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

MARCH 2013

Code_Saturne documentation

Code_Saturne **version 2.0 tutorial - Fluid-structure interaction**

contact: saturne-support@edf.fr

Executive Summary

The present document aims at describing the use of the main features of *Code_Saturne* available for fluid-structure interaction calculations. The key points adressed here concern:

- performing a calculation on a mobile mesh using the ALE (Arbitrary Lagrangian-Eulerian) frame- work;
- how to parameter the mesh deformation and properly define the parameters of the boundary conditions for a mobile mesh;
- how to impose the mesh deformations or how to link a solid body in the fluid domain to a mass-spring system (with fluid-structure coupling);
- how to perform some basic post-processing related to such calculations.

The tutorial has been created based on *Code_Saturne 2.0*. Adjustments might be needed for other versions.

Sommaire / Summary

1 Objectives

The present tutorial aims at describing the use of the main features of *Code_Saturne* available for fluid- structure interaction calculations. The key points adressed here concern:

- performing a calculation on a mobile mesh using the ALE (Arbitrary Lagrangian-Eulerian) frame- work;
- how to parameter the mesh deformation and properly define the parameters of the boundary condi- tions for a mobile mesh;
- how to impose the mesh deformations or how to link a solid body in the fluid domain to a mass-spring system (with fluid-structure coupling);
- how to perform some basic post-processing related to such calculations.

The tutorial has been created based on *Code_Saturne 2.0*. Adjustments might be needed for other versions.

2 Description of the test case

The present test case focuses on the numerical simulation of the transverse response of an elastically mounted cylinder subjected to vortex-induced vibrations (VIV). A sketch of the flow setup is provided in figure 1: the flow is uniform and the mechanical dynamics of the cylinder is modeled by a simple mass-spring system.

The physical parameters are chosen so that the system configuration may be simulated with a reasonable computational cost. The Reynolds number based on the cylinder diameter is taken to be 100 so that the flow is laminar and a 2D calculation may then be able to capture the whole features of the phenomenon. The parameters of the simulation are given throughout the tutorial but a sum-up is provided in table 1.

As shown in figure 2 the calculation domain is 2D with only one cell in spanwise direction.

Table 1: Simulation parameters of the vortex-induced vibration test case.

Parameter	Description	
D	cylinder diameter	0.025 m
L	cylinder spanwise length	0.005 m
U_∞	inflow velocity	0.004 m.s^{-1}
ρ	density	1000 kg.m^{-3}
μ	dynamic viscosity	0.001 Pa.s
m	cylinder mass (kg)	*see* $m*$
k	cylinder stiffness (N.m^{-1})	*see* $k*$
$m* = \frac{m}{1/2\rho D^2 L}$	normalized mass	3.3
$K* = \frac{k}{1/2\rho U_\infty^2 L}$	normalized stiffness	12.02

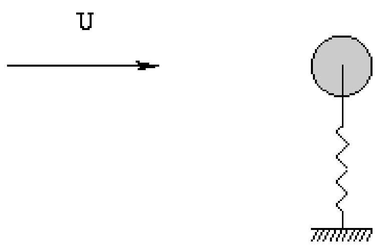

Figure 1: Sketch of the vortex-induced vibration test case.

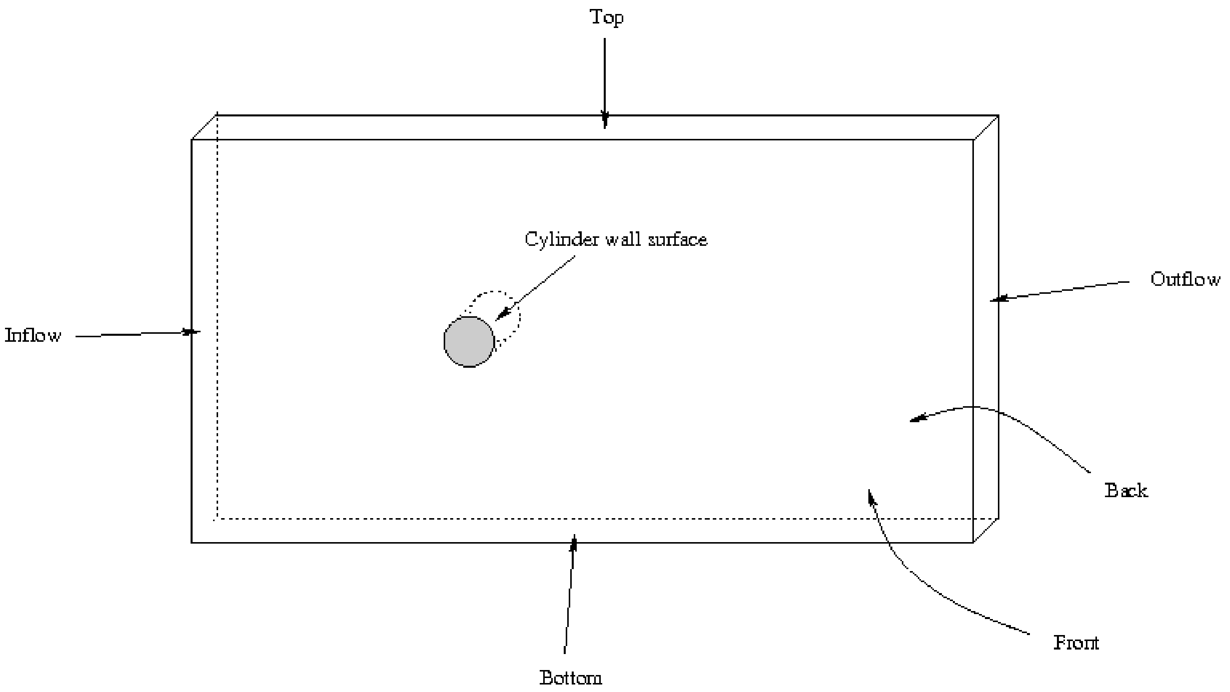

Figure 2: Overview of the computational domain.

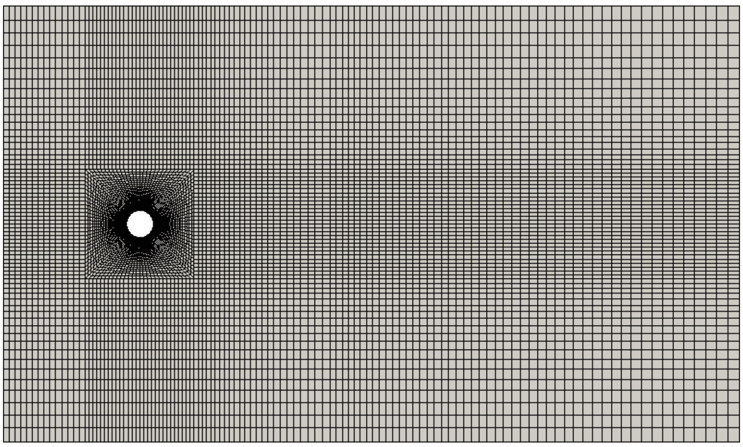

Figure 3: View of the mesh.

3.1 How to setup a case involving fluid-structure interactions (GUI)

3.1.1 Setting up and running the test case

The first step is to create the *Code_Saturne* case. The following command creates a study directory, named `Tuto_VIV`, and a new case, referred to as `VIV` :

```
code_saturne create -s Tuto_VIV -c VIV
```

It is then necessary to copy the mesh file `mesh_viv.des` into the mesh directory of the study whose path is `Tuto_VIV/MESH` (the mesh file is located into the tutorial directory `Tutorial_Files`). Note that the mesh `mesh_viv.des` is rather coarse so that it will lead to short calculations. Another mesh, referred to as `mesh_viv_file.des`, is also available. It will give more accurate results but will requier more computational time. One may first start with the coarse mesh to setup the test case and then switch to the fine one.

The graphical user interface (GUI) can now be started. Just go into the `DATA` directory of the case (with the path `Tuto_VIV/VIV/DATA/`) and type in the command:

```
./SaturneGUI
```

in order to run the GUI. The next steps and commentaries are provided in figures 4 to 41.

3.1.2 Post-processing

As concern calculations with a mobile mesh, basic flow visualization can be performed. As pointed out during the tutorial (in figure 37), as long as the outputs are based on a deformable mesh one may visualize the mesh deformations.

Some others valuable outputs related to the structure motions are also available in the working directory of the run. These data are copied into the `RESU` directory of the case, in the `HIST` directory corresponding to the run:

- go into the history directory of the run which should be `Tuto_VIV/VIV/RESU/HIST.010100`. The last eight digits corresponds to the date and time at which the calculation has started;
- a series of data files are available on the time history of the acceleration (`str_acceleration_x.dat`), velocity (`str_vitesse_x.dat`), displacement (`str_deplacement_x.dat`) of the structure and on the fluid forces acting on it (`str_force_x.dat`). Note that each direction, x, y and z, are treated in separated files;
- these data can easily be plotted using standard plotting software. The following command shows how to plot the displacement of the structure in the y-direction as a function of time, using the plotting tool `xmgrace`:

  ```
  xmgrace -block str_deplacement_y.dat -bxy 2:3
  ```

  which plots the displacement in the y-direction as a function of time. The figure 42 illustrates the result that should be obtained. Remark that during the initialization period, for t < 500 s, the structure does not move. See figure 10 for the definition of the initialization period in the GUI.

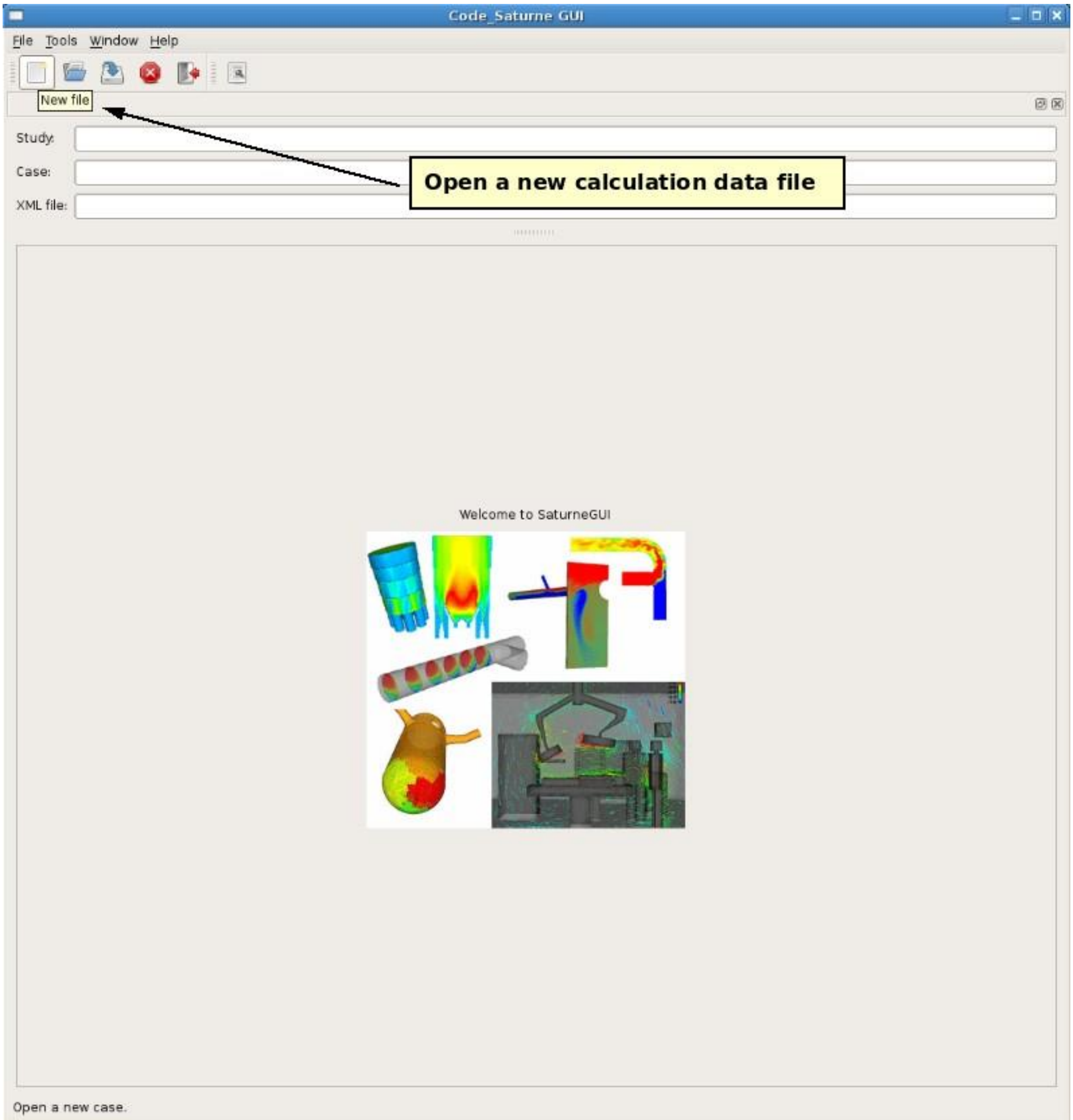

Figure 4: VIV test case. First steps of the case opening.

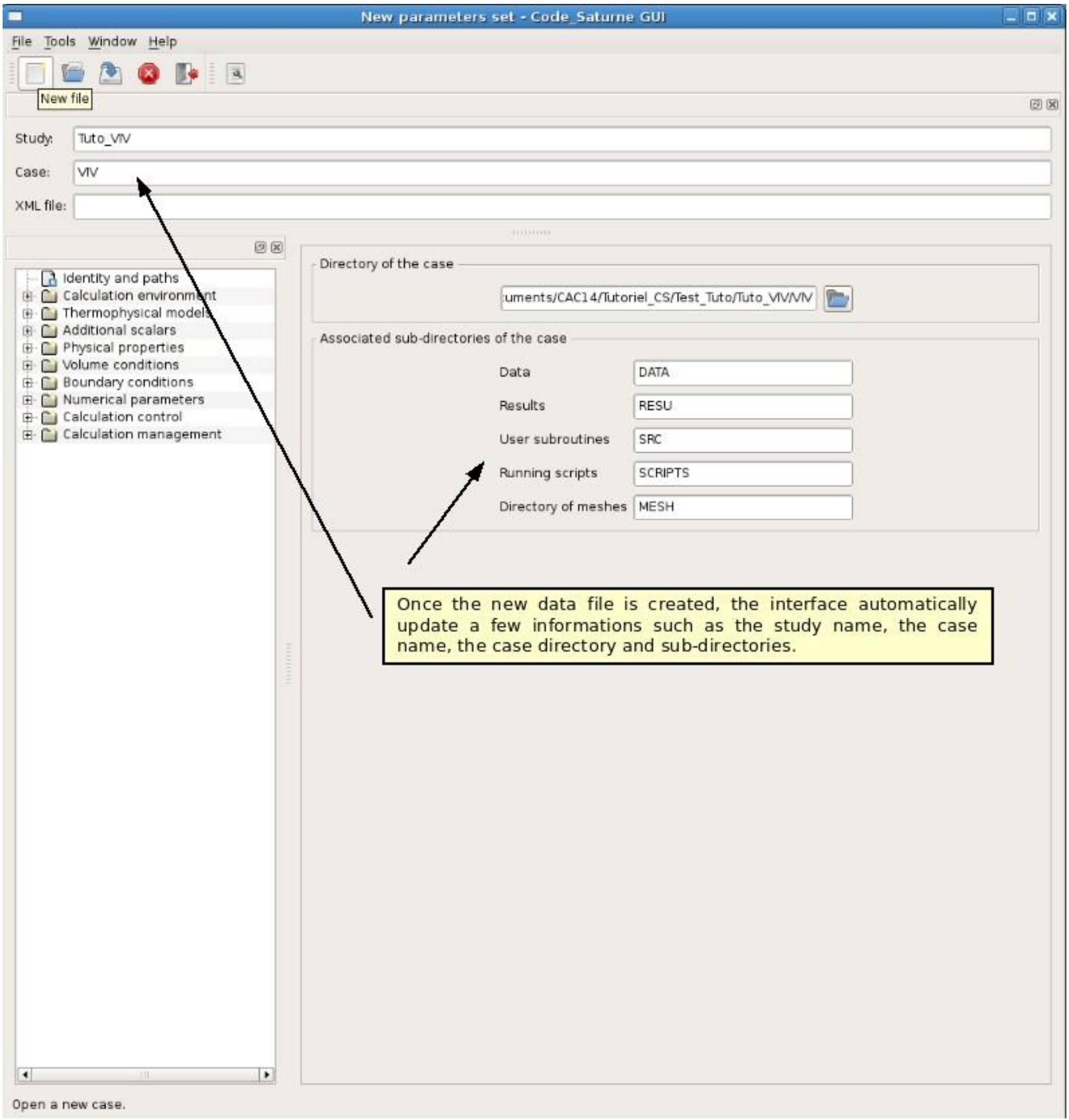

Figure 5: VIV test case. First steps of the case opening.

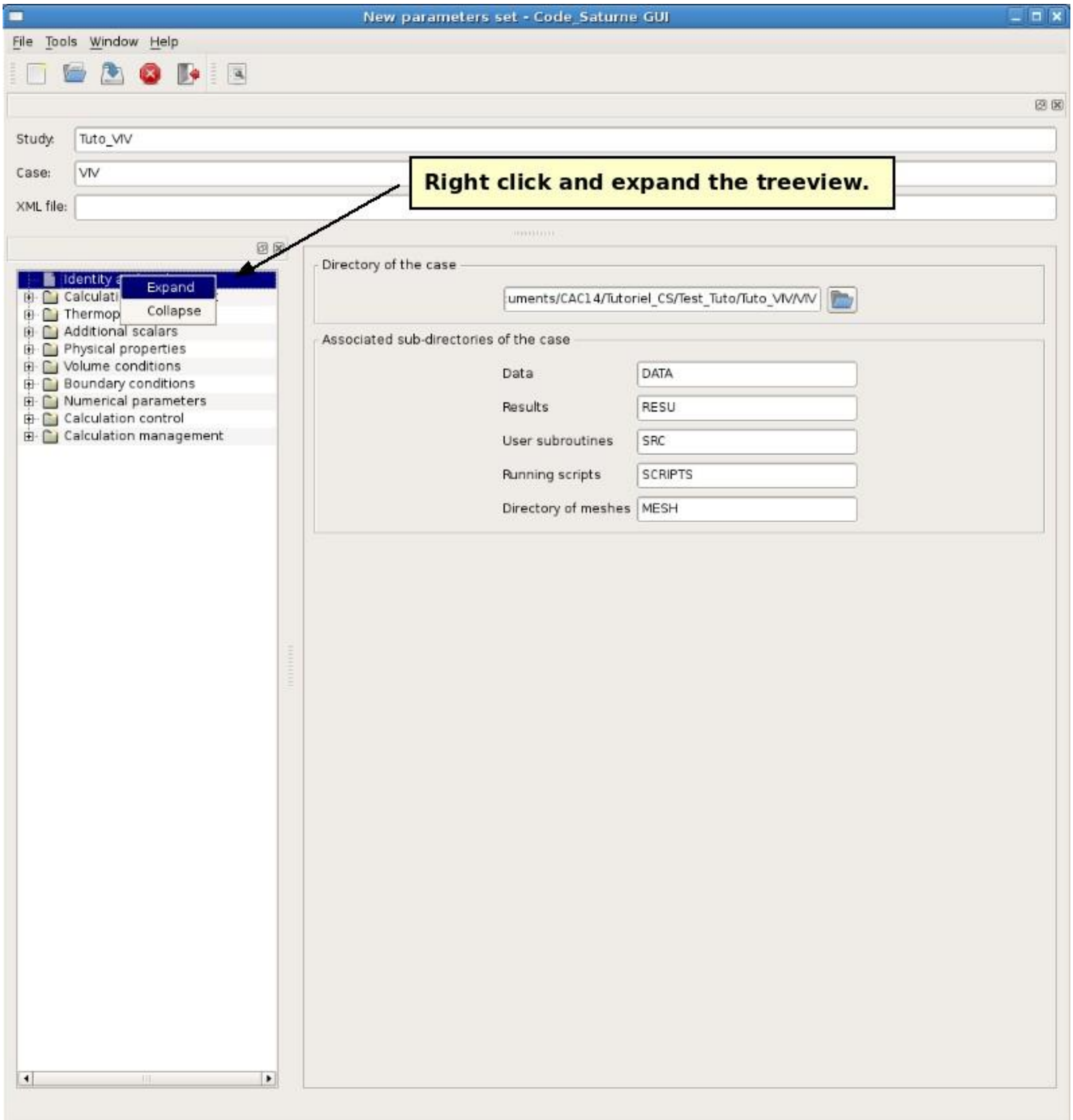

Figure 6: VIV test case. First steps of the case opening.

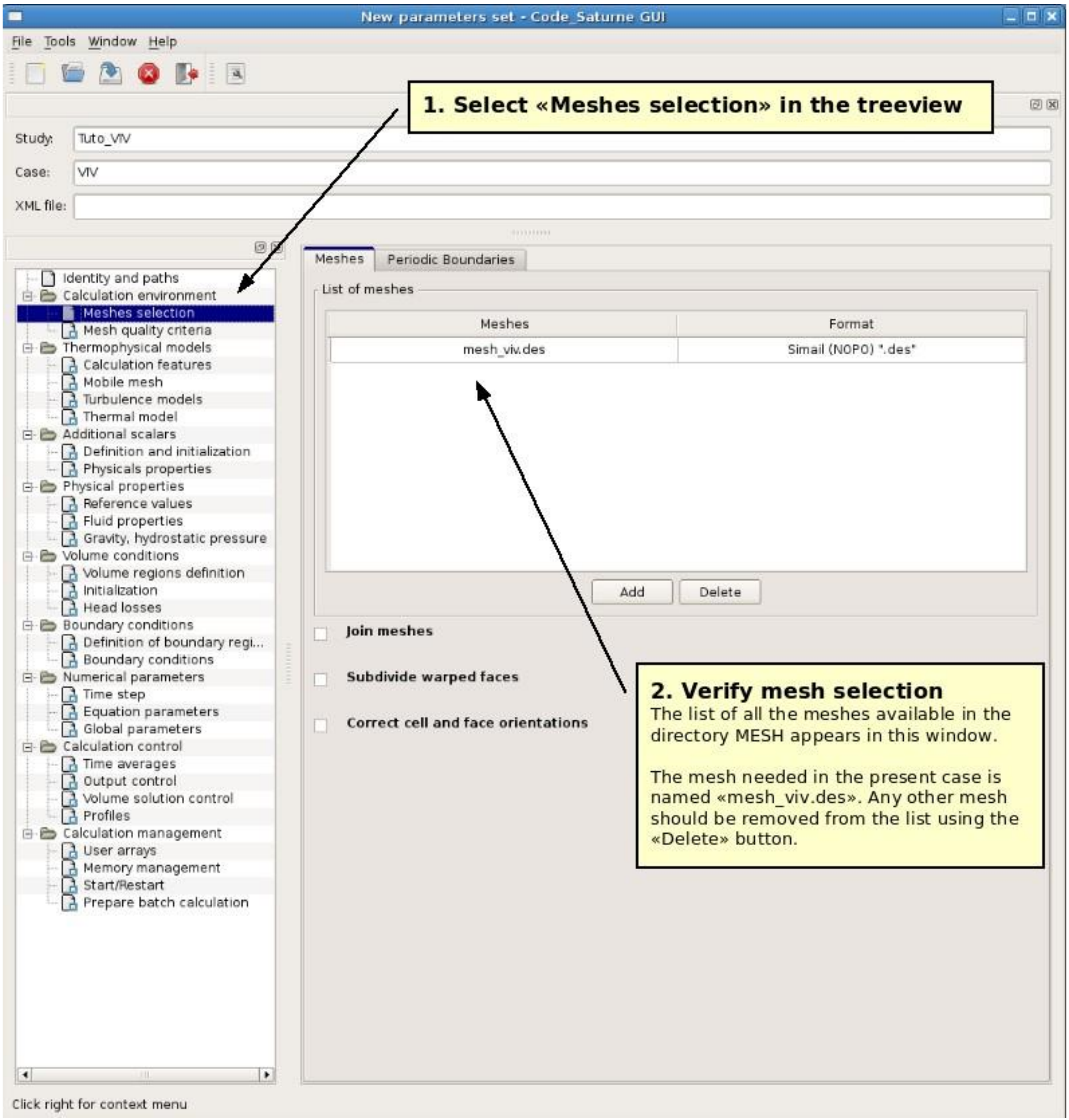

Figure 7: VIV test case. Meshes selection.

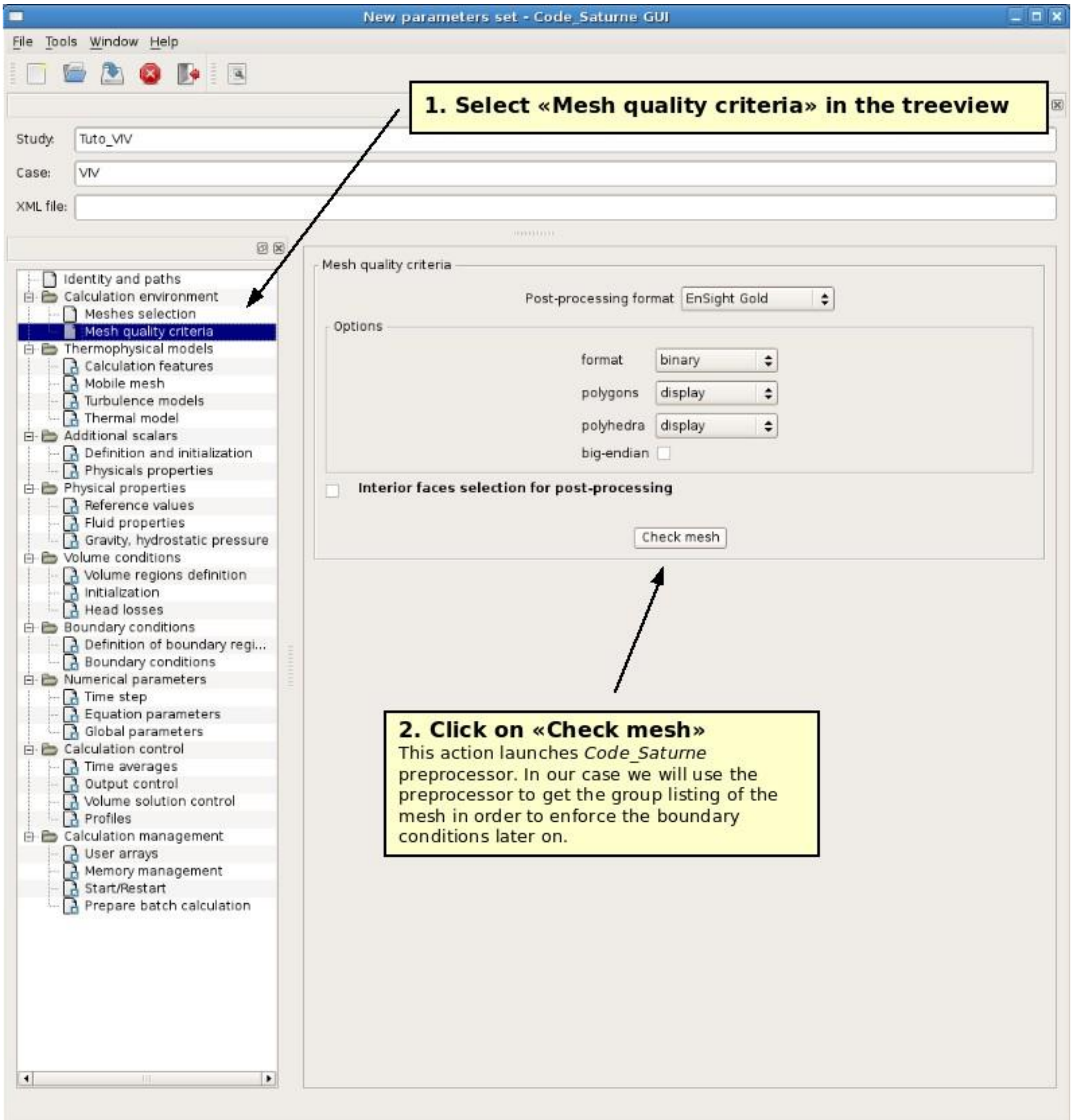

Figure 8: VIV test case. Mesh quality criteria.

