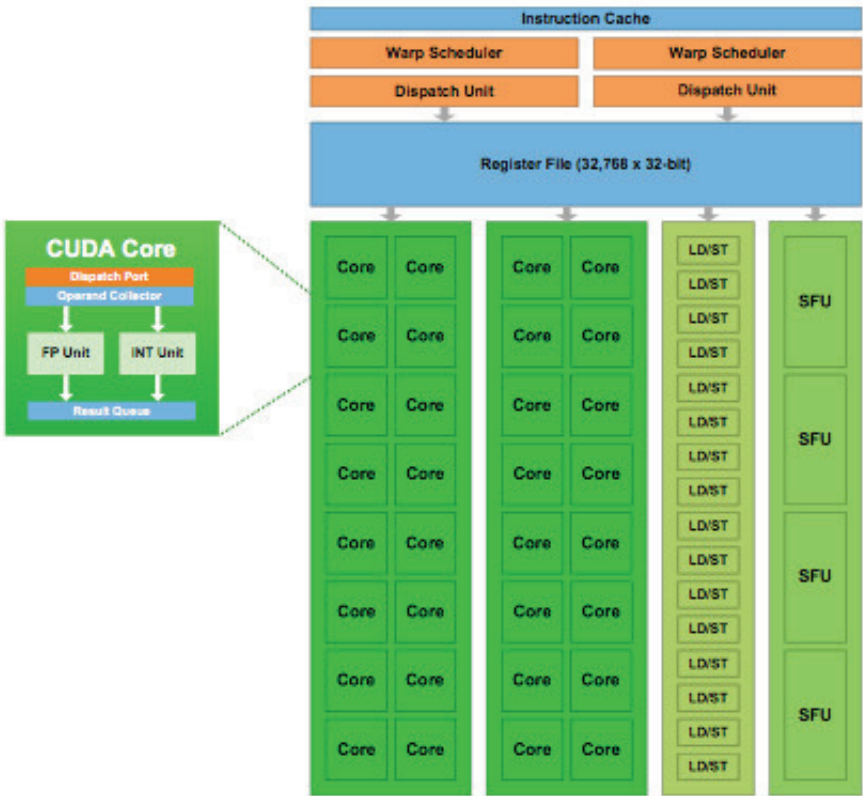


Figure 3.53: Streaming multiprocessor (SM) GPU structure

To understand about GPGPU (General Purpose Graphics Processing Unit) it is necessary to explain what are the components of a GPU and how programs interact with them. Below is a table with specific characteristics of the graphics card available at the Montecuccolino DIENCA laboratory where the tests for the FEM-LCORE

