What's up in *Code_Saturne* V5.0(unreleased)

Code_Saturne development team [1]

[1] Fluid Mechanics, Energy and Environment,

2017/04/21

Overview

1 User functionalities: Graphical User Interface – SALOME _CFD

2 Physical modelling

- Compressible module
- Volume of Fluid module
- Cooling Tower module
- Lagrangian module
- Turbulence modelling
- Atmospheric module
- Internal coupling
- Others

3 Numerics and linear solvers

- Compatible Discrete Operator (CDO) schemes
- Iterative solvers
- Others

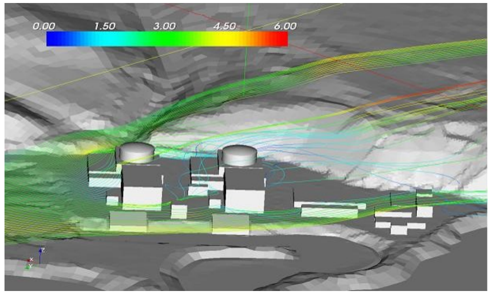
4 Architecture

Development of *Code_Saturne* at EDF

Multiphysics modules fused into *Code_Saturne* framework

Arbitrary Lagrangian Eulerian (ALE)

Atmospheric flows

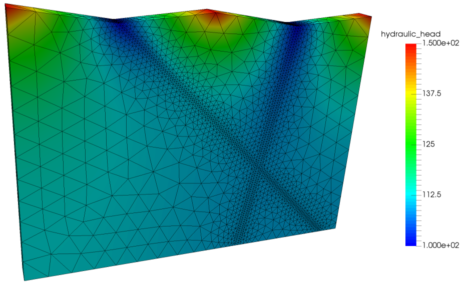


Combustion (fuel, coal, gas)

Electric Arc

Fire modelling

Groundwater flows



Lagrangian particle tracking

Thermohydraulics for Nuclear applications

Turbomachinery

Development of *Code_Saturne* at EDF

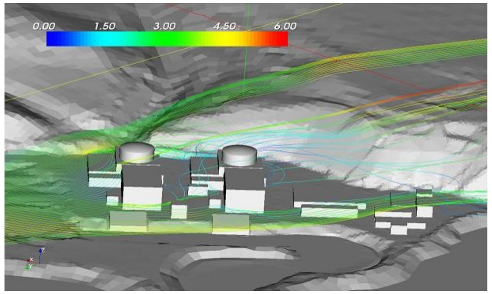
Multiphysics modules fused into *Code_Saturne* framework

Arbitrary Lagrangian Eulerian (ALE)

Electric Arc

Lagrangian particle tracking

Atmospheric flows

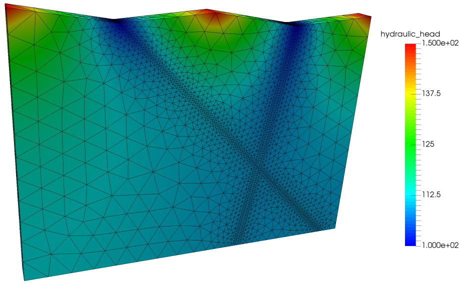


Fire modelling

$$\left\{\begin{aligned} \frac{\partial \rho}{\partial t} + \text{div}\, \rho \underline{u} &= 0 \\ \frac{\partial \rho \underline{u}}{\partial t} + \underline{\text{div}}\, (\underline{u} \otimes \rho \underline{u}) &= -\underline{\nabla} P \\ +\underline{\text{div}} \left(\mu \left(\underline{\nabla u} + \underline{\nabla u}^T\right)\right) + \rho \underline{g} \end{aligned}\right.$$

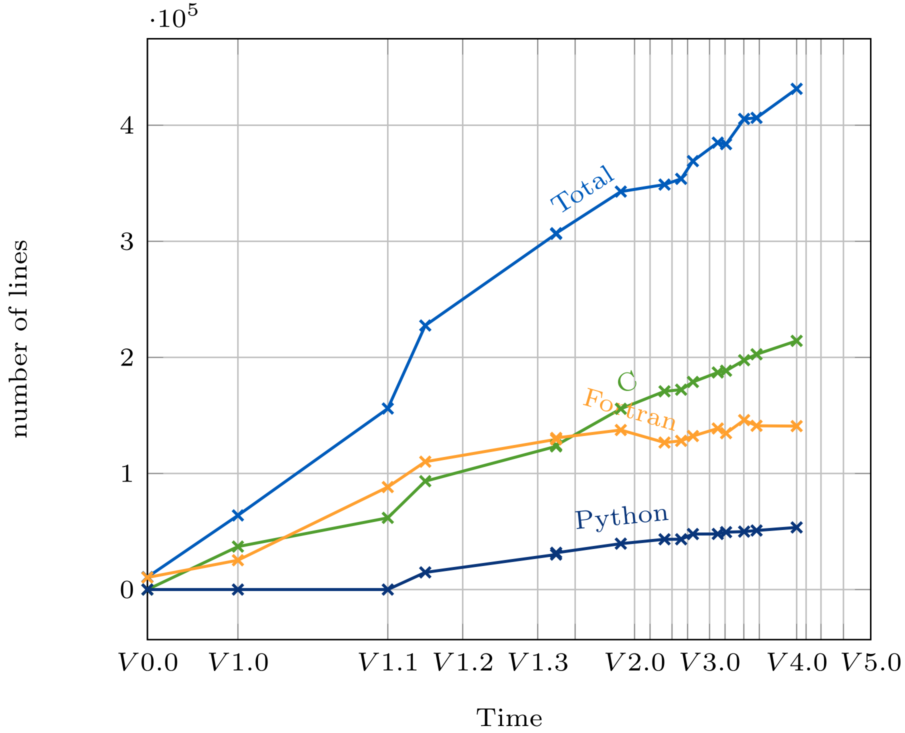
Combustion (fuel, coal, gas)

Groundwater flows



Turbomachinery

What's up since version 4.0?

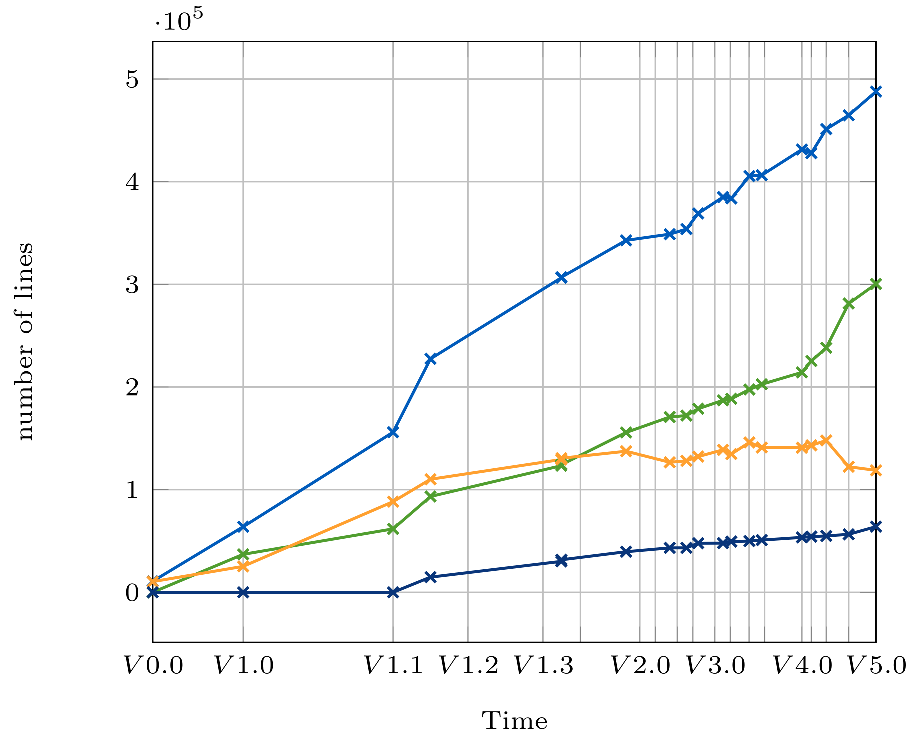


Number of lines of code...

Code_Saturne $V5.0.0$ released at the end of May

in the **SALOME_CFD** platform in September

What's up since version 4.0?

