EDF R&D

Fluid Dynamics, Power Generation and Environment Department
Single Phase Thermal-Hydraulics Group

6, quai Watier
F-78401 Chatou Cedex

Tel: 33 1 30 87 75 40
Fax: 33 1 30 87 79 16

SEPTEMBER 2017

Code_Saturne documentation

Code_Saturne version 4.0 tutorial: Heated Square Cavity Flow

contact: saturne-support@edf.fr

TABLE OF CONTENTS

Chapter I

Introduction

1 Tutorial Components

This tutorial makes use of:

- The SALOME [1] platform for geometry generation, meshing, and post-processing
- *Code_Saturne* [2, 3] for CFD calculations
- References [4, 5, 6] for comparison with published results

To work through this tutorial you will need a computer on which these two software applications are already available or on which you have permission to install them.

You will also need to know how to create and setup a *Code_Saturne* case with the CFDStudy module. For instructions on how to do so, please see [7].

2 Tutorial Structure

This tutorial is made of two complementary parts:

- Part 1 (Section II) illustrates how to setup, conduct and analyse a natural buoyancy CFD simulation with temperature-dependent density using SALOME and *Code_Saturne* and its user coding.
- Part 2 (Section III) illustrates how the user subroutine 'cs_user_extra_operations.f90' may be used to compute additional custom data from the calculated data in order to analyse a simulation and compare its results with published data.

If you are already familiar with setting up CFD simulations with *Code_Saturne* and SALOME and you only want to learn how to compute custom data with 'cs_user_extra_operations.f90', you may skip Part 1 and go directly to Part 2 (Section III).

Chapter II

Part 1 - Heated Square Cavity Flow CFD Study

The preparation, simulation and analysis of the 'Rayleigh1Million' case of the 'HeatedSquareCavity' study are described in section II, from the construction of the computational domain and mesh to the preparation, running and post-processing of the CFD simulation.

A familiarity with Fortran programming would be desirable to understand the user coding in detail, however it is not strictly a requirement as the user subroutines may be used in the form delivered with this tutorial.

1 What You Will Learn

Through this tutorial, you will learn how to perform an end-to-end CFD study using the CFDStudy module of the SALOME [1] platform. The tutorial illustrates how to:

- Create a computational domain using available shapes and groups
- Create an hexahedral mesh with different mesh refinement in the X, Y and Z directions
- Setup a *Code_Saturne* [2, 3] steady-state, viscous, laminar CFD simulation with
  - Buoyancy
  - Heat transfer
  - Variable density
  - Non-slip walls with a combination of imposed heat flux and temperature
  - Symmetry planes
- Model natural convection with the Boussinesq approximation and gravity
- Program user coding to:
  - Model the density as a function of temperature
  - Compute user defined variables and extract quantities of interest for post-processing
- Control and run the *Code_Saturne* simulation from the CFDStudy module
- Examine the *Code_Saturne* output and results files
- Analyse and visualise the results
- Visually and quantitatively compare the case results with available data

2 Case Description

The tutorial refers to the classic natural convection benchmark cases of de Vahl Davies and co-workers [4, 5, 6].

A viscous fluid is contained in a two-dimensional square box, or cavity (Figure II.1). All the walls of the cavity are stationary. Whilst the horizontal walls are isolated and adiabatic (zero heat flux), the vertical walls are kept at constant temperature. The left vertical wall is the hot wall, with a fixed temperature of T_h. The right vertical wall is the cold wall, with a fixed temperature of T_c.

The acceleration of gravity points downward in the vertical (y-axis) direction.

The temperature values form part of the specification of the problem and are specified in Sections 2.3 and 2.5 along with the box's dimension for this tutorial.

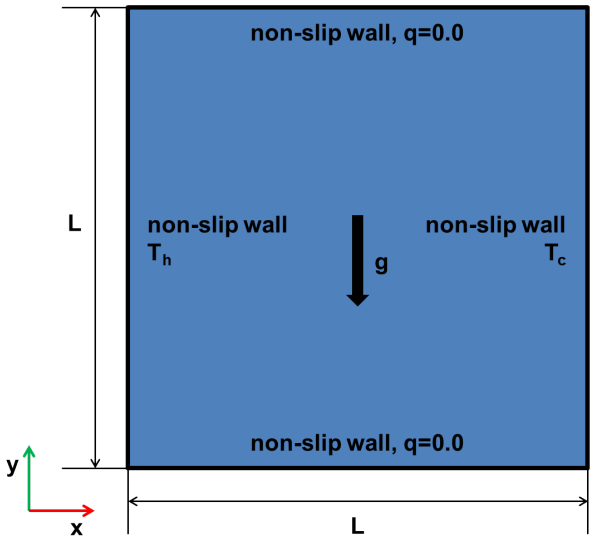

Figure II.1: Schematic of the 2D Heated Square Cavity.

The flow physics are described next.

2.1 Flow Physics

The problem is setup to simulate the natural convection of a fluid in the presence of a body force (gravity) as it changes density with temperature. In this example, a basic pattern is obtained where the fluid rises along the hot wall, is driven by its momentum along the top wall and towards the cold wall where it cools down and moves downwards. More or less complex flow patterns may be obtained depending on the size of the cavity, the fluid properties, and the temperature difference between the two walls.

The dominant physics of the problem being buoyancy and heat transfer, the problem may be characterised in terms of the Rayleigh number based on the cavity size, defined as the product of the Prandtl and Grashof numbers:

$$R_a = P_r G_r = \frac{\mu C_p}{k} . \frac{g\beta(\Delta T)L^3}{\nu^2} = \frac{\rho^2 C_p g\beta(\Delta T)L^3}{k\mu} \tag{II.1}$$

where ρ is the density, C_p is the specific heat at constant pressure, g is the acceleration of gravity, β is the thermal expansion coefficient defined as $\beta = -\frac{1}{\rho}\left(\frac{\partial \rho}{\partial T}\right)_p$, ΔT is the characteristic temperature difference, k is the thermal conductivity, and μ is the viscosity. In this problem, $\Delta T = T_h - T_c$, the temperature difference between the heated walls of cavity.

The flow patterns and characteristics of the heated cavity, such as velocity extremas and heat fluxes, can all be classified as functions of the R_a number. Turbulent flow occurs when $R_a \geq 10^9$.

For the purpose of this tutorial, you will perform laminar flow simulations for $R_a = 10^6$ and compare the results to the laminar flow benchmarks of de Vahl Davis et al. [5, 6]. In this regime, the flow is essentially incompressible. However, variations of density with temperature must be taken into account in order to simulate buoyancy. This combination of an incompressible fluid assumption with a density variation explicit in the gravitational term is known as the Boussinesq approximation [8], which is described next.

2.2 Boussinesq Buoyancy Approximation - Theory

Applying the definition of the thermal expansion coefficient to a perfect gas:

$$\left(\frac{\partial \rho}{\partial T}\right)_p = -\frac{p}{RT^2} = -\frac{p}{T} \quad \longrightarrow \quad \beta = \frac{1}{T} \tag{II.2}$$

With the Boussinesq approximation, the density is calculated as a function of temperature, with respect to the reference state (ρ_0, T_0) :

$$\rho - \rho_0 = -\rho_0 \beta_0 (T - T_0) \qquad \beta_0 = \frac{1}{T_0} \tag{II.3}$$

This density formulation is not already coded in *Code_Saturne*. Instead, you will need to program it in user subroutine 'usphyv' in file 'cs_user_physicalproperties.f90', as described in Sections 2.7 and 6 below.

2.3 Geometry

The cavity is a square of length L = 1.0 (m).

2.4 Fluid Properties

The fluid is given the properties of air, specified at the cold wall temperature, T_c , as listed in Table II.1 below.

Density ($\mathbf{kg/m^3}$)	**Viscosity** (**Pa.s**)	**Specific Heat** (**J/kg.K**)	**Thermal Conductivity** (**W/m.K**)
1.2039	1.83×10^{-5}	1004.84	0.0259

Table II.1: Fluid Properties.

The properties are calculated for a perfect gas at the reference state of $T = 293.15K$ and $P = 101325Pa$, with $\gamma = 1.4$ and to ensure that $P_r = 0.71$, as per the benchmark specifications.

2.5 Boundary Conditions

As described in above, the domain is fully enclosed by non-slip walls. This means that, exactly at the surface of the walls, the fluid inside the box attaches to the walls and has exactly the same velocity as the walls. Therefore, for the momentum equations, the problem is fully defined by specifying that all the walls are stationary. For the energy equation, the two vertical walls are kept at constant temperature whilst the horizontal walls are adiabatic. The boundary conditions are summarised in Table II.2 below.