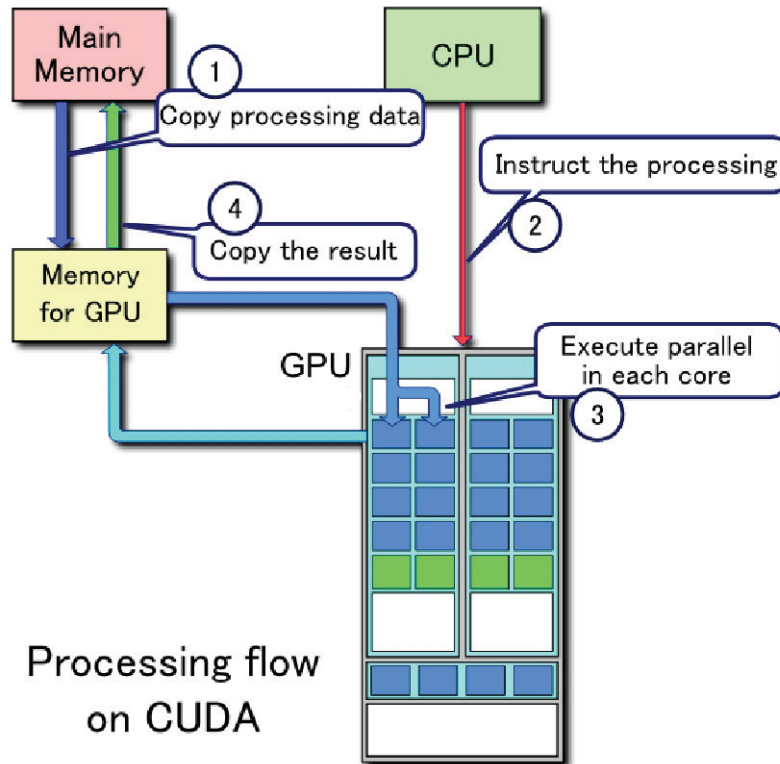


Figure 3.54: Workflow for GPU computing

code are performed. The NVIDIA Quadro 4000 card is composed of 8 streaming

CUDA cores	256
Gigaflops (Single Precision)	486.4
Gigaflops (Double Precision)	243.2
Total Frame Buffer	2 GB GDDR5
Memory Bandwidth	89.6 GB/s

Table 3.4: Specific characteristics of the NVIDIA Quadro 4000 graphics card

multiprocessors (SM) and shared memory. The SM are composed of 32 CUDA cores and therefore can manage and process groups of 32 threads; in total a GPU like Quadro 4000 is able to perform simultaneously groups of 256 threads (8 SM \* 32 threads), see Figure 3.53. This type of architecture is named Fermi.

The complexity of a Fermi architecture is managed by a multi-level programming model that allows developers to focus on the algorithms in general rather than on the details of the parallelization of the computations. In programming models for GPU like CUDA and OpenCL the elementary computational algorithm takes the name of *kernel*, which can be written in C language with appropriate modifications. Once kernels are compiled, they are made up of several threads that run the same

program in parallel: one thread can be seen as a small group of instructions. Groups of threads are combined into blocks each of which runs in a single SM in order to access and manage shared memory. In turn, the blocks are divided into groups of 32 threads called warps which constitute the calculation step for a SM. In the Fermi architecture two warps can be executed in parallel from different blocks. As shown in Figure 3.53, one Streaming Multiprocessor includes the following components: 32 CUDA cores (each of these cores can perform floating point operations), LS-16 units (the LS units are necessary for memory loading and unloading), special function units (such as the reciprocal function, square power, etc.) and a 64 Kbytes local SRAM. The SRAM is divided between the cache and local memory. In Figure 3.54 it is possible to observe the workflow of a program that uses the GPU. We have three main stages: copying data from main memory to GPU memory, GPU calculations and copying the results from GPU back to main memory. By analyzing the time spent by the program to run every step, we can get an idea how the GPU works.

To test the computation ability of the above mentioned graphics card we performed tests to see how the solution algorithms often used in the code behave with the GPU computing technology. The FEM-LCORE solver is a Krylov solver implemented in the PETSC libraries. Given a matrix  $A$  and a vector  $b$ , the subspace

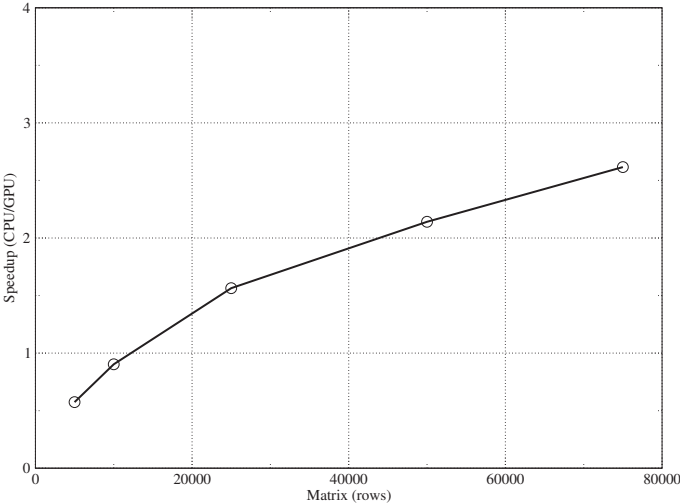


Figure 3.55: Speedup of the KSP solver as a function of the matrix dimension

created by the linear closure of the columns of the matrix is called a Krylov subspace. To evaluate the time resolution of the system we put the counter of time around the Krylov solver call and we parametrize with respect to the size of the matrix. The results are in Figure 3.55. The figure indicates that GPUs are ineffective for matrices of order less than 15000 rows. By analyzing the performance with the NVIDIA GPU profiler, one realizes that in these cases the time for copying data from the host to the GPU is greater than the computation time. Increasing the size of the matrix increases the allocation time but it also drastically increases

