Introduction to *Code_Saturne*

J. Uribe

School of MACE, University of Manchester, UK
**Code_Saturne** is a 3D unstructured finite volume code developed by Electricté de France (EDF) since 2000. It was made open-source in 2007 under the GPL license.

- Single phase turbulent flows: $k - \varepsilon$, $k - \omega$, $\bar{v}^2 - f$, $R_{ij} - \varepsilon$, *LES*.
- Radiative heat transfer.
- Combustion coal, heavy fuel oil, gas.
- Electric arc and Joule effect.
- Lagrangian module for dispersed particle tracking.
- Compressible flow.
- ALE method for deformable meshes.
- Conjugate heat transfer (*SYRTHES* and 1D module).
- Specific modules for nuclear waste storage and cooling towers.
Introduction

- Written in FORTRAN (49%), C (41%) and Python (10%).
- Graphical User Interface available in Qt (xml format files).
- Parallel coding using mpi.
- Each release version is validated in around 30 cases with 1 to 15 simulations per case.
- Highly portable: Linux/UNIX, Solaris, Irix64, HP-UX, Mac OS.
- Validated on different architectures: Opteron, Itanium, Power5-6, BlueGene.
Introduction

Structure of Code_Saturne

- Preprocessor (Envelope)
  - mesh import
  - mesh pasting
  - domain decomposition

- Parallel Kernel
  - ghost cells creation
  - periodicity
  - CFD Solver

- XML data file
- GUI

- FVM library
  - parallel mesh management

- BFT library
  - I/O memory management

- External libraries (EDF, LGPL):
  - BFT: Base Functions and Types
  - FVM: Finite Volume Mesh

- Code_Saturne
  - code coupling
    - parallel treatment
      - Code_Saturne
      - SYRTHES
      - Code_Aster
      - Salome platform
      - ...

- Post-processing output

- Restart files

- Meshes
Graphical interface written in Qt.

Allows to set up parameters for the case.

Generates a xml file.

v2.0 has Mathematical Expression Interpreter.

Not all specific physics available.

User subroutines have precedence.
From version 2.0 the command `code_saturne` accesses all scripts of the code:

```
[mcjijju3@flair4gba:bin$] code_saturne
Usage: code\_saturne <topic>

Topics:
  help
    check\_consistency
    check\_mesh
    config
    create
    gui
    info
    plot\_probes

Options:
  -h, --help show this help message and exit
```
Help and Info

From version 2.0 the command `code_saturne` accesses all scripts of the code:

```
[mcjijju3@flair4gba:bin$] code_saturne info -help
Usage: cs info [options]
Options:
  -h, --help                show this help message
                            and exit
  -r <pdfreader>, --reader=<pdfreader>
                            define a pdf reader
  -g <guide>, --guide=<guide>
                            open a given manual [user, theory, tutorial]
  --version
                            print Code_Saturne
                            version number
```
From version 2.0 the command `code_saturne` accesses all scripts of the code:

```bash
[mcjijju3@flair4gba:bin]$ code_saturne info -help
Usage: cs info [options]
Options:
  -h, --help show this help message and exit
  -r <pdfreader>, --reader=<pdfreader> define a pdf reader
  -g <guide>, --guide=<guide> open a given manual [user, theory, tutorial]
  --version print Code_Saturne version number
```

The command `code_saturne info -g user` will bring the pdf version of the user’s manual.
From version 2.0 the command `code_saturne` accesses all scripts of the code:

```
[mcjijju3@flair4gba:bin$] code_saturne info -help
Usage: cs info [options]
Options:
  -h, --help show this help message
      and exit
  -r <pdfreader>, --reader=<pdfreader> define a pdf reader
  -g <guide>, --guide=<guide> open a given manual [user, theory, tutorial]
  --version print Code_Saturne version number
```

The command `code_saturne info -g user` will bring the pdf version of the user’s manual. The manual can be access from Help menu of the graphical interface.
From version 2.0 the command `code_saturne` accesses all scripts of the code:

```bash
[mcjijju3@flair4gba:bin$] code_saturne info -help
Usage: cs info [options]
Options:  
  -h, --help        show this help message  
                   and exit  
  -r <pdfreader>, --reader=<pdfreader>  
                   define a pdf reader  
  -g <guide>, --guide=<guide>            
                   open a given manual [user, theory, tutorial]  
--version            print Code_Saturne  
                    version number
```

The command `code_saturne info -g user` will bring the pdf version of the user’s manual.