Wall	Velocity component(m/s)		Energy	
	$\mathbf{v_x}$	$\mathbf{v_y}$	$\mathbf{T(K)}$	$\mathbf{Q(J/m^2)}$
Left	0.0	0.0	303.15	N/A
Top	0.0	0.0	N/A	0.0
Right	0.0	0.0	293.15	N/A
Bottom	0.0	0.0	N/A	0.0

Table II.2: Wall boundary conditions.

The temperature difference between the two walls is kept under 20 (K) which ensures the validity of the Boussinesq approximation.

2.6 Flow Regime

For this tutorial, the setup is chosen so that $R_a = 10^6$, for which the flow is steady and laminar . This Rayleigh number corresponds to the highest Rayleigh number reported in [5, 6] making comparisons with benchmark results possible and ensuring significant fluid motion in the cavity and complex flow patterns.

2.7 Boussinesq Buoyancy Approximation – *Code_Saturne* Formulation

As the Boussinesq approximation is not directly available in *Code_Saturne*, it will need to be implemented with user coding. The actual programming of user subroutine 'usphyv' for this purpose is described in Section 6. Here, we describe the formulation chosen for this tutorial.

The reference state is taken at the cold wall temperature. Therefore, the density law becomes:

$$\rho = \rho_c(1 - \beta(T - T_c)) \tag{II.4}$$

Further, in order to make the simulations flexible, in this tutorial we make β a variable of the imposed Rayleigh number, instead of using the physical definition of β as derived for a perfect gas:

$$\beta = \frac{k\mu}{\rho^2 C_p g(T_h - T_c)L^3} R_a \tag{II.5}$$

This way, it will be possible to simulate the flow for different Rayleigh numbers just by changing the value of the imposed Rayleigh number in the user subroutine and without altering the geometric or physical setup.

3 Creating the *Code_Saturne* case

The 'HeatedSquareCavity' study and 'Rayleigh1Million' case are created using the procedure detailed in Part I of tutorial 1 [7]. Start SALOME with the command '*code_saturne salome*', select the CFD-Study module, and go through all the steps detailed in Tutorial 1, Part 2 to:

- Create the CFD case structure with the CFDStudy module
- Save the new file as 'HeatedSquareCavity'

- Customise the background settings

The next sections describe how to setup and run the heated cavity simulation for a Rayleigh number of 10^6.

4 Creating the Computational Domain

The computational domain is created with the GEOM module of SALOME. As the geometry is the same as for tutorial 1 [7] (shear driven 2D cavity), in what follows the geometry creation will be referring to this tutorial and only the elements which differ from the setup of Tutorial 1 will be highlighted. Therefore, if you are not already familiar with GEOM, you should first go through Tutorial 1 [7].

4.1 Geometry

To create the geometry, open the GEOM module and go through all the steps detailed in Tutorial 1, Part 2 to:

- Create a thin box of dimensions 1.0 m x 1.0 m x 0.01 m
- 'Explode' the box into its constituting faces and edges

Next, group together the faces which make up surfaces on which the same boundary conditions will be applied following the steps described in Tutorial 1, Part 2 [7].

Here, two groups of faces are built for the adiabatic walls ('Adiabatic_wall', faces 2 and 5) and for the symmetry planes ('Symmetry_plane', faces 3 and 4) (Figure II.2). To identify them more easily, rename the left vertical face (face 1) 'Hot_wall' and the right vertical face (face 6) 'Cold_wall'. As a final preparation step before meshing, group and rename the edges according to the main X, Y, and Z directions (Figure II.2).

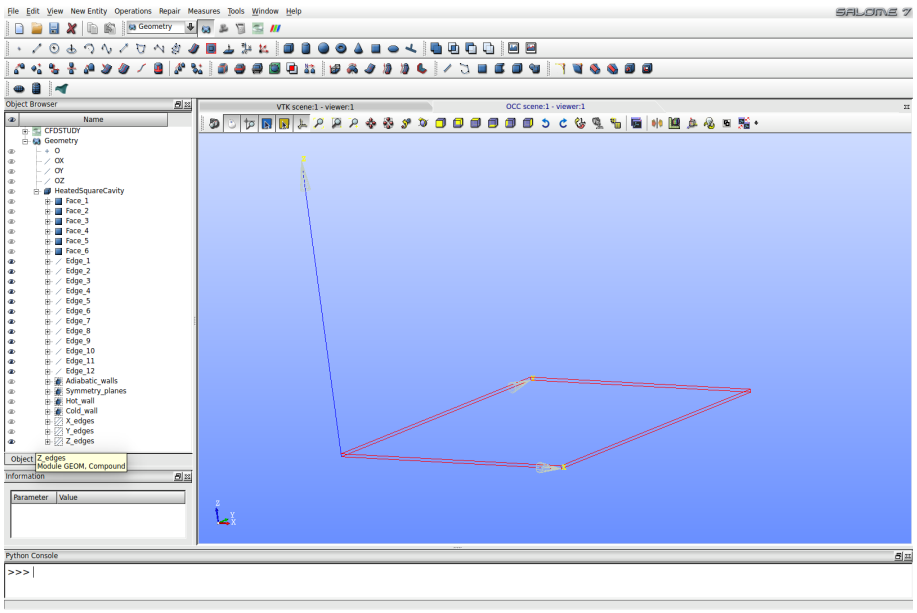

Figure II.2: Grouping and renaming faces and edges.

Remember to save the file at regular intervals during this tutorial to preserve your work before making changes.

'HeatedSquareCavity' is now ready to be meshed. Save the file and proceed to Meshing.

4.2 Meshing

Switch to the meshing module, which opens the 'VTK scene viewer'. Create the 'Mesh' and rename it 'HSC_mesh'. Similarly to Tutorial 1 [7], create a uniform, rectangular mesh with one cell in the Z direction by following the methodology described in Part 2 of Tutorial 1 [7]. However, this time specify 80 cells for the discretisation along the 'X' and 'Y' edges, as shown in Figure II.3 below.

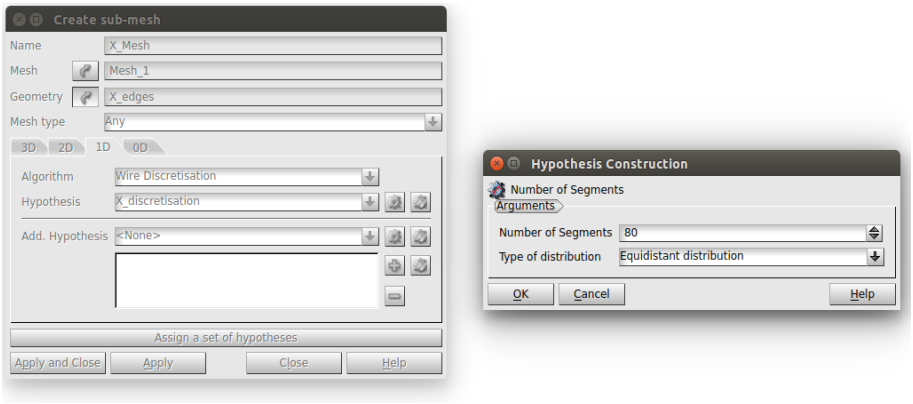

Figure II.3: Edge discretisation for the edges along the X and Y directions.

When all the sub-meshes have been specified, right-click on 'HSC_mesh' and select 'Compute' to generate the mesh. The HSC_mesh is now created and the relevant mesh faces have automatically been ascribed to the previously created face groups. Two more groups need to be added directly into the mesh in order to complete the definition of the boundary conditions on the mesh.

Right-click on 'HSC_mesh' and select 'Create Groups from Geometry' from the drop-down menu. Under 'Elements' click on the arrow icon next to 'Geometry'. Then, using the cursor, in the 'Heated-SquareCavity' object in the 'Geometry' list, left-click on 'Hot_wall' (Figure II.4). Repeat the process for the 'Cold_wall' faces.