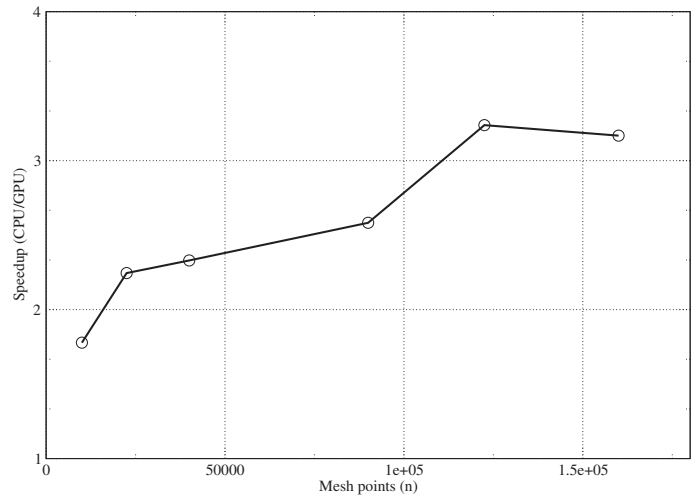


Figure 3.56: Speedup for incompressible Navier-Stokes and energy equations as a function of mesh points

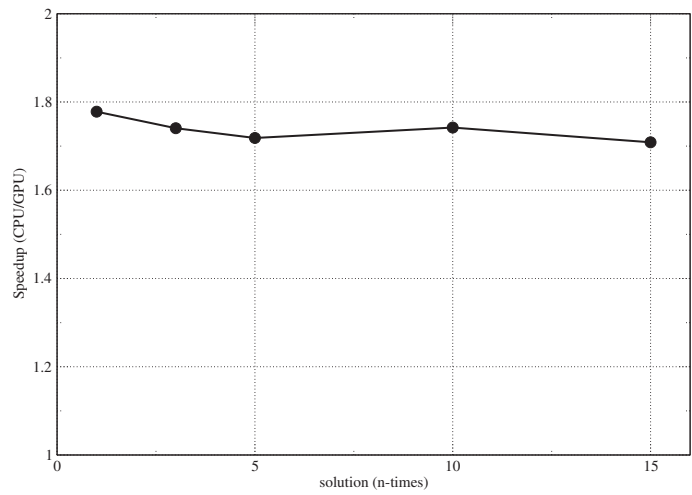


Figure 3.57: Speedup for incompressible Navier-Stokes and energy equations as a function of the number of solutions with fixed number of unknowns (10000)

the amount of calculations to perform, so that the time for performing arithmetic operations is dominant.

We have then analyzed the behavior of the GPU with an example that contains the Navier-Stokes and the energy equations. The test has been performed starting from the example 19 of the PETSC library. Details on equations and boundary conditions can be found in [25]. Since we solve with this library we expect similar results also for the FEM-LCORE code. By using the PETSC libraries we parametrize the GPU time as a function of the number of grid points. The results are shown in Figure 3.56. We see that even for small grids GPUs take less time to calculate the solution. Since the solution of the system requires a large number of iterations,

even for small systems we can appreciate the advantage of using the GPU. In applications like FEM-LCORE we repeat several times the solution of the system for each time step. Therefore it is important to test the GPU behavior for repeated iterations. In Figure 3.57 we see what happens if one repeats the solution of the system for  $n$  times. The ratio between the computation time with only the CPU and the computation time with the GPU which remains roughly constant. This indicates that the calculation time obtained using the GPU is already in the linear growth range and repeating this operation several times would not change the time of each single solution. The use of PETSC with GPUs is still under development and therefore also the implementation of GPU computing on FEM-LCORE is still in an early stage. However the results are sufficiently good to keep developing the implementation of the GPU technology on FEMuS and FEM-LCORE.