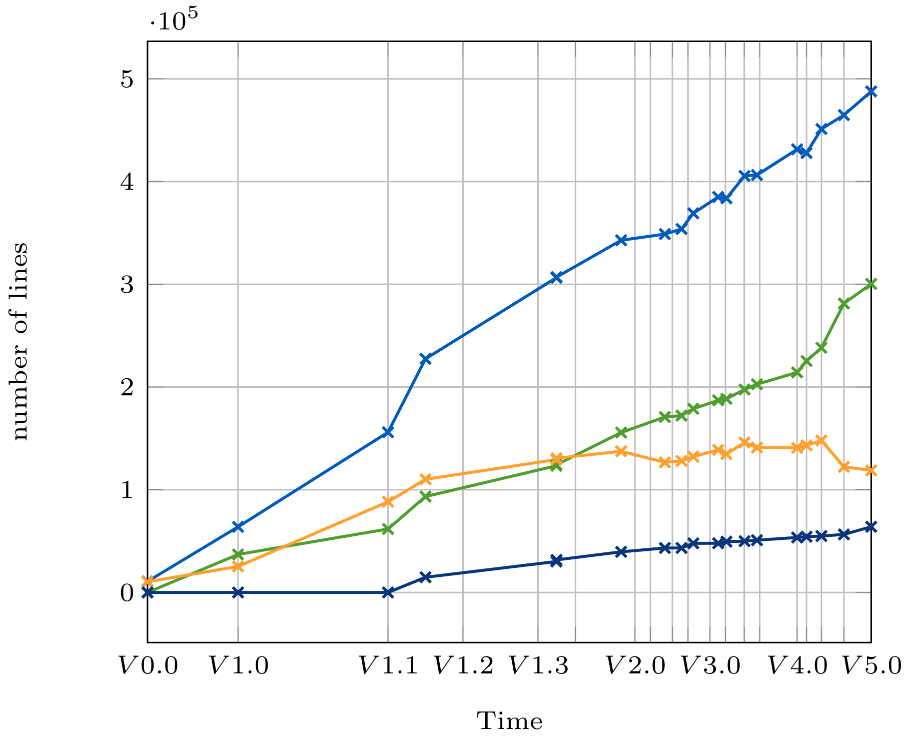


Number of lines of code...

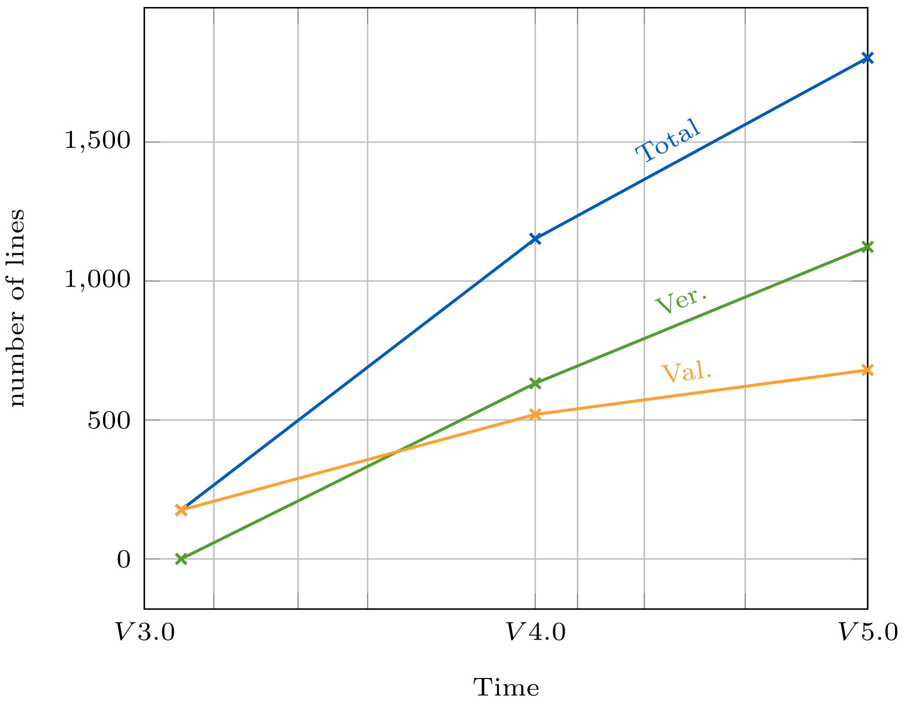
Code_Saturne *V*5.0.0 released at the end of May

in the **SALOME_CFD** platform in September

What's up since version 4.0?



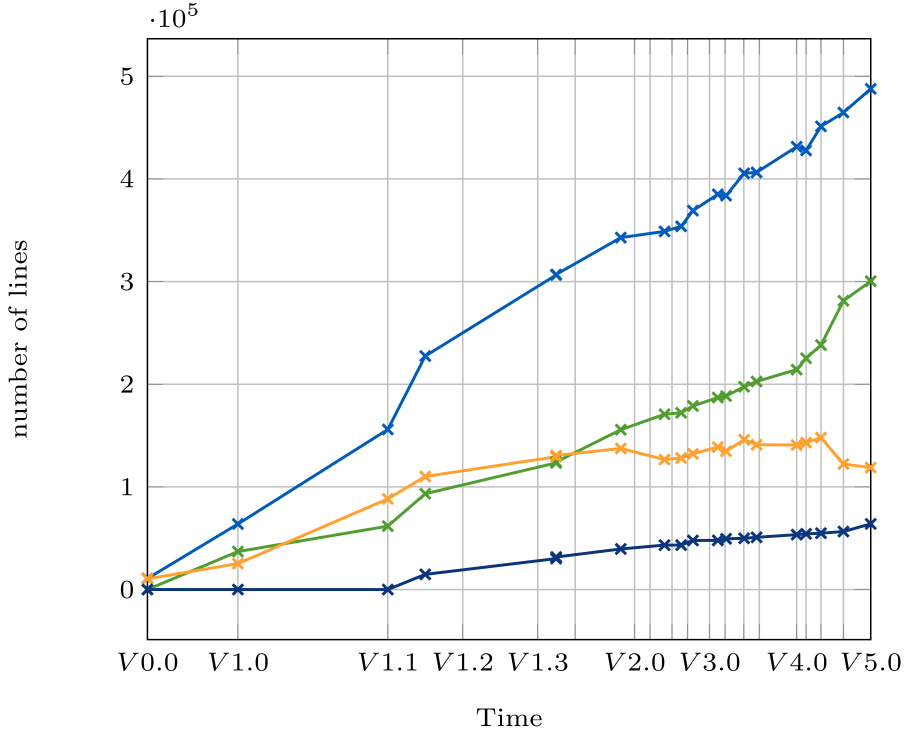
Number of lines of code...



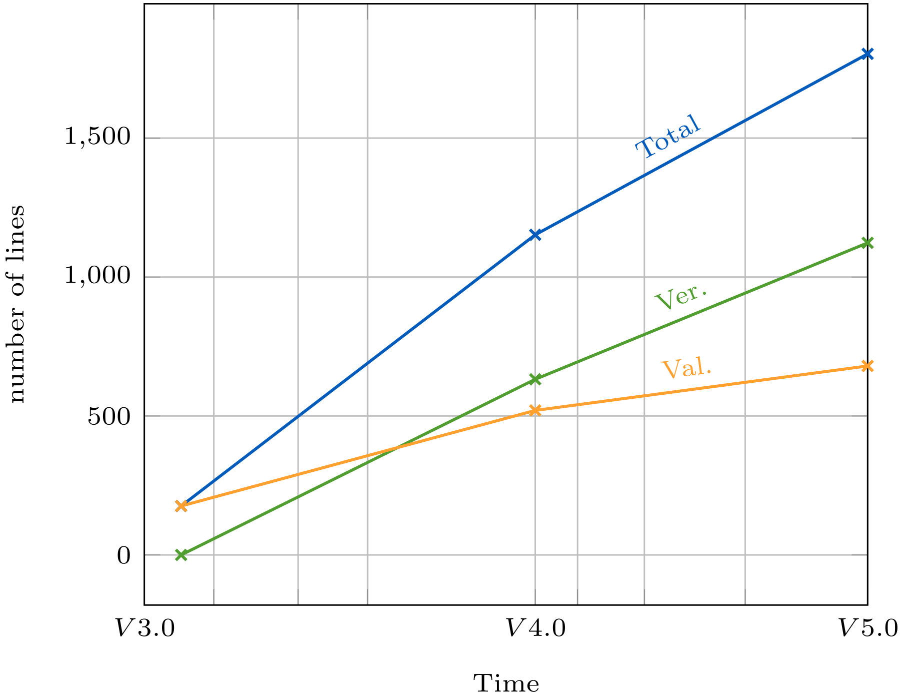
Number of Verif. and Valid. runs...