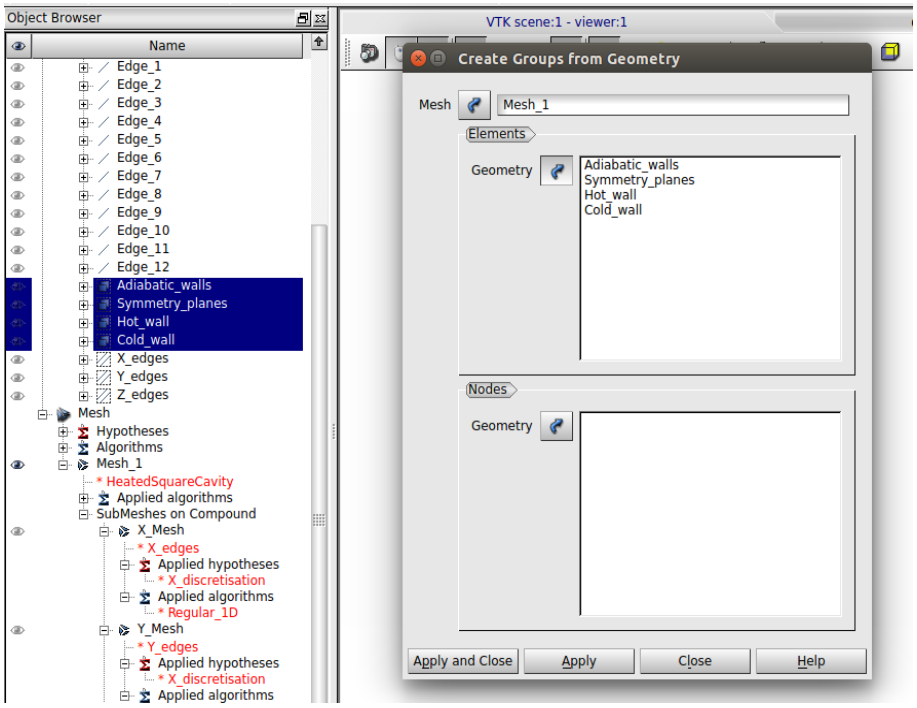

Figure II.4: Creating the mesh face group 'Hot_wall'.

The mesh is now completed and the different boundaries may be assigned different colours for visualisation, as described in Tutorial 1, Part 2 [7].

Save the SALOME file and export the mesh file in '.med' format as per the methodology detailed in Tutorial 1, Part 2 [7]. The file should be placed in the 'MESH' directory of the 'HeatedSquareCavity' study, where *Code_Saturne* will expect the file to be situated by default.

For the file name, choose 'HSC_mesh'; the '.med' extension is automatically added. You are now ready to set up the CFD simulation with the CFDStudy module.

5 Setting up the CFD Simulation

The CFD case is setup using the CFDStudy module. Go back into CFDStudy using either the drop-down menu or the module's icon in the tool bar. Next, open the *Code_Saturne* GUI, either by pressing on the *Code_Saturne* GUI button in the toolbar or by right-clicking on the case in the object browser and selecting 'Launch GUI'. These steps are detailed in Tutorial 1, Part 2, Section 5 [7].

In the *Code_Saturne* GUI, create a new *Code_Saturne* xml file with 'New File'. Verify that the case directory structure has been recognised and save the file as 'Rayleigh1Million'.

You can now proceed with setting up the *Code_Saturne* case, in the top down order of the folders in the left-hand side, starting with the mesh.

In the 'Meshes' tab of the 'Meshes Selection' folder, add the 'HSC_mesh.med' to the initially empty list of meshes (Figure II.5).

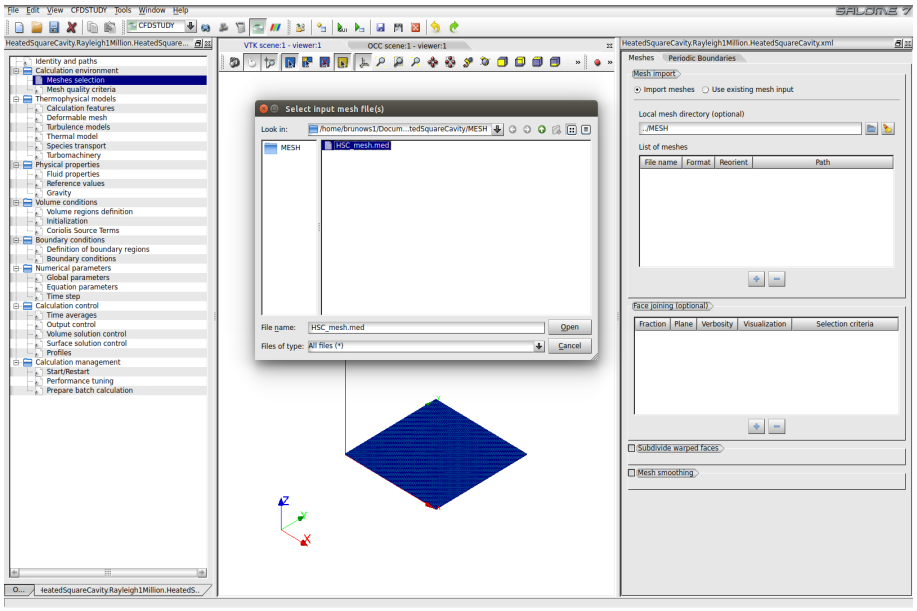

Figure II.5: Selecting the 'HSC_Mesh.med' file for the calculations.

No further input is necessary. Next, move to 'Thermophysical Models' to specify the flow physics for the calculations.

In the 'Calculation features' folder, change the algorithm to 'steady flow' in the drop down menu at the top. Leave all the other default values unchanged: multiphase flow, atmospheric flows, combustion and the electrical and compressible models are all inactive.

Next, change 'Turbulence models' to 'No model' for this laminar flow case.

In 'Thermal model', choose 'Temperature (Kelvin)' for the 'Thermal scalar'. This will activate solution of the energy equation and designate Temperature as the scalar specified at the boundary conditions (Figure II.6).

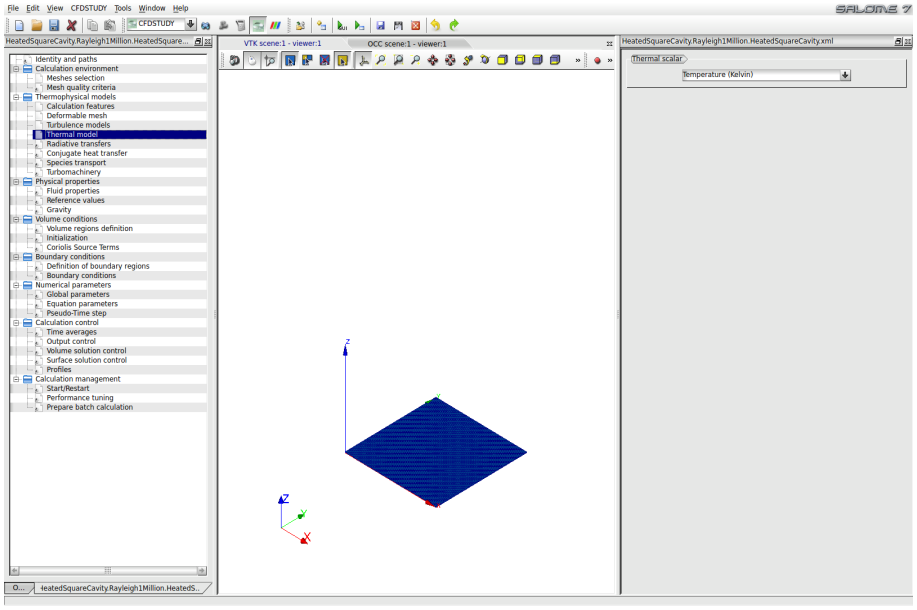

Figure II.6: Activating solution of the energy equation.

Move to the 'Physical Properties' folder where the 'Fluid Properties' must be modified for our imaginary fluid. Enter the values from Table II.1 above in the text boxes (Figure II.7). For density, specify that it will be calculated according to the formulation that you will program in user subroutine 'cs_user_physical_properties'.

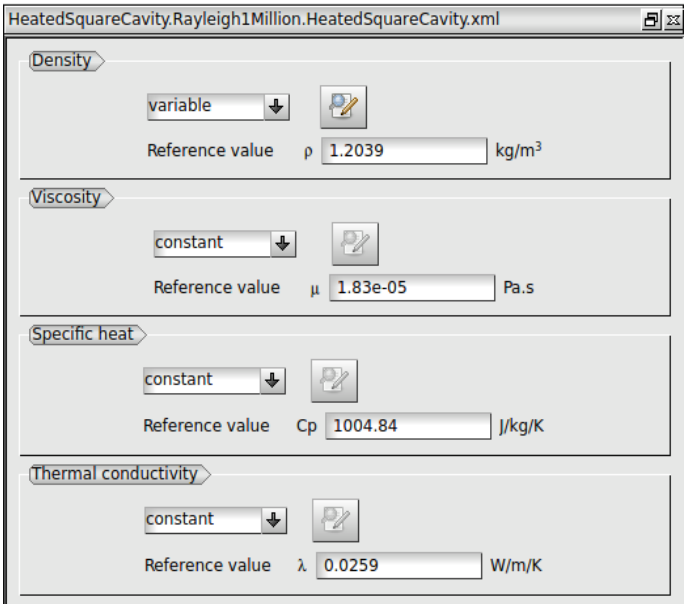

Figure II.7: Selecting the fluid properties.