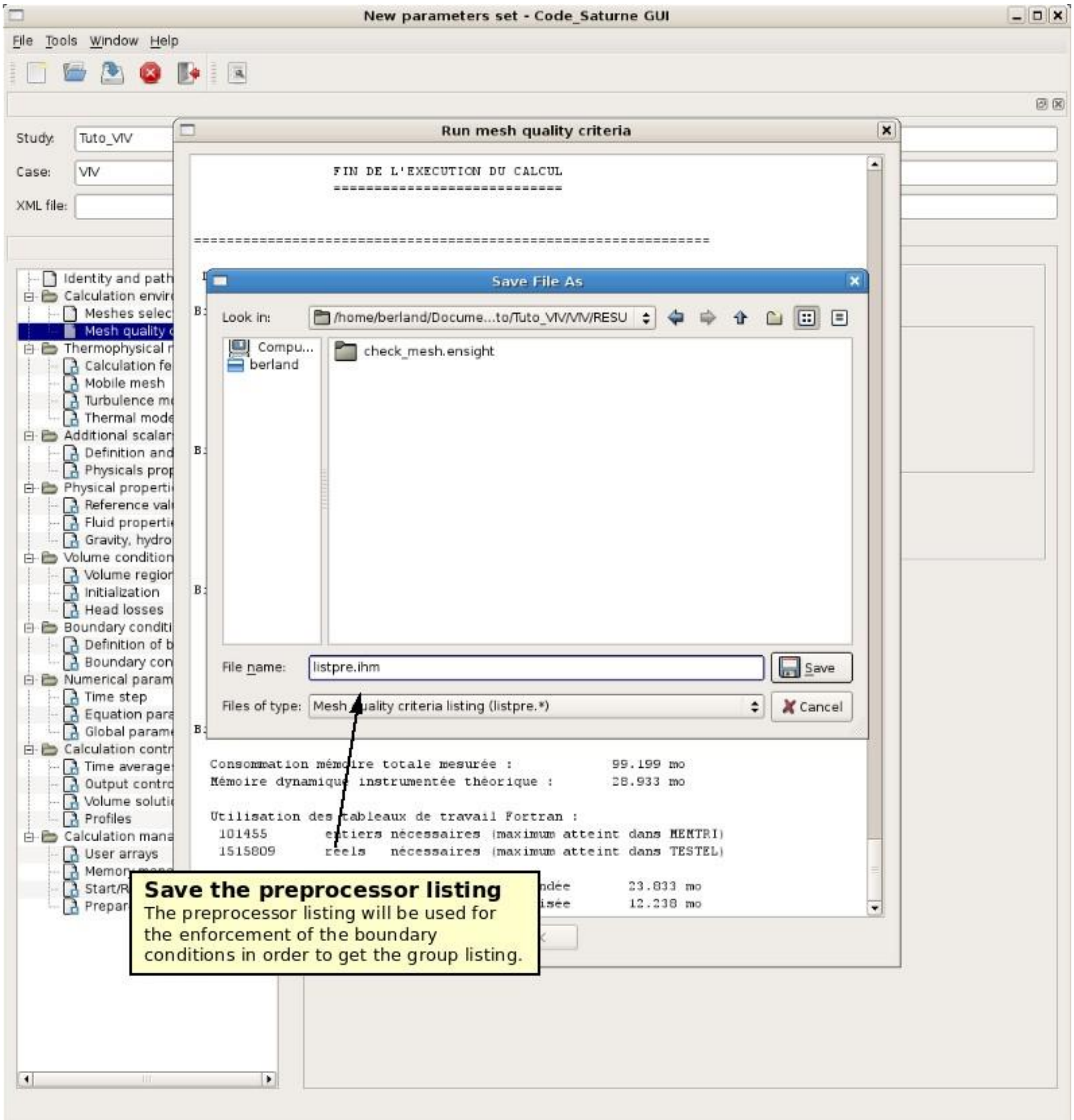

Figure 9: VIV test case. Mesh quality criteria.

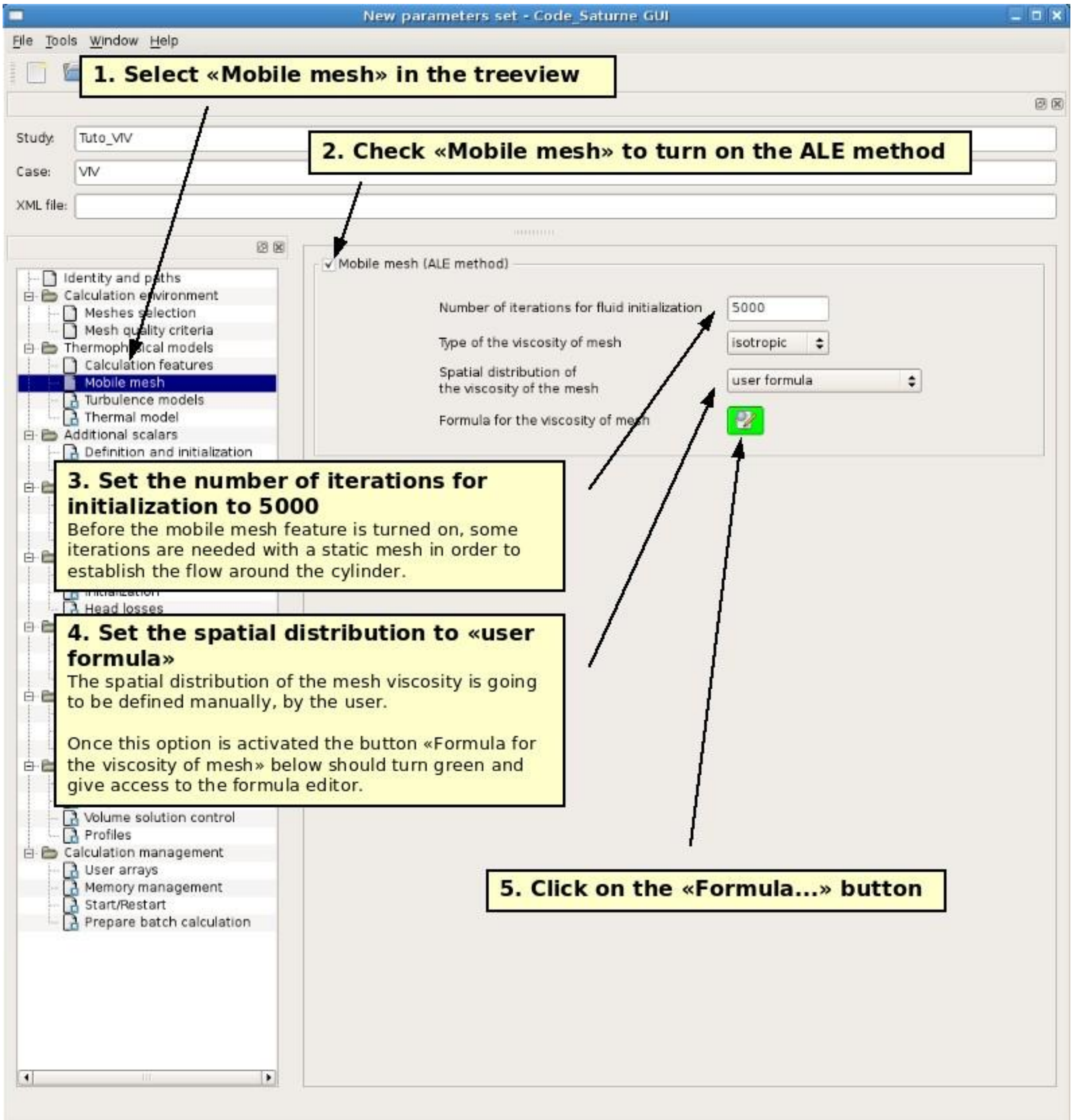

Figure 10: VIV test case. Mobile mesh.

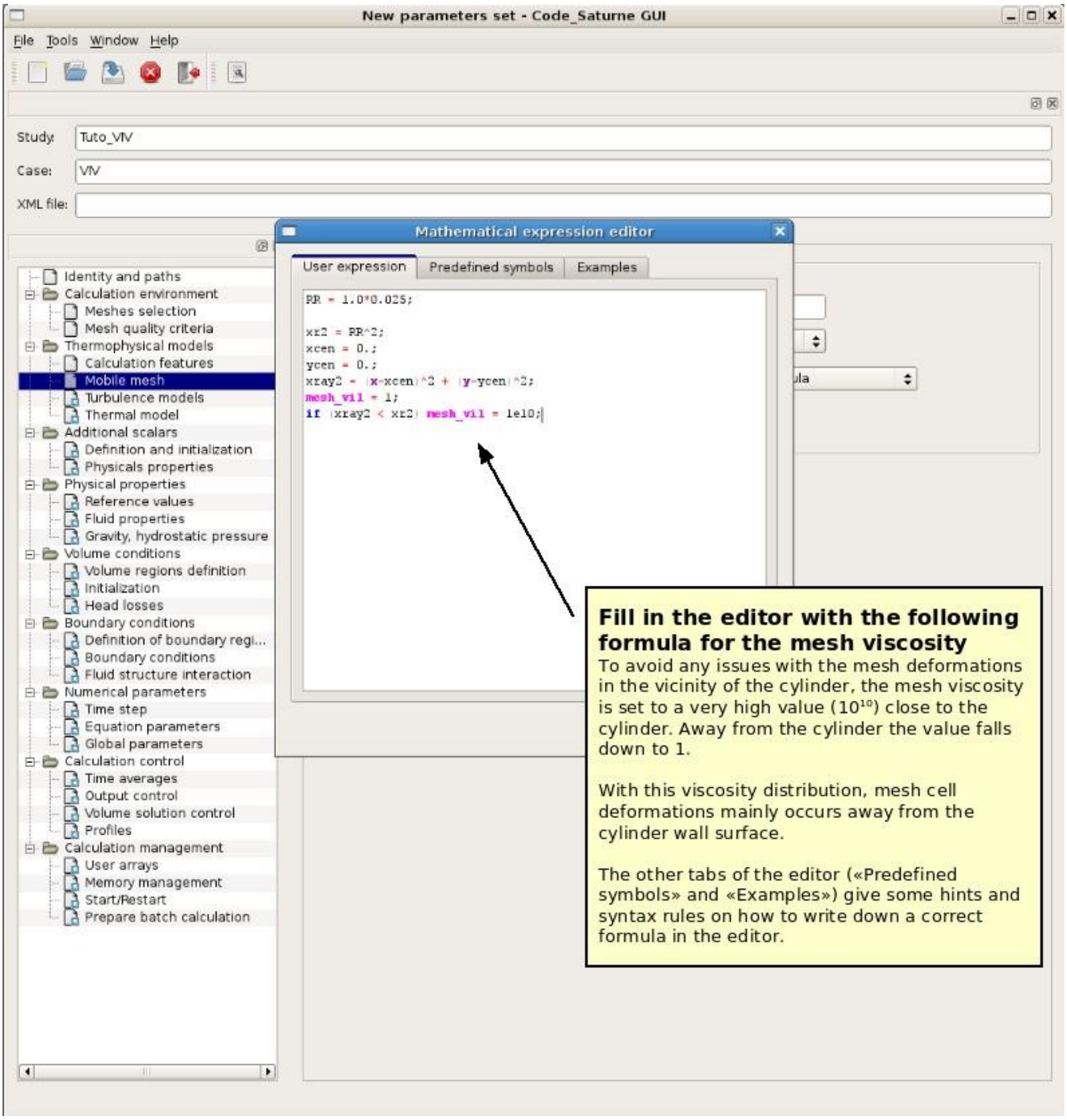

Figure 11: VIV test case. Mobile mesh.

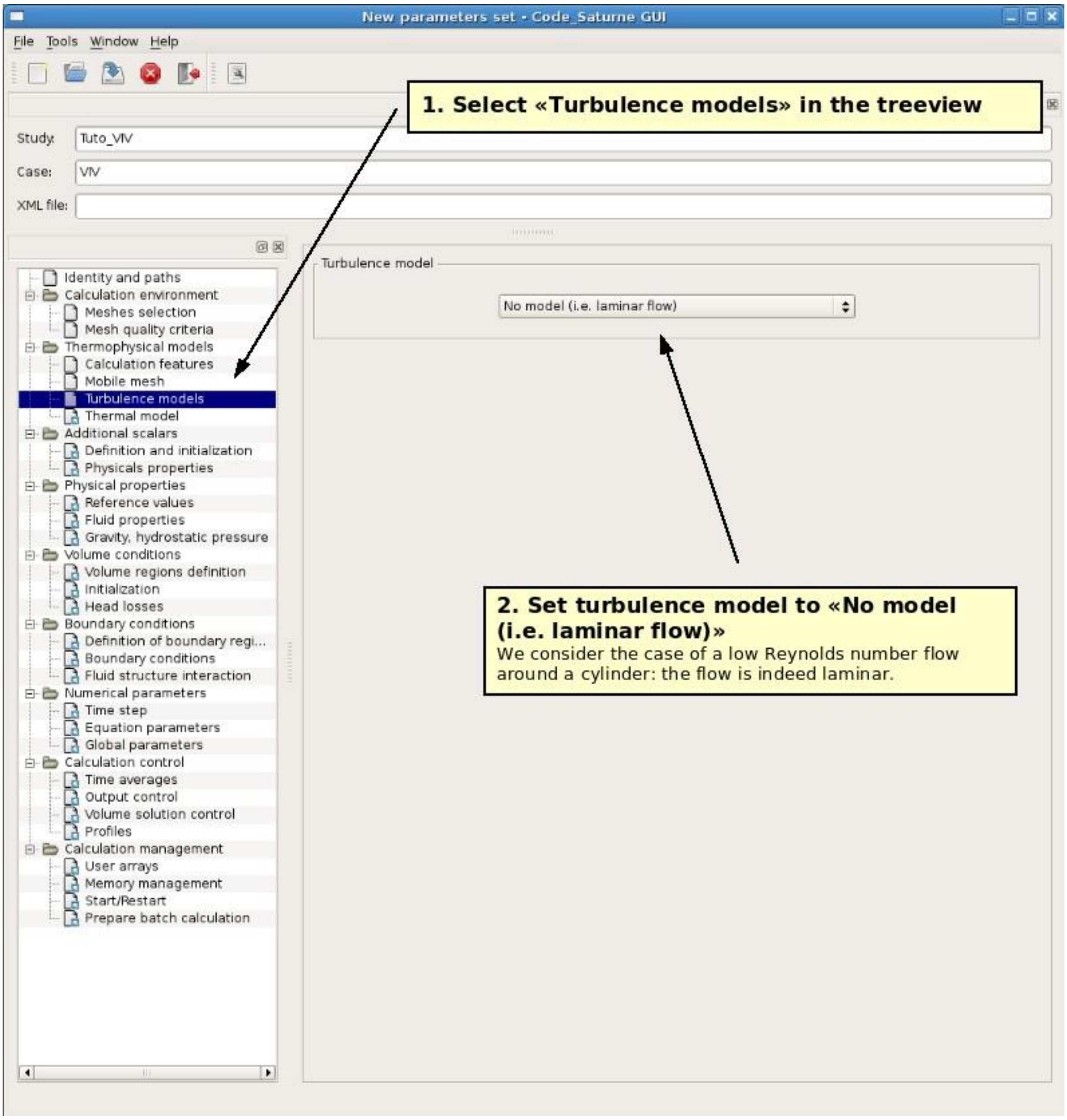

Figure 12: VIV test case. Turbulence models.

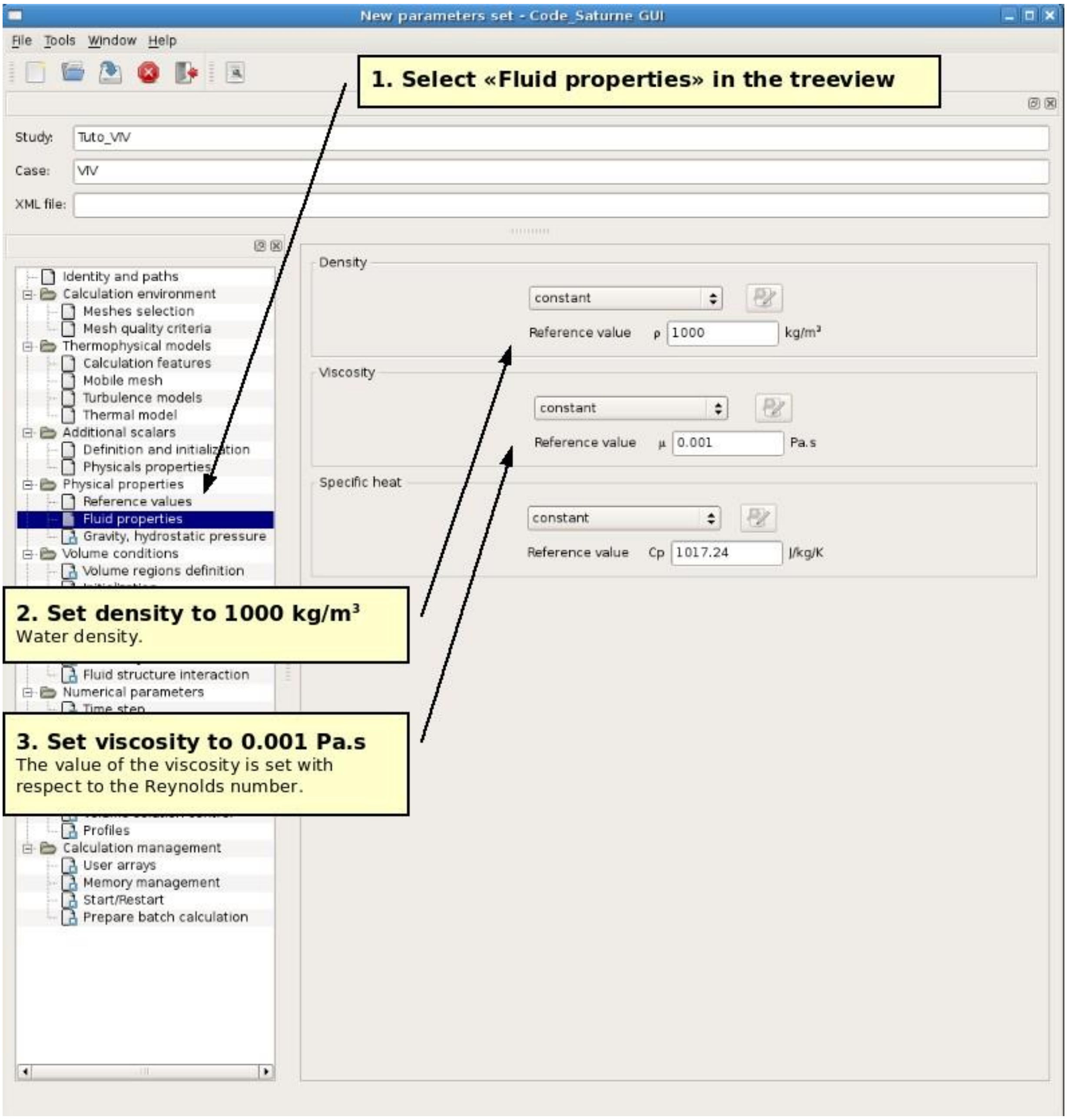

Figure 13: VIV test case. Fluid properties.

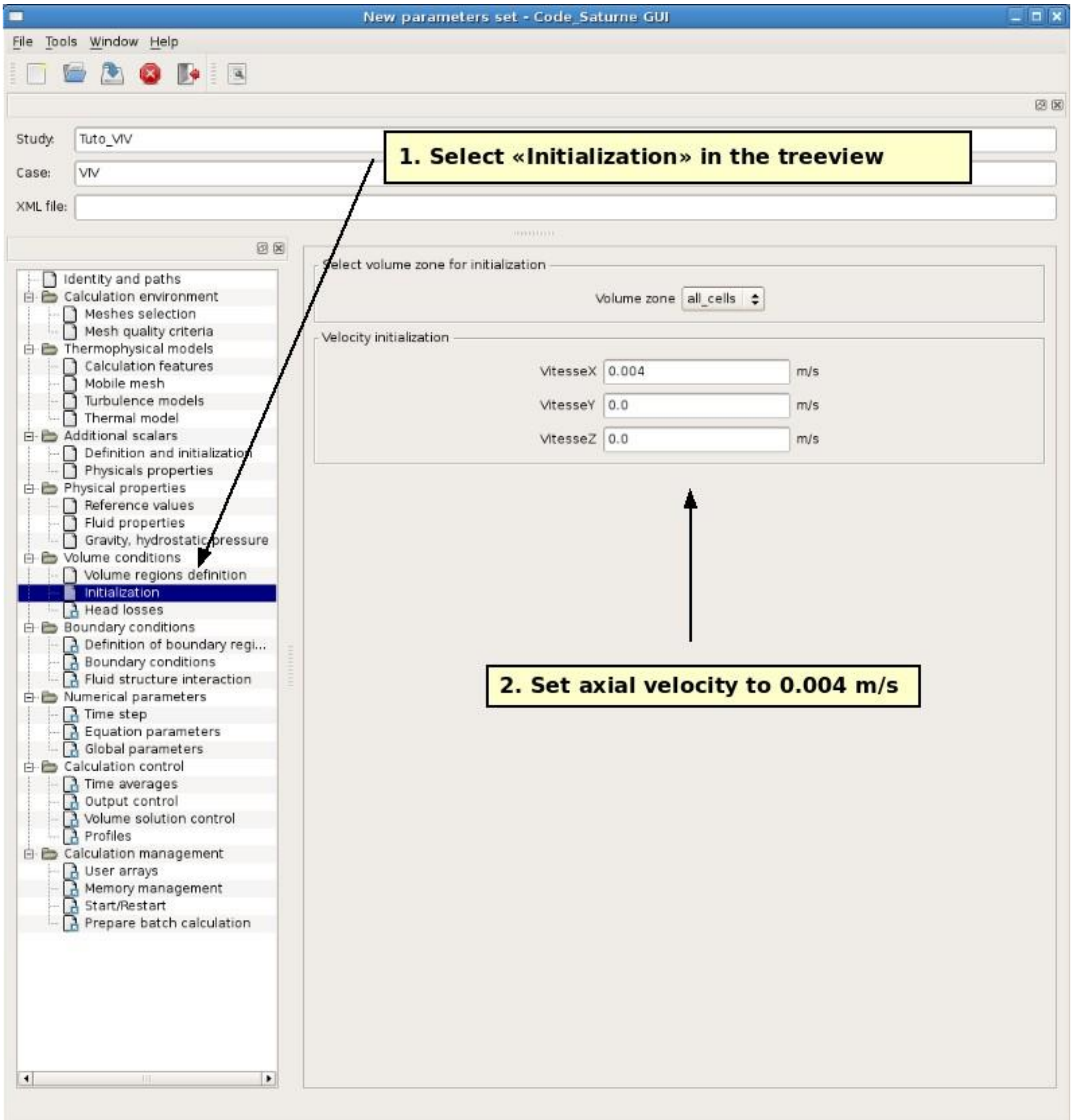

Figure 14: VIV test case. Initialization.

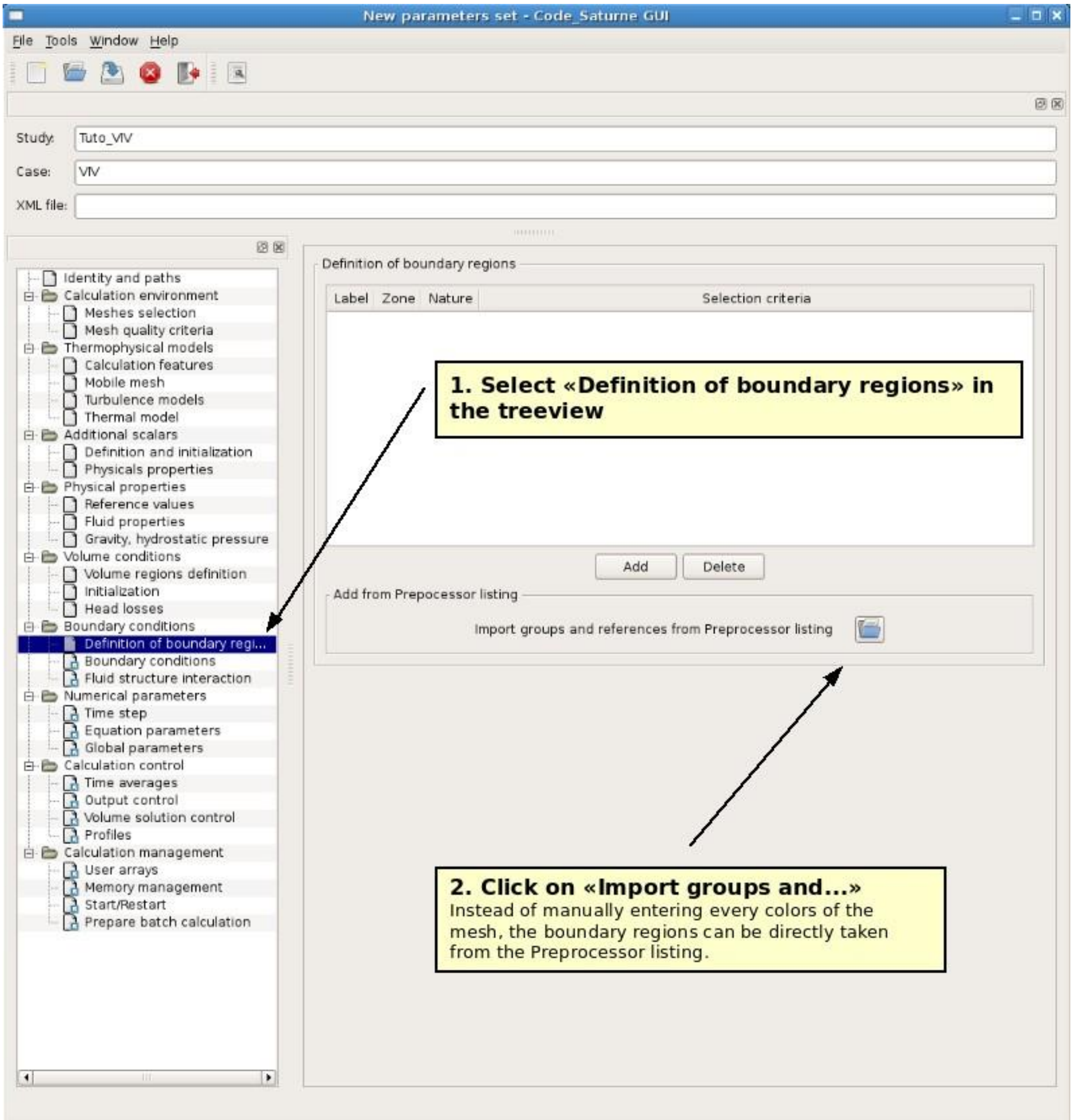

Figure 15: VIV test case. Definition of boundary regions.