The manual can be access from Help menu of the graphical interface.

More help via support email `saturne-support@edf.fr` or on the forums:

https://code-saturne.info and at
http://cfd.mace.manchester.ac.uk/Forum/WebHome
No mesh generator included in *Code_Saturne* but it can read a wide range of formats:

- Universal I-DEAS (.unv). From NX or ICEM software.
- Hexa NUMECA (.hex)
- MED 2.2 (.med). From Salomé platform
- CGNS (.cgns). Generic format.
- Gmesh.
- Gambit Neutral (.neu).
- StarCD (.ccm). Via libccmio available from StarCD on request.
The *Code_Saturne* preprocessor works independently from the Kernel. To check your mesh use the command:

```
code_saturne check_mesh [options] -m meshfile
```
Preprocessing

The *Code_Saturne* preprocessor works independently from the Kernel. To check your mesh use the command:

```
code_saturne check_mesh [options] -m meshfile
```

- Checks all aspects of the mesh, including periodicity.
The *Code_Saturne* preprocessor works independently from the Kernel. To check your mesh use the command:

```
code_saturne check_mesh [options] -m meshfile
```

- Checks all aspects of the mesh, including periodicity.
- Lists all boundaries.
The *Code_Saturne* preprocessor works independently from the Kernel. To check your mesh use the command:

```python
code_saturne check_mesh [options] -m meshfile
```

- Checks all aspects of the mesh, including periodicity.
- Lists all boundaries.
- Quality information.
The *Code_Saturne* preprocessor works independently from the Kernel. To check your mesh use the command:

```bash
code_saturne check_mesh [options] -m meshfile
```

- Checks all aspects of the mesh, including periodicity.
- Lists all boundaries.
- Quality information.
- Creates Ensight files for visualisation that can be open with ParaView.
Running a case

*Code_Saturne* needs specific directory structure. It is organised in `STUDY/CASE` hierarchy.
Running a case

_Code_Saturne_ needs specific directory structure. It is organised in **STUDY/CASE** hierarchy.

- To create the necessary structure use the command:
  
  ```bash
  code_saturne create --study STUDY1 CASE1 CASE2
  
  ... which creates a study named **STUDY1** with cases **CASE1** and **CASE2**.
  ```
Running a case

*Code_Saturne* needs specific directory structure. It is organised in **STUDY/CASE** hierarchy.

- To create the necessary structure use the command:
  ```
  code_saturne create --study STUDY1 CASE1 CASE2
  ... which creates a study named **STUDY1** with cases **CASE1** and **CASE2**.
  ```

- Cases can be added by using:
  ```
  code_saturne create --case CASE3
  ```
**Code_Saturne** needs specific directory structure. It is organised in **STUDY/CASE** hierarchy.

- To create the necessary structure use the command:
  ```bash
code_saturne create --study STUDY1 CASE1 CASE2
  ...
  ```
  which creates a study named **STUDY1** with cases **CASE1** and **CASE2**.

- Cases can be added by using:
  ```bash
code_saturne create --case CASE3
  ```

- The option **--nogui** copies user subroutines for easier editing without the graphical interface.
- **STUDY** contains the cases, the mesh and a post-processing (empty) directories.
• STUDY contains the cases, the mesh and a post-processing (empty) directories.