*Code_Saturne V*5.0.0 released at the end of May
in the **SALOME_CFD** platform in September

What's up since version 4.0?



Number of lines of code...

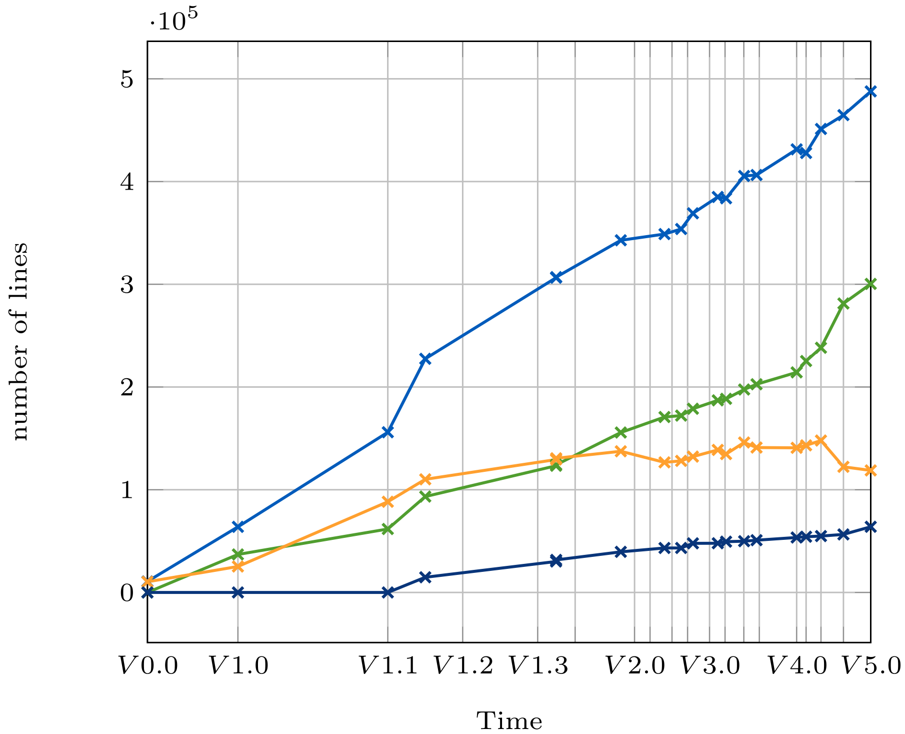


Number of Verif. and Valid. runs...

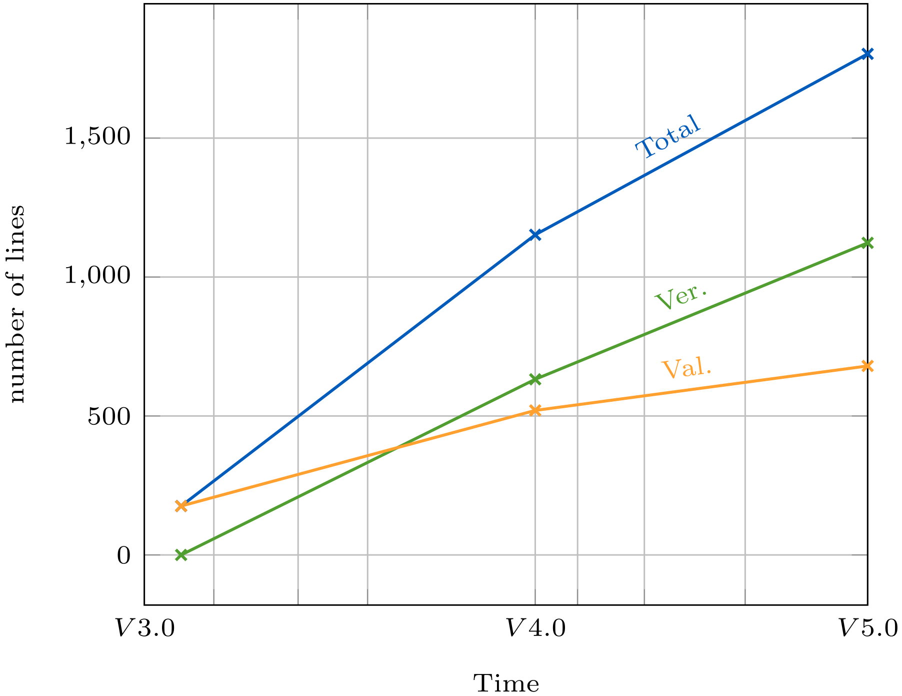
Code_Saturne $V5.0.0$ released at the end of May

in the **SALOME_CFD** platform in September

What's up since version 4.0?



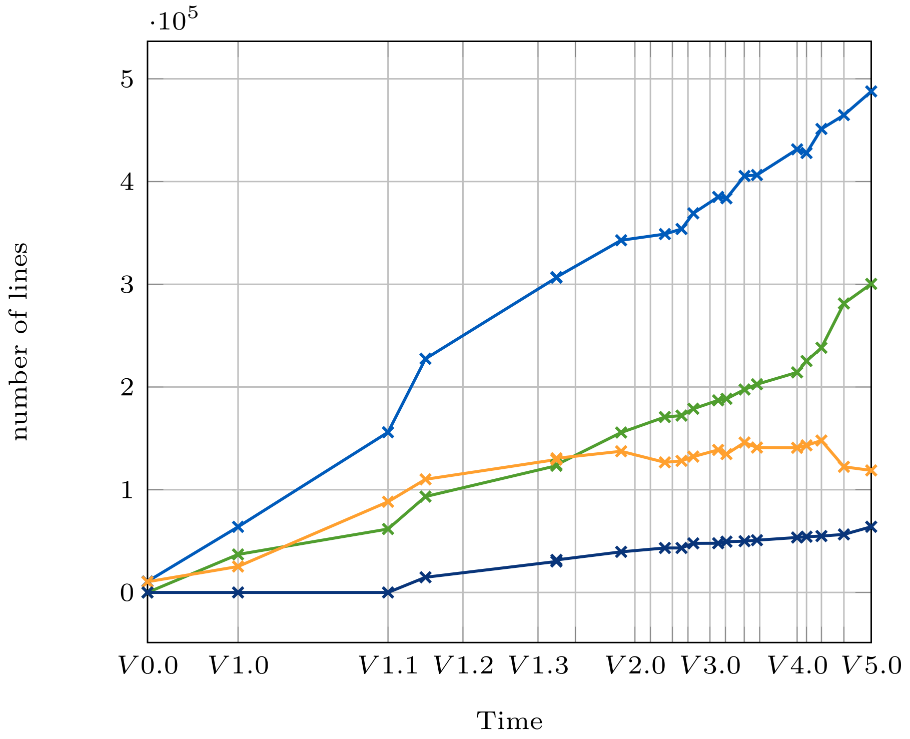
Number of lines of code...



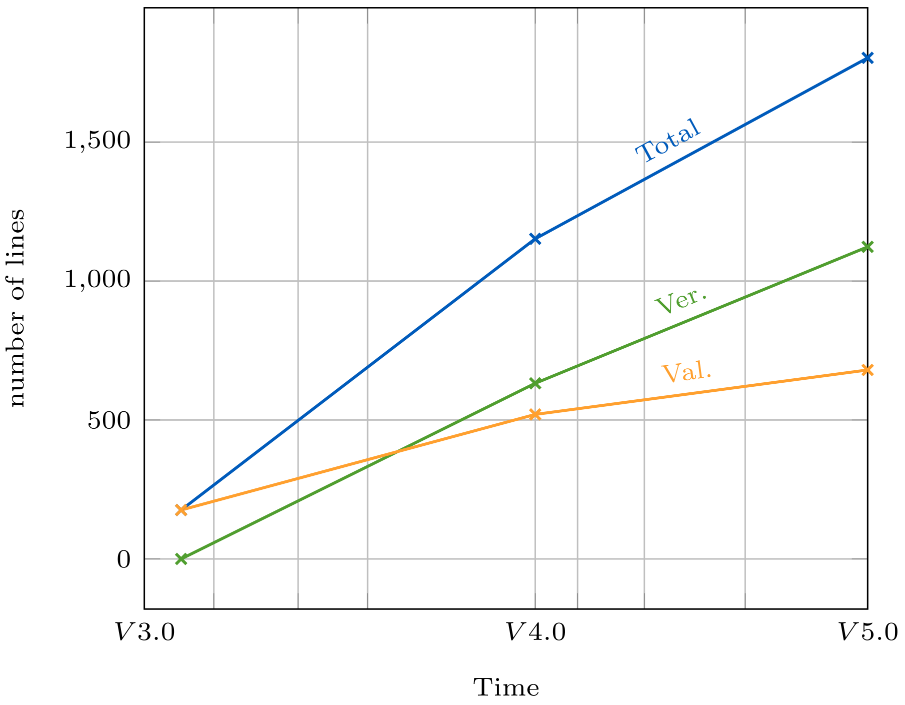
Number of Verif. and Valid. runs...