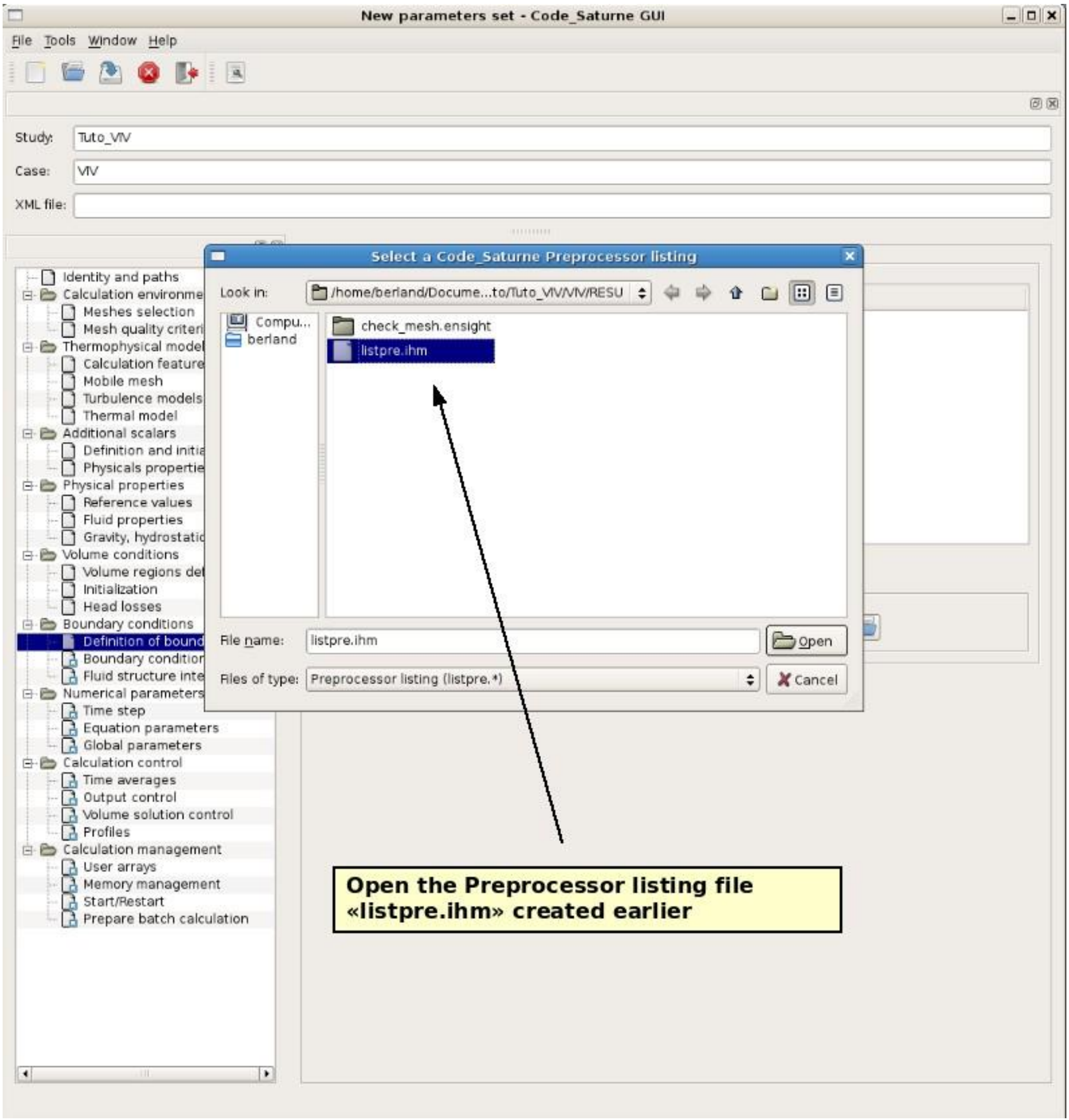

Figure 16: VIV test case. Definition of boundary regions.

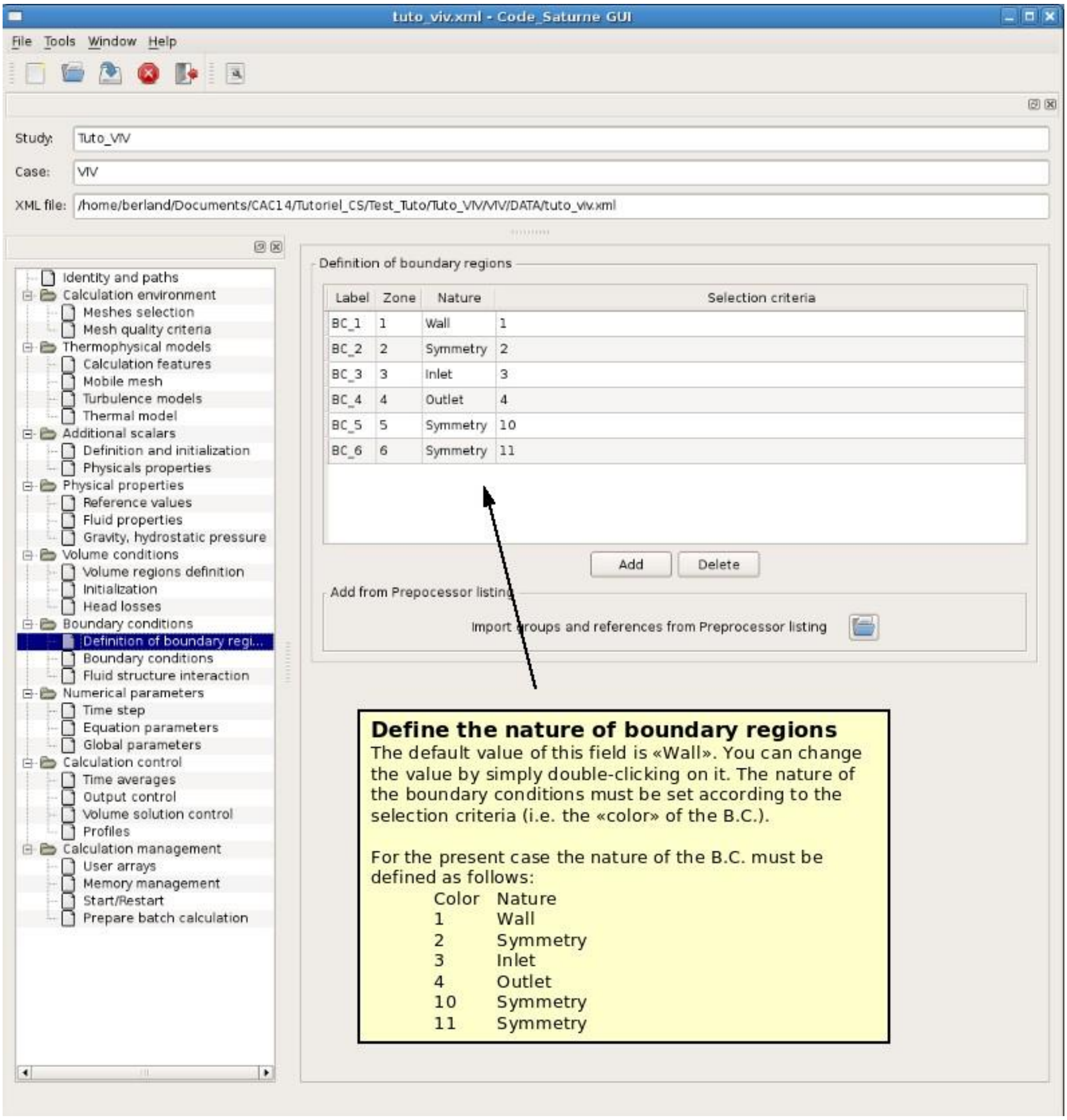

Figure 17: VIV test case. Definition of boundary regions.

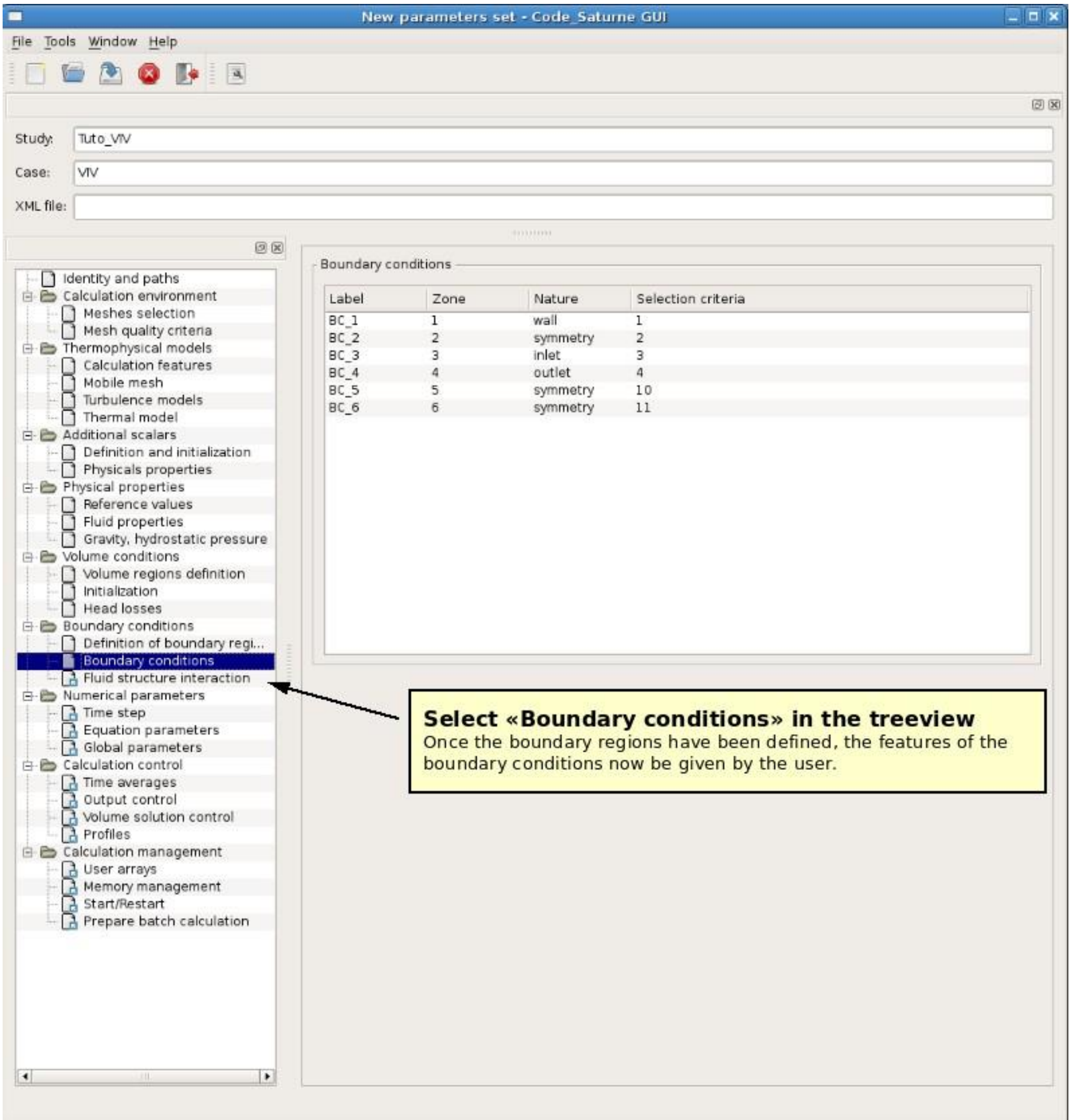

Figure 18: VIV test case. Boundary conditions.

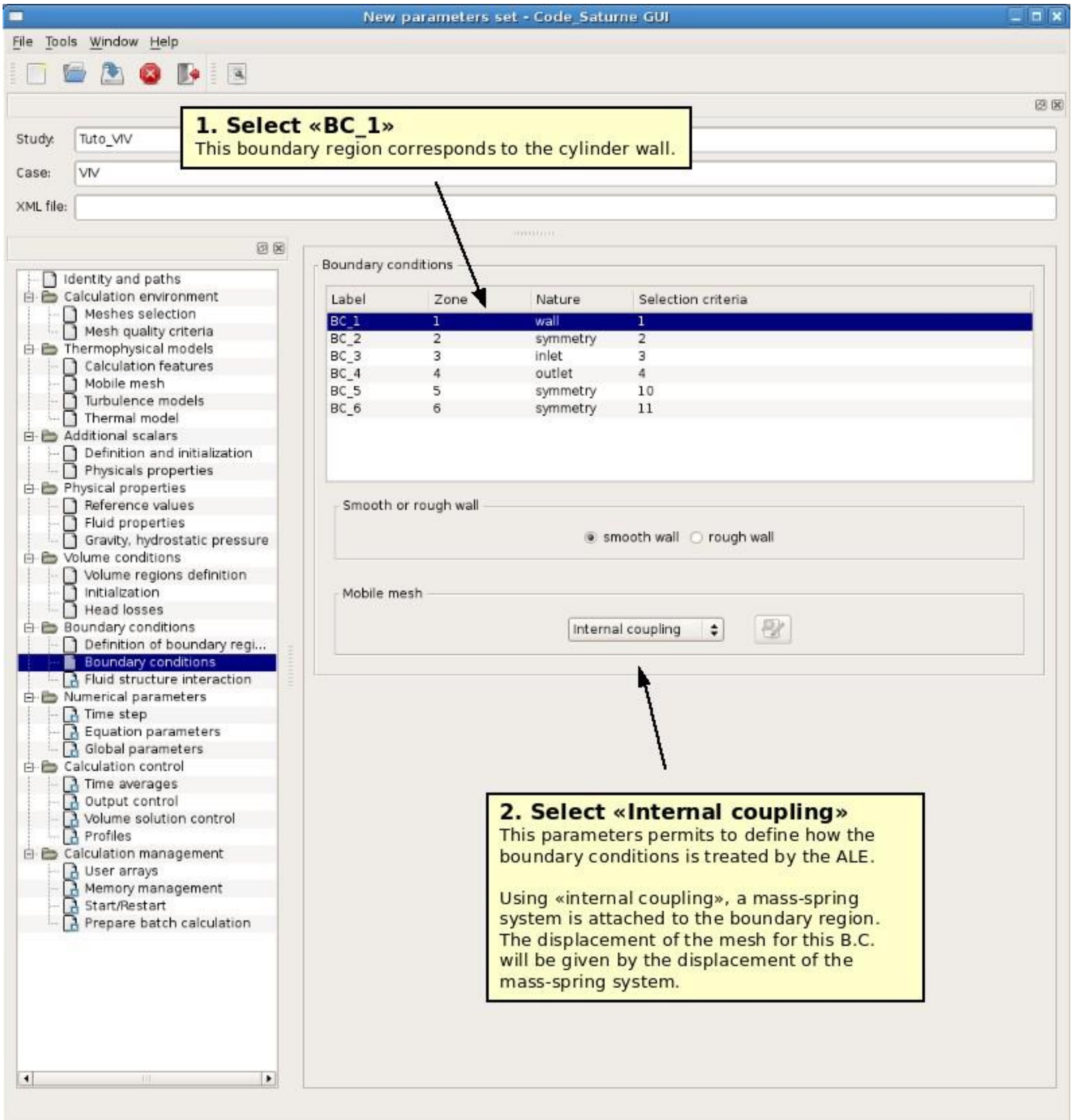

Figure 19: VIV test case. Boundary conditions.

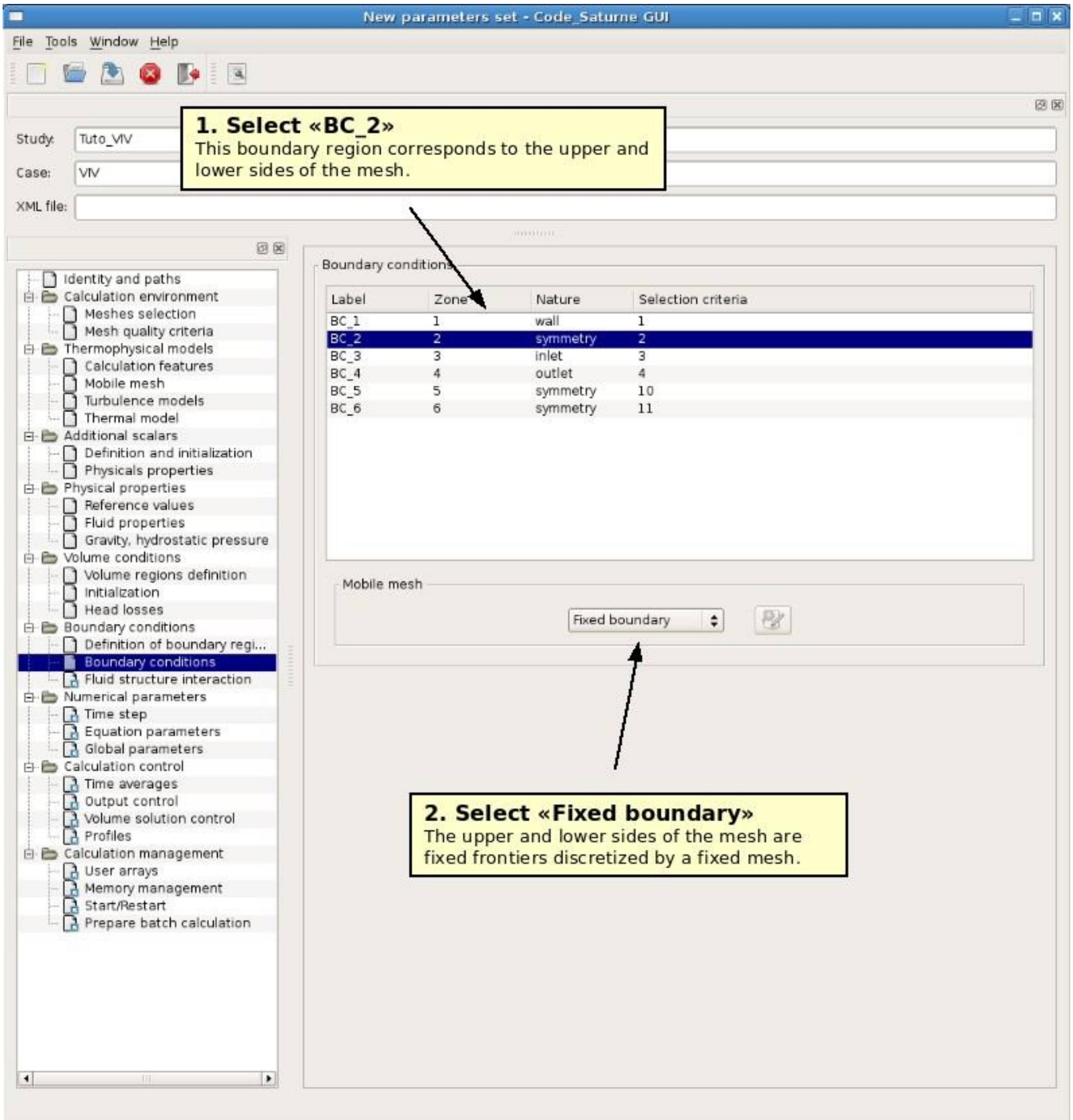

Figure 20: VIV test case. Boundary conditions.

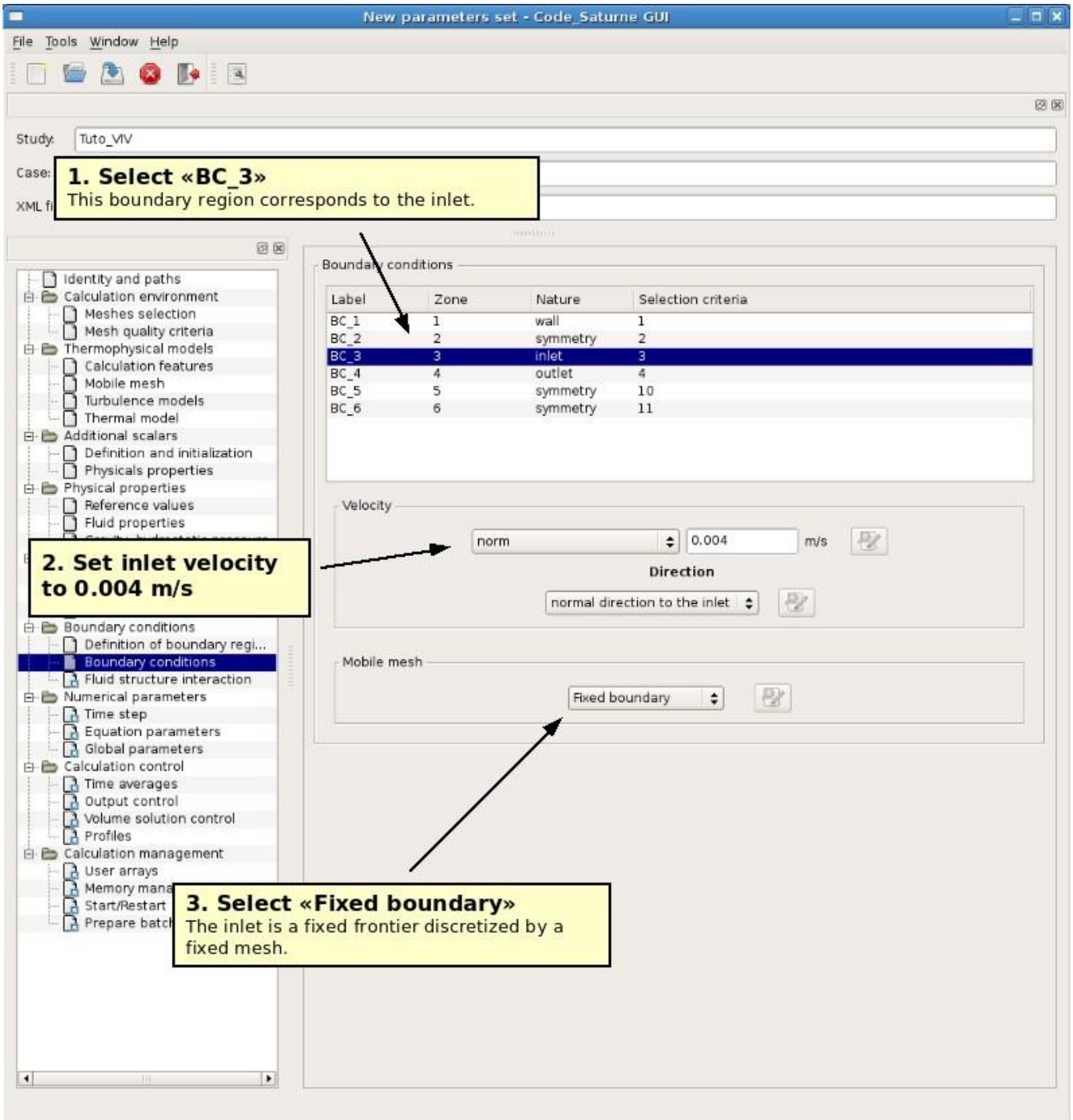

Figure 21: VIV test case. Boundary conditions.

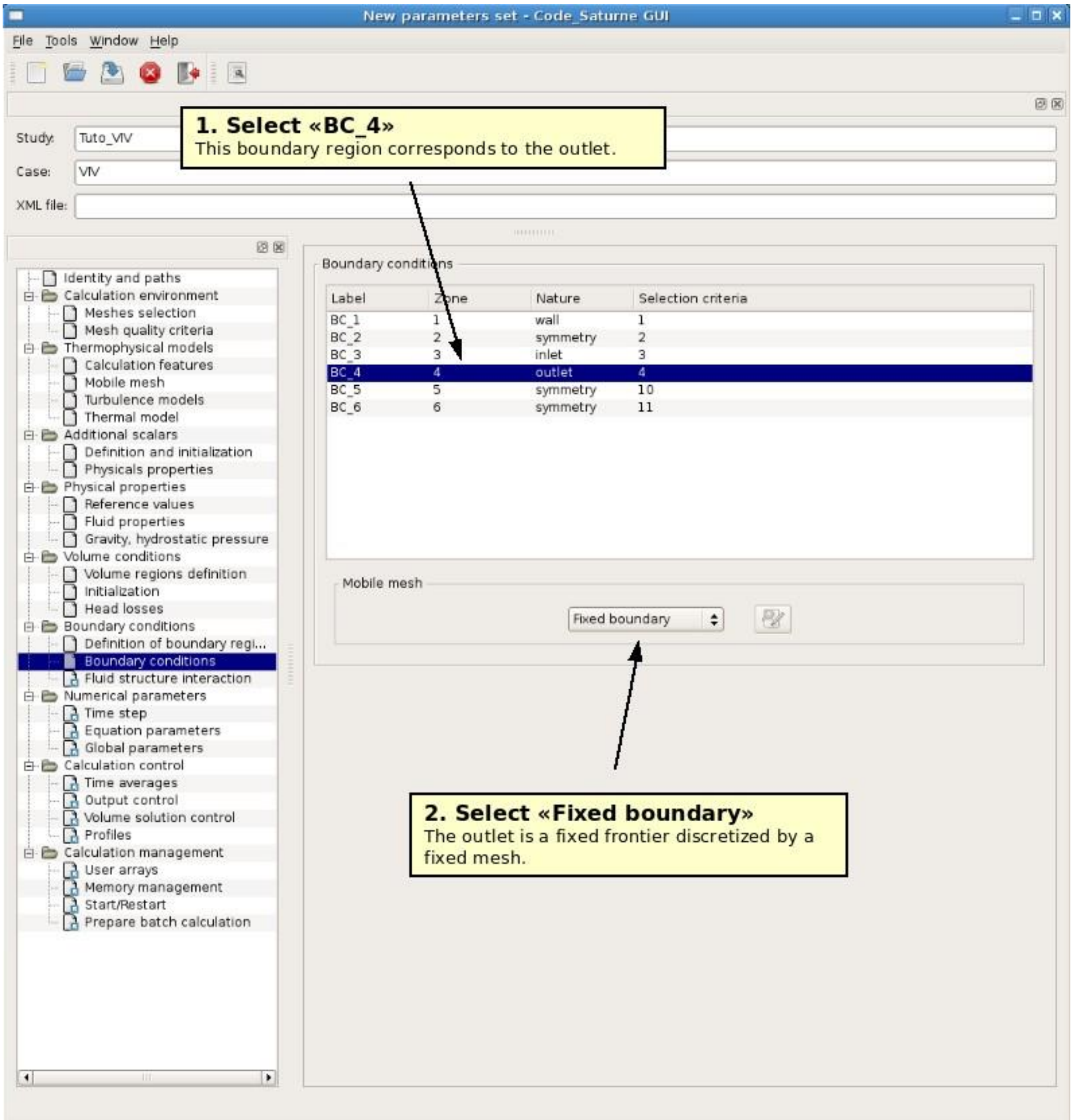

Figure 22: VIV test case. Boundary conditions.