• CASE contains all input data, running scripts and results files.
- **STUDY** contains the cases, the mesh and a post-processing (empty) directories.
- **CASE** contains all input data, running scripts and results files.
- **DATA** has the graphical interface script and xml file.
Running a case

- STUDY contains the cases, the mesh and a post-processing (empty) directories.
- CASE contains all input data, running scripts and results files.
- DATA has the graphical interface script and xml file.
- RESU has the saved results after the simulation has finished.
- STUDY contains the cases, the mesh and a post-processing (empty) directories.
- CASE contains all input data, running scripts and results files.
- DATA has the graphical interface script and xml file.
- RESU has the saved results after the simulation has finished.
- SCRIPT contains the runcase file to launch the calculation.
- STUDY contains the cases, the mesh and a post-processing (empty) directories.
- CASE contains all input data, running scripts and results files.
- DATA has the graphical interface script and xml file.
- RESU has the saved results after the simulation has finished.
- SCRIPT contains the runcase file to launch the calculation.
- SRC contains the user subroutines.
Running a case - GUI

To use the graphical interface go to the **DATA** directory and execute

```
./SaturneGUI
```

- Create a new file.
Running a case - GUI

To use the graphical interface go to the **DATA** directory and execute **./SaturneGUI**
Running a case - GUI

To use the graphical interface go to the **DATA** directory and execute 
`./SaturneGUI`

- Create a new file.
- Go through the folders and fill the necessary information for the case (physical properties, boundary conditions, numerical options, etc.).
Running a case - GUI

To use the graphical interface go to the DATA directory and execute 
./SaturneGUI

Create a new file. Go through the folders and fill the necessary information for the case (physical properties, boundary conditions, numerical options, etc.). Save an .xml file (by default in the data directory).

Click on Code Saturne batch running and check the terminal.

Not all modules are available on the GUI. Sometimes use of the FORTRAN files is required as complement.
Running a case - GUI

To use the graphical interface go to the DATA directory and execute 
./SaturneGUI

- Create a new file.
- Go through the folders and fill the necessary information for the case (physical properties, boundary conditions, numerical options, etc.).
- Save an .xml file (by default in the data directory).
Running a case - GUI

To use the graphical interface go to the DATA directory and execute 
./SaturneGUI

Create a new file.
Go through the folders and fill the necessary information for
the case (physical properties, boundary conditions, numerical
options, etc.).
Save an .xml file (by default in the data directory).
Click on Code Saturne batch running and check the terminal.

Not all modules are available on the GUI. Sometimes use of
the FORTRAN files is required as complement.
Running a case - GUI

To use the graphical interface go to the DATA directory and execute
```bash
./SaturneGUI
```

- Create a new file.
- Go through the folders and fill the necessary information for the case (physical properties, boundary conditions, numerical options, etc.).
- Save an `.xml` file (by default in the data directory).
- Click on `Code_Saturne batch running` and check the terminal.

Not all modules are available on the GUI. Sometimes use of the FORTRAN files is required as complement.
Running a case - GUI

To use the graphical interface go to the DATA directory and execute ./SaturneGUI

Create a new file. Go through the folders and fill the necessary information for the case (physical properties, boundary conditions, numerical options, etc.). Save an .xml file (by default in the data directory).

Click on CodeSaturne batch running and check the terminal.

Not all modules are available on the GUI. Sometimes use of the FORTRAN files is required as complement.
Running a case - GUI

To use the graphical interface go to the DATA directory and execute 
./SaturneGUI

```
[mcjijju3@flair4gba:DATA$] nohup: ignoring input and redirecting stderr to stdout

Single processor Code_Saturne simulation

Code_Saturne is running
****************************

Working directory (to be periodically cleaned):
/home/mcjijju3/TEMP/STUDY1/CASE1/tmp_Saturne/STUDY1.CASE1.04161729

Kernel version: /usr/local/saturne/v2.0rc/cs-2.0-beta2
Preprocessor: /usr/local/saturne/v2.0rc/cs-2.0-beta2/bin

****************************************************
Preparing calculation
****************************************************

****************************************************
Starting calculation
****************************************************
```

Not all modules are available on the GUI. Sometimes use of the FORTRAN files is required as complement.
To use the graphical interface go to the DATA directory and execute 
`./SaturneGUI`

- Create a new file.
- Go through the folders and fill the necessary information for 
the case (physical properties, boundary conditions, numerical 
options, etc.).
- Save an .xml file (by default in the data directory).
- Click on Code_Saturne batch running and check the terminal.
- Not all modules are available on the GUI. Sometimes use of 
the FORTRAN files is required as complement.
If the GUI is not used, all necessary parameters have to be filled in the FORTRAN subroutines.
If the GUI is not used, all necessary parameters have to be filled in the FORTRAN subroutines.