Code_Saturne V5.0.0 released at the end of May
in the **SALOME_CFD** platform in September

What's up since version 4.0?



Number of lines of code...



Number of Verif. and Valid. runs...

Code_Saturne V5.0.0 released at the end of May
in the **SALOME_CFD** platform in September

Overview

1 User functionalities: Graphical User Interface – SALOME _CFD

Salome_CFD distribution

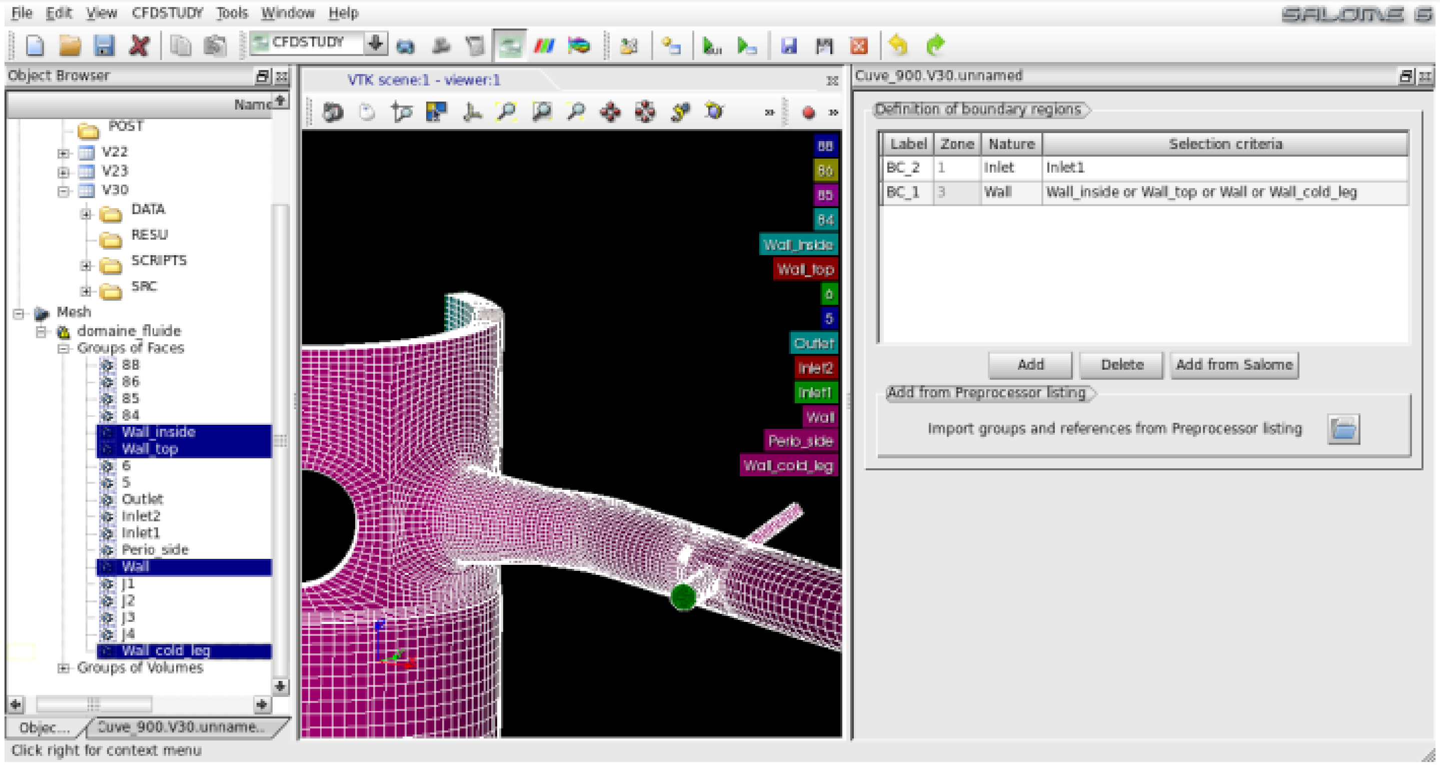
Context

- Increasing links and dependencies between *Code_Saturne* and other tools
- CFDStudy module for the Salome platform extends *Code_Saturne* GUI
  - visual selection of boundary zones for setting BCs.
  - visual verification of probes placement
  - GUI for handling of user functions
- *Code_Saturne* interoperability with other tools often based on tools from the Salome platform
  - OpenTURNS
  - code_aster coupling
  - ADAO, ...
- Setting up, building, or deploying environment with all the prerequisites increasingly complex

Salome_CFD workbench

Improved integration between components

- connection with SMESH for BC selections and probes placement
- graphical study creation and browsing
  - recently added support for studies with SYRTHES coupling

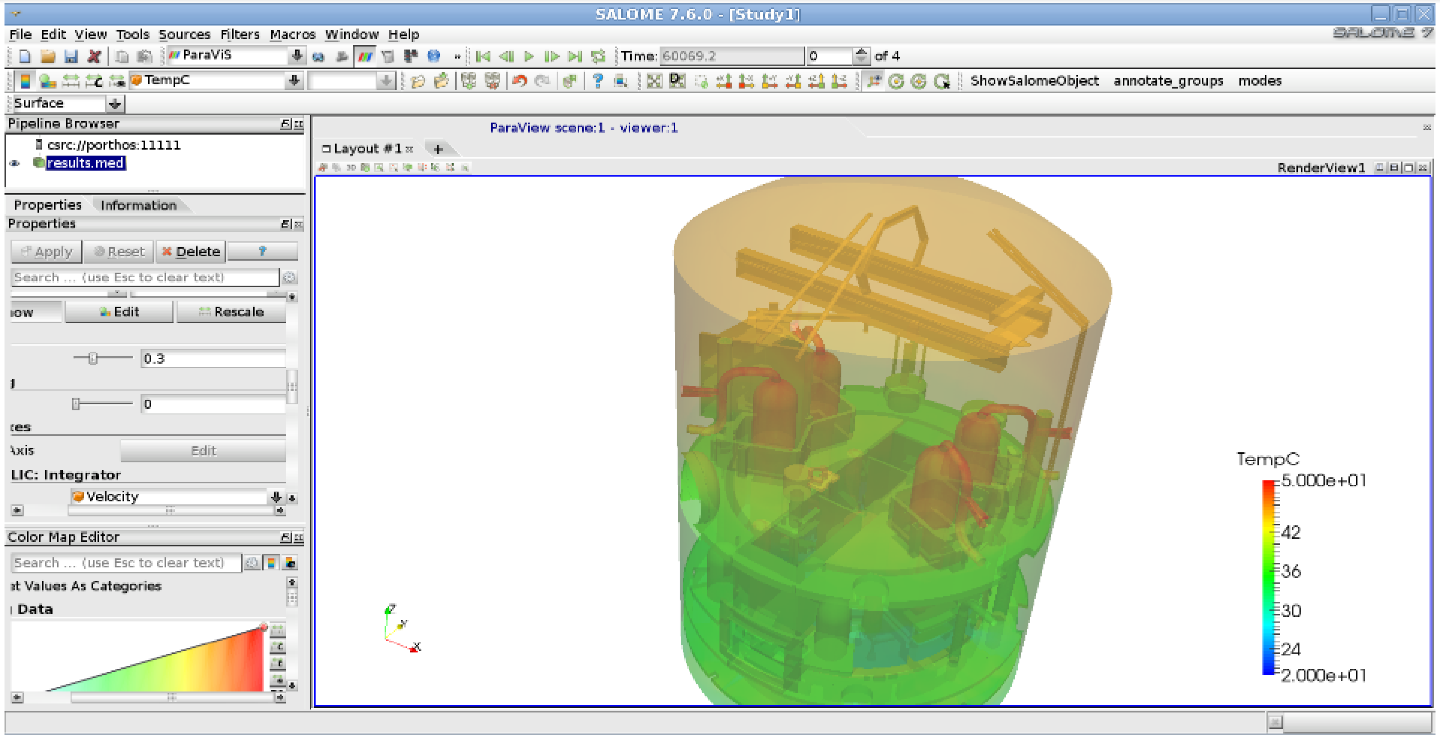


Label	Zone	Nature	Selection criteria
BC_2	1	Inlet	Inlet1
BC_1	3	Wall	Wall_inside or Wall_top or Wall or Wall_cold_leg