Next, activate the acceleration of gravity by entering the value magnitude '$-9.81 m/s^2$' for its component in the vertical (Y) direction (Figure II.8).

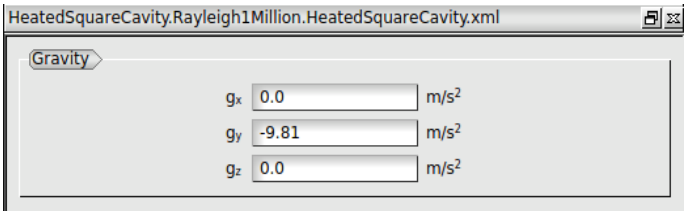

Figure II.8: Specifying gravity and hydrostatic pressure.

The initial values for the velocity and temperature are defined in the 'Initialization' tab of the 'Volume conditions' folder. The flow is initially stagnant by default. To set the initial temperature, click on the Mathematical Expression Editor button marked 'Thermal' and enter the temperature of the cold wall (Table II.2) in the pop-up editor panel (Figure II.9).

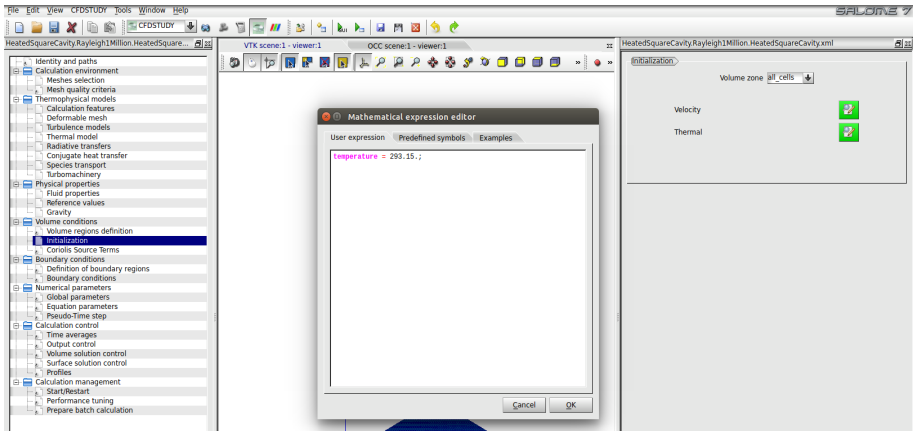

Figure II.9: Specifying the initial temperature.

Under the 'Boundary Conditions' folder, select 'Definition of Boundary Regions' and then press 'add' three times to add three boundary conditions which are given the 'Wall' type by default. Change the name of each wall to reflect exactly the name of each of the walls in the mesh: 'Hot_wall', 'Cold_wall', and 'Adiabatic_wall'. Click on the 'Add' button once more to define the remaining boundary condition. Change the latter's type to 'Symmetry' in the 'Nature' drop-down menu for the boundary and type in 'Symmetry_plane' as its name in 'Selection Criteria'. The boundary regions are now fully defined (Figure II.10).

HeatedSquareCavity.Rayleigh1Million.HeatedSquareCavity.xml

Definition of boundary regions

Label	Zone	Nature	Selection criteria
BC_1	1	Wall	Adiabatic_walls
BC_2	2	Wall	Cold_wall
BC_3	3	Wall	Hot_wall
BC_4	4	Symmetry	Symmetry_planes

Figure II.10: Defining the boundary conditions.

Where required depending on the boundary type, the conditions applied on the region must also be specified under the 'Boundary Conditions' sub-folder of 'Boundary Conditions'. Boundaries of type 'Symmetry' do not require further specification. Therefore, they are not listed in this folder. However, 'Walls', may also be given a surface roughness, velocity (sliding wall) and thermal characteristics (either heat flux or temperature).

By default, 'Walls' are assumed to be smooth, stationary, and adiabatic. Therefore, no further changes are required for boundary 'Adiabatic_wall', as shown in Figure II.11(left) below. For the heated vertical walls, click on each wall in succession, specify 'Prescribed value' for the type and enter the imposed temperature in 'Value' according to Table II.2 and as illustrated for the 'Hot_wall' boundary in Figure II.11(right) below.

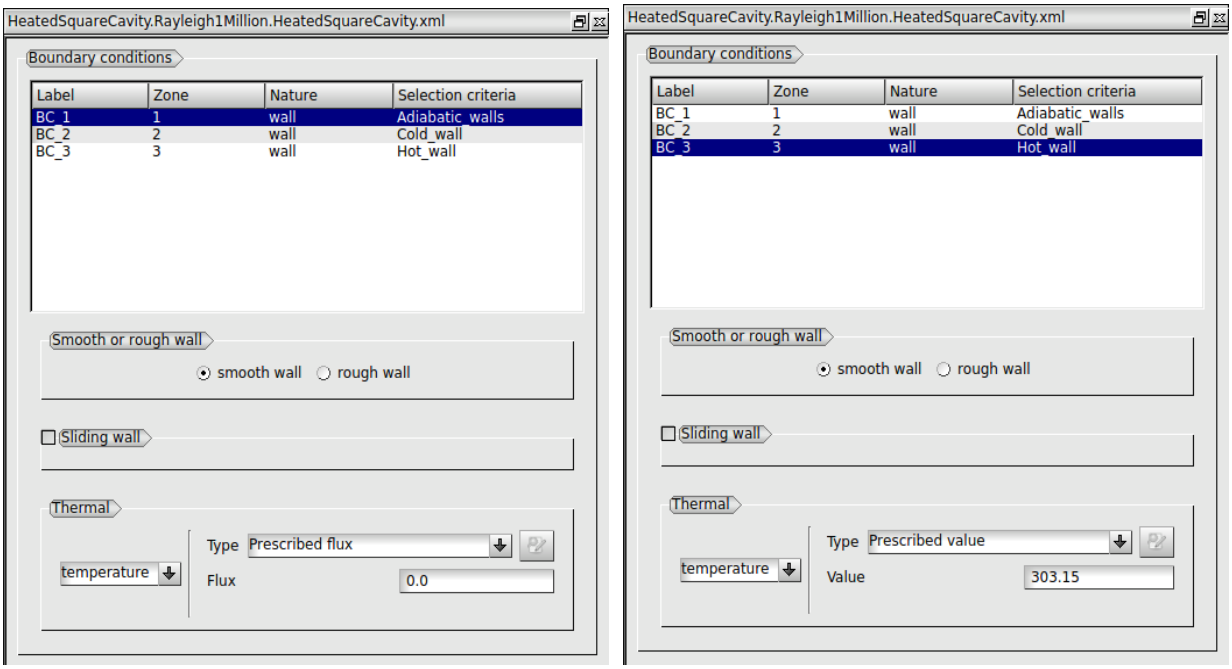

Figure II.11: Specifying the wall boundaries. Adiabatic wall (left), Hot wall (right).

Having set up the mesh and physics of the problem, the numerical parameters may now be specified.

In the 'Numerical Parameters' folder, leave all the default settings in the 'Global parameters' sub-folder, including the SIMPLEC Velocity-Pressure algorithm. Move to the 'Equation parameters' folder. The 'Solver' tab shows that pressure, velocity and temperature are solved for. Click on the 'Scheme' tab. For better convergence, and as the flow in the cavity will involve small temperature and density variations, keep the 'Centered' scheme for all variables but unselect the 'Slope test' (Figure II.12).

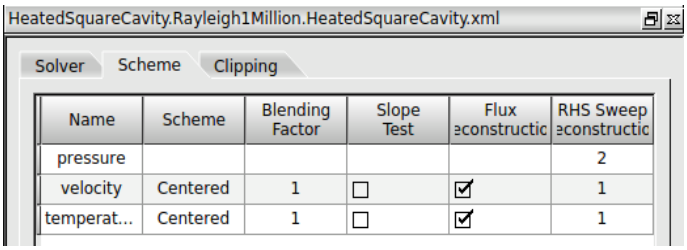

Figure II.12: Defining the numerical schemes.

The 'Clipping' tab is used to set the temperature bounds, making it possible to instruct the code to clip the temperature to these values if it strays outside the defined range. This feature can be very useful to prevent spurious, low or high values, from poisoning a calculation, for example when convergence has not yet been achieved. However, it is not required for this tutorial and you may leave the default values unchanged.

Move down the selection tree to the 'Pseudo-Time step' folder and increase the 'Number of iterations' to 450 and the 'Maximal CFL number' to 8.0.