	Sigla di identificazione	Rev.	Distrib.	Pag.	di
	NNFISS-LP3-059	0	L	103	106

## Conclusions

In this report we have presented the development of the FEM-LCORE code for the study of three-dimensional thermal hydraulics of liquid metal reactors. The current version of the FEM-LCORE code is integrated in the SALOME platform and therefore it can be launched and configured inside the the SALOME GUI. All the preprocessing, running and postprocessing steps can be achieved inside the platform framework. We have tested this version with some simple examples with different fuel assembly distributions and pressure drops. Parallel performance with CPU and GPU has been discussed and some tests with partial assembly blockage have been studied. From all these preliminary tests the code can be easily used to study a large range of different LFR reactors but further improvement and validation should be carried out. The physical modeling on the porous reactor model core or turbulence modeling in the plenum must be improved by using experimental or computational data. Parallel performance can be enhanced by introducing ghost cell methods over the boundary of the partitioned domains. Future computing architectures will be hybrid systems with parallel-core GPUs working in tandem with multi-core CPUs. A Graphical Processor Unit (GPU) may contain several hundreds of processors and therefore can be used to speed up fluidynamics computations. GPUs have evolved to the point that many real-world applications can be implemented and run significantly faster than multi-core systems. In order to maximize the GPU performance a lot of development efforts are required. The purpose of FEM-LCORE was to use a mutiscale approach to investigate the behavior of the liquid metal reactors. The reactor primary circuit, the plenum and the core may be investigated at three different scale levels. The original plan was to study the reactor with 3D-CFD tools and the rest of the primary circuit with monodimensional system codes. The coupling between the core and the plenum can be studied with the current FEM-LCORE code but the interface with the rest of the primary loop is still under investigation. A first step in this direction might be the coupling of FEM-LCORE with a system code, like CATHARE, to simulate the entire primary loop.

	Sigla di identificazione	Rev.	Distrib.	Pag.	di
	NNFISS-LP3-059	0	L	104	106

## Bibliography

- [1] 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) [51](#), [52](#)
- [2] 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. [40](#), [43](#), [57](#)
- [3] 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. [40](#), [78](#), [90](#), [91](#)
- [4] F. Bassenghi, G. Bornia, L. Deon and S. Manservisi, *Implementation and validation of the NURISP platform*, Technical report CIRTEN (2011) [92](#)
- [5] F. Bassenghi, G. Bornia, A. Cervone and S. Manservisi, *The ENEA-CRESCO platform for simulating liquid metal reactor*, Technical report LIN-THRG 210, (2010) [92](#)
- [6] *ELSY Work Program. European Lead-cooled SYstem (ELSY) Project*, Technical report, EURATOM, Management of Radioactive Waste (2006) [42](#), [43](#), [57](#), [58](#)
- [7] OECD/NEA, *Handbook on Lead-bismuth Eutectic Alloy and Lead Properties, Materials Compatibility, Thermal-hydraulics and Technologies*, OECD/NEA No. 6195, ISBN 978-92-64-99002-9 (2007) [22](#)
- [8] LASPACk (Linear Algebra Sparse Matrix Package):  
<http://www.mgnet.org/mgnet/Codes/laspac/html/laspac.html>
- [9] LIBMESH package: <http://libmesh.sourceforge.net/>
- [10] R. N. Lyon, *Liquid-Metals Handbook*, 2nd ed., Atomic Energy Comm., Washington D.C. (1952) [22](#)
- [11] F.R. Menter, *Zonal Two Equation  $\kappa$ - $\omega$  Turbulence Models for Aerodynamic Flows*, AIAA Paper. pp.93-2906 (1993) [49](#)