- All used subroutines have to be copied into the SRC directory from the REFERENCE sub-directory.
If the GUI is not used, all necessary parameters have to be filled in the FORTRAN subroutines.

- All used subroutines have to be copied into the SRC directory from the REFERENCE sub-directory.
- The user subroutines are organised by modules.
If the GUI is not used, all necessary parameters have to be filled in the FORTRAN subroutines.

- All used subroutines have to be copied into the SRC directory from the REFERENCE sub-directory.
- The user subroutines are organised by modules.
- The base module contains the basic subroutines for incompressible flow.
If the GUI is not used, all necessary parameters have to be filled in the FORTRAN subroutines.

- All used subroutines have to be copied into the SRC directory from the REFERENCE sub-directory.
- The user subroutines are organised by modules.
- The base module contains the basic subroutines for incompressible flow.
- At least `usini1.f90` (main physical and numerical parameters) and `usclim.f90` (boundary conditions) are mandatory if not using the GUI.
Running a case - FORTRAN

If the GUI is not used, all necessary parameters have to be filled in
the FORTRAN subroutines.

- All used subroutines have to be copied into the SRC directory from the
  REFERENCE sub-directory.
- The user subroutines are organised by modules.
- The base module contains the basic subroutines for incompressible flow.
- At least usini1.f90 (main physical and numerical parameters) and
  usclim.f90 (boundary conditions) are mandatory if not using the GUI.
- FORTRAN files have precedence over the GUI xml file.
Once all subroutines are complete, go to the `SCRIPTS` directory and edit the `runcate` file.

```fortran
# BEGINNING OF USER MODIFIABLE ZONE FOR STANDARD CALCULATIONS
#
# runcate.help gives more details about the different variables.
#
SOLCOM=0
#
# On some systems, some external libraries may require TERM to be defined.
export TERM=xterm
#
STUDY=STUDY1
CASE=CASE1
PARAM=
MESH="studymesh.unv"
COMMAND_REORIENT=
COMMAND_JOIN=
COMMAND_COPY=
COMMAND_PERIO="--perio --trans 1 0 0 --color 2 3"
THERMOCHEMISTRY_DATA=
METEO_DATA=
#
# Choose the total number of processors used (if empty, automatic detection
# through the batch system if possible, set to 1 otherwise).
# When coupling with SYNTHERS with COUPLING_MODE=mpi, the 1st processor is
# used by SYNTHERS, so the effective number of processors assigned to the
# Kernel is reduced by 1.
# The processors list is only usable when not running on a batch system
# (as such a system usually already defines a similar list)
NUMBER_OF_PROCESSORS=1
PROCESSOR_LIST=
#
PARTITION_LIST=
#
USER_INPUT_FILES="inlet.dat"
USER_OUTPUT_FILES="output.dat"
```

J. Uribe  (University of Manchester)  Introduction to Code_Saturne
Running a case - FORTRAN

Once all subroutines are complete, go to the SCRIPTS directory and edit the `runcase` file.