In the 'Calculation control' folder, select 'Surface solution control' and unselect 'Yplus', 'Efforts', and 'Tplus' from the list of variables to output on surfaces as they are not relevant to this tutorial. Note that 'Thermal flux' and 'Boundary temperature' have been automatically added to the list when the thermal model was activated. Add 'Boundary layer Nusselt' to the selection (Figure II.13) and save

the file.

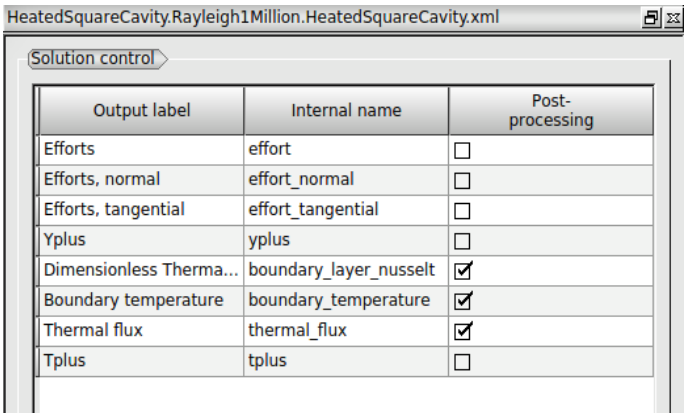

Figure II.13: Defining the output variables on surfaces.

The analysis of the wall and domain data will be done in user coding as described in section III below and no further specification of 'Time averages', 'Output control' and 'Profiles' is required.

The *Code_Saturne* calculation is now fully specified from the standpoint of the GUI and the file should be saved again. However, before you can run the simulation, the density law must be programmed in the user coding file 'cs_user_physicalproperties.f90'. This step is described next.

6 Programming the Boussinesq Model with User Coding

To begin with, copy the sample file 'cs_user_physical_properties.f90' from the tutorial's /../HeatedSquareCavity/Rayleigh1Million/SRC/REFERENCE directory to the SRC directory itself in order to create a local copy which you will be able to customise and which will be automatically recompiled and linked with the 'cs_solver' executable at run time.

Then, open your local version of the file in your editor of choice.

The files contain a number of subroutines. The specification of a physical property is done in subroutine 'usphyv'. Scrolling through this subroutine, you can see that several examples of implementations for different physical properties are available. In this tutorial, you are going to amend 'Example 1' with your own implementation of the density as a function of temperature (Eq. 3.4). For clarity, you may remove all the other examples from the file. The customised code available with the tutorial is already commented. Here we describe the main parts and the logic behind them.

1. Declare your own local variables at the top, as double precision real values or integer.
2. Activate the example (replace '.false.' with '.true.')
3. Specify the desired Rayleigh number for the simulation
4. Compute the magnitude of the gravity vector from the information entered in the CFDStudy module
5. Impose the domain size and wall temperatures consistently with the problem setup
6. Compute the expansion coefficient for the physical properties entered in the CFDStudy module and the imposed Rayleigh number
7. Cycling through the internal cells of the computational domain:

(a) Compute the density as a function of temperature, using the cold wall temperature and density as the reference state

(b) Compute the density maxima and minima for output purposes

8. Write the values of the density maxima and minima to the output file 'listing' with the keyword 'usphyv' for easy identification

7 Running and Analysing the Simulation

In the folder 'Calculation management', go directly to the 'Prepare batch calculation' sub-folder. By default, the calculation-restart is deactivated in 'Start/Restart' and so this sub-folder does not need to be visited. In the 'Prepare batch calculation' sub-folder panel (Figure II.14), the default selections of 'runcase' for the 'Script file', 'standard' (straight calculations without mesh or partitions import or pre-processing) and '1' processor for the 'Calculation script parameters' are already set.

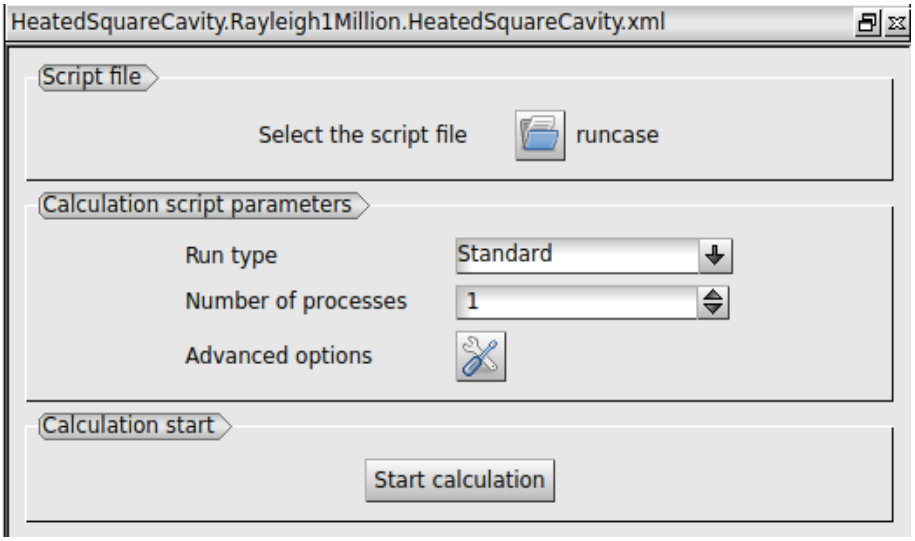

Figure II.14: Batch calculation settings.

Press the 'Start calculation' button to run *Code_Saturne*. The pop-up panel for the run opens, listing in real time the different stages of the calculation, from user-subroutines compilation to saving the results.

Wait for the calculations to complete and open the 'listing' file in your 'Rayleigh1Million/RESU/DateOfRunTimeOfRun/' directory. Verify that the residuals listed under 'derive' in the 'Information on Convergence' table have dropped several order of magnitudes for all the solved variables (pressure, velocity, temperature), showing that the calculations have fully converged to a steady-state.

```
   ** INFORMATION ON CONVERGENCE
      --------------------------
-------------------------------------------------------------------------------
   Variable    Rhs norm      N_iter  Norm. residual      derive Time residual
-------------------------------------------------------------------------------
c  Velocity     0.29892E-13       0   0.00000E+00   0.00000E+00 0.00000E+00
c  Velocity[X]                                      0.00000E+00
c  Velocity[Y]                                      0.00000E+00
c  Velocity[Z]                                      0.00000E+00
c  Pressure     0.00000E+00       0   0.00000E+00  -0.00000E+00 0.00000E+00
c  TempK        0.40330E+03       4   0.31120E-09   0.94582E-02 0.10003E-02
-------------------------------------------------------------------------------
```

Figure II.15: *Code_Saturne* Convergence history from the 'listing' file, after 1 iteration.

```
   ** INFORMATION ON CONVERGENCE
      --------------------------
-------------------------------------------------------------------------------
   Variable    Rhs norm      N_iter  Norm. residual      derive Time residual
-------------------------------------------------------------------------------
c  Velocity     0.13226E-07      88   0.54113E-06   0.18651E-17 0.17772E-06
c  Velocity[X]                                      0.10527E-17
c  Velocity[Y]                                      0.81242E-18
c  Velocity[Z]                                      0.46235E-48
c  Pressure     0.23505E-10      87   0.89656E-08   0.29538E-05 0.15469E-06
c  TempK        0.27688E+01      28   0.91219E-08   0.13607E-10 0.24400E-08
-------------------------------------------------------------------------------
```

Figure II.16: *Code_Saturne* Convergence history from the 'listing' file, after 450 iterations.

Scanning through the listing file for the 'usphyv' output, inspect how the density has changed during the calculations, to verify that the user-coding has been visited and validate your implementation. As required, the density has decreased as a consequence of the temperature of the fluid increasing next to the hot wall.

```
 INSTANT    0.100000000E+00   TIME STEP NUMBER               1
 ============================================================


 USPHYV: rho_min =    1.2038999999756232      , rho_max =    1.2038999999756232
```

Figure II.17: *Code_Saturne* Density extremas history from the 'listing' file, after 1 iteration.

```
 INSTANT    0.450000000E+02   TIME STEP NUMBER             450
 ============================================================


 USPHYV: rho_min =    1.2038603236819787      , rho_max =    1.2038997567079919
```

Figure II.18: *Code_Saturne* Density extremas history from the 'listing' file, after 400 iterations.

You can now proceed with examining and post-processing the results by returning to the SALOME platform.

8 Post-processing the Results

The post-processing is performed with the ParaView [9] visualisation package, which is integrated in the ParaVis SALOME module. To start the analysis, select the ParaVis module from the drop-down module selector in the top menu bar. The name of the module will add itself to the Object Browser list and the ParaView-specific panels and menus will be activated, including a new 'ParaView scene viewer' window.

Before loading the run data in ParaView, follow the steps described in Tutorial 1, Part 2 [7] to modify the default colour schemes.