Salome_CFD workbench

Improved integration between components

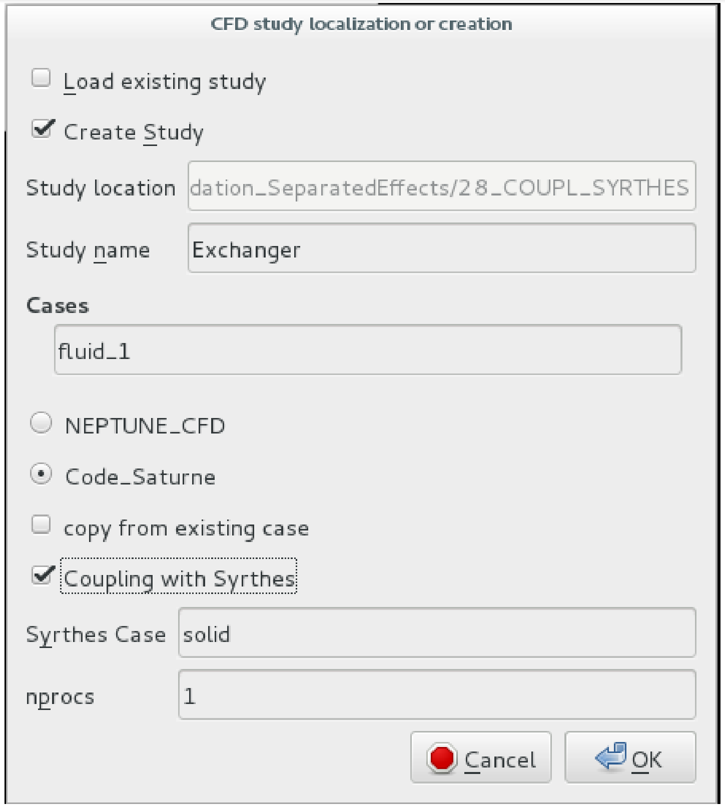
- connection with SMESH for BC selections and probes placement
- graphical study creation and browsing
    - recently added support for studies with SYRTHES coupling



Salome_CFD workbench

Improved integration between components

- connection with SMESH for BC selections and probes placement
- graphical study creation and browsing
  - recently added support for studies with SYRTHES coupling



CFD study localization or creation

□ Load existing study

☑ Create Study

Study location: dation_SeparatedEffects/28_COUPL_SYRTHES

Study name: Exchanger

Cases

fluid_1

○ NEPTUNE_CFD

● Code_Saturne

□ copy from existing case

☑ Coupling with Syrthes

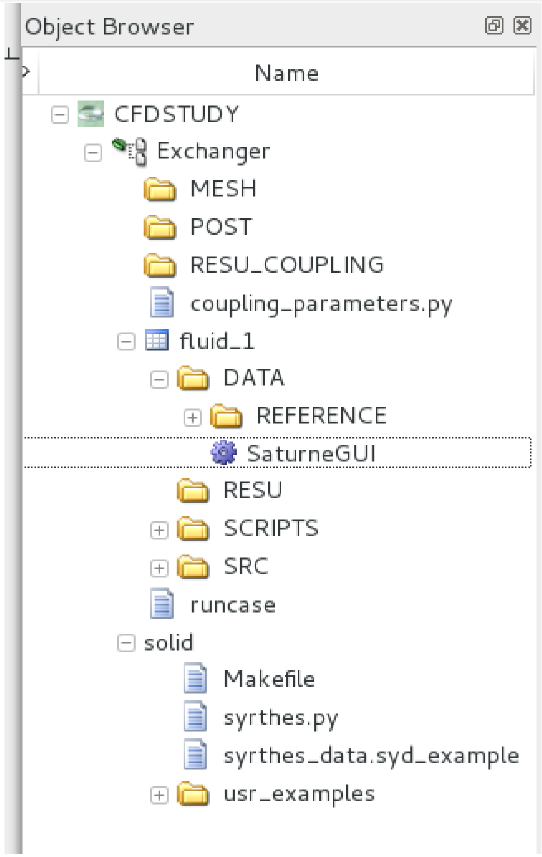
Syrthes Case: solid

nprocs: 1

Salome_CFD workbench

Improved integration between components

- connection with SMESH for BC selections and probes placement
- graphical study creation and browsing
  - recently added support for studies with SYRTHES coupling



Object Browser

Name

- CFDSTUDY
  - Exchanger
    - MESH
    - POST
    - RESU_COUPLING
    - coupling_parameters.py
    - fluid_1
      - DATA
        - REFERENCE
        - SaturneGUI
      - RESU
      - SCRIPTS
      - SRC
      - runcase
    - solid
      - Makefile
      - syrthes.py
      - syrthes_data.syd_example
      - usr_examples

Salome_CFD tools and modules

- General Salome modules
  - GEOM
  - SMESH
  - PARAVIS (visualization cluster connection preconfigured)
  - Homard
  - ...
- *Code_Saturne*
  - both production and debug (with additional checks and instrumentation) builds included
  - ParaView-based Catalyst in-situ visualization included
- NEPTUNE_CFD (optional, restricted distribution)
- SYRTHES
- OpenTURNS
- ADAO