- Add the name of the mesh file to the `MESH` variable.

```fortran
# BEGINNING OF USER MODIFIABLE ZONE FOR STANDARD CALCULATIONS
#
# runcase.help gives more details about the different variables.
#
# SOLCOM=0
#
# On some systems, some external libraries may require TERM to be defined.
# export TERM=xterm
#
# STUDY=STUDY1
# CASE=CASE1
# MESH="studymesh.unv"
#COMMAND_ROOT=""
#COMMAND_JOIN=""
#COMMAND_CWI=""
#COMMAND.PERIO="--perio --trans 1 0 0 --color 2 3"
#THERMOCHEMISTRY_DATA=""
#METEO_DATA=""
#
# Choose the total number of processors used (if empty, automatic detection
# through the batch system if possible, set to 1 otherwise).
# When coupling with SYRTHERES with COUPLING_MODE=mpi, the 1st processor is
# used by SYRTHERES, so the effective number of processors assigned to the
# Kernel is reduced by 1.
# The processors list is only usable when not running on a batch system
# (as such a system usually already defines a similar list)
#NUMBER_OF_PROCESSORS=1
#PROCESSOR_LIST=""
#
# PARTITION_LIST=""
#
# USER_INPUT_FILES="inlet.dat"
# USER_OUTPUT_FILES="outlet.dat"
```
Once all subroutines are complete, go to the SCRIPTS directory and edit the `runcase` file.

- Add the name of the mesh file to the **MESH** variable.
- Add commands for periodicity or mesh pasting.

```fortran
# BEGINNING OF USER MODIFIABLE ZONE FOR STANDARD CALCULATIONS
# runcase.help gives more details about the different variables.
# ---------------------------------------------
SOLCOM=0
# On some systems, some external libraries may require TERM to be defined.
exe TERM=xterm
# STUDY=STUDY1
# CASE=CASE1
PARAM=
MESH="studylmesh.unv"
COMMAND_REORIENT=
COMMAND_JOIN=
COMMAND_PASTE="--perio --trans 1 0 0 --color 2 3"
# Choose the total number of processors used (if empty, automatic detection
# through the batch system if possible, set to 0 otherwise).
# When coupling with SYRTHES with COUPLING_MODE=MPI, the last processor is
# used by SYRTHES, so the effective number of processors assigned to the
# Kernel is reduced by 1.
# The processors list is only usable when not running on a batch system
# (as such a system usually already defines a similar list)
NUMBER_OF_PROCESSORS=1
PROCESSOR_LIST=
# PARTITION_LIST=
# USER_INPUT_FILES="inlet.dat"
USER_OUTPUT_FILES="outlet.dat"
```

J. Uribe (University of Manchester) Introduction to Code_Saturne
Running a case - FORTRAN

Once all subroutines are complete, go to the SCRIPTS directory and edit the `runcase` file.

- Add the name of the mesh file to the `MESH` variable.
- Add commands for periodicity or mesh pasting.
- Add number of processors if running in parallel.
Running a case - FORTRAN

Once all subroutines are complete, go to the SCRIPTS directory and edit the `runcase` file.

- Add the name of the mesh file to the `MESH` variable.
- Add commands for periodicity or mesh pasting.
- Add number of processors if running in parallel.
- Add any input/output files.

```plaintext
# BEGINNING OF USER MODIFIABLE ZONE FOR STANDARD CALCULATIONS
# runcase.help gives more details about the different variables.
#
SOLCOM=0
#
# On some systems, some external libraries may require TERM to be defined.
# export TERM=xterm
#
STUDY=STUDY1
CASE=CASE1
PARAM=
MESH="studymesh.unv"
COMMAND_REORIENT=
COMMAND_JOIN=
COMMAND_CW=
COMMAND_PERI="--perio --trans 1 0 0 --color 2 3"
THERMOCHEMISTRY_DATA=
MTEQO_DATA=

# Choose the total number of processors used (if empty, automatic detection
# through the batch system if possible, set to 1 otherwise).
# When coupling with SYRTHES with COUPLING_MODE=MP1, the 1st processor is
# used by SYRTHES, so the effective number of processors assigned to the
# Kernel is reduced by 1.
# The processors list is only usable when not running on a batch system
# (as such a system usually already defines a similar list)
NUMBER_OF_PROCESSORS=1
PROCESSOR_LIST=
#
PARTITION_LIST=

USER_INPUT_FILES="inlet.dat"
USER_OUTPUT_FILES="outlet.dat"
```
Once all subroutines are complete, go to the SCRIPTS directory and edit the `runcase` file.