In the 'Pipeline Browser' panel on the left-hand side, right click and select 'Open' in the drop-down menu. Point to the 'RESULTS.case' file in the RESU directory for the run you have just concluded:

/../Rayleigh1Million/RESU/DateOfRunTimeOfRun/postprocessing/RESULTS.case.

The new Paraview object 'RESULTS.case' will now become visible in the 'Pipeline Browser'. Press 'Apply' in its 'Object Inspector' panel in order to make it active. Next, apply the 'Extract Block' filter to 'RESULTS.case' in order to access the fluid domain data and the boundary data. With the new object 'ExtractBlock1' highlighted, choose 'Fluid domain' in the 'Block Indices' category of the 'Properties' tab. You may then press 'Apply' to load the calculated fluid domain data.

To visualise the results and compare them qualitatively to the benchmarks of [5, 6], you will need to create isoline contours of temperature and streamlines which show the flow patterns. To create these plots, the data output by *Code_Saturne* given at the centres of the cells first needs to be interpolated to the cell vertices. As described in Tutorial 1, Part 2 [7], with 'ExtractBlock1' selected, in 'Filters' at the top of the Paraview window, select the 'CellDatatoPointData' filter, which will create the 'CellDatatoPointData1' object. Click on the object and select 'Contours' in the list of filters. Again this will create a new object based on its originator, this time name 'Contour1' (Figure II.19).

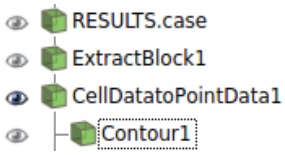

Figure II.19: The chain of Paraview visualisation objects.

To create the temperature isolines, click on 'Contour1' and click on 'Apply' in its 'Object Inspector' 'Properties' tab. Choose 'TempK' in the 'Contour' category as shown in Figure II.20 below.

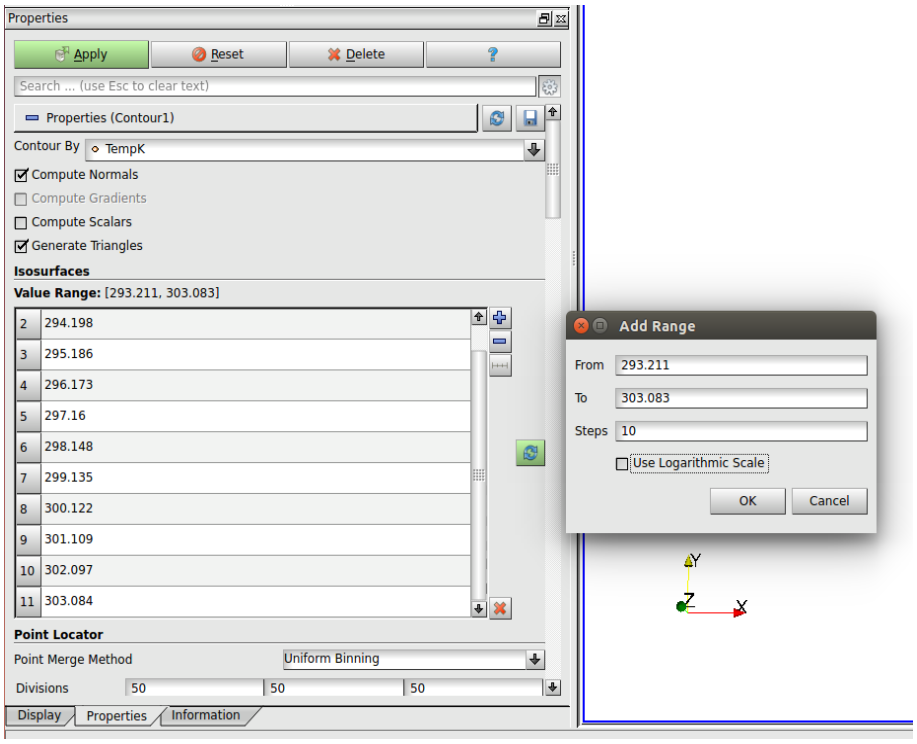

Figure II.20: Specifying the temperature isolines in the Contour filter.

In the 'Isosurfaces' category, press 'Delete All' to reset the contour levels. Then press 'New Range' and specify 11 levels in the pop-up window in order to display the same number of isolines as in the plots published in [5], [6]. The correct value range should already be set in the panel. For better rendering, the colour scale should be changed to the 'Blue to Red' 'HSV' in the 'Display' tab, as per the instructions of Tutorial 1, Part 2 [7]. Add the colour scale to the plot by clicking on its icon on the left of the top Paraview menu bar.

To complete the image, we want to superimpose the contour lines on top of the computational domain. Select 'CellDataToPointData1' and 'Solid Color' for the display. Change the representation from 'Surface' to 'Outline' to visualise the cavity's perimeter. Make sure that the icon in the form of an eye to the left of 'CellDataToPointData1' is active for the object to be visible. If the isolines have become invisible, unclick on the eye icons on the 'Contour1' and 'CellDatatoPointData1' objects, then make them visible again starting with 'CellDatatoPointData1' to keep it in the background.

A plot of 9 visible isolines of temperature is then obtained which compares very well with the published results of [5], [6] (Figure II.21). If required, reset the temperature scale manually for the bounds to match exactly the range of the isolines (Figure II.20).

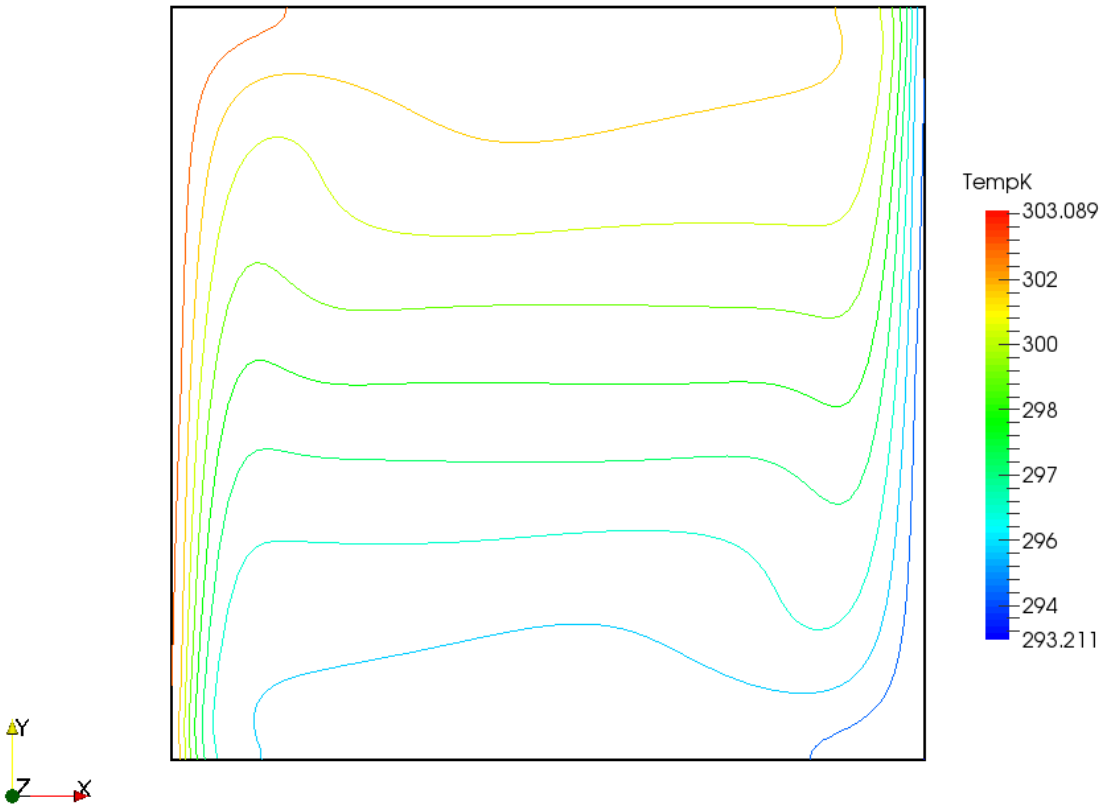

Figure II.21: Temperature isolines.

Next, prepare a visualisation of the streamlines which can also be compared directly with the benchmark results. The procedure to generate streamlines superimposed on a view of the computational domain has already been described in Tutorial 1, Part 2 [7] and only the features specific to this tutorial are detailed here.