Salome_CFD distribution

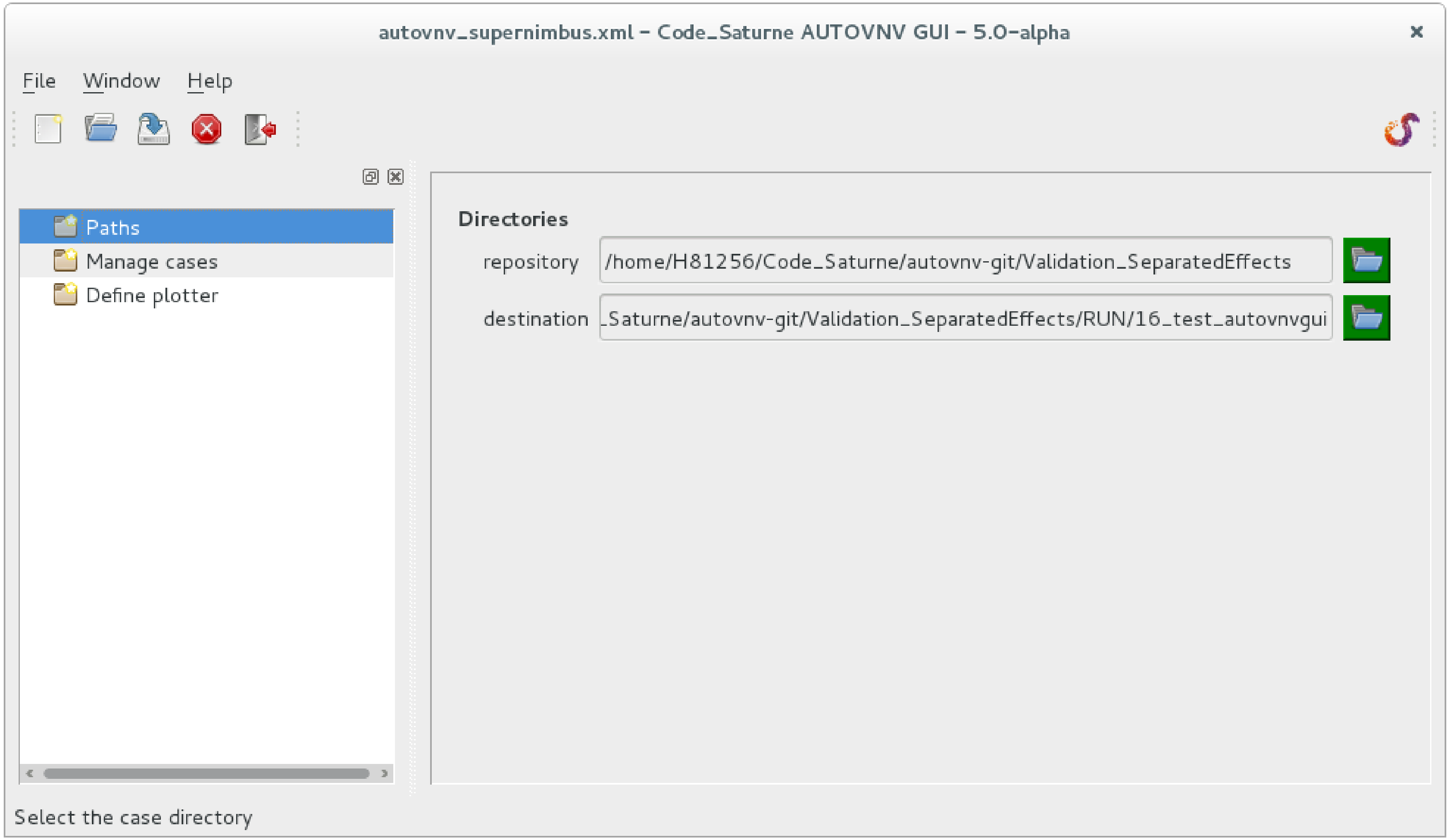
Status

- First EDF internal release of Salome_CFD October 2015
- managed in collaboration with Salome team and maintenance since 2016
- automated builds of multiple distributions
  - EDF Linux workstation (Calibre 9/Scibian 8)
  - EDF Linux workstation with NEPTUNE_CFD (Calibre 9/Scibian 8)
  - "universal" Linux workstation
- currently in testing
- release date for Salome-8/*Code_Saturne* 5.0-based version: September 2017
- "universal" version's technology may evolve in the future (Docker ?)
- Windows builds may become available in the future

Simplify data setting

New GUI for studymanager tool (previously autovnv)

- Add a GUI for StudyManaGeR launcher: `code_saturne studymanagergui` (V5.0)



autovnv_supernimbus.xml – Code_Saturne AUTOVNV GUI – 5.0-alpha

Directories

repository /home/H81256/Code_Saturne/autovnv-git/Validation_SeparatedEffects

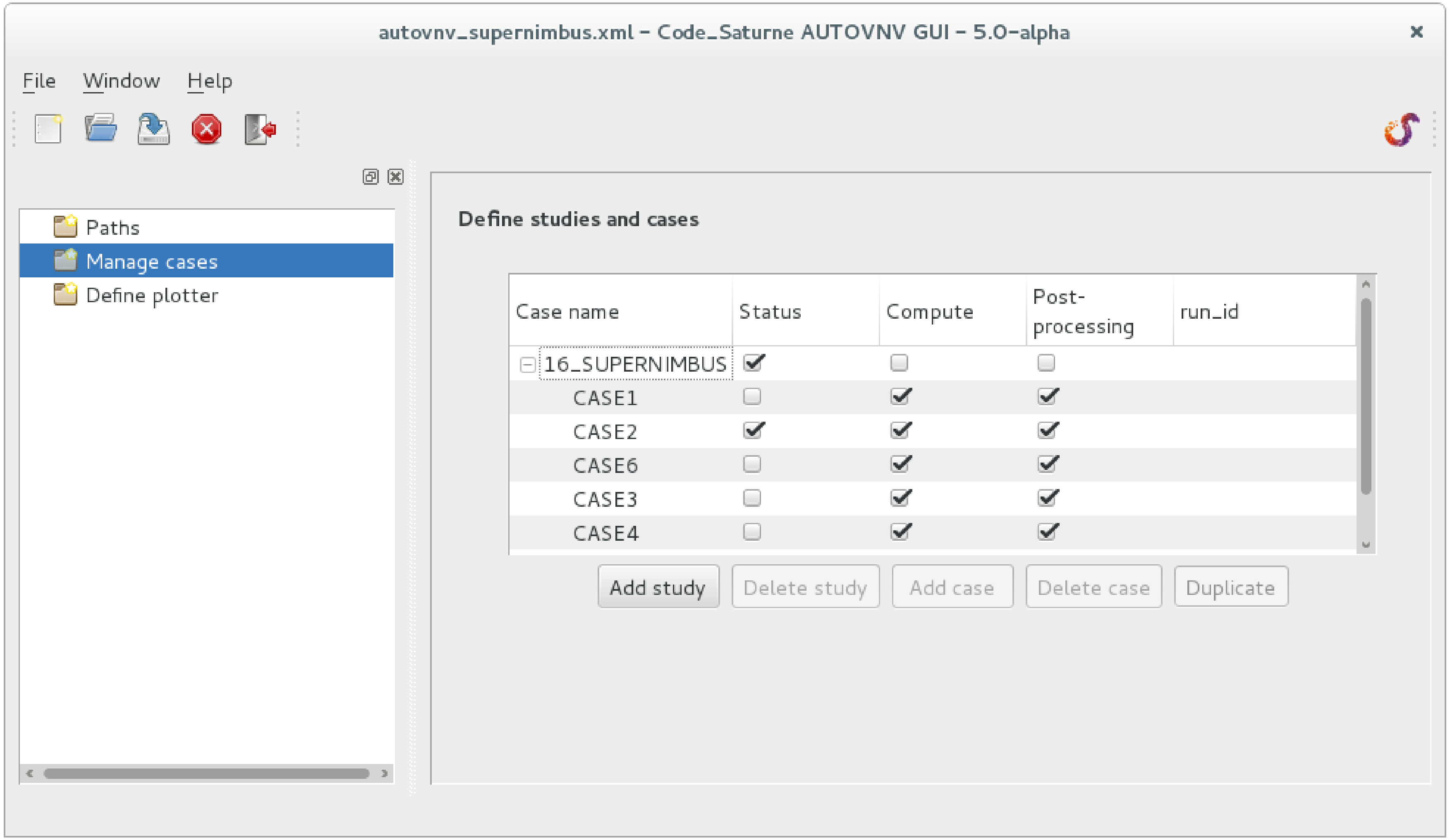
destination _Saturne/autovnv-git/Validation_SeparatedEffects/RUN/16_test_autovnvgui

Select the case directory

Simplify data setting

New GUI for studymanager tool (previously `autovnv`)

- Add a GUI for StudyManaGeR launcher:
  `code_saturne studymanagergui` (V5.0)