- Add the name of the mesh file to the `MESH` variable.
- Add commands for periodicity or mesh pasting.
- Add number of processors if running in parallel.
- Add any input/output files.
- Run the script `./runcase`
Running a case. Check info

- After running the `runcase` either from the terminal or through the GUI, check the output window.

```
[mcjijju3@flair4gba:SCRIPTS$] ./runcase
Code_Saturne is running

Working directory (to be periodically cleaned):
/home/mcjijju3/TEMP/STUDY1/CASE1/tmp_Saturne/STUDY1.CASE1.04201335
Kernel version:    /usr/local/saturne/v2.0rc/cs-2.0-beta2
Preprocessor:      /usr/local/saturne/v2.0rc/cs-2.0-beta2/bin

Compilation of user subroutines and linking of Code_Saturne

Preparing calculation

Starting calculation
```
Running a case. Check info

- After running the `runcase` either from the terminal or through the GUI, check the output window.
- The calculation will run on a temporary directory (set in the variable `CS_TMP_PREFIX` on the `runcase` file).

```
[mcjijju3@flair4gba:SCRIPTS$] ./runcase
Code_Saturne is running
                        ***********************
Working directory (to be periodically cleaned):
/home/mcjijju3/TEMP/STUDY1/CASE1/tmp_Saturne/STUDY1.CASE1.04201335
Kernel version:  /usr/local/saturne/v2.0rc/cs-2.0-beta2
Preprocessor:     /usr/local/saturne/v2.0rc/cs-2.0-beta2/bin
                        ********************************************
Compilation of user subroutines and linking of Code_Saturne
                        ********************************************
Preparing calculation
                        ********************************************
Starting calculation
                        ********************************************
```
Running a case. Check info

- After running the `runcase` either from the terminal or through the GUI, check the output window.
- The calculation will run on a temporary directory (set in the variable `CS_TMP_PREFIX` on the `runcase` file).
- The listing file contains all input parameters and information on the variables at each iteration.

```bash
[mcjijju3@flair4gba:SCRIPTS$] ./runcase
Code_Saturne is running

Working directory (to be periodically cleaned):
/home/mcjijju3/TEMP/STUDY1/CASE1/tmp_Saturne/STUDY1.CASE1.04201335

Kernel version: /usr/local/saturne/v2.0rc/cs-2.0-beta2
Preprocessor: /usr/local/saturne/v2.0rc/cs-2.0-beta2/bin

Compilation of user subroutines and linking of Code_Saturne

Preparing calculation

Starting calculation
```
Running a case. Check info

- After running the `runcase` either from the terminal or through the GUI, check the output window.
- The calculation will run on a temporary directory (set in the variable `CS_TMP_PREFIX` on the `runcase` file).
- The listing file contains all input parameters and information on the variables at each iteration.
- Once the calculation is finished, all relevant files are copied to the `RESU` directory.

```
[mcjijju3@flair4gba:RESU$] ls
  case1.xml.09091117  listing.09091117  runcase.09091117
  CHR.ENSIGHT.09091117/  listpre.09091117  SRC.09091117/
  compil.log.09091117  RESTART.09091117/  summary.09091117
  HIST.09091117/  RES_USER.09091117/
```
In case of problems ...

- Check compilation log.
- Check preprocessing output file listpre.
- Check listing file.
In case of problems ...

- Check compilation log.

```
[mcjijju3@flair4gba:SCRIPTS$] ./runcase
  Code_Saturne is running
  ********************

Working directory (to be periodically cleaned):
  /home/mcjijju3/TEMP/STUDY1/CASE1/tmp_Saturne/STUDY1.CASE1.04201328

Kernel version: /usr/local/saturne/v2.0rc/cs-2.0-beta2
Preprocessor:  /usr/local/saturne/v2.0rc/cs-2.0-beta2/bin

  ********************
  Compilation of user subroutines and linking of Code_Saturne
  ********************
COMPILE OR LINK ERROR

[mcjijju3@flair4gba:SCRIPTS$] ls ../RESU/
[mcjijju3@flair4gba:SCRIPTS$]
```
In case of problems ...