With 'CellDatatoPointData1' selected in the Pipeline Browser, choose 'Steam tracer' from the 'Filters' list. Select the 'StreamTracer1' object in the Pipeline Browser and make it visible. In the 'Object Inspector', select the 'Properties' tab and modify the default settings for 'Seeds'. Change 'Point Source' to 'Line Source'. The diagonal line showing the location of the streamline seeds becomes visible in the visualisation window. For the purpose of post-processing, unselect 'Show Line' under 'Seed Type'. In the 'Display' tab, choose 'Velocity' and 'Magnitude' in the 'Color' sub-category. The streamlines show the expected flow symmetry and very good agreement with the published benchmarks (Figure II.22).

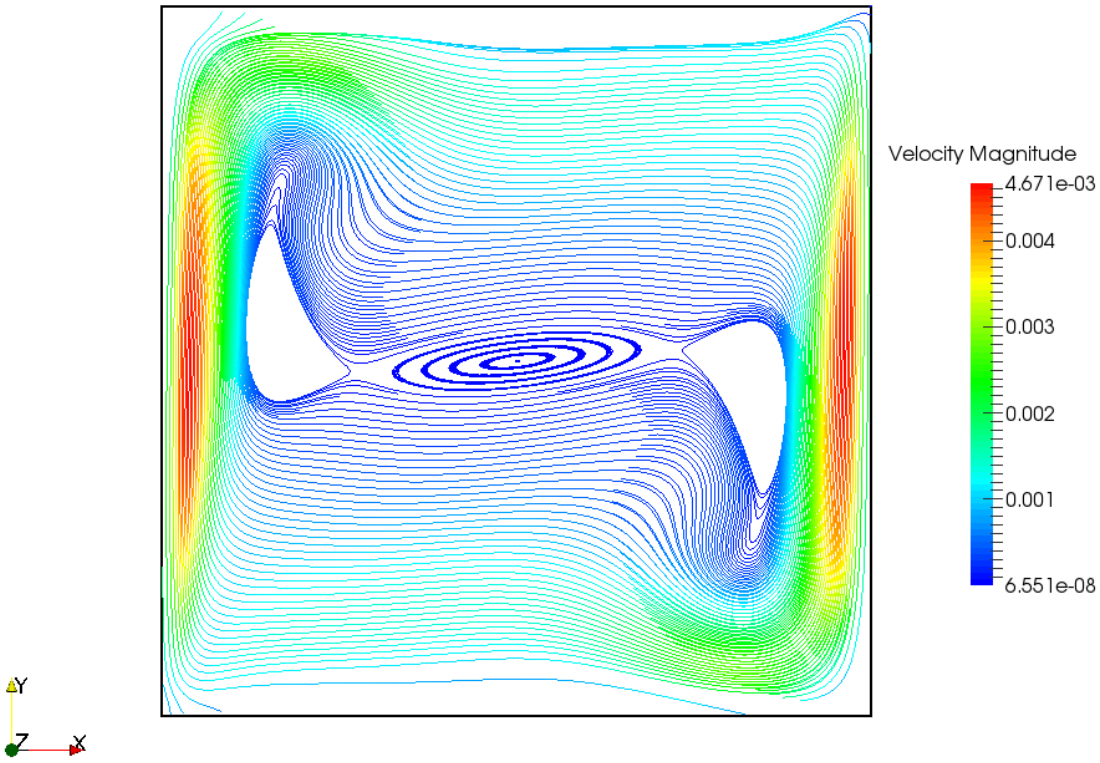

Figure II.22: Streamlines coloured by velocity magnitude.

Chapter III

Part 2 - Data Analysis with user subroutine

The first part of the tutorial describes how to setup the heated cavity case so that it can be compared directly to published benchmark results. In the second part of the tutorial, we illustrate how users may introduce their own coding in order to calculate additional data of specific relevance to their case based on variables internal to *Code_Saturne* and whilst the calculations are running.

1 What you will learn

Through section III, you will learn how:

- *Code_Saturne* gives access to all the internal variables which are used in the calculations, whether related to the domain geometry, such as the cells' positions, or the physical properties or the primitive variables for the calculations, such as density or velocity
- User subroutine 'cs_user_extra_operations.f90' provides an entry point to use or modify these variables at each iteration or time step
- 'cs_user_extra_operations.f90' may be programmed to calculate new, custom variables which are of significance to your case based on the code's internal variables
- To format and add the results to the 'listing' file for clean outputs
- To post-process the output to produce files which can be used easily to produce line plots

2 Comparing the Results with Available Data

We would now like to perform quantitative comparisons with the benchmark results [5, 6] and compare the maximum x and y velocities at the half-way planes and the Nusselt numbers computed at the hot wall and in the vertical mid-plane to the published values. Computing these quantities at each time step will also give us a good indication of the convergence of the calculations.

For $R_a = 10^6$, de Vahl Davis et al. [5, 6] report:

- $\widetilde{U}_{x,max} = 64.63$ at $\widetilde{x} = 0.5$ and $\widetilde{y} = 0.85$
- $\widetilde{U}_{y,max} = 219.36$ at $\widetilde{x} = 0.0379$ and $\widetilde{y} = 0.5$
- $N_{u,0} = 8.817$ at $\widetilde{x} = 0.0$, integrated from $\widetilde{y} = 0.0$ to $\widetilde{y} = 1.0$
- $N_{u,\frac{1}{2}} = 8.799$ at $\widetilde{x} = 0.5$, integrated from $\widetilde{y} = 0.0$ to $\widetilde{y} = 1.0$

where the symbol $\widetilde{}$ has been used to denote non-dimensional quantities. In [5, 6], these are defined as follows:

$$\widetilde{T} = \frac{T-T_c}{T_h-T_c}, \qquad \widetilde{u} = \frac{uL}{\kappa} = \frac{\rho C_p}{k} uL, \qquad \widetilde{x} = \frac{x}{L}, \qquad \widetilde{y} = \frac{y}{L}$$

In the calculations quoted by [5], [6], the full Boussinesq model is used in which the density is assumed to be constant and its variation with temperature is taken into account only in the buoyancy term. For consistency, we use the density at the cold wall temperature as the reference value and specify the non-dimensional velocity vector as: $\widetilde{u} = \frac{\rho_c C_p}{k} uL$.

The Nusselt number, N_u, is evaluated along vertical planes from the definitions:

$$N_u = \int_0^1 \widetilde{Q}(\frac{1}{2}, \widetilde{y}) \mathrm{d}\widetilde{y} \qquad \widetilde{Q} = \widetilde{u_x}\widetilde{T} - \frac{\partial \widetilde{T}}{\partial \widetilde{x}} \tag{III.1}$$

Expressed in dimensional variables and applied at the mid-plane ($\widetilde{x} = \frac{1}{2}$), this definition becomes:

$$\begin{aligned} N_{u,\frac{1}{2}} &= \int_{y=0}^{y=L} (\frac{\rho_c C_p}{k} L u_x \frac{T - T_c}{T_h - T_c} - \frac{L}{T_h - T_c} \frac{\partial T}{\partial x}) \frac{\mathrm{d}y}{L} \\ &= \frac{1}{T_h - T_c} \int_{y=0}^{y=L} (\frac{\rho_c C_p}{k} u_x (T - T_c) - \frac{\partial T}{\partial x}) \mathrm{d}y \end{aligned} \tag{III.2}$$

In discretised form on the computational mesh, the integral in Eq.III.2 will be calculated as a sum of piecewise constant values on all the mesh faces and over the height of the mid-plane.

$$\begin{aligned} N_{u,\frac{1}{2}} &= \frac{1}{T_h - T_c} \sum_{y=0}^{y=L} (\frac{\rho_c C_p}{k} u_x (T - T_c) - \frac{\partial T}{\partial x})_f \Delta y \\ &= \frac{1}{T_h - T_c} \sum_{y=0}^{y=L} (\frac{\rho_c C_p}{k} u_x (T - T_c) - \frac{\partial T}{\partial x})_f \frac{S_f}{\Delta z} \\ &= \frac{1}{(T_h - T_c)\Delta z} \sum_{y=0}^{y=L} \left[S_f \left\{ \frac{\rho_c C_p}{k} (u_x)_f (T - T_c)_f - (\frac{T_{if2} - T_{if1}}{x_{if2} - x_{if1}})_f \right\} \right] \end{aligned} \tag{III.3}$$

where the subscript f indicates a face value, $F_{m,x} = (\rho \overline{u} \bullet \overline{S_f}) n_x$ represents the mass flux through the cell face in the X direction, and Δz is the mesh thickness in the Z direction. The temperature derivative is evaluated with a central-difference formulation across the two cells on each side of the interface, as schematised in Figure III.1 below.