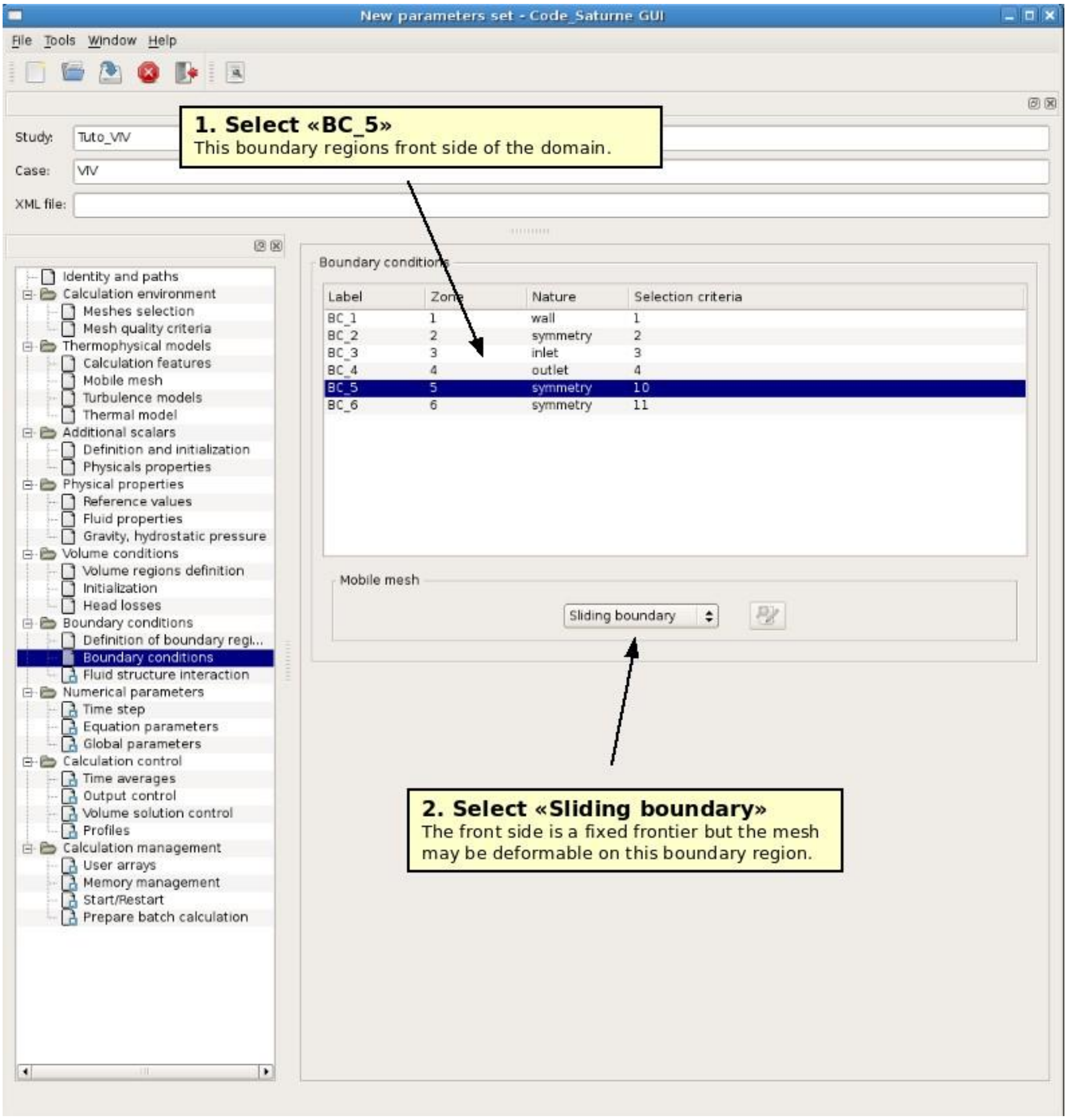

Figure 23: VIV test case. Boundary conditions.

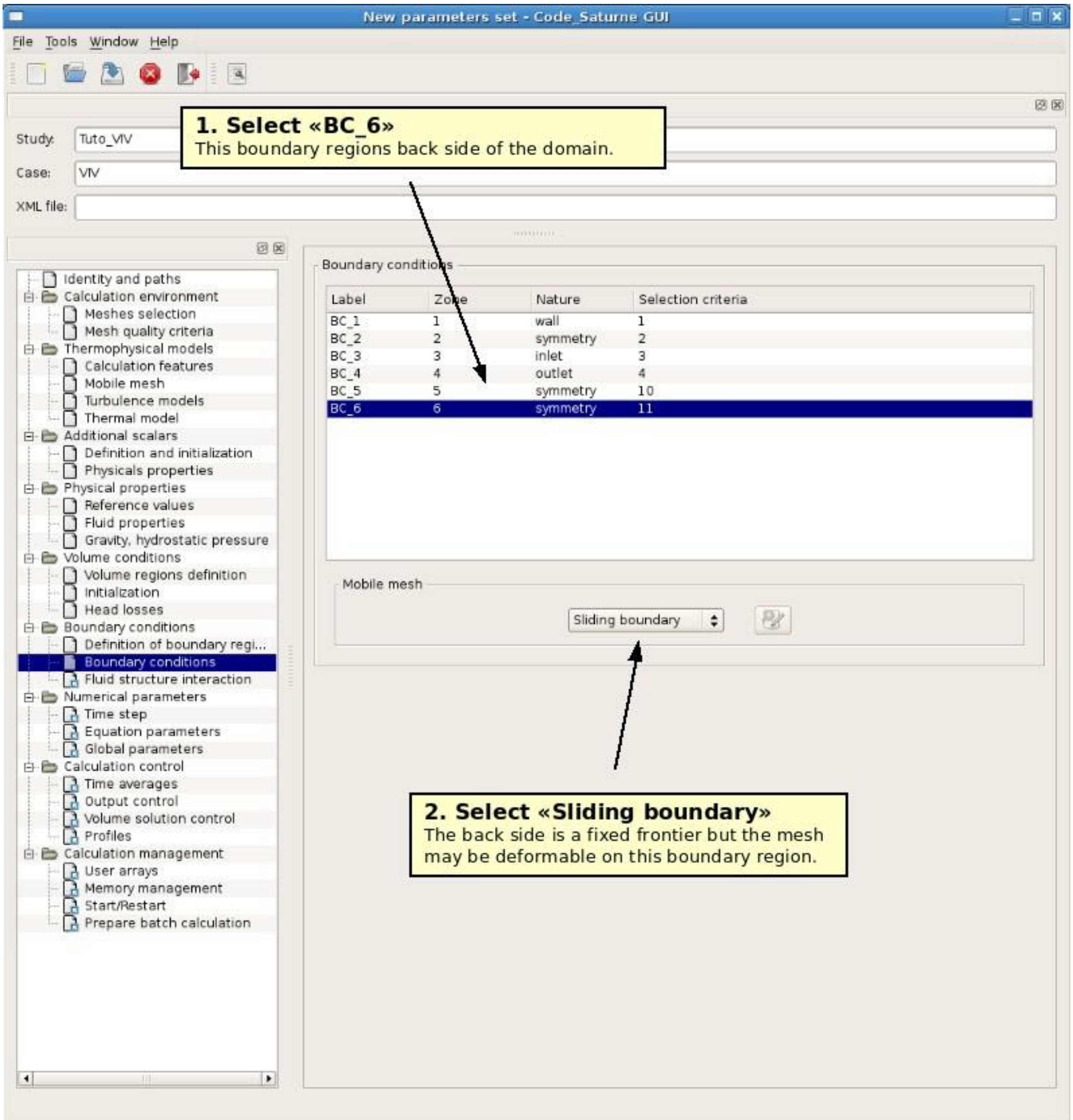

Figure 24: VIV test case. Boundary conditions.

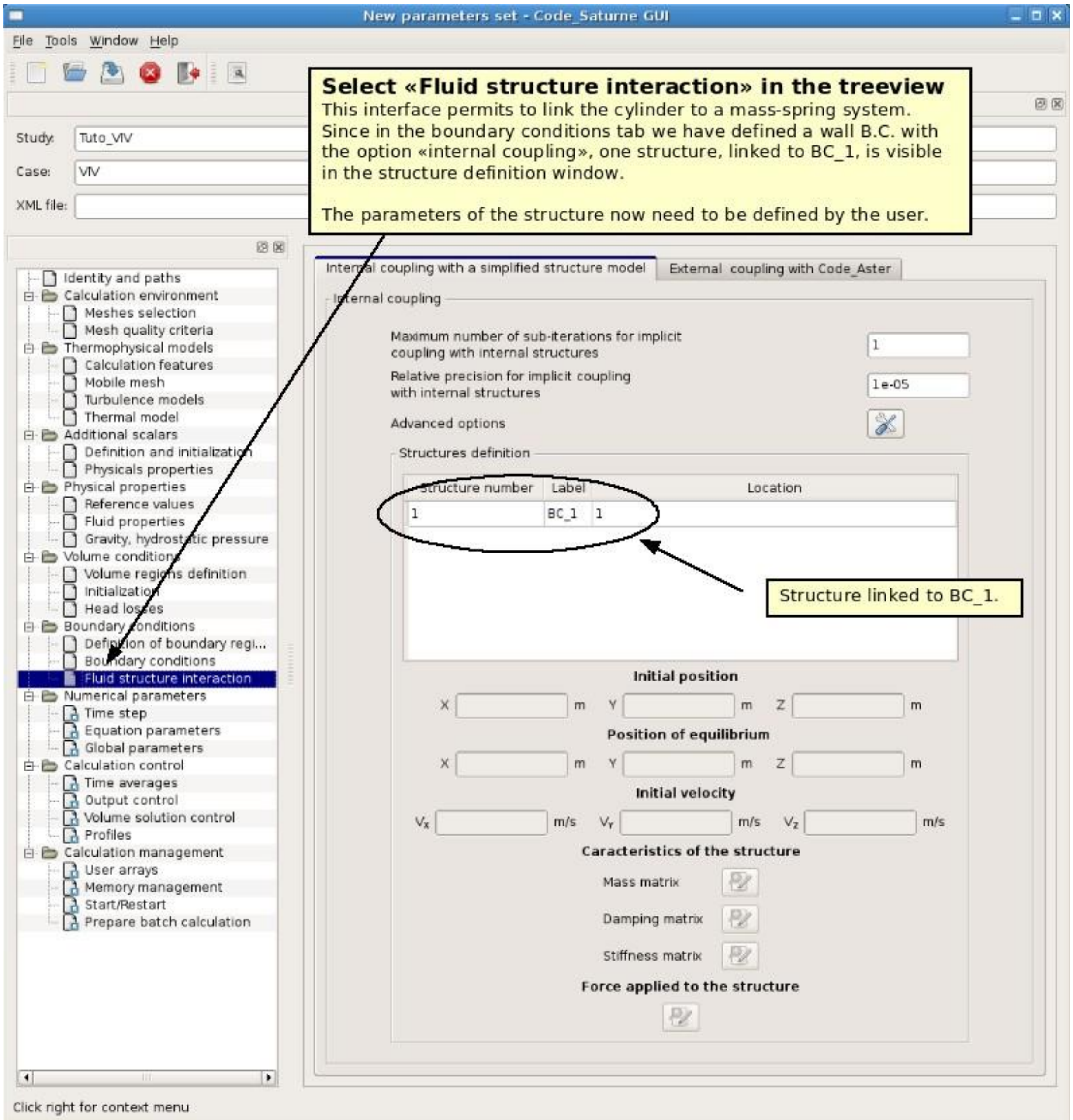

Figure 25: VIV test case. Fluid structure interaction.

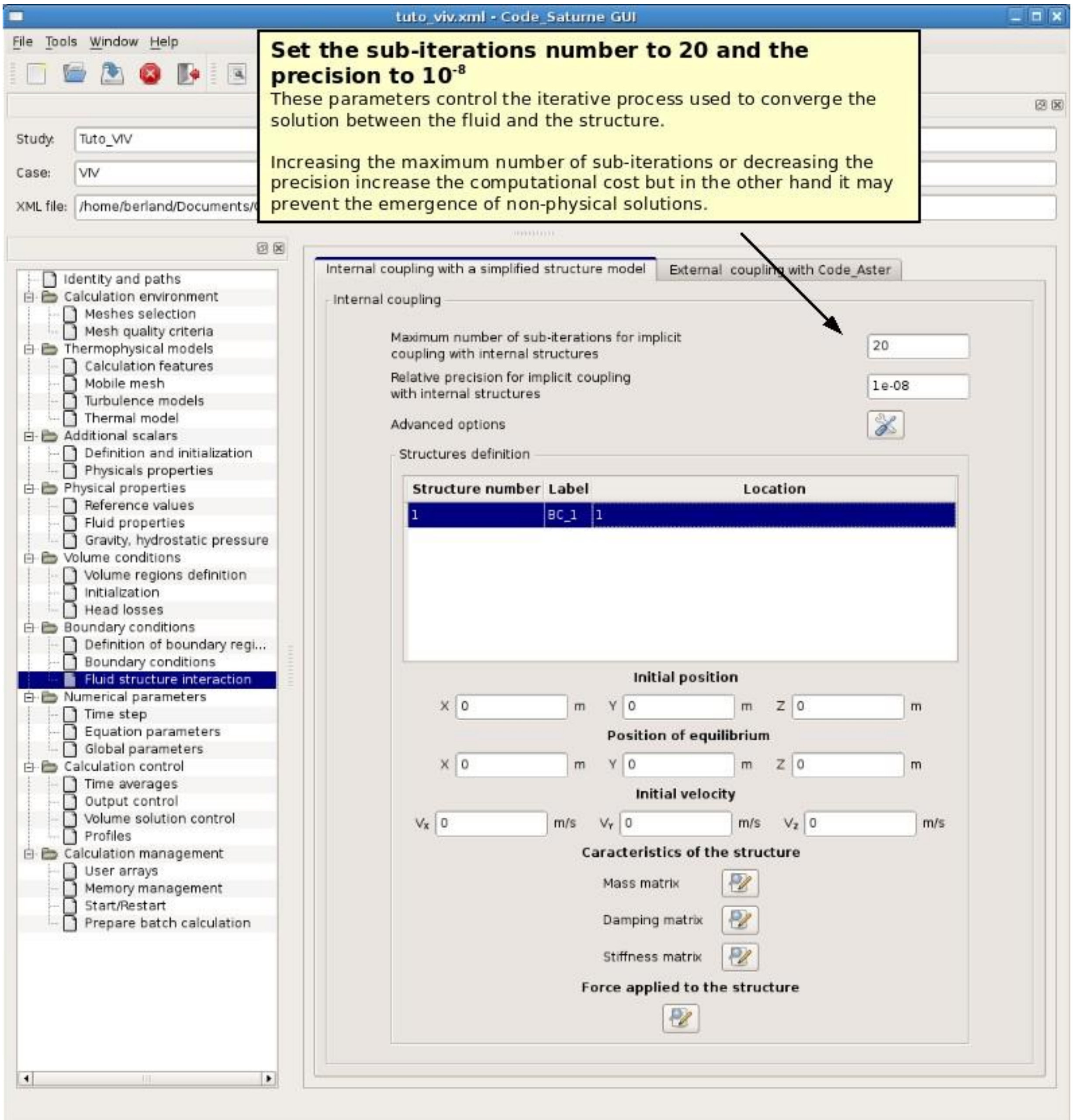

Figure 26: VIV test case. Fluid structure interaction.

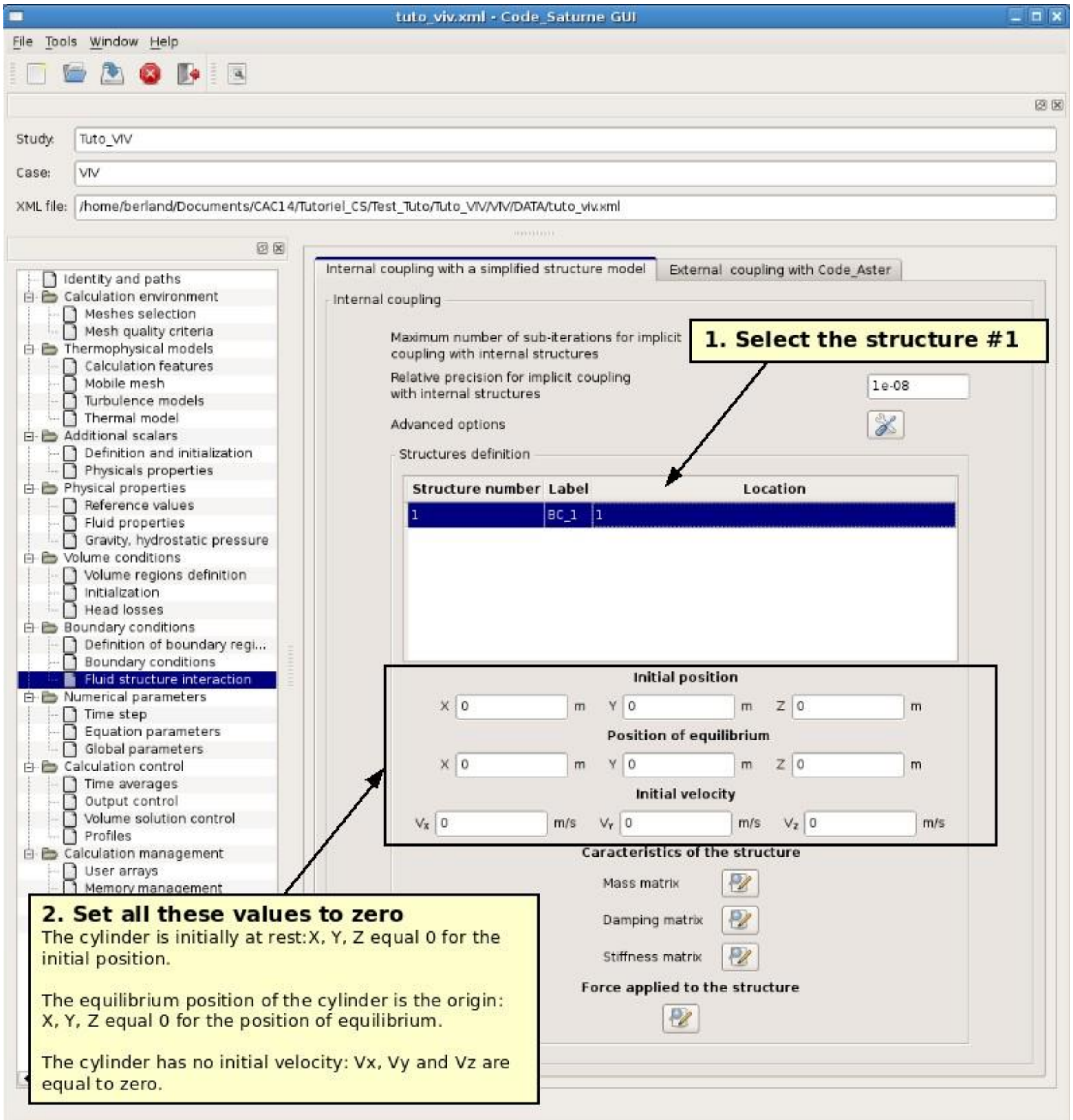

Figure 27: VIV test case. Fluid structure interaction.

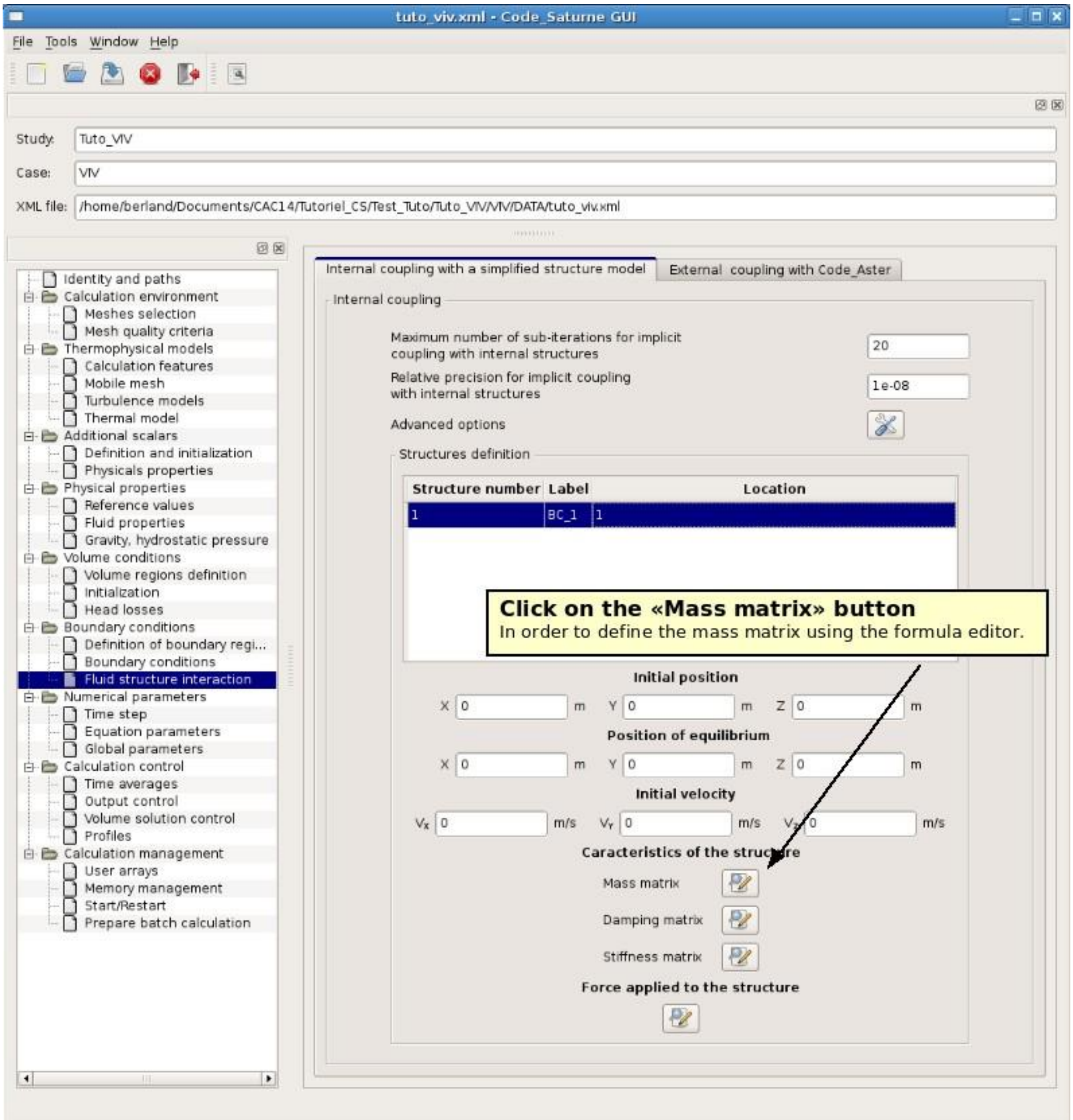

Figure 28: VIV test case. Fluid structure interaction.

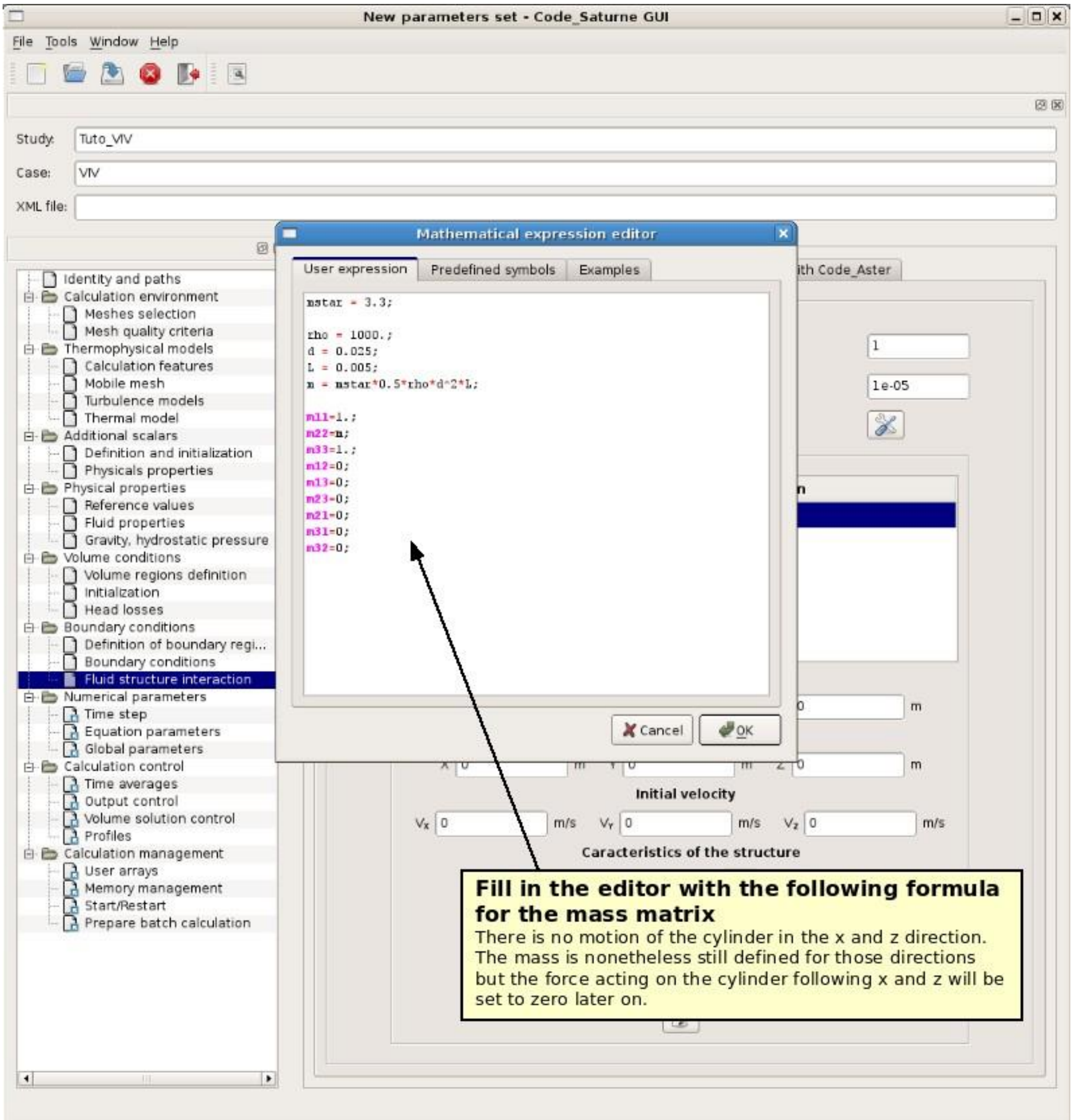

Figure 29: VIV test case. Fluid structure interaction.