- Check compilation log.
- Check preprocessing output file listpre.

```bash
[mcjijju3@flair4gba:SCRIPTS$] ./runcase
  Code_Saturne is running
  *************************************************
Working directory (to be periodically cleaned):
  /home/mcjijju3/TEMP/STUDY1/CASE1/tmp_Saturne/STUDY1.CASE1.04201319
Kernel version:   /usr/local/saturne/v2.0rc/cs-2.0-beta2
Preprocessor:     /usr/local/saturne/v2.0rc/cs-2.0-beta2/bin
                        *************************************************
Compilation of user subroutines and linking of Code_Saturne
                        *************************************************
Preparing calculation
                        *************************************************
Starting calculation
                        *************************************************
Error running the preprocessor.
Check preprocessor log (listpre) for details.
                        *************************************************
Error in preprocessing stage.
                        *************************************************
```
In case of problems ...

- Check compilation log.
- Check preprocessing output file listpre.
- Check listing file.

```bash
[mcjijju3@flair4gba:SCRIPTS$] ./runcase
  Code_Saturne is running
  ********************************************
Working directory (to be periodically cleaned):
  /home/mcjijju3/TEMP/STUDY1/CASE1/tmp_Saturne/STUDY1.CASE1.04201335
Kernel version: /usr/local/saturne/v2.0rc/cs-2.0-beta2
Preprocessor: /usr/local/saturne/v2.0rc/cs-2.0-beta2/bin
  ************************************************
Compilation of user subroutines and linking of Code_Saturne
  ************************************************
Preparing calculation
  ************************************************
Starting calculation
  ************************************************
Error running the calculation.
Check Kernel log (listing) and error* files for details
  ************************************************
Error in calculation stage.
  ************************************************
```
Checking the results.

After the calculation has finished, the RESU directory contains files with the date appended, i.e. DDMMHHMM:

- **CHR.ENSIGHT.DDMMHHMM**: Has the files for visualisation using Ensight or ParaView.
Checking the results.

After the calculation has finished, the RESU directory contains files with the date appended, i.e. DDMMHHMM:

- CHR.ENSIGHT.DDMMHHMM: Has the files for visualisation using Ensight or ParaView.
- HIST.DDMMHHMM: Contains history files of variables at given locations (probes).
Checking the results.

After the calculation has finished, The RESU directory contains files with the date appended, i.e. DDMMHHMM:

- **CHR.ENSIGHT.DDMMHHMM**: Has the files for visualisation using Ensight or ParaView.
- **HIST.DDMMHHMM**: Contains history files of variables at given locations (probes).
- **RESTART.DDMMHHMM**: Has the necessary files to restart a calculation.
- **RESU.USER.DDMMHHMM**: Any user output files.
- **SRC.DDMMHHMM**: A copy of the FORTRAN files used to run the case.
- **listpre** and **listing** files for the preprocessor and kernel executions.
- Copies of runcase and .xml files used.
Checking the results.

After the calculation has finished, The **RESU** directory contains files with the date appended, i.e. DDMMHHMM:

- **CHR.ENSIGHT.DDMMHHMM**: Has the files for visualisation using Ensight or ParaView.
- **HIST.DDMMHHMM**: Contains history files of variables at given locations (probes).
- **RESTART.DDMMHHMM**: Has the necessary files to restart a calculation.
- **RES_USER.DDMMHHMM**: Any user output files.
Checking the results.

After the calculation has finished, The **RESU** directory contains files with the date appended, i.e. DDMMHHMM:

- **CHR.ENSIGHT.DDMMHHMM**: Has the files for visualisation using Ensight or ParaView.
- **HIST/DDMMHHMM**: Contains history files of variables at given locations (probes).
- **RESTART/DDMMHHMM**: Has the necessary files to restart a calculation.
- **RES_USER/DDMMHHMM**: Any user output files.
- **SRC/DDMMHHMM**: A copy of the FORTRAN files used to run the case.
Checking the results.