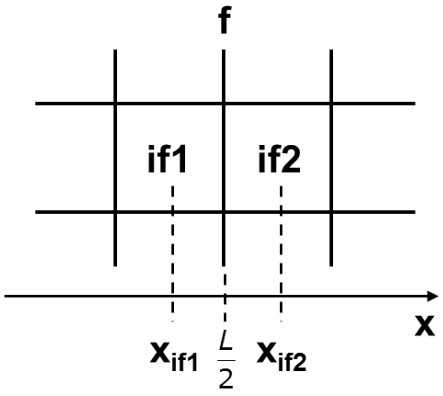

Figure III.1: Mesh cell face schematic.

The density, temperature, and velocity at the cell faces are not stored quantities in *Code_Saturne*. Therefore, they must be calculated. In this tutorial which makes use of a uniform mesh, the properties are calculated as means across the cell face whilst the velocity is extracted from the face density and the mass flux:

$$\rho_f = \frac{\rho_{if2} + \rho_{if1}}{2} \qquad U_{x,f} = \frac{(\rho \overline{u} \bullet \overline{S_f}) n_x}{\rho_f} = \frac{F_m n_x}{\rho_f} \tag{III.4}$$

The mass flux through the face, $F_{m,x} = (\rho \overline{u} \bullet \overline{S_f}) n_x$, is one of the stored variables at boundary and internal faces which are available in 'cs_user_extra_operations.f90'.

For the Nusselt number along the hot wall, the expression simplifies to:

$$\begin{aligned} N_{u,\frac{1}{2}} &= \frac{1}{T_h - T_c} \int_{y=0}^{y=L} (-\frac{\partial T}{\partial x}) \mathrm{d}y \\ &= \frac{1}{(T_h - T_c)\Delta z} \sum_{y=0}^{y=L} \left[S_f \left\{ -(\frac{T_i - T_b}{x_i - x_b}) \right\} \right] \\ &= \frac{-1}{(T_h - T_c)\Delta z} \sum_{y=0}^{y=L} \left[S_f \left\{ \frac{T_i - coefap(b)}{x_i - x_b} \right\} \right] \end{aligned} \tag{III.5}$$

The temperature at each boundary face is computed based on *Code_Saturne* 's 'coefap' boundary conditions definition array.

The implementation of the formulae in the user coding is described next.

3 Customising 'cs_user_extra_operations.f90'

Similar to the steps described in Section II for 'usphyv.f90', start by copying:

/HeatedSquareCavity/Rayleigh1Million/SRC/REFERENCE/cs_user_extra_operations.f90

to your local SRC directory:

/HeatedSquareCavity/Rayleigh1Million/SRC/.

to create a local copy which you will be able to customise and which will be automatically recompiled and linked with the 'cs_solver' executable at run time.

The customised code available with the tutorial is already commented. Here we describe the main parts and the logic behind them.

1. Declare your own local variables at the top, as double precision real values or integer.
2. Collect information about the case, geometrical and physical. This makes the subroutine setup independent and flexible:
   (a) Identify wall faces
   (b) Find their position and the temperature which has been ascribed to them
   (c) From their position, compute the size of the domain (assumed to be of square shape) and the position of the vertical and horizontal mid-planes
3. Compute the Nusselt number for the left vertical wall by cycling through the faces for that wall, identified using a combination of wall-boundary face indices and their boundary condition
4. Compute the Nusselt number and the maximum x-axis velocity in the vertical mid-plane
   (a) Cycling through the internal faces, identify the faces which join cells on each side of the mid-plane. Because of inevitable numerical inaccuracies, this is more reliable than finding faces which are exactly at the mid-plane.
   (b) Compute the face values of density, temperature and velocity
   (c) Compute the face contribution and accumulate the Nusselt number
   (d) Compute the maximum x-axis velocity, store its non-dimensional value and its location
5. Compute the maximum y-axis velocity in the horizontal mid-plane
   (a) Cycling through the internal faces, identify the faces which join cells on each side of the mid-plane. Because of inevitable numerical inaccuracies, this is more reliable than finding faces which are exactly at the mid-plane.

(b) Compute the face value of density and velocity

(c) Compute the maximum y-axis velocity, store its non-dimensional value and its location

6. Output the Nusselt numbers, and the maximum velocities and their locations in adimensional form in the 'listing' file. The output is formatted and uses the keyword 'us_extraops' to flag the data in 'listing'.

4 Running the case

To run the case, simply repeat the calculations that are described in Section II. 'cs_user_extra_operations.f90' will automatically be detected, compiled and linked with the *Code_Saturne* executable.

At the end of the calculation, open the 'listing' file and look for the keyword 'us_extraops'. Data should appear as below:

```
   ** Heated Cavity Solution **
      ----------------------
-------------------------------------------------------------------------------------------------
              Iter   Nu_mid       Nu_lw        Ux_max       y(Ux_max)    Uy_max       x(Uy_max)
us_extraops:    1   0.0000E+00   0.1554E+03  -0.1000E-11  -0.1000E-11  -0.1000E-11  -0.1000E-11
-------------------------------------------------------------------------------------------------
```

Figure III.2: Output from 'cs_user_extra_operations.f90', after 1 iteration.

```
   ** Heated Cavity Solution **
      ----------------------
-------------------------------------------------------------------------------------------------
              Iter   Nu_mid       Nu_lw        Ux_max       y(Ux_max)    Uy_max       x(Uy_max)
us_extraops:  450   0.8987E+01   0.8988E+01   0.6508E+02   0.8562E+00   0.2186E+03   0.3125E-01
-------------------------------------------------------------------------------------------------
```

Figure III.3: Output from 'cs_user_extra_operations.f90', after 450 iterations.

Aside from direct, quantitative comparisons with existing data, computing representative variables for a case is also very useful to evaluate calculation convergence and stability. For example, in Linux you may use the command:

```
grep -n us_extraops /../listing > HeatedSquareCavity.txt
```

which will output all the data in the text file specified. The file may then be edited to remove 'us_extraops' and line numbers and imported in a plotting package to graph the evolution of all the variables as a function of the number of iterations.

5 Comparison with Benchmark Data

The calculated values at the last iteration (450) are compared with the published data in Table III.1 below, showing an excellent quantitative agreement.

	$\mathbf{Nu_{\frac{1}{2}}}$	$\mathbf{Nu_0}$	$\mathbf{\widetilde{u}_{max}}$	$\mathbf{\widetilde{y}_{max}}$	$\mathbf{\widetilde{v}_{max}}$	$\mathbf{\widetilde{x}_{max}}$
Code_Saturne	8.99	8.99	65.12	0.86	218.7	0.0313
Benchmark	8.8	8.82	64.63	0.85	219.36	0.0379
Error (%)	2.16	1.93	0.76	1.2	−0.3	17.4

Table III.1: Comparison between *Code_Saturne* 's and the benchmark results.

The error is larger for the position of the maximum Y-axis velocity in the horizontal mid-plane. However, the benchmark maximum values were obtained by fourth-order interpolation whereas here they are taken at the centre of the cell faces. The absolute difference between the *Code_Saturne* and benchmark value is equal to $0.0379 - 0.0313 = 6.6 \times 10^{-3}$, which is less than the interval size between two cell centres, $\Delta\widetilde{x} = \frac{1}{L}\frac{L}{80} = 1.25 \times 10^{-2}$, and shows that the location of the maximum is captured accurately in *Code_Saturne*.

To test your coding, you are encouraged to repeat the tests for the lower values of $R_a = 10^4, R_a = 10^5$ which are also listed in [5, 6].

Chapter IV

References

1 References

[1] www.salome-platform.org

[2] F. Archambeau, N. Méchitoua, M. Sakiz,
Code_Saturne: a Finite Volume Code for the Computation of Turbulent Incompressible Flows - Industrial Applications,
International Journal on Finite Volumes, Vol. 1, 2004.

[3] www.code-saturne.org

[4] G.D. Mallinson and G. de Vahl Davis,
Three-dimensional natural convection in Box: a numerical study,
J. Fluid Mech., Vol. 83, part 1, pp. 1-31, 1977.

[5] G. de Vahl Davis and I.P. Jones,
Natural Convection in a Square Cavity: A Comparison Exercise,
International Journal for Numerical Methods in Fluids, Vol. 3, pp 227-248, 1983.

[6] G. de Vahl Davis and I.P. Jones,
Natural Convection of Air in a Square Cavity: A Benchmark Numerical Solution,
International Journal for Numerical Methods in Fluids, Vol. 3, pp. 249-264, 1983.

[7] EDF,
Tutorial 1: Shear Driven Cavity Flow,
Code_Saturne Tutorial Series

[8] P. Oosthuizen and D. Naylor,
Introduction to Convective Heat Transfer Analysis,
WCB/McGraw-Hill, 1999

[9] U. Ayachit,
The ParaView Guide: A Parallel Visualization Application,
Kitware, 2015, ISBN 978-1930934306