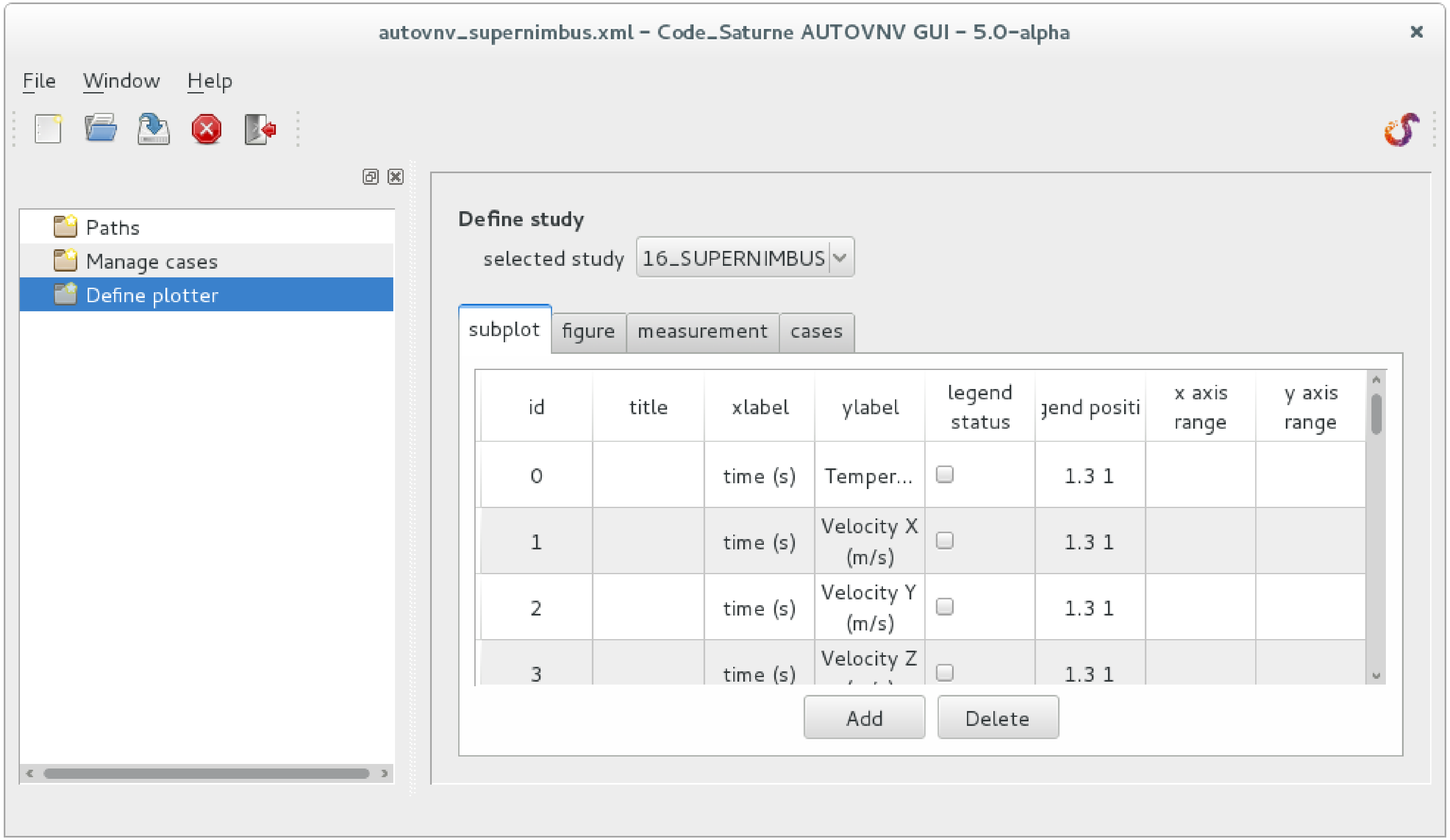
Define studies and cases

Case name	Status	Compute	Post-processing	run_id
⊟ 16_SUPERNIMBUS	☑	□	□	
CASE1	□	☑	☑	
CASE2	☑	☑	☑	
CASE6	□	☑	☑	
CASE3	□	☑	☑	
CASE4	□	☑	☑	

Simplify data setting

New GUI for studymanager tool (previously `autovnv`)

- Add a GUI for StudyManaGeR launcher: `code_saturne studymanagergui` (V5.0)



autovnv_supernimbus.xml – Code_Saturne AUTOVNV GUI – 5.0-alpha

Define study

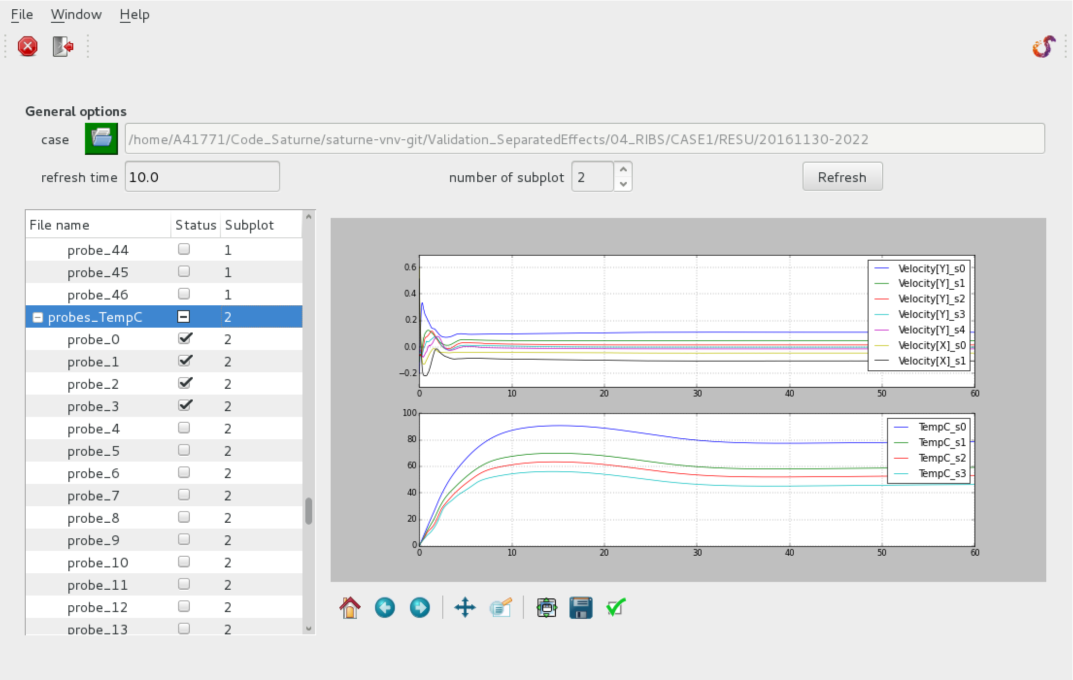
selected study: 16_SUPERNIMBUS

id	title	xlabel	ylabel	legend status	jend positi	x axis range	y axis range
0		time (s)	Temper...	☐	1.3 1		
1		time (s)	Velocity X (m/s)	☐	1.3 1		
2		time (s)	Velocity Y (m/s)	☐	1.3 1		
3		time (s)	Velocity Z	☐	1.3 1		

Simplify convergence analysis

New GUI to visualize data at probes and time residuals

- Add tracking convergence tool: `code_saturne trackcvg` (V5.0)



General options

case /home/A41771/Code_Saturne/saturne-vnv-git/Validation_SeparatedEffects/04_RIBS/CASE1/RESU/20161130-2022

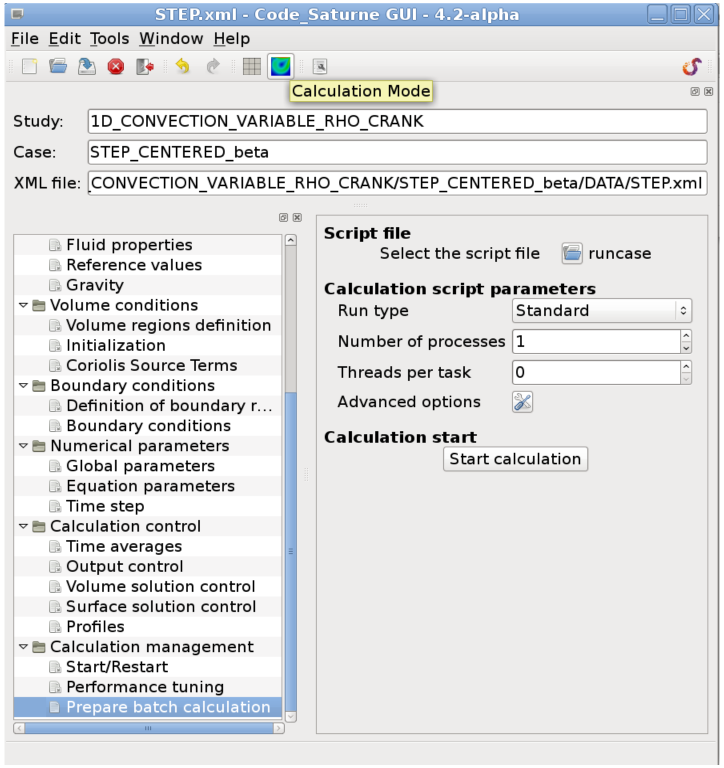
refresh time 10.0

number of subplot 2

File name	Status	Subplot
probe_44	□	1
probe_45	□	1
probe_46	□	1
⊟ probes_TempC	⊟	2
probe_0	☑	2
probe_1	☑	2
probe_2	☑	2
probe_3	☑	2
probe_4	□	2
probe_5	□	2
probe_6	□	2
probe_7	□	2
probe_8	□	2
probe_9	□	2
probe_10	□	2
probe_11	□	2
probe_12	□	2
probe_13	□	2

New preprocessor view

- "check mesh" option is replaced by a new "preprocessor view" in the GUI: when building a new case, the GUI only shows sections relative to mesh selection and preprocessing, showing only the steps necessary up to preprocessing (V4.2)
- "Tools" menu entries and toolbar icons allow switching from the preprocessing mode to the computation mode.
- the "preprocessing" run type handles batch runs and user subroutines, which the "check mesh" option did not.