	Sigla di identificazione	Rev.	Distrib.	Pag.	di
	NNFISS-LP3-059	0	L	105	106

- [12] F.R. Menter, *Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications*, AIAA Journal, vol. 32, no 8. pp. 1598-1605 (1994) 49
- [13] Sato H., Shimada M., Nagano Y, *A two-equation turbulence model for predicting heat transfer in various Prandtl number fluids*, Proceedings of the Tenth International Heat Transfer Conference, Brighton, UK, Vol.2, p.443-448 (1994) 50, 51
- [14] V.I. Mikhin, Second-order  $\kappa$ - $\epsilon$  turbulence model, The VI International Congress on Mathematical Modeling, Nizhny Novgorod, Russia (2004) 50, 51
- [15] V.I Mikhin, *Two  $\kappa$ - $\epsilon$  turbulence models (first & second order respectively), not containing model functions*, ENEA-RT-FIS; 05-50, ISSN 0393-3016, ENEA, Italy (2005) 50, 51
- [16] V.I Mikhin, Second-order  $\kappa$  -  $\epsilon$  turbulence model not containing model functions, Proceedings of the Fourth Russian National Conference on Heat Exchange, 2006, Moscow. Vol. 2, pp.202-205. 50
- [17] V.I Mikhin, *Turbulence models  $\kappa$  -  $\epsilon$  of first and second orders. Test results*, Proceedings of the Fourth Russian National Conference on Heat Exchange, Vol. 2, pp.206-209, Moscow (2006) 50
- [18] V.I Mikhin, Voukelatou K, *Four-parametric  $\kappa$  -  $\epsilon$  -  $\kappa_\theta$  -  $\epsilon_\theta$  turbulence model for compressible fluids*, ENEA-RT-FPN; ISSN 0393-3016, ENEA, Italy (2007) 50
- [19] H. Schlichting , K. Gersten, *Boundary-Layer Theory*, 8th Revised and Enlarged Edition, Springer (2000) 50
- [20] T. Tsuji and Y. Nagano, *Characteristics of a turbulent natural convection boundary layer along a vertical flat plate*, Int. J. Heat Mass Transfer. Vol. 31, n. 8, pp. 1723-1734 (1988) 50, 51
- [21] T. Tsuji, Y. Nagano, *Turbulence measurements in a natural convection boundary layer along a vertical flat plate*, Int. J. Heat Mass Transfer. Vol. 31, n. 10, pp. 2101-2111 (1988) 50, 51
- [22] MPI (Message Passing Interface) forum, <http://www.mpi-forum.org/> 94
- [23] OpenMPI library, <http://www.open-mpi.org/> 94
- [24] *PARAVIEW visualization software*, Official documentation at <http://www.paraview.org> 34
- [25] PETSc (Portable Extension Toolkit for Scientific Computation), <http://www.mcs.anl.gov/petsc/petsc-as/> 95, 101

	Sigla di identificazione	Rev.	Distrib.	Pag.	di
	NNFISS-LP3-059	0	L	106	106

- [26] CEA/DEN, *SALOME Documentation*, CEA/DEN, EDF R&D, OPEN CASCADE, (2007-2008) 35
- [27] *VTK library*, <http://www.vtk.org> 39
- [28] D.C. Wilcox, *Re-assessment of the scale-determining equation for advanced turbulence models*, AIAA Journal, vol. 26, no. 11, pp. 1299-1310 (1988) 49
- [29] D.C. Wilcox, *Turbulence Modeling for CFD*, ISBN 1-928729-10-X, 2nd Ed., DCW Industries Inc. (2004) 49

### Notes on the Working Group of the University of Bologna

The working group consists of a Researcher in Nuclear Plants from the University of Bologna (DIENCA Department), Sandro Manservisi, by a student enrolled in DIENCA M.D. program, Daniele Cerroni, and by the PostDoc researcher Giorgio Bornia.

Dr. Sandro Manservisi has carried out research activities at the University of Bologna for several years in the field of Nuclear Engineering, with particular reference to thermohydraulics and computational fluidynamics. He is the author of several reports in the past PAR activities regarding the development of the FEM-LCORE code. Daniele Cerroni is taking a MD degree with thesis at the University of Bologna and he works on simulations with the FEM-LCORE code. Dr. Giorgio Bornia recently took his doctorate at the University of Bologna and will be holding a Visiting Professor position at Texas Tech University in the next academic year.

More details and a list of recent publications can be found in the Web site of the University of Bologna (<http://www.unibo.it>).