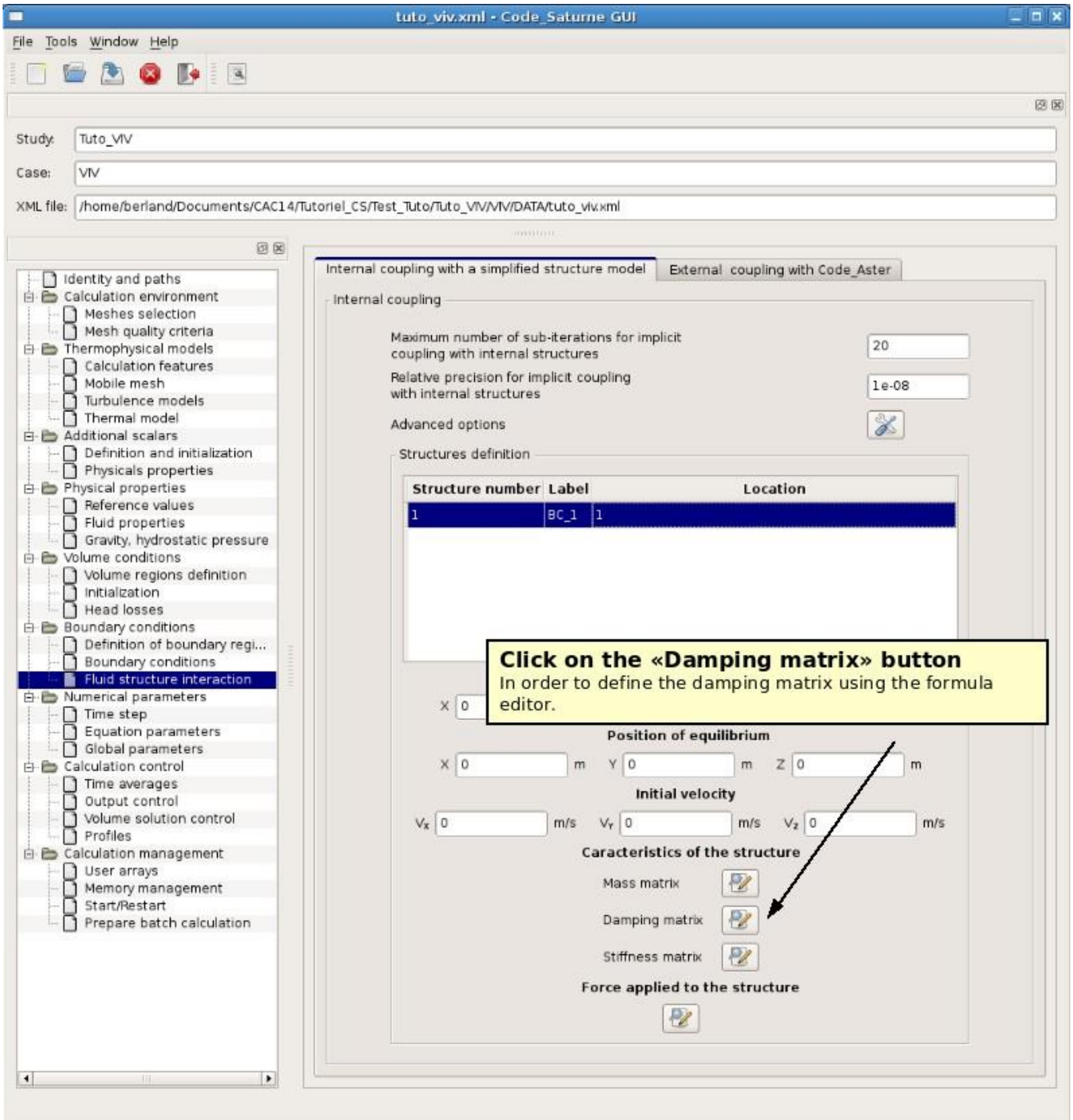

Figure 30: VIV test case. Fluid structure interaction.

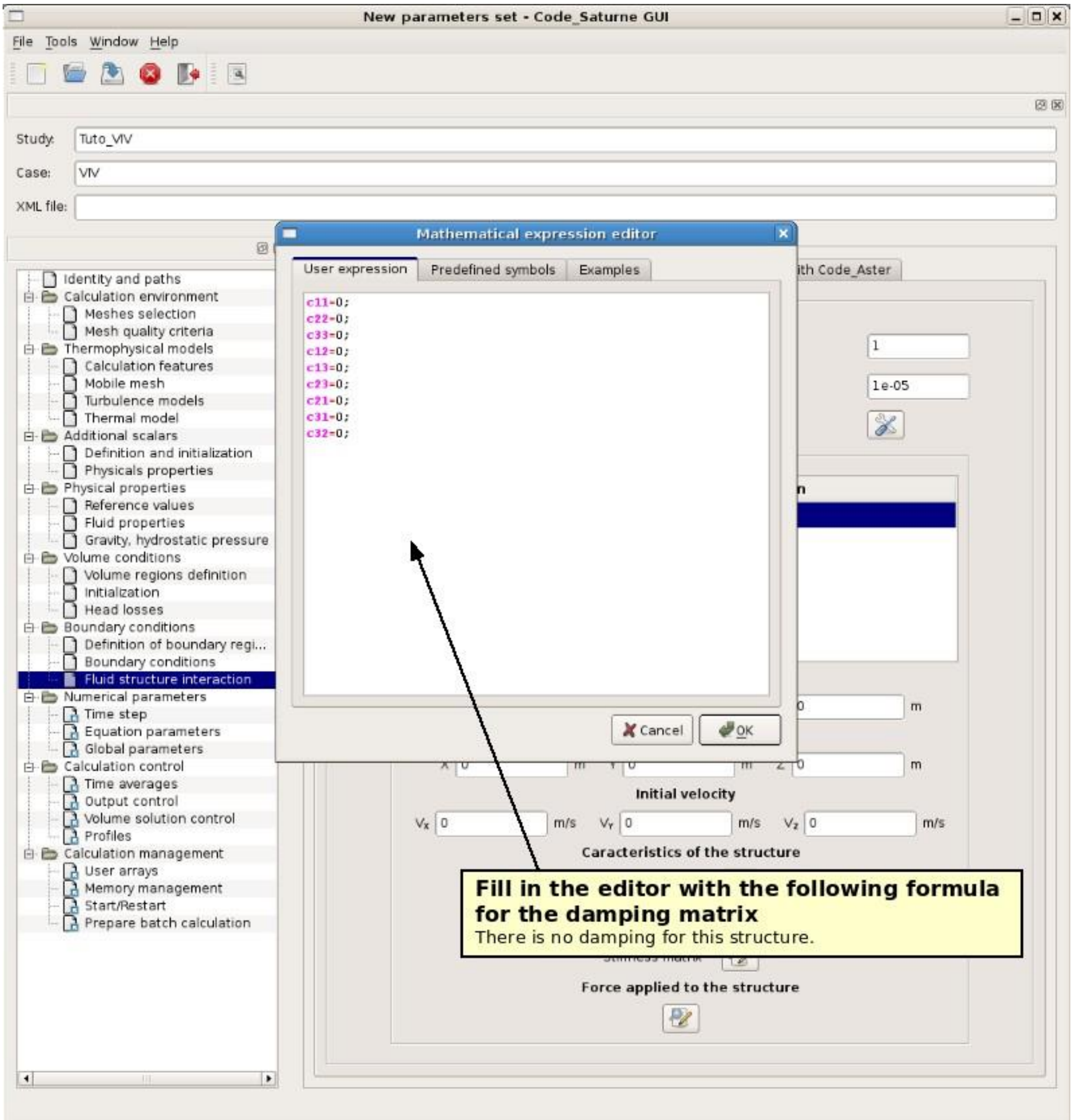

Figure 31: VIV test case. Fluid structure interaction.

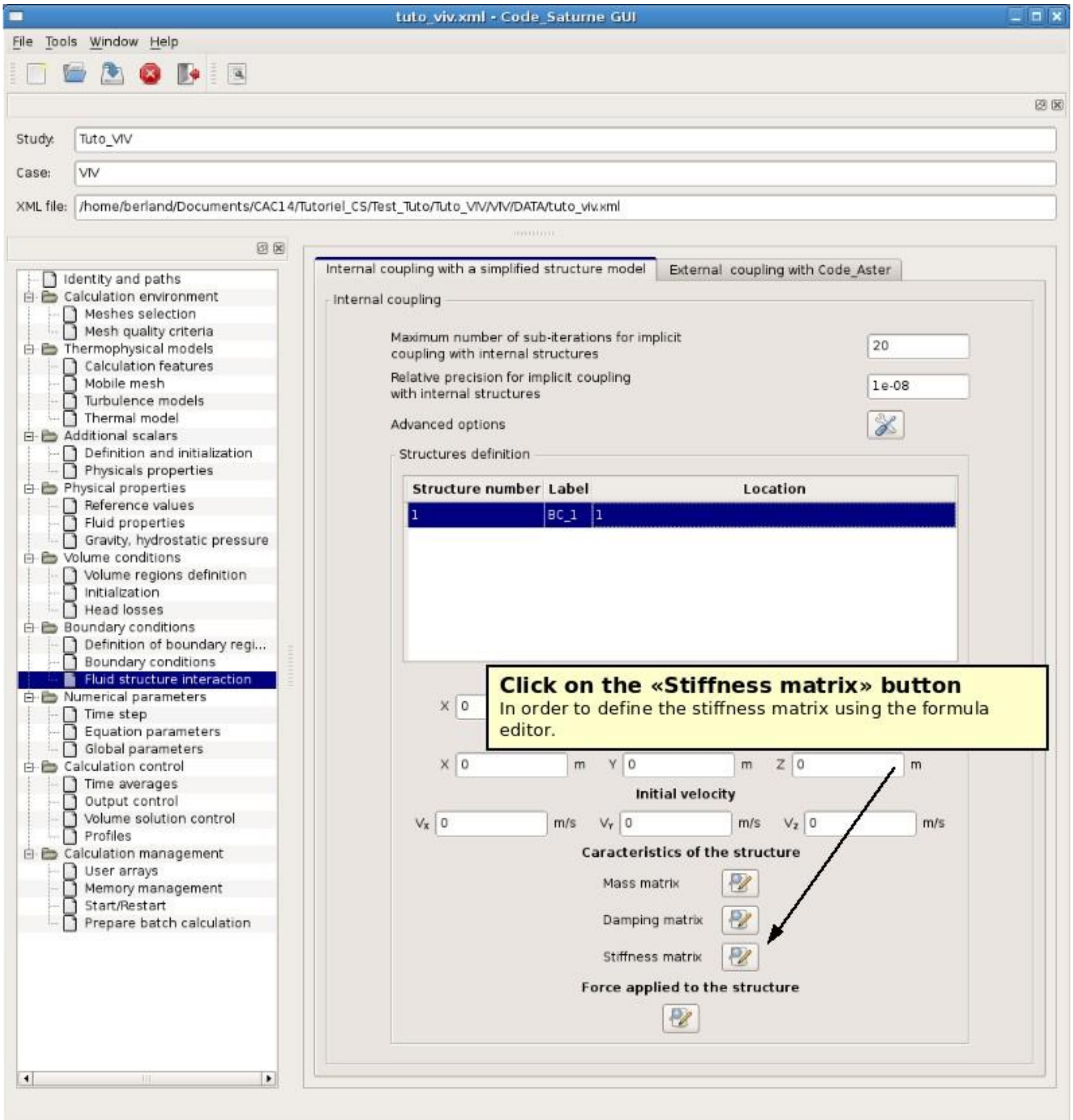

Figure 32: VIV test case. Fluid structure interaction.

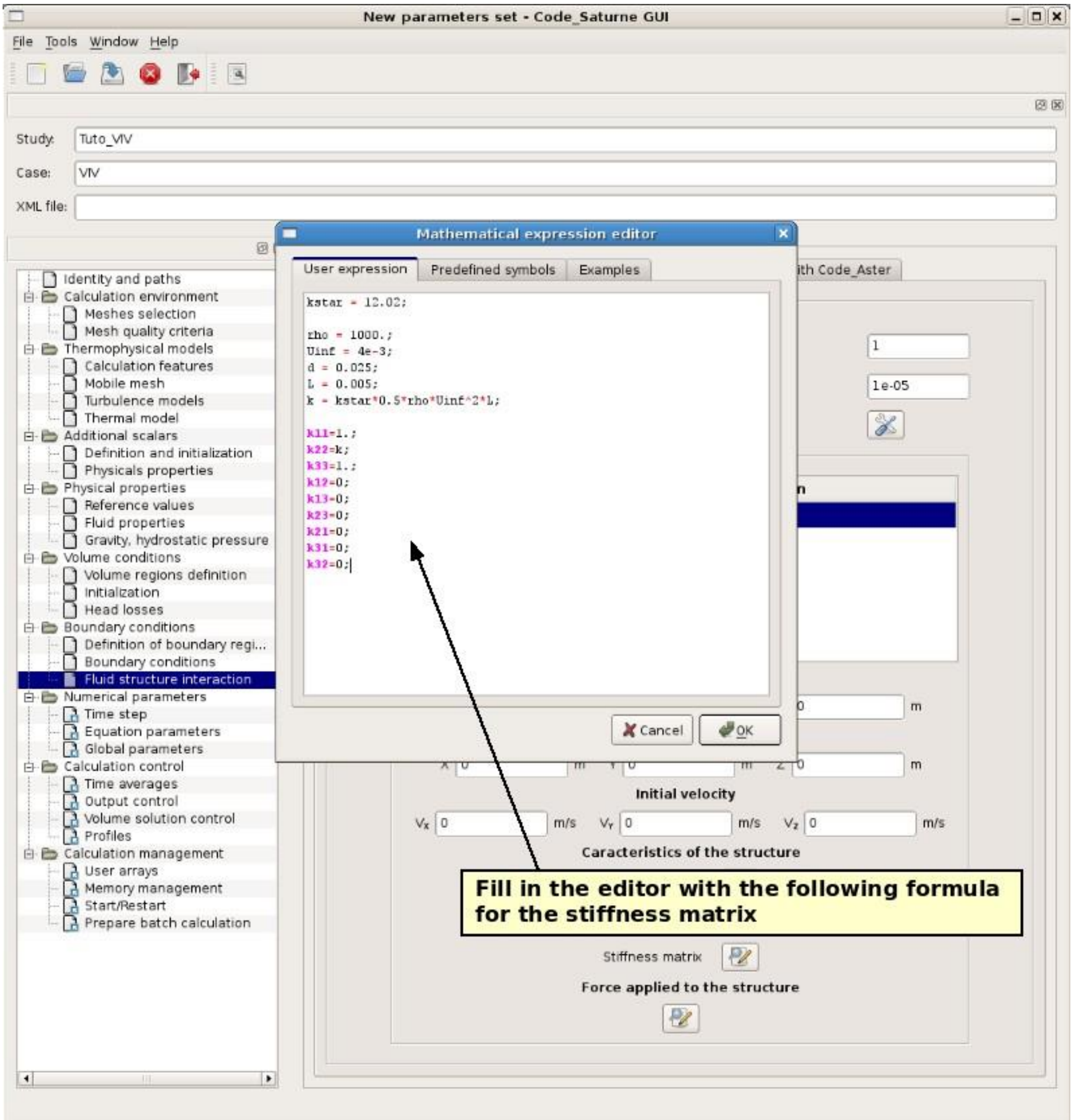

Figure 33: VIV test case. Fluid structure interaction.

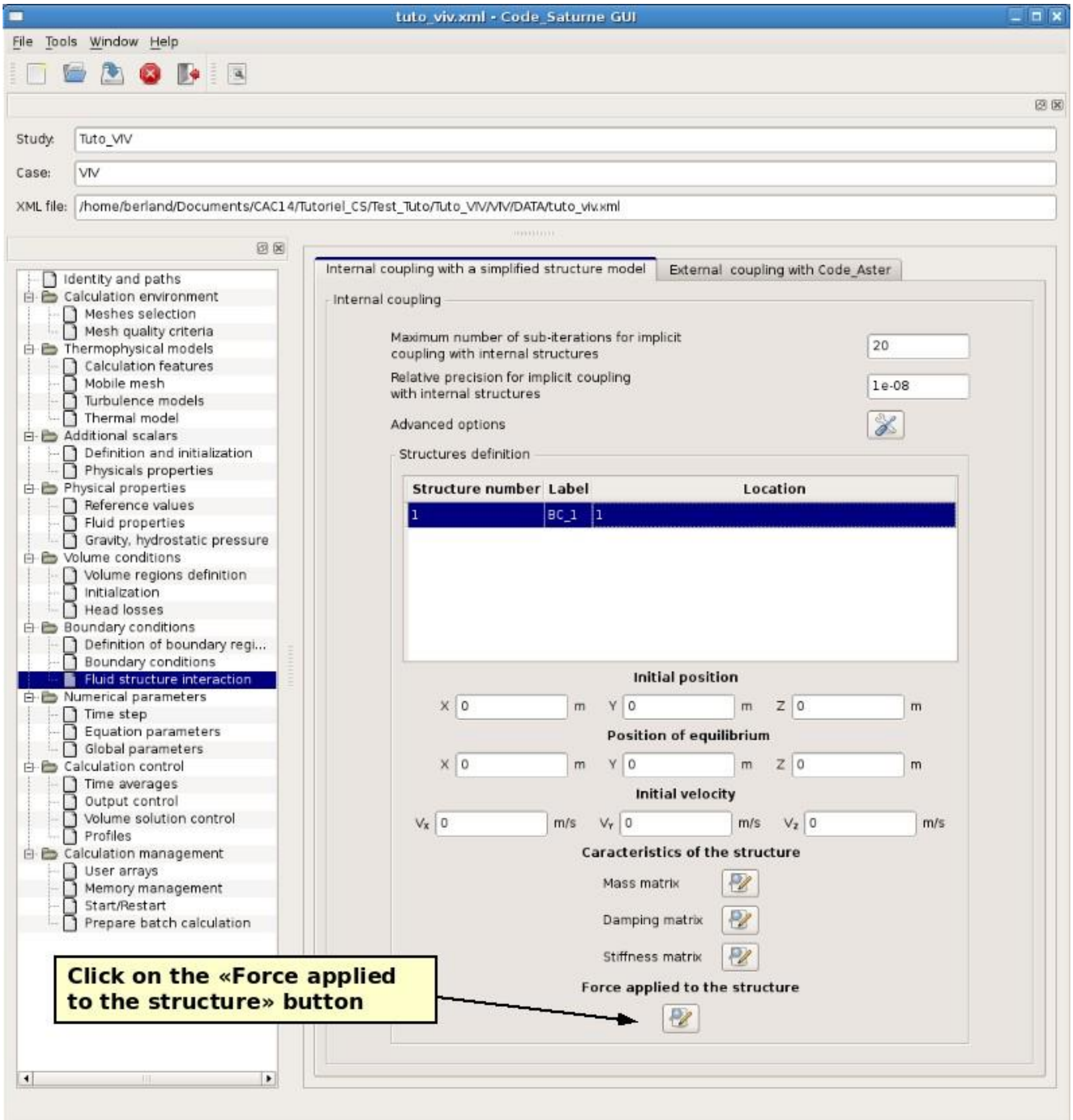

Figure 34: VIV test case. Fluid structure interaction.

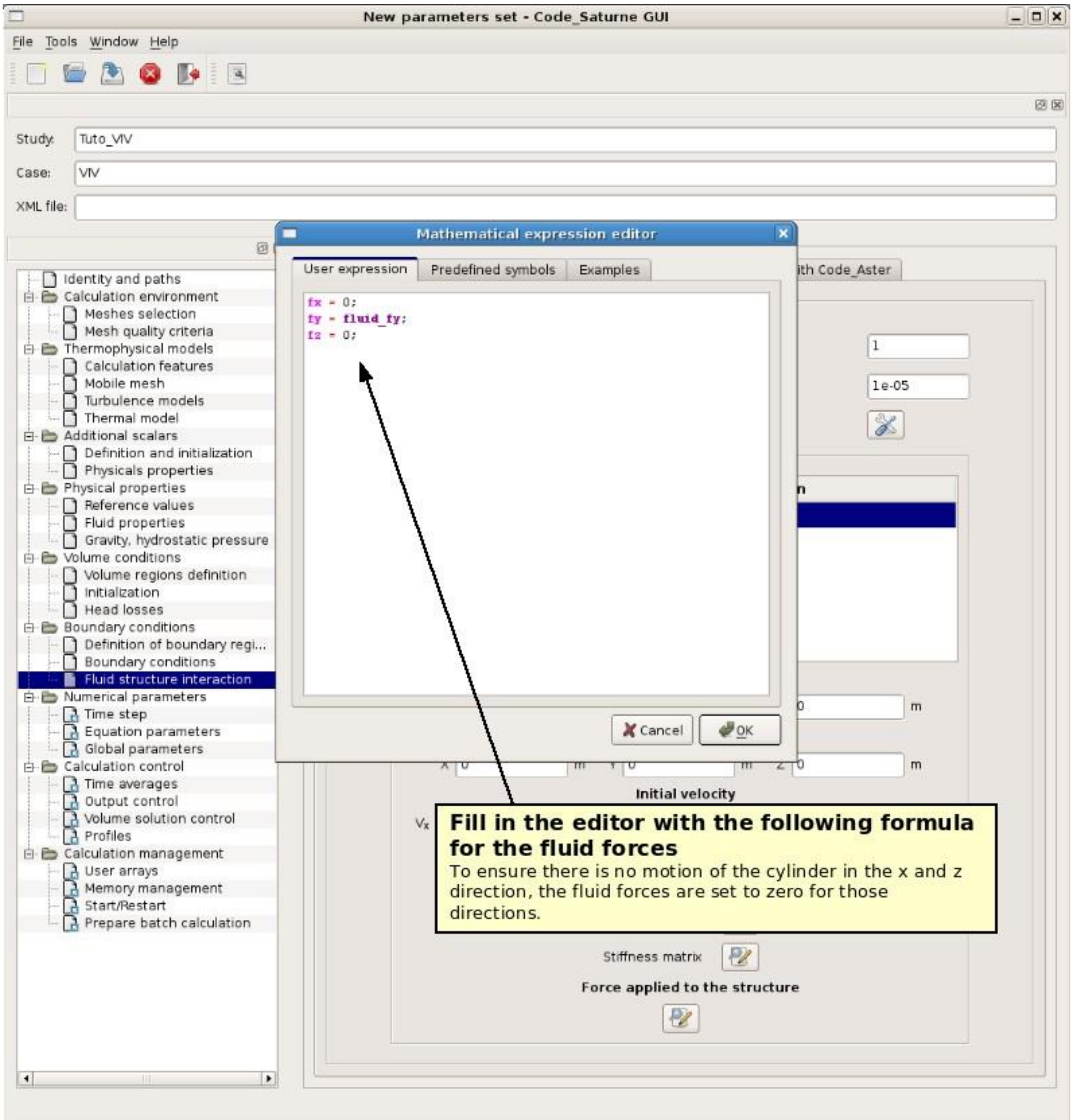

Figure 35: VIV test case. Fluid structure interaction.

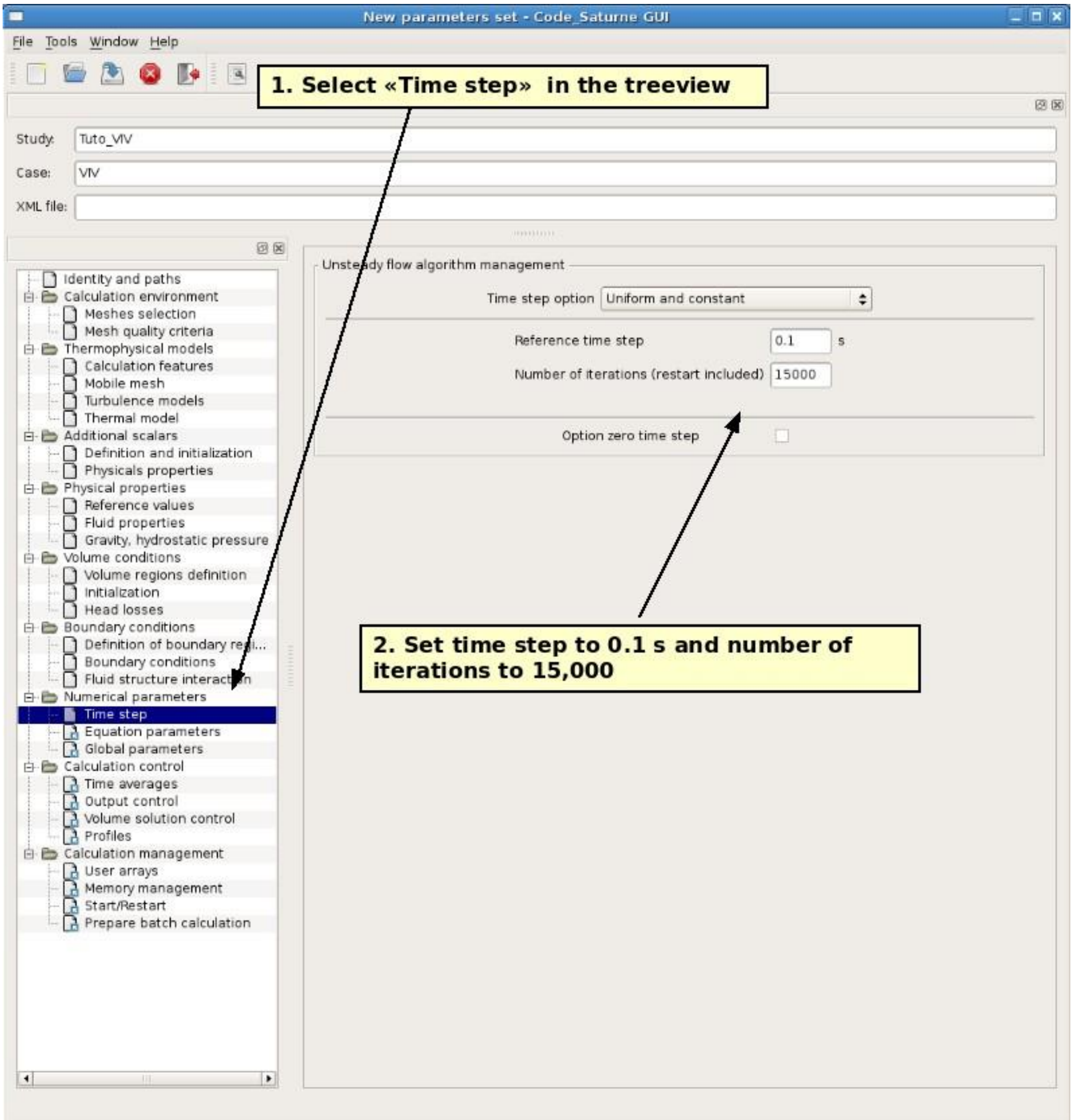

Figure 36: VIV test case. Time step.