After the calculation has finished, the RESU directory contains files with the date appended, i.e. DDMMHHMM:

- CHR.ENSIGHT.DDMMHHMM: Has the files for visualisation using Ensight or ParaView.
- HIST.DDMMHHMM: Contains history files of variables at given locations (probes).
- RESTART.DDMMHHMM: Has the necessary files to restart a calculation.
- RES_USER.DDMMHHMM: Any user output files.
- SRC.DDMMHHMM: A copy of the FORTRAN files used to run the case.
- listpre and listing files for the preprocessor and kernel executions.
Checking the results.

After the calculation has finished, The **RESU** directory contains files with the date appended, i.e. DDMMHHMM:

- **CHR.ENSIGHT.DDMMHHMM**: Has the files for visualisation using Ensight or ParaView.
- **HIST.DDMMHHMM**: Contains history files of variables at given locations (probes).
- **RESTART.DDMMHHMM**: Has the necessary files to restart a calculation.
- **RES_USER.DDMMHHMM**: Any user output files.
- **SRC.DDMMHHMM**: A copy of the FORTRAN files used to run the case.
- **listpre and listing** files for the preprocessor and kernel executions.
- **Copies of runcase and .xml files used.**
A case file with info on files CHR.case.

- CHR.ENSiGHT.09091117
  - CHR.case
  - chr.courant_nb.0001
  - chr.dissip.0001
  - chr.efforts.0001
  - chr.fourier_nb.0001
  - chr.geo
  - chr.pressure.0001
  - chr.total_pressure.0001
  - chr.turbener.0001
  - chr.turb._vi.0001
  - chr.velocitu.0001
  - chr.yplus.0001

Files can be opened using ParaView (open-source under GPL).

Parts can be created by the user for specific sections of the geometry.
A case file with info on files CHR.case.
A geometry file chr.geo.
Results: CHR.ENSIGHT files

- A case file with info on files CHR.case.
- A geometry file chr.geo.
- One file per variable per time iteration (frequency set by the variable NTCHR in usini1.f90 or in the Calculation Control - Output control page of the GUI.

Files can be opened using ParaView (open-source under GPL).

Parts can be created by the user for specific sections of the geometry.
Results: CHR.ENSIHGHT files

- A case file with info on files CHR.case.
- A geometry file chr.geo.
- One file per variable per time iteration (frequency set by the variable NTCHR in usini1.f90 or in the Calculation Control - Output control page of the GUI.
- Files can be opened using ParaView (open-source under GPL).
Results: CHR.ENSIGHT files

- A case file with info on files CHR.case.
- A geometry file chr.geo.
- One file per variable per time iteration (frequency set by the variable NTCHR in usini1.f90 or in the Calculation Control - Output control page of the GUI.
- Files can be opened using ParaView (open-source under GPL).
- Parts can be created by the user for specific sections of the geometry.
Results: CHR. ENSIGHT files
Results: History files

- One file per variable.

```
HIST.03131727

Density.hst
Lam._vis.hst
Pressure.hst
total_pressure.hst
VelocitU.hst
VelocitV.hst
VelocitW.hst
```
- One file per variable.
- Variation over number of iterations.

```plaintext
HIST.03131727
  Density.hst
  Lam._vis.hst
  Pressure.hst
  total_pressure.hst
  VelocitU.hst
  VelocitV.hst
  VelocitW.hst
```
Results: History files

- One file per variable.
- Variation over number of iterations.
- Easily visualised via `code_saturne` `plot_probes` command. (Requires `xmgrace` installed).
Results: History files

- One file per variable.
- Variation over number of iterations.
- Easily visualised via `code_saturne plot_probes` command. (Requires `xmgrace` installed).
- Monitoring points defined in `usini1.f90` or in `Calculation control - Output control` page in the GUI.
Results: History files