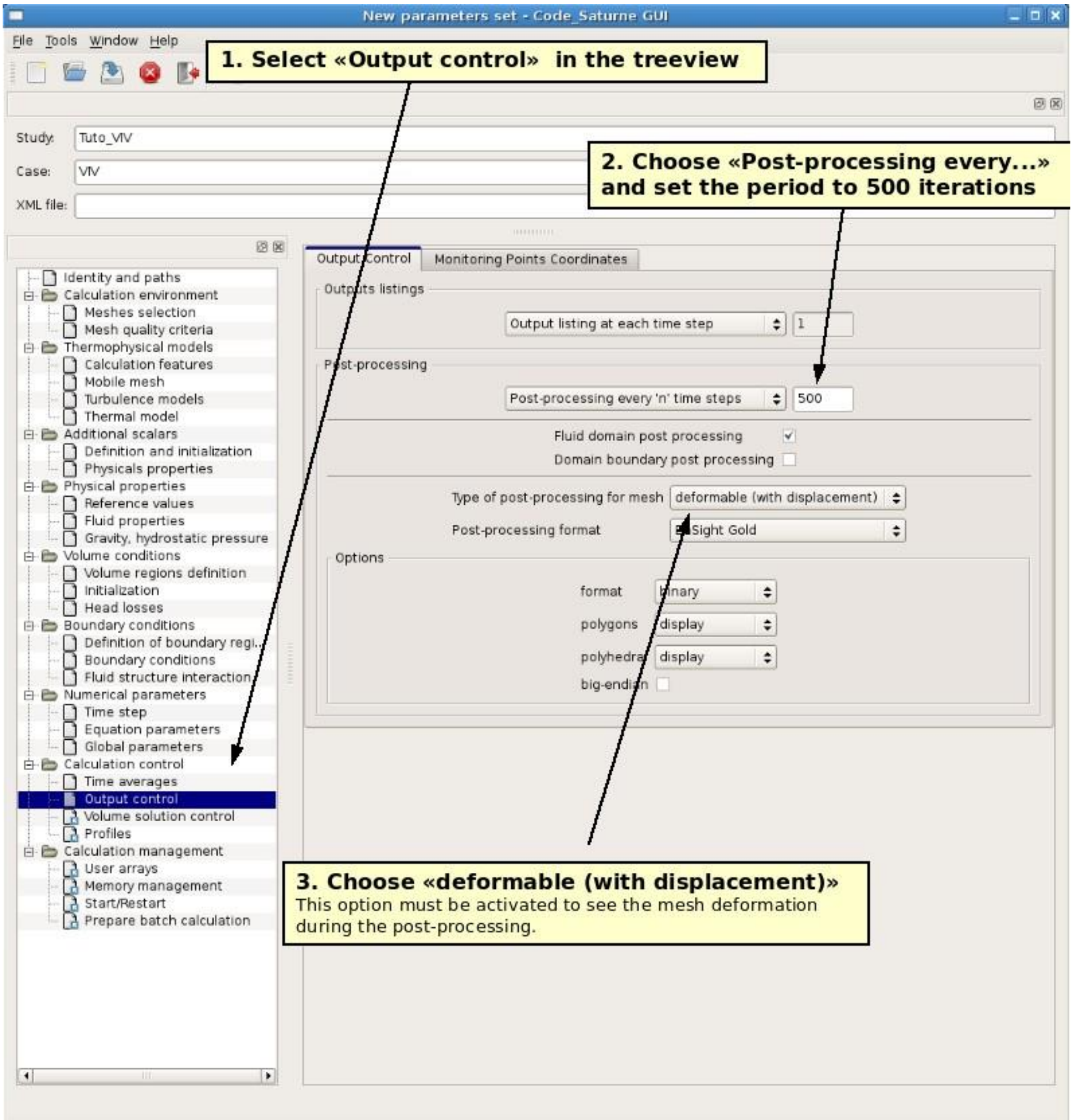

Figure 37: VIV test case. Output control.

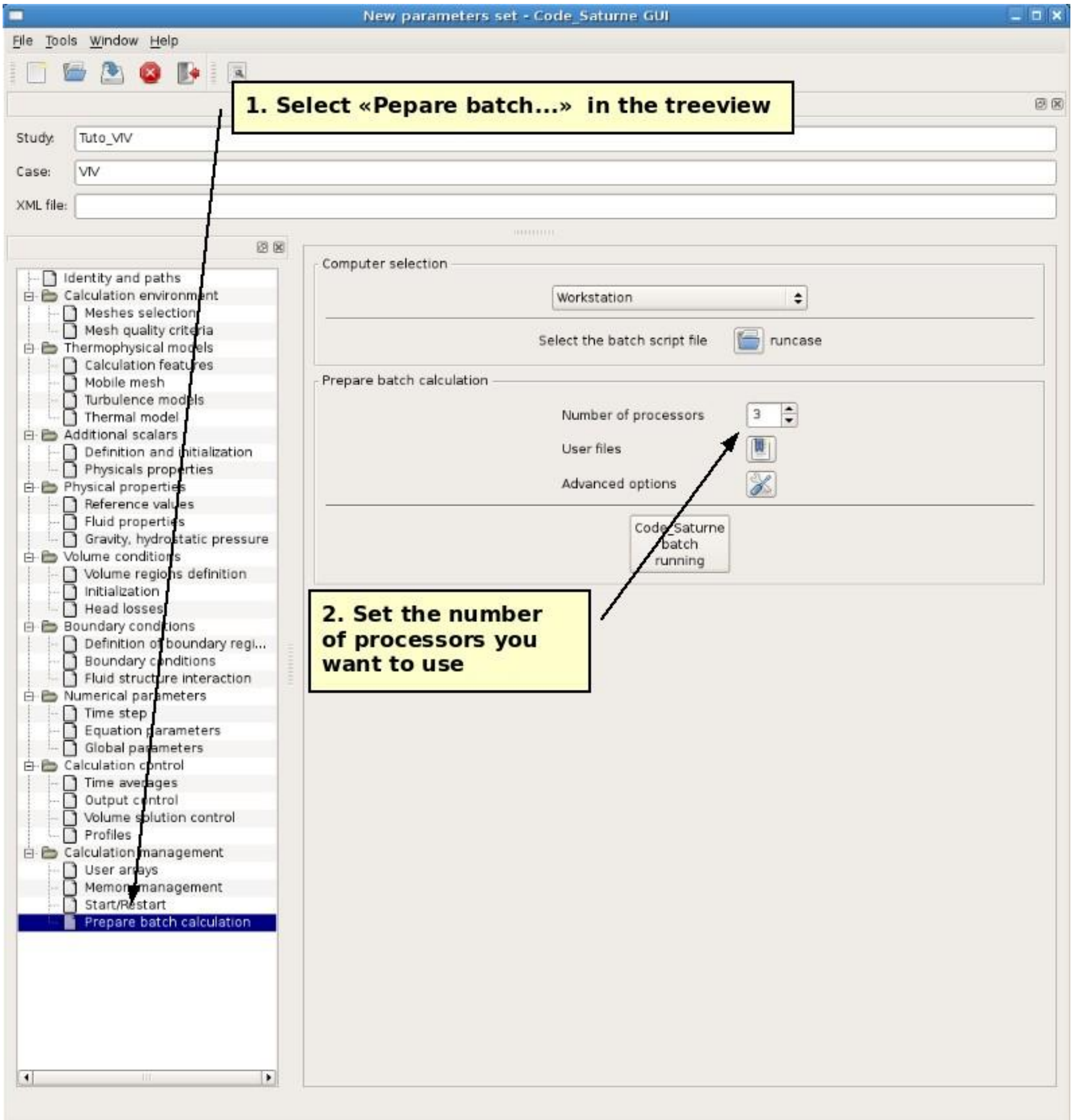

Figure 38: VIV test case. Prepare batch calculation.

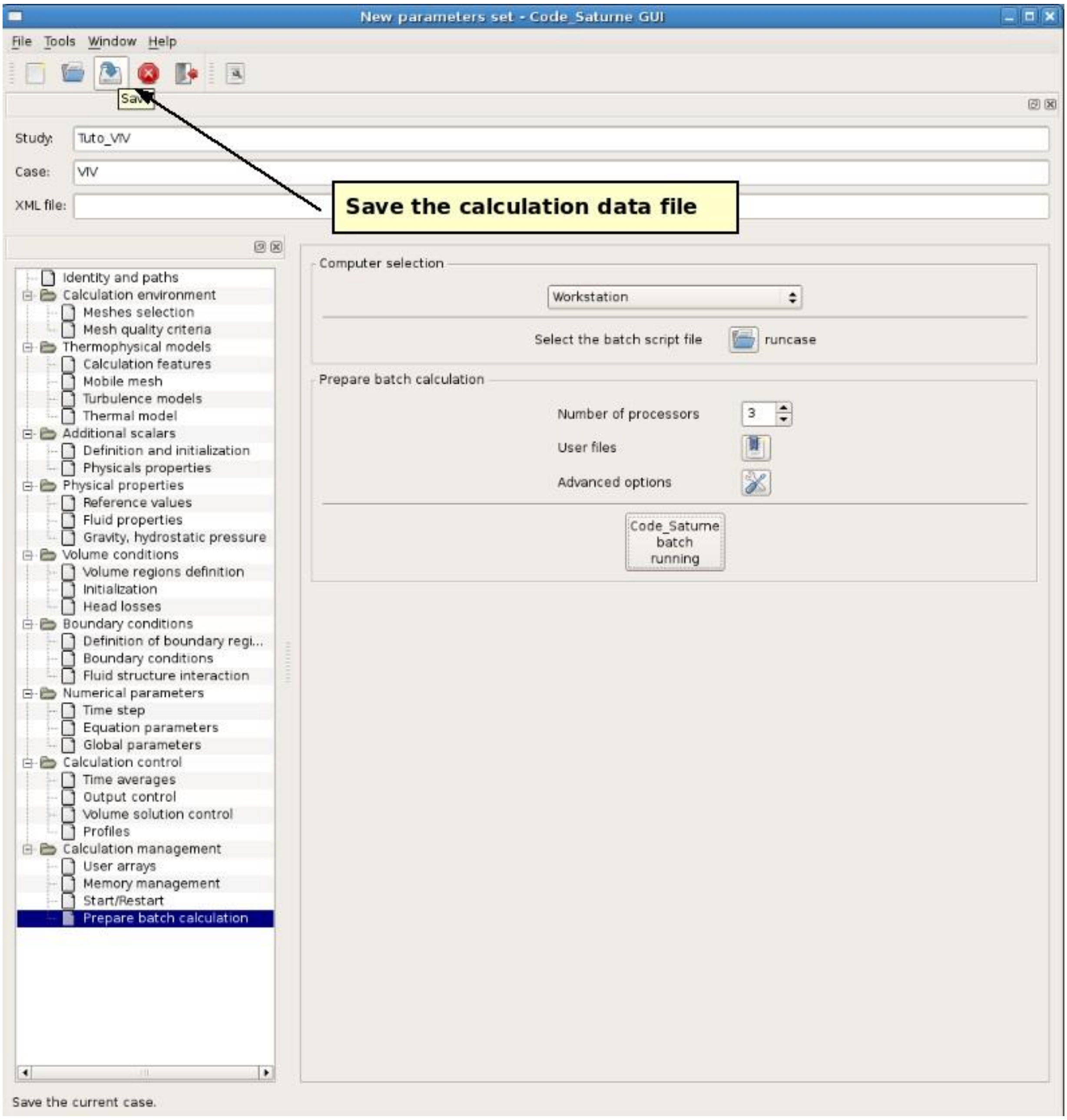

Figure 39: VIV test case. Prepare batch calculation.

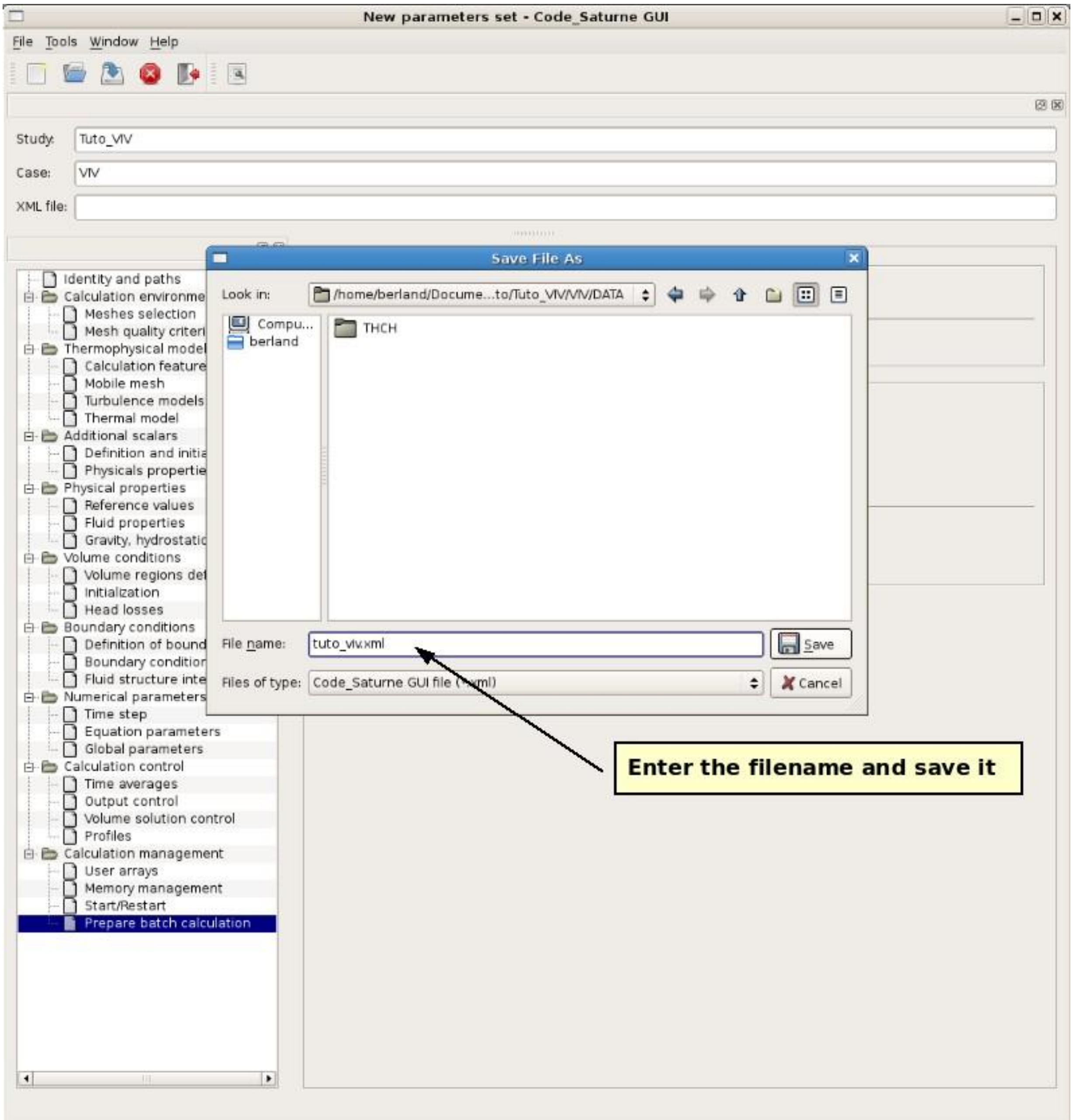

Figure 40: VIV test case. Prepare batch calculation.

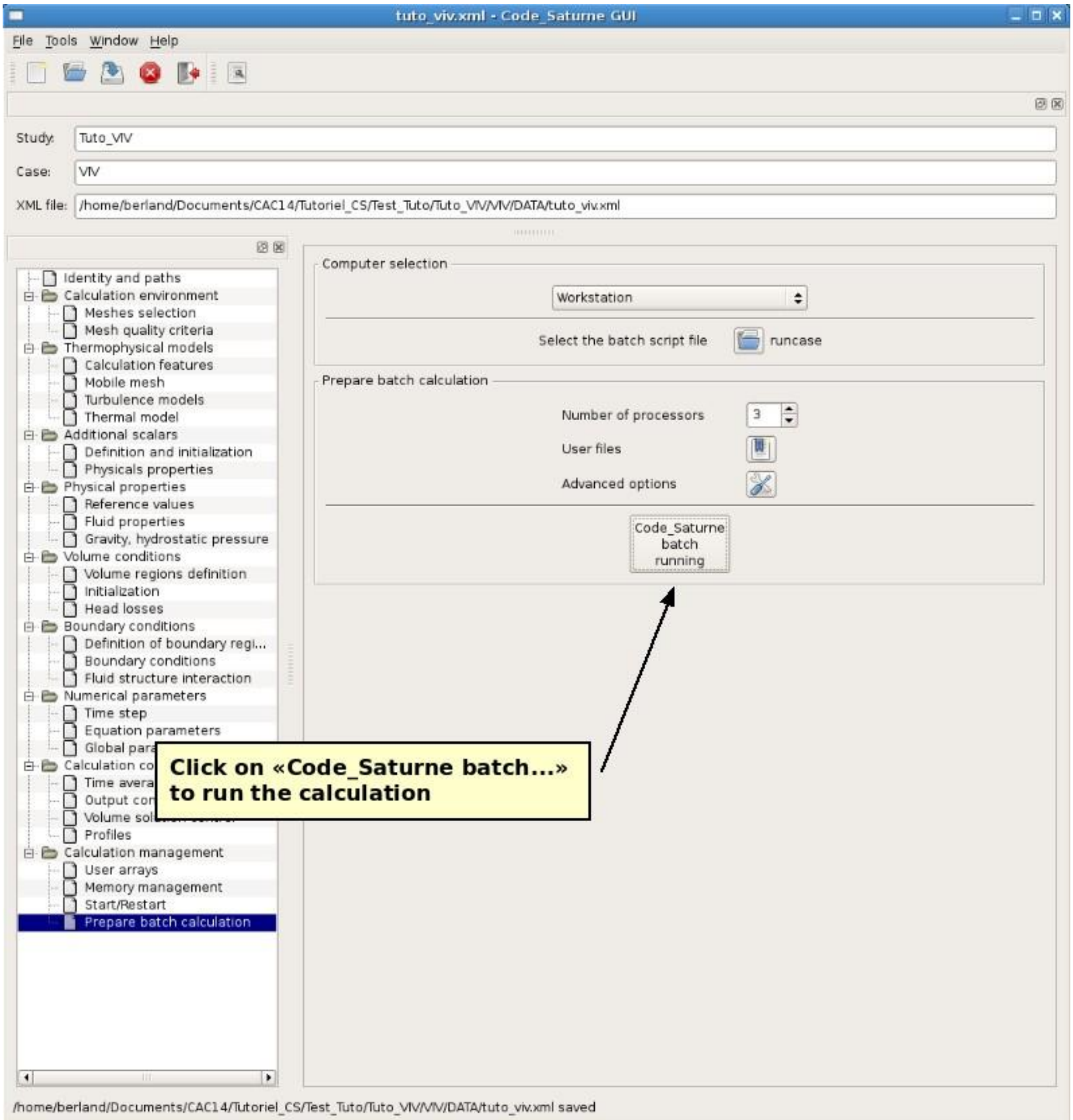

Figure 41: VIV test case. Prepare batch calculation.

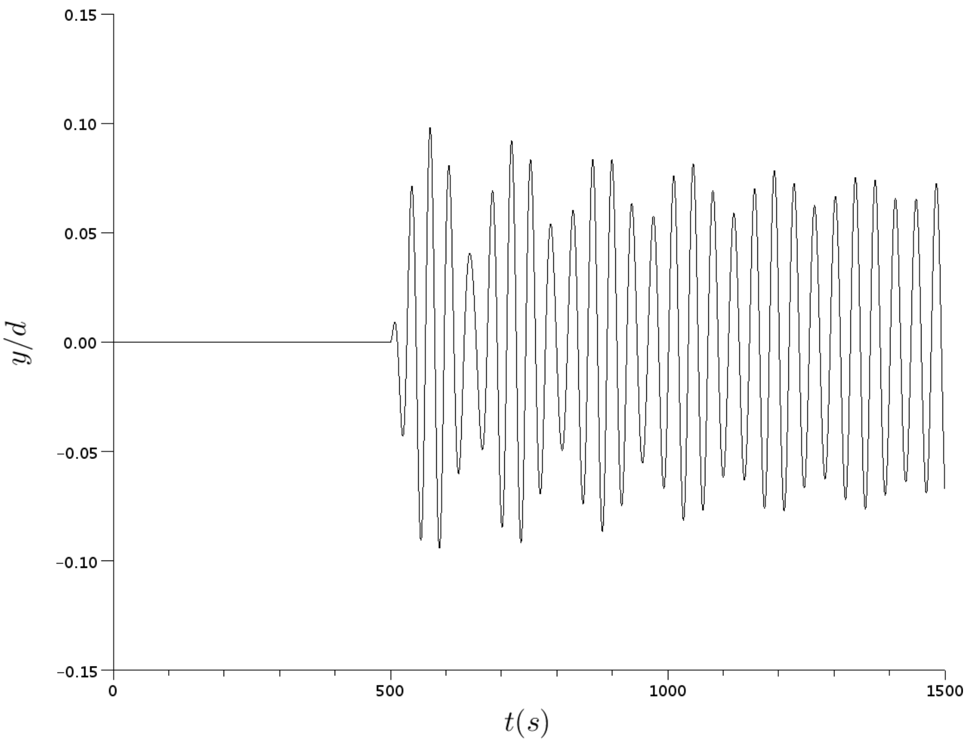

Figure 42: VIV test case. Time history of the structure displacement in the y direction.

3.2 How to setup a case involving fluid-structure interactions with user sub- routines

In order to fully define the present test case the following source files are required and must be copied in the `SRC` directory:

- `usini1.f90`: definition of the standard calculation parameters;
- `usclim.f90`: definition of the nature of the boundary conditions;
- `usalin.f90`: definition of the specific parameters of the ALE (to turn it on and provide the parameters of the internal fluid-structure coupling);
- `usalcl.f90`: definition of the behavior of the boundary regions when the mesh is mobile (fixed b.c., sliding b.c.......);
- `usstru.f90`: to link a mass-spring structure to a boundary condition ("internal" fluid-structure coupling);
- `usvima.f90`: definition of the mesh viscosity.

3.2.1 Standard parameters definition

The standard calculation parameters are defined in the user source file `usini1.f90`. Nonetheless, these variables will not be described here since they fall outside the scope of a tutorial on fluid-structure interaction in *Code_Saturne*. However, as concern the outputs, one should not forget to set the keyword `ichrmd` to 1:

```
ichrmd = 1
```

in order to obtain output data on a deformable mesh. This feature makes possible to observe the mesh motions with a visualization software (Ensight or Paraview for instance).

3.2.2 Boundary conditions

In a similar manner, even though a mobile mesh is used, the setup of the boundary conditions does not require any specific treatment in the file `usclim.f90`. This source file will not be described here.

3.2.3 Definitions specific to the ALE

The features of the mobile mesh procedure, as well as those of the fluid-structure internal coupling algorithm, are defined in the user source file `usalin.f90`. For the present case the various parameters of the ALE should be set to the following values:

```
iale = 1      ! turn on the ALE
nalinf = 5000! number of iterations for fluid initialization
nalimx = 20   ! number of sub-cycling iteration for fluid-structure
interaction epalim = 1.d-8 ! relative precision of sub-cycling fluid-
structure coupling iortvm = 0     ! isotropic mesh viscosity
```

As pointed out earlier in section 3.1 the key parameters are:

- `iale`: equals 0 when the mobile mesh feature is deactivated and 1 when it is activated;
- `nalinf`: is the number of iteration before the fluid-structure coupling is actually performed. Some iterations might indeed be necessary in order to have a well-established flow before allowing any motion of the structure;
- `nalimx` and `epalim`: control the convergence of the iterative resolution of the fluid-structure coupling. These parameters should be chosen carefully to avoid any unphysical behavior of the solution (see for instance section 3.8);
- `iortvm`: defined whether the mesh viscosity is orthotropic (the mesh viscosity is a vector) or isotropic (the mesh viscosity is the same in all three directions).

3.2.4 Mobile mesh boundary conditions

Since the mesh is mobile one can define if a boundary region is fixed or mobile. These definitions can be done in the source file `usalcl.f90`. For each face `ifac` of the mesh, the keyword `ialtyb(ifac)` can be set to `igliss` if the mesh is sliding, or to `ibfixe` if the mesh is fixed. In our present case, only boundary regions 10 and 11, which corresponds to the front and back frontiers of the mesh, are sliding:

```
! --- For boundary faces of color 10 and 11 assign a sliding boundary
call getfbr('10 and 11', nlelt, lstelt)
!==========

do ilelt = 1, nlelt
   ifac = lstelt(ilelt) ialtyb(ifac) = igliss
enddo

! --- prescribe elsewhere a fixed boundary
call getfbr('not (10 or 11)', nlelt, lstelt)
!==========

do ilelt = 1, nlelt
   ifac = lstelt(ilelt)
   ialtyb(ifac) = ibfixe
enddo
```

The final source file is provided in the tutorial directory `Tutorial_Files/User_Sources`.

3.2.5 Structure definition

The mechanical properties of the structure and the definition of the forces acting on it can be provided by the user in the file `usstru.f90`.

The first subroutine `usstr1` is called at the beginning of the calculation. It first permits to link a structure to a boundary region. For this structure a few parameters mau be defined:

- `xstr0(idim,istructure)`: initial position of the structure (`idim=1,2,3` is the index of the dimension and `istructure` the number of thestructure);
- `vstr0(idim,istructure)`: initial velocity of the structure;
- `xstreq(idim,istructure)`: equilibrium position of the structure;
- `aexxst`, `bexxst` and `cfopre`: advanced users may also change some parameters of the internal fluid-structure coupling procedure.

For the present test case the source code for this subroutine should read:

```
! --- Assign faces of boundary region 1 to a structure
call getfbr('1', nlelt, lstelt)

do ilelt = 1, nlelt
   ifac = lstelt(ilelt)
   idfstr(ifac) = 1
enddo

! --- Some structure parameters
xstr0(2,1)  = 0.d0
xstreq(2,1)  = 0.d0
vstr0(3,2)  = 0.d0

! --- Here one can modify the values of the prediction coefficients for
!     displacements and fluid forces in internal FSI coupled algorithm.
aexxst = 0.5d0
bexxst = 0.0d0
cfopre = 2.d0

! --- Activation of structural history output
ihistr = 1
```

Then, in subroutine `usstr2` which is called at each time step the mass, stiffness and damping of the structure, as well as the forces acting on the structure can de defined:

- `xmstru(i,j,istr)`,`xcstru(i,j,istr)` and `xkstru(i,j,istr)` correspond to the mass, damping and stiffness matrices of the structure number `istru`;
- `forstr(idim,istr)` is the force acting on the structure. By default this vector is filled in with the fluid-forces but the user can modify these values if needed.

As concerns the case treated in this tutorial the source code of the subroutine is given by:

```
! --- Define matrices istr = 1

cyl_m = 5.16e-3
cyl_k = 4.81e-4

xmstru(1,1,istr) = 1.
xmstru(2,2,istr) = cyl_m
xmstru(3,3,istr) = 1.

xcstru(1,1,istr) = 0.
xcstru(2,2,istr) = 0.
xcstru(3,3,istr) = 0.

xkstru(1,1,istr) = 1.
xkstru(2,2,istr) = cyl_k
xkstru(3,3,istr) = 1.

! --- Define forces acting on the structure
! --- only the force in the lift direction (2)
! --- is kept. The others are set to zero istr = 1
forstr(1,istr) = 0.
forstr(3,istr) = 0.

! --- Structural time step istr = 1
dtstr(istr) = dtcel(1)
```

The final source file is provided in the tutorial directory `Tutorial_Files/User_Sources`.

3.2.6 Mesh viscosity definition

A shown in figure 11, the mesh viscosity is set to a very high value close to the cylinder whereas it is equal to 1 away from the surface of the cylinder. This can be done using the source file `usvima.f90` with the following code: in the declarations add the line,

```
double precision rad, xr2, xcen, ycen
```

and the subroutine body should contain:

```
if (ntcabs.eq.0) then
   rad = (0.025)**2
   xcen = 0.d0
   ycen = 0.d0
   do iel = 1, ncel
      xr2 = (xyzcen(1,iel)-xcen)**2 + (xyzcen(2,iel)-ycen)**2
      if (xr2.lt.rad) viscmx(iel) = 1.d10
   enddo
endif
```

The aim here is to loop over all the cells `(do iel = 1, ncel)` and apply a given mesh viscosity `(viscmx(iel))` according to the location of the center of the cell `(xyzcen(1,iel)` for x location and `xyzcen(2,iel)` for y location).

The viscosity is isotropic so that only `viscmx(iel)` needs to be defined. In the general case the three viscosity components, `viscmx(iel)`, `viscmy(iel)` and `viscmz(iel)` have to be provided by the user.

3.2.7 Runcase script

No special step need to be performed for the runcase when the mobile mesh features is turned on. Simply define the mesh used, and the number of processors required:

```
MESH='mesh_viv.des'
NUMBER_OF_PROCESSORS=3
```

3.3 How to impose the displacement of the structure (GUI)

Instead of using the internal coupling to obtain the motions of the structure one may want to impose the displacement of the cylinder. The steps required in the GUI to perform such an operation are explained in figures 43 to 48. Note that the steps described in section 3.1 also need to be performed to setup the calculation.

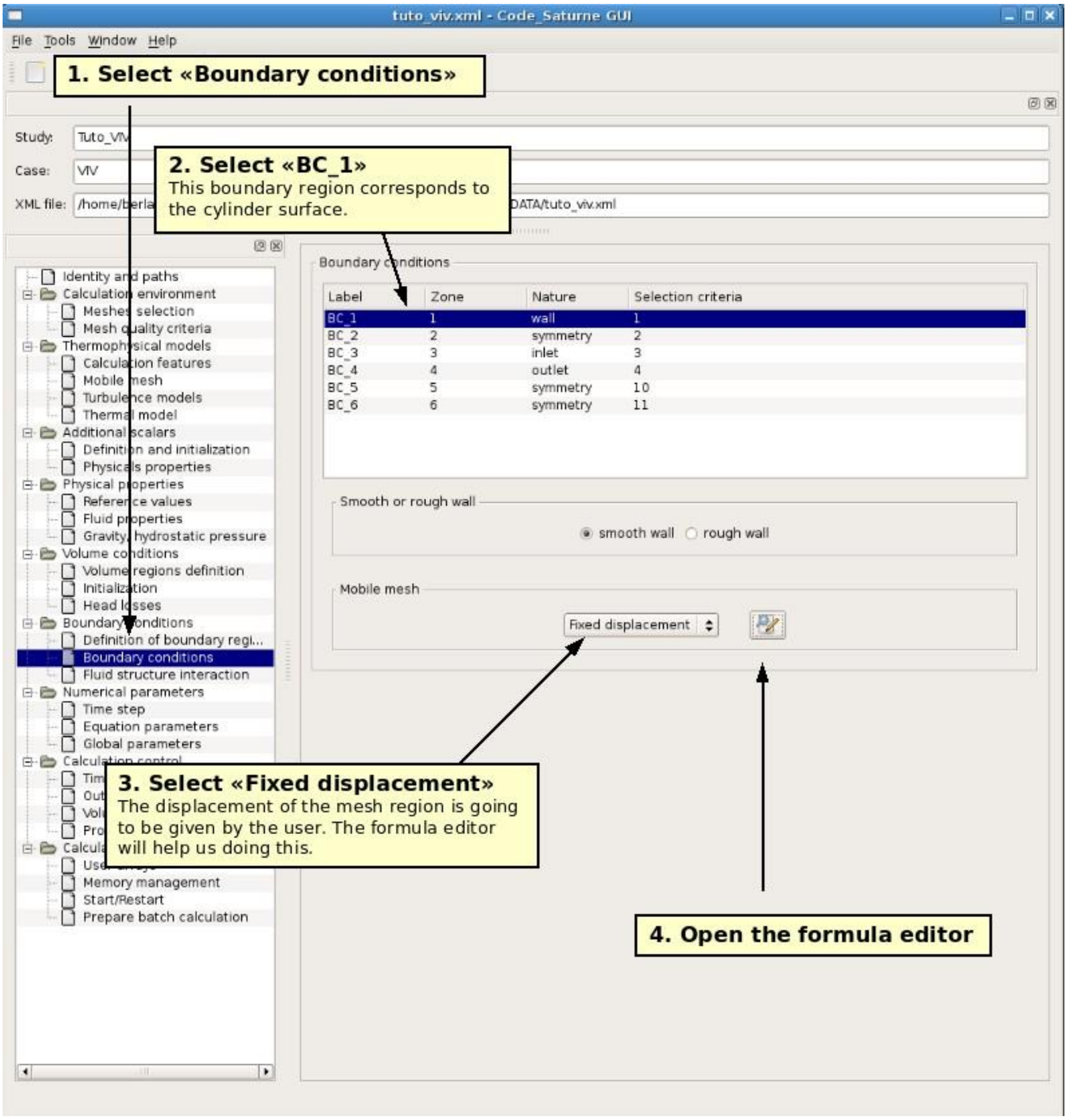

Figure 43: Imposed displacement case. Boundary conditions.

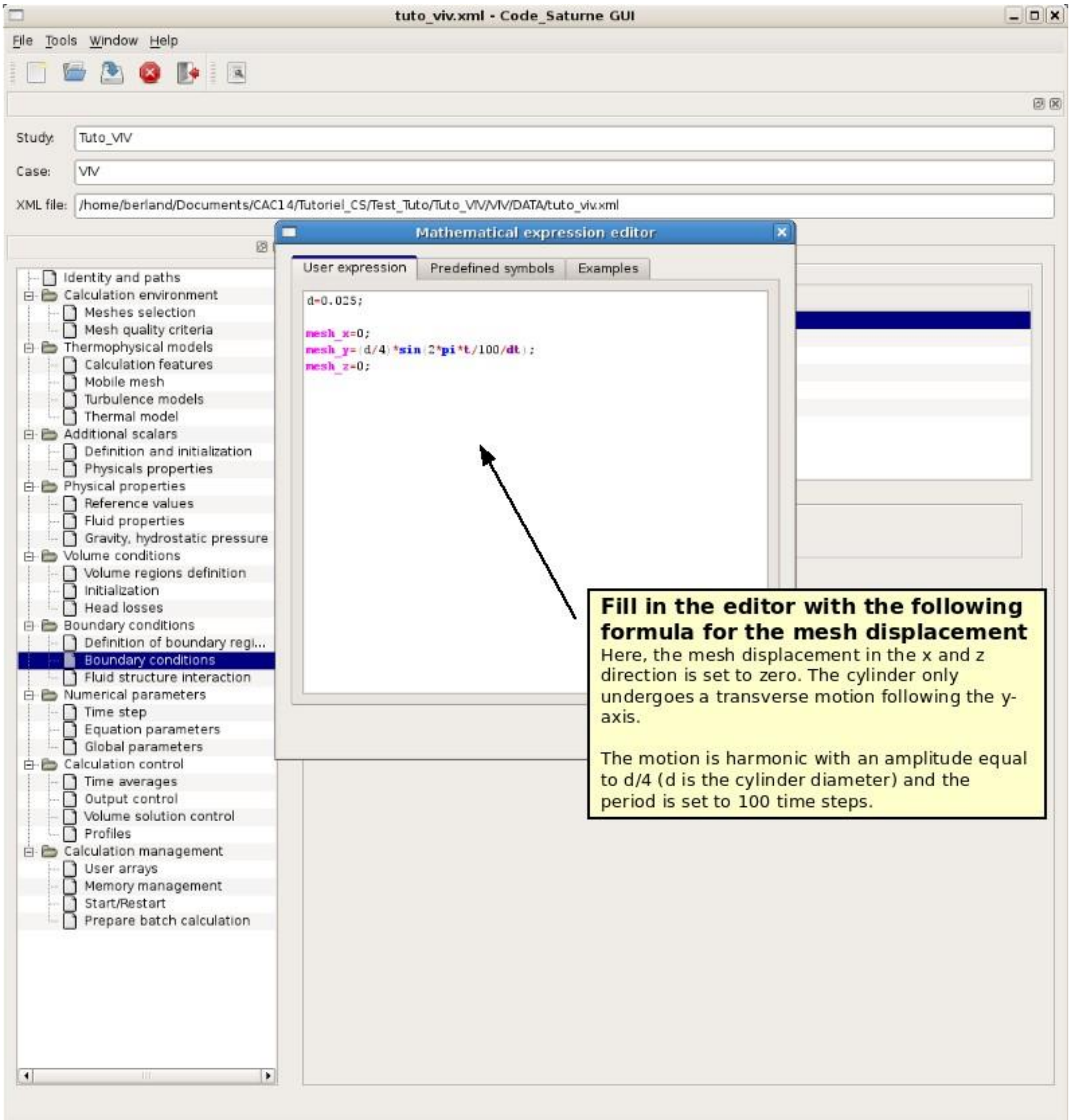

Figure 44: Imposed displacement case. Boundary conditions.

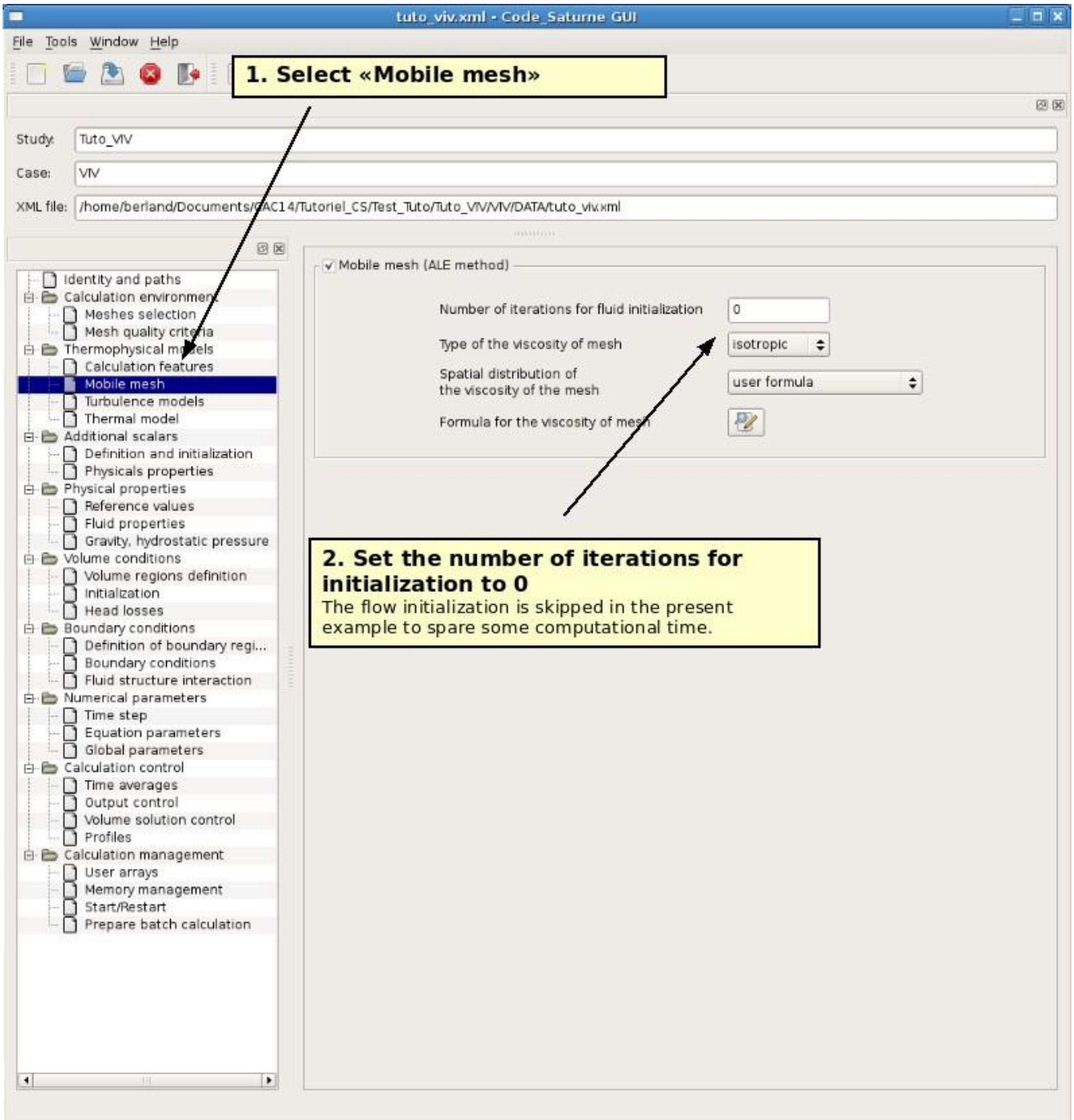

Figure 45: Imposed displacement case. Mobile mesh.

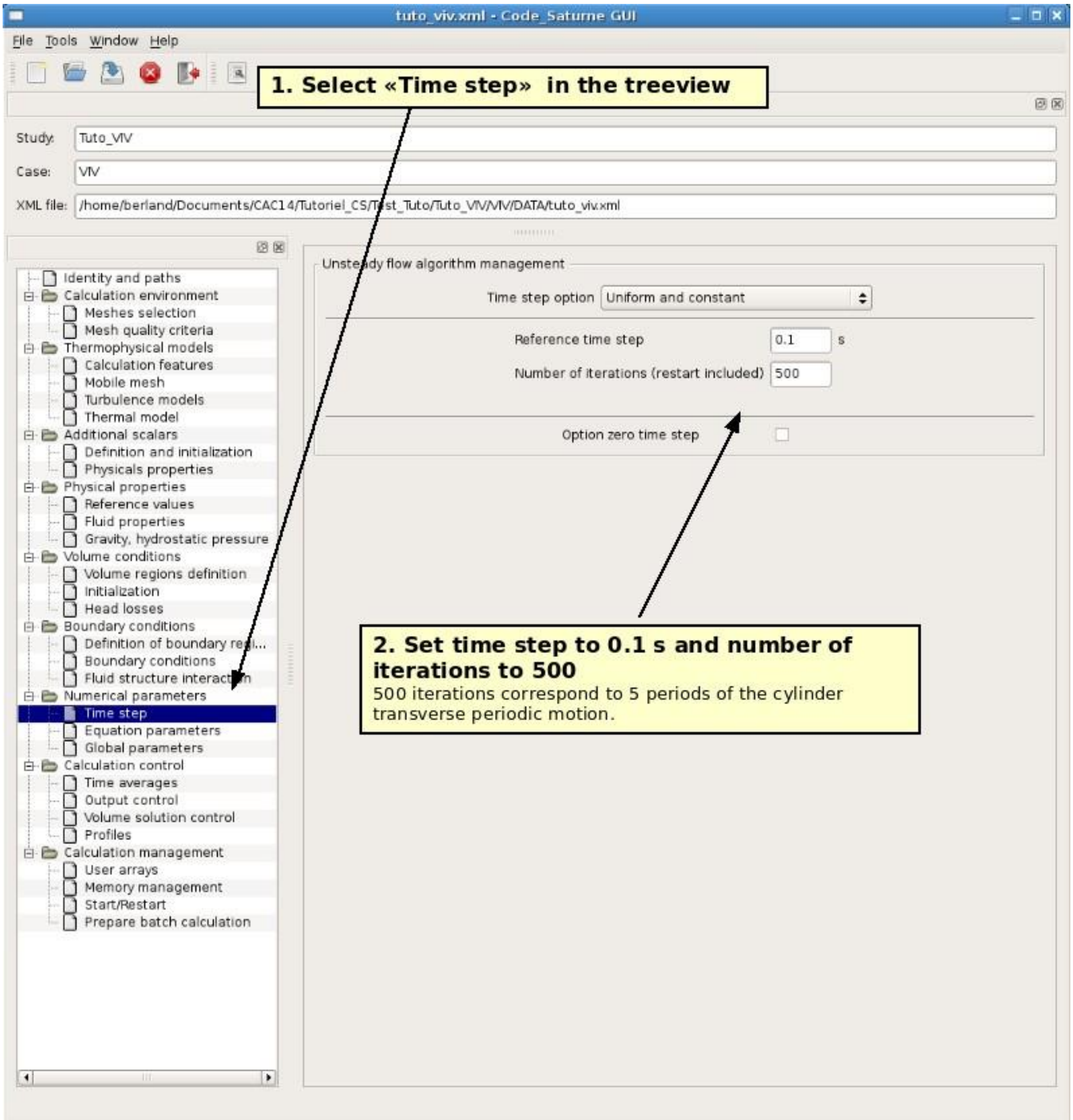

Figure 46: Imposed displacement case. Time step.

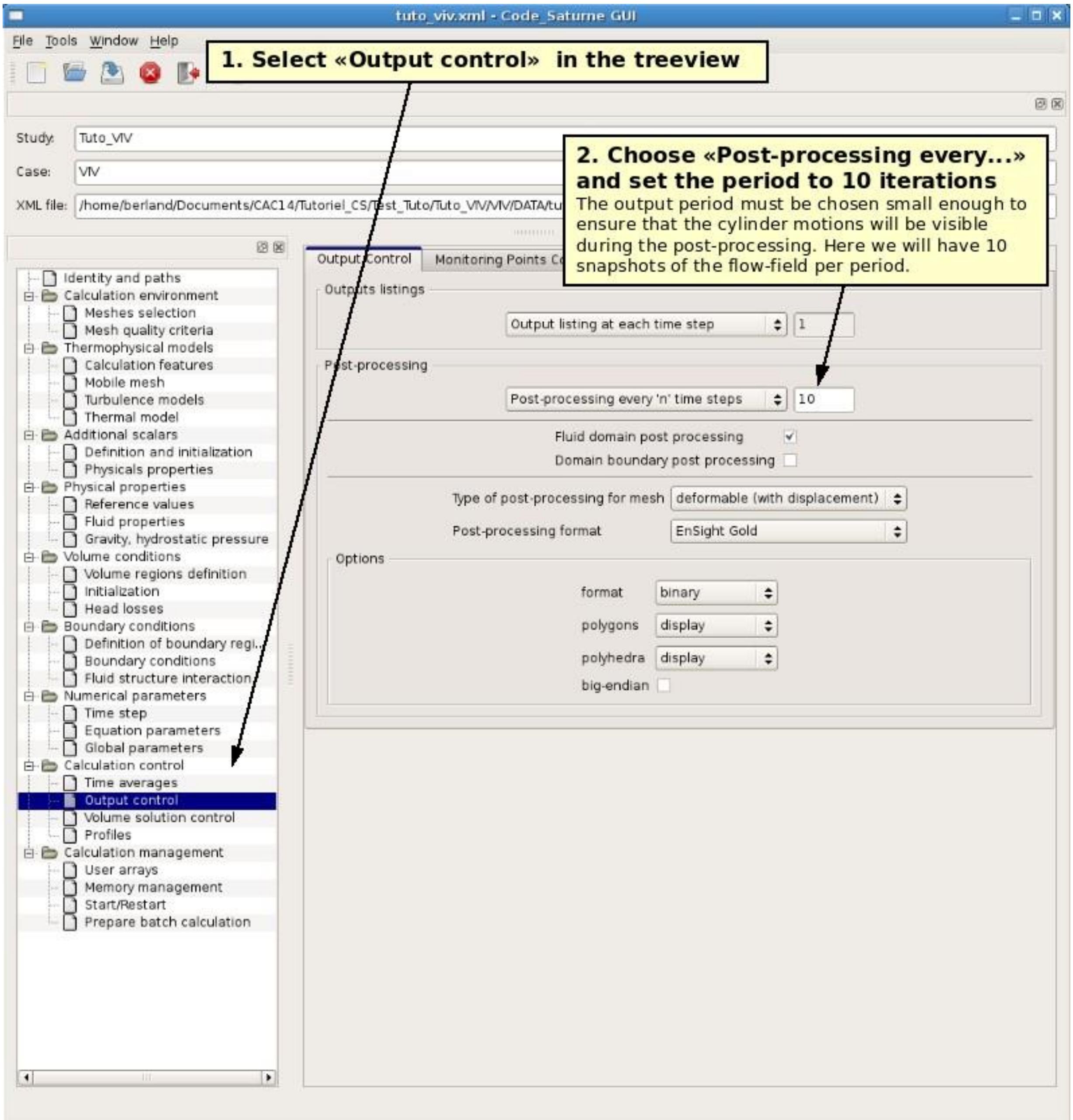

Figure 47: Imposed displacement case. Output control

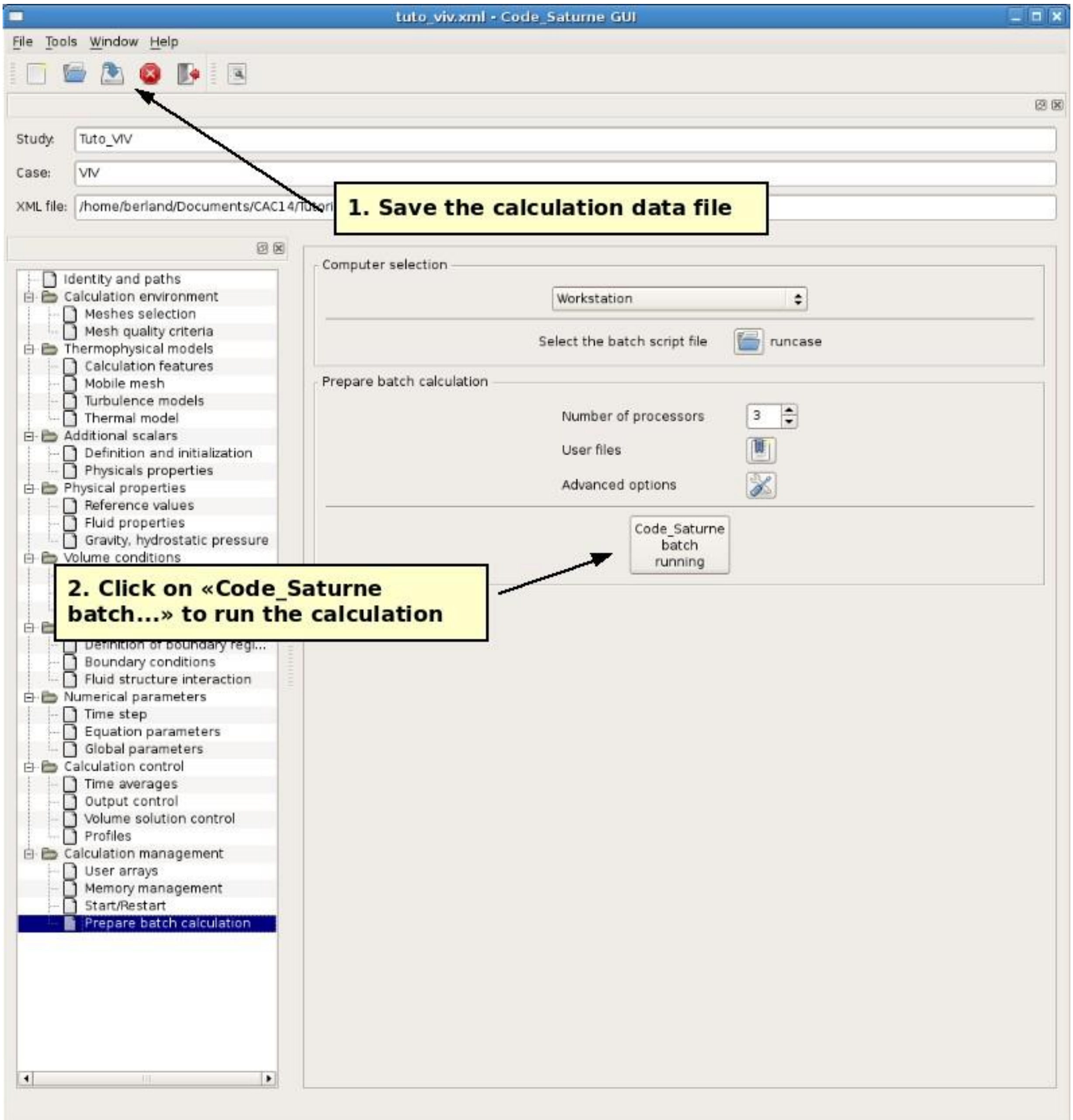

Figure 48: Imposed displacement case. Prepare batch calculation.

3.4 How to impose the displacement of the structure (user subroutines)

Copy the user-source `usalcl.f90` in the source directory:

```
cp Tuto_VIV/VIV/SRC/REFERENCE/base/usalcl.f90 Tuto_VIV/VIV/SRC/.
```

Edit the source file and add in the “local variables” declaration zone the following line to add a variable for the displacement of the mesh:

```
double precision delta
```

Then, in the body of the subroutine, add the following source code in order to impose to each node a given displacement:

```
! displacement amplitude (the diameter is 0.025)
delta = (0.025d0/4.d0)*sin(2.d0*pi*ttcabs/100.d0/dtref)

! get the cell faces corresponding to color 1
call getfbr('1',nlelt,lstelt)
!==========

! loop over these faces and impose them some
! displacement
do ilelt = 1, nlelt
   ifac = lstelt(ilelt)
   do ii = ipnfbr(ifac), pnfbr(ifac+1)-1
      inod = nodfbr(ii)
      if (impale(inod).eq.0) then
         depale(inod,1) = 0.d0  ! displacement /x
         depale(inod,2) = delta ! displacement /y
         depale(inod,3) = 0.d0  ! displacement /z
         impale(inod) = 1
      endif
   enddo
enddo
```

Note that the displacement is defined at the nodes, whereas the mesh velocity is defined at the cell faces (see section 3.6). Finally run *Code_Saturne* using the GUI.

3.5 How to impose the velocity of the structure (GUI)

The velocity of the cylinder may also be imposed. The steps required in the GUI to perform such an operation are explained in figures 49 to 54. Note that the steps described in section 3.1 also need to be performed to setup the calculation.

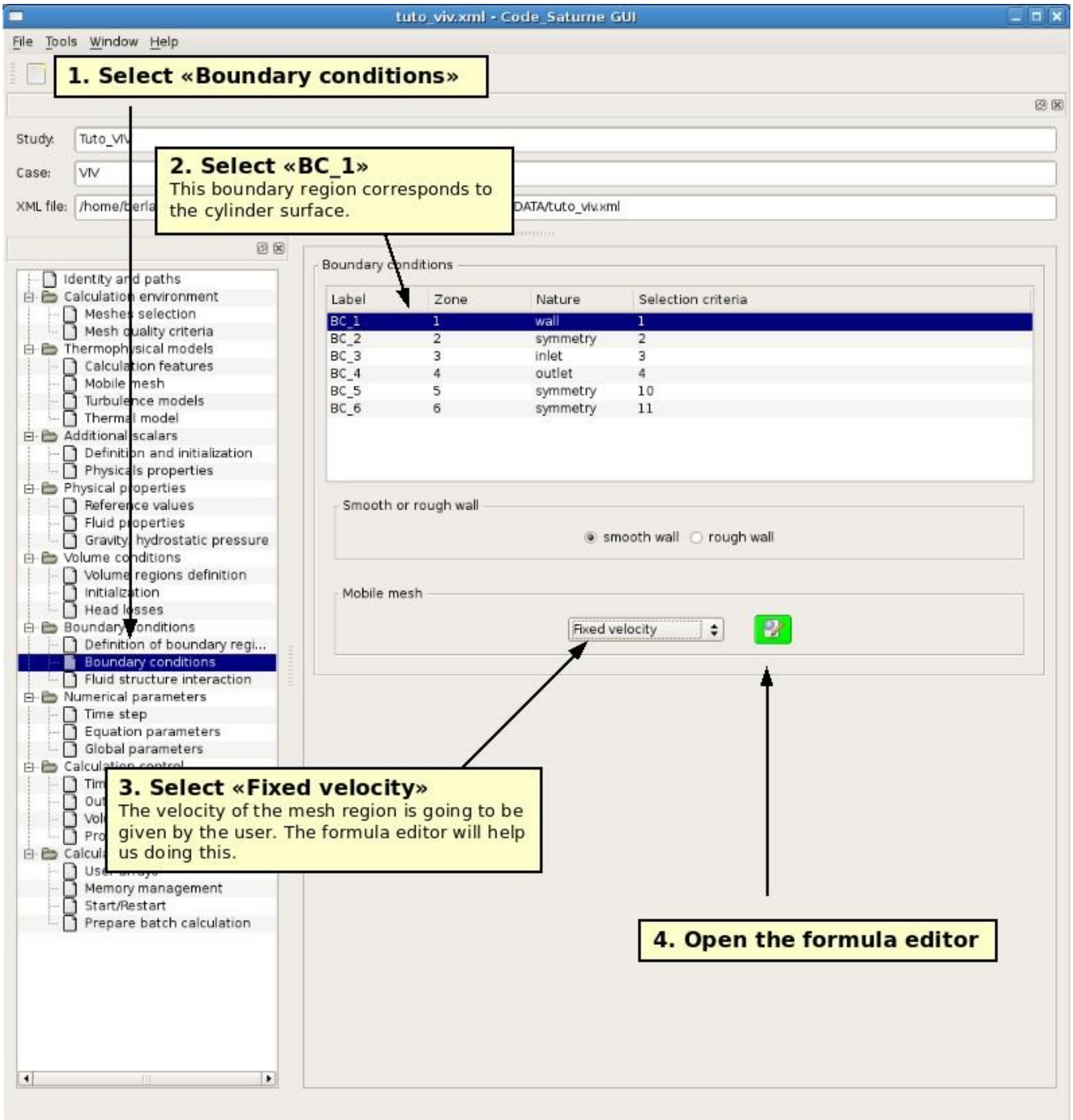

Figure 49: Imposed velocity case. Boundary conditions.

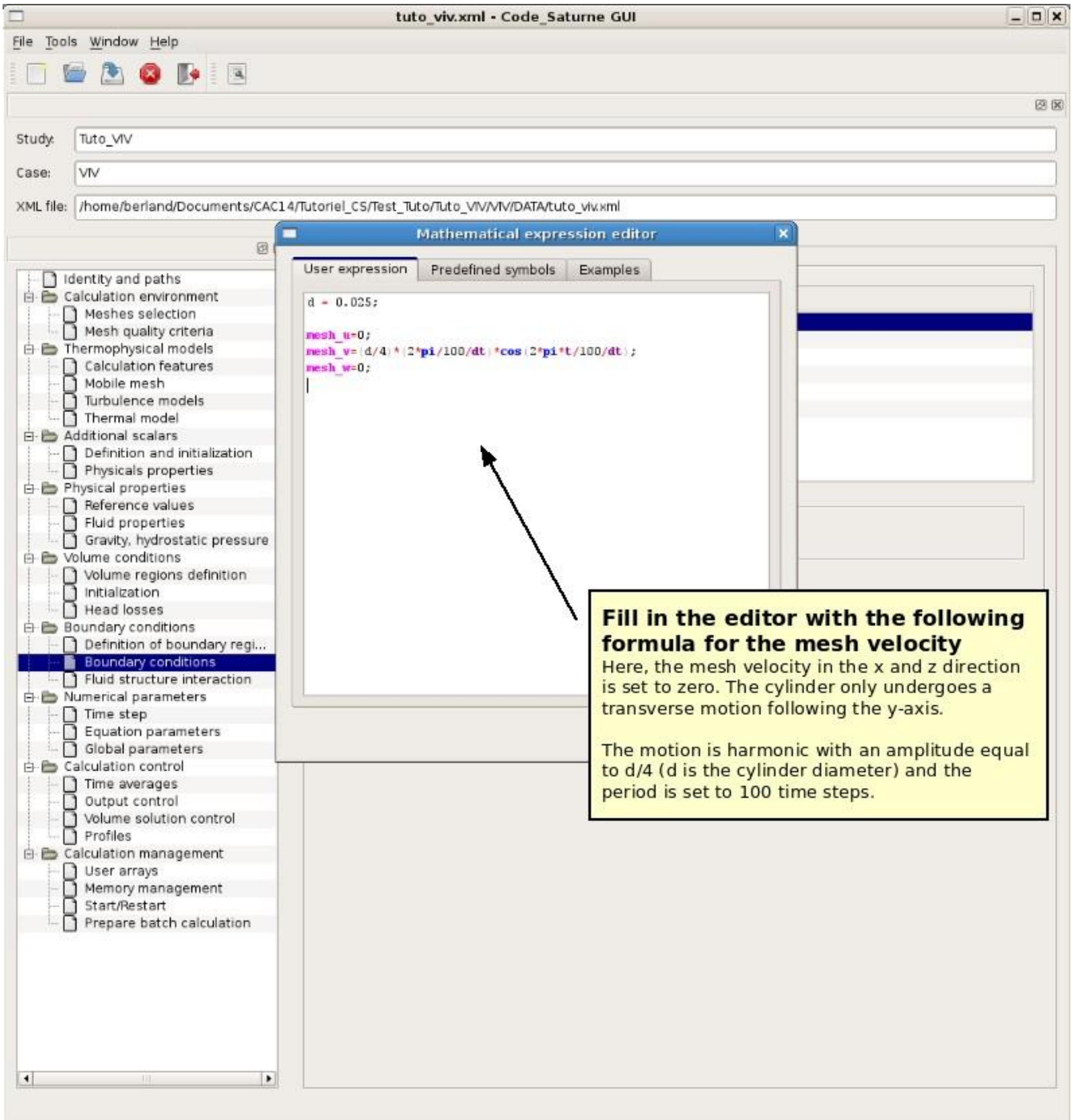

Figure 50: Imposed velocity case. Boundary conditions.

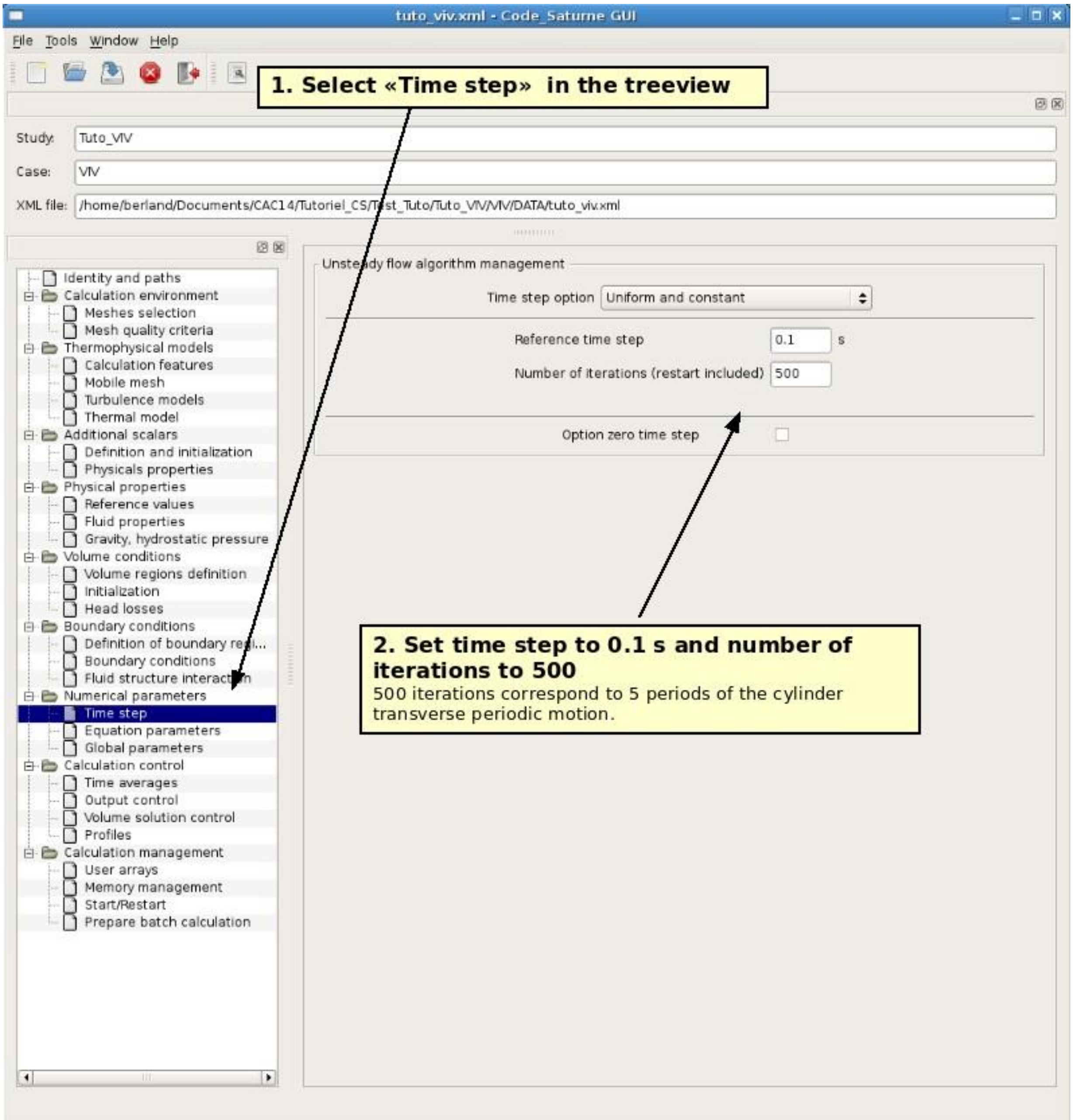

Figure 51: Imposed velocity case. Time step.

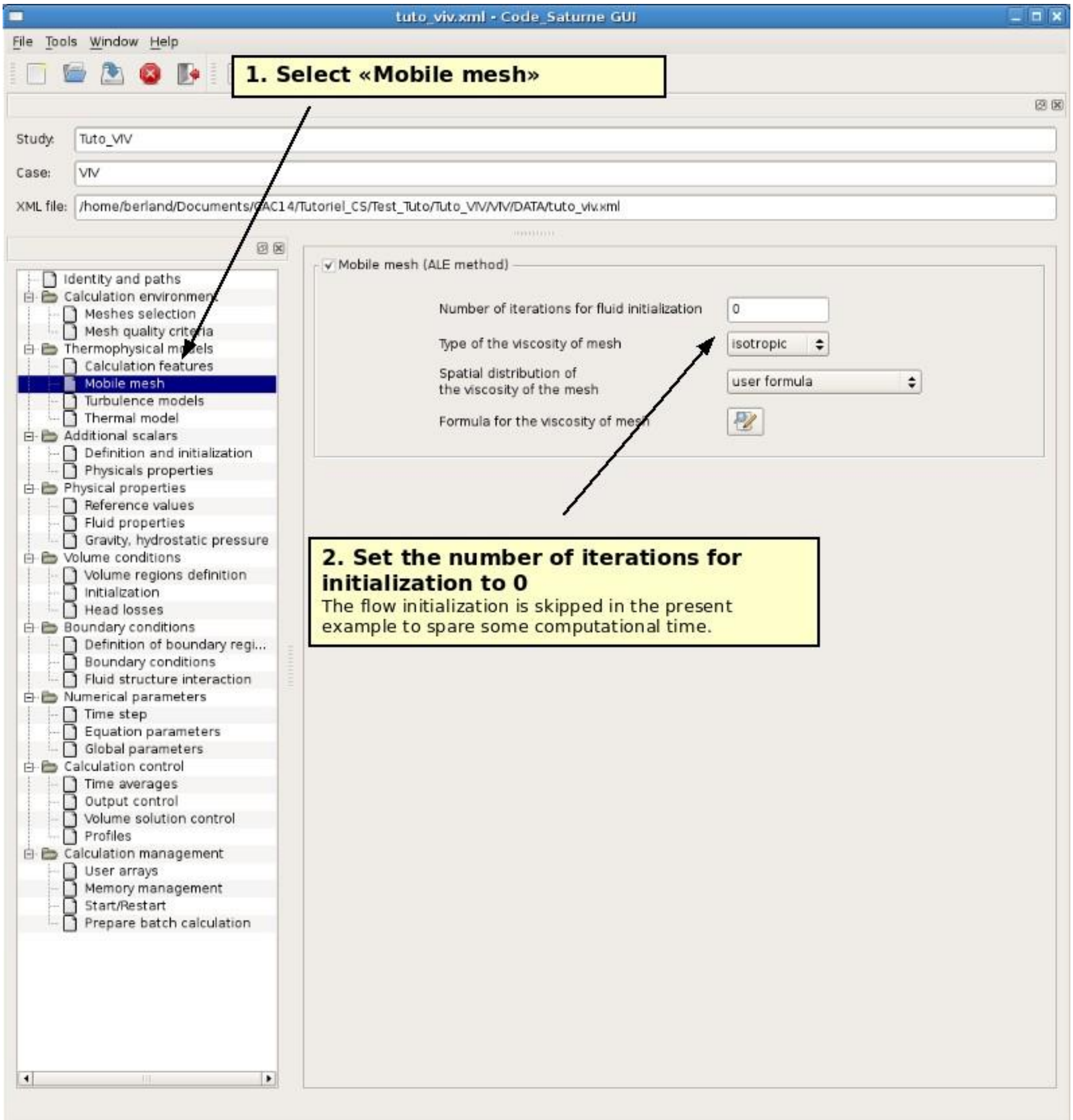

Figure 52: Imposed velocity case. Mobile mesh.

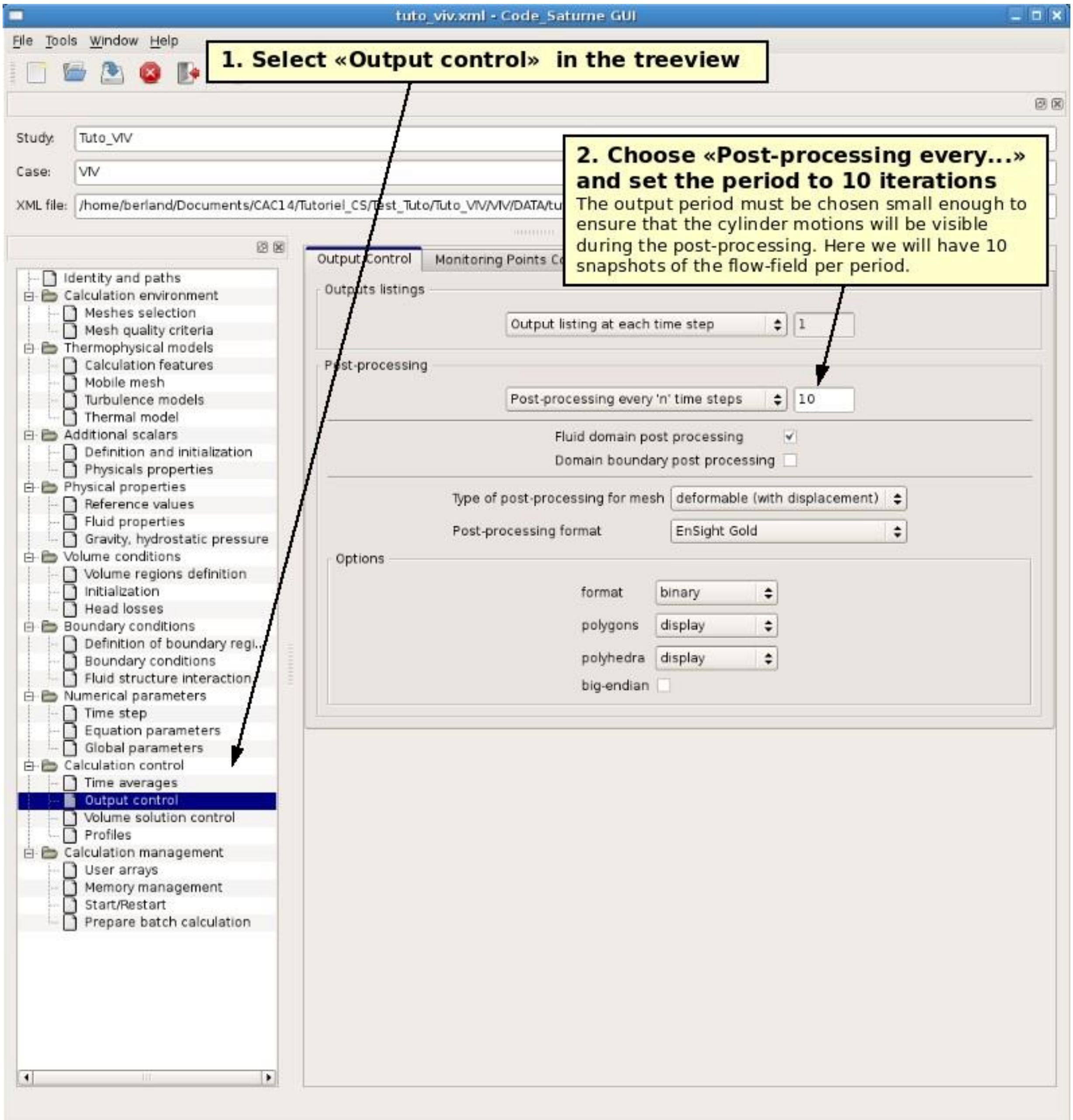

Figure 53: Imposed velocity case. Output control.

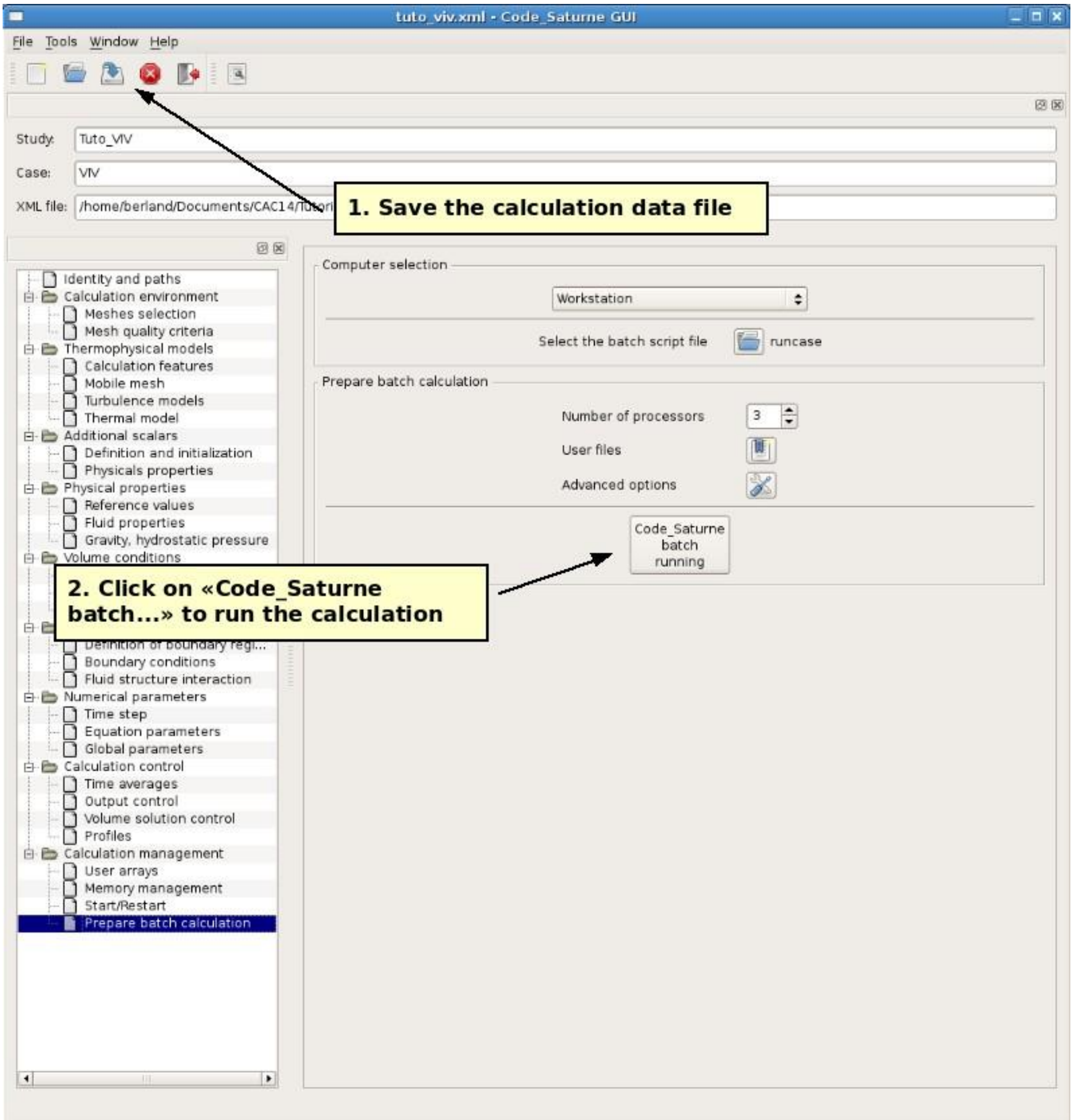

Figure 54: Imposed velocity case. Prepare batch calculation.

3.6 How to impose the velocity of the structure (user subroutines)

Copy the user-source `usalcl.f90` in the source directory:

```
cp Tuto_VIV/VIV/SRC/REFERENCE/base/usalcl.f90 Tuto_VIV/VIV/SRC/.
```

Edit the source file and add in the "local variables" declaration zone the following line to add a variable for the velocity of the mesh:

```
double precision deltav
```

Then, in the body of the subroutine, add the following source code in order to impose to each a given displacement:

```
! velocity magnitude (the diameter is 0.025)
deltav =    (0.025/4.)*(2.*3.141596d0/100.d0/dtref)     &
          * cos(2.*3.141596d0*ttcabs/100.d0/dtref)

! get the cell faces corresponding to color 1
call getfbr('1', nlelt, lstelt)
!==========

! loop over these faces and impose them some
! velocity

do ilelt = 1, nlelt
  ifac = lstelt(ilelt)
  iel = ifabor(ifac)
  ialtyb(ifac) = ivimpo
  rcodcl(ifac,iuma,1) = 0.d0
  rcodcl(ifac,ivma,1) = deltav
  rcodcl(ifac,iwma,1) = 0.d0
enddo
```

Note that velocity is defined at the cell faces, whereas the displacement is defined at the nodes (see section 3.4). Finally run *Code_Saturne* using the GUI.

3.7 How to compute the force acting on the structure (usersubroutines)

As pointed out in section 3.1, some outputs specific to fluid-structure interactions, such as the forces acting on the structure, are available in the `RESULTS`.

However it should be noted that these quantities are calculated only when the fluid-structure coupling is activated (as shown in figure 25). When the displacement is imposed, the forces acting on the structure has to be computed by the user if needed.

To do so, copy the user-source `usproj.f90` in the source directory. Edit the source file and add in the "local variables" declaration zone the following lines:

```
  double precision xfor(3)
```

Then, in the body of the subroutine, add the following source code:

```
  ! set the force components to zero
  do ii = 1, ndim
     xfor(ii) = 0.d0
  enddo

  ! get the cells corresponding to the cylinder surface
  call getfbr('1', nlelt, lstelt)
  !==========

  ! loop over the cells to integrate the force
  ! over the structure surface
  do ilelt = 1, nlelt
     ifac = lstelt(ilelt)

     ! update the force
     do ii = 1, ndim
        xfor(ii) = xfor(ii) + ra(iforbr + (ifac-1)*ndim + ii-1)
     enddo

  enddo

  ! if the calculation is parallel, add the data from the
  ! other processes
  if (irangp.ge.0) then
     call
     parrsm(ndim,xfor)
  endif
```

We eventually get a variable `xfor` containing the three components of the force acting on the cylinder.

3.8 How to control the convergence of the internal fluid-structure coupling procedure (advanced user)

As shown in figure 26, the number of sub-iterations and the precision for the fluid-structure are some parameters that need to be adjusted to obtain a relevant solution. In order to control wether the iterative of the coupling has fully converged, it may be valuable to print out the number of sub-iterations that have indeed been performed.

To do so, get the source file `strdep.f90` in the source themselves of *Code_Saturne*. This file is not available in the `SRC/REFERENCE` directory of the case but may be found in the installation directory of *Code_Saturne*. Edit the `strdep.f90` and add at the end of the convergence test labelled "`5. TEST DE CONVERGENCE`" the following lines:

```
if (irangp==1) then
   if (icv==1) then
      open(unit=impusr(1),file='cv.dat',position='append')
      write(impusr(1),*) ntcabs, italim
      close(impusr(1))
   endif
endif
```

in order to create a data file named `cv.dat` containing the time history of the convergence of the coupling iterative process (`impusr(1)` is the unit id corresponding to the user file). The following *Code_Saturne* variables are used:

- `icv`: when equal to 1, indicates that the algorithm has indeed converged (else equal to 0);
- `ntcabs`: current time-step;
- `italim`: sub-iteration number.

The first column of the file contains the iteration number and second column corresponds to the number of sub-iteration required for the fluid-structure coupling to converge. Make sure this number is always lower than the maximum number of sub-iteration defined in the GUI (see figure 26), otherwise unwanted unphysical behaviors may occur.

To allow the file `cv.dat` to be copied in the result directory at the end of the calculation, one also need to add its name in the `runcase` script located in the `SCRIPTS` directory of the case.