STEP.xml - Code_Saturne GUI - 4.2-alpha

Study: 1D_CONVECTION_VARIABLE_RHO_CRANK

Case: STEP_CENTERED_beta

XML file: CONVECTION_VARIABLE_RHO_CRANK/STEP_CENTERED_beta/DATA/STEP.xml

Script file

Select the script file: runcase

Calculation script parameters

Run type: Standard

Number of processes: 1

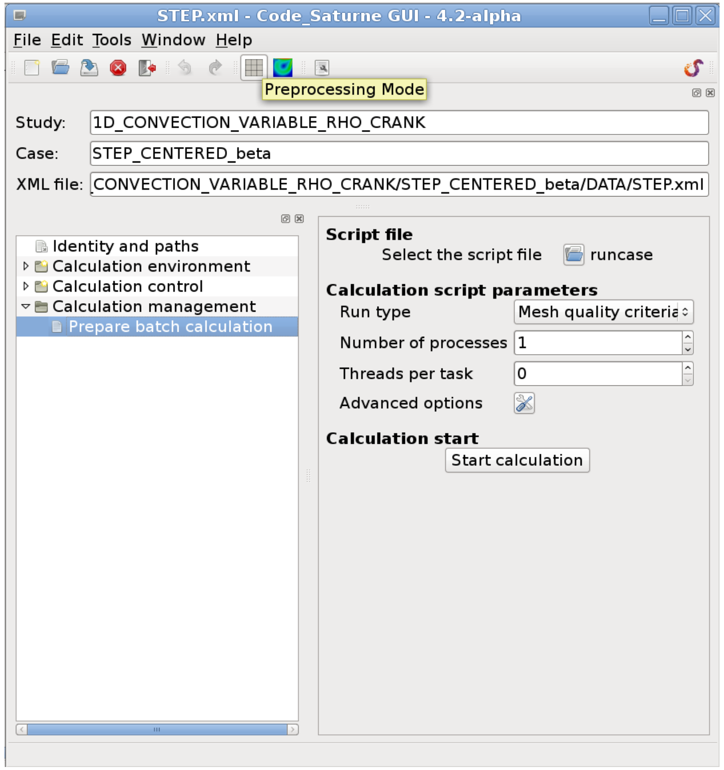
Threads per task: 0

Advanced options

Calculation start

New preprocessor view

- "check mesh" option is replaced by a new "preprocessor view" in the GUI: when building a new case, the GUI only shows sections relative to mesh selection and preprocessing, showing only the steps necessary up to preprocessing (V4.2)
- "Tools" menu entries and toolbar icons allow switching from the preprocessing mode to the computation mode.
- the "preprocessing" run type handles batch runs and user subroutines, which the "check mesh" option did not.



Study: 1D_CONVECTION_VARIABLE_RHO_CRANK

Case: STEP_CENTERED_beta

XML file: CONVECTION_VARIABLE_RHO_CRANK/STEP_CENTERED_beta/DATA/STEP.xml

Script file

Select the script file runcase

Calculation script parameters

Run type: Mesh quality criteria

Number of processes: 1

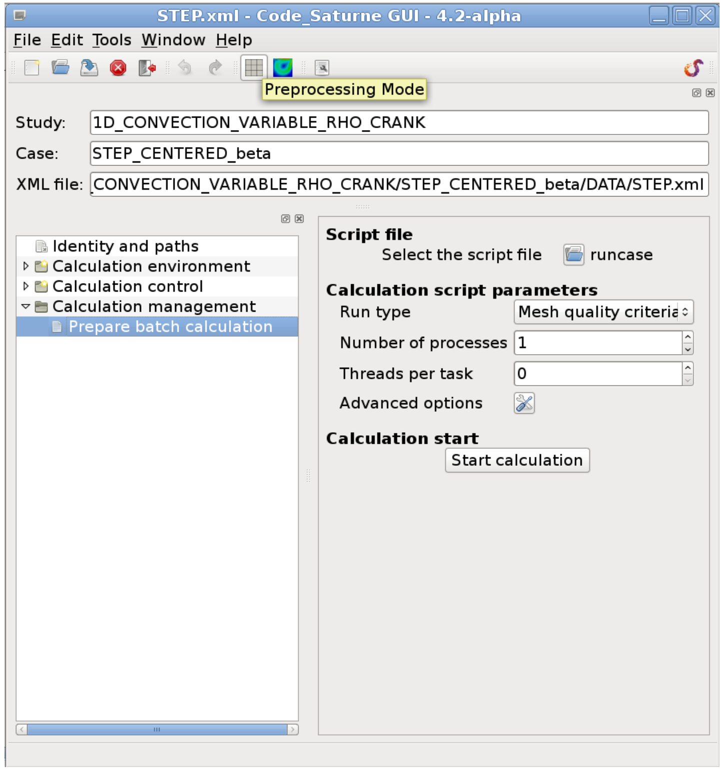
Threads per task: 0

Advanced options

Calculation start

New preprocessor view

- "check mesh" option is replaced by a new "preprocessor view" in the GUI: when building a new case, the GUI only shows sections relative to mesh selection and preprocessing, showing only the steps necessary up to preprocessing (V4.2)
- "Tools" menu entries and toolbar icons allow switching from the preprocessing mode to the computation mode.
- the "preprocessing" run type handles batch runs and user subroutines, which the "check mesh" option did not.



STEP.xml - Code_Saturne GUI - 4.2-alpha

Study: 1D_CONVECTION_VARIABLE_RHO_CRANK

Case: STEP_CENTERED_beta

XML file: CONVECTION_VARIABLE_RHO_CRANK/STEP_CENTERED_beta/DATA/STEP.xml

Script file

Select the script file: runcase

Calculation script parameters

Run type: Mesh quality criteria

Number of processes: 1

Threads per task: 0

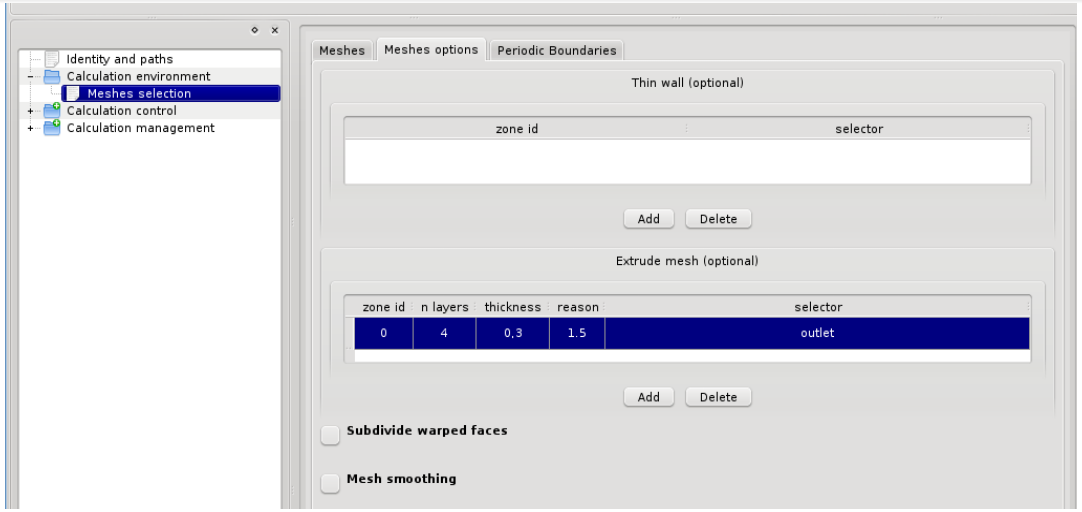
Advanced options

Calculation start

Pre-processing

Extrusion

- Add selected mesh boundary extrusion algorithm to extend a mesh (V4.3).
    - Available through simple and advanced user functions and through the GUI (for the simple variant).
    - done by the solver, so works in parallel
    - in case of periodicity, rebuilding the periodicity in a later preprocessing stage may be necessary).



Thin wall (optional)

zone id	selector

Extrude mesh (optional)

zone id	n layers	thickness	reason	selector
0	4	0,3	1.5	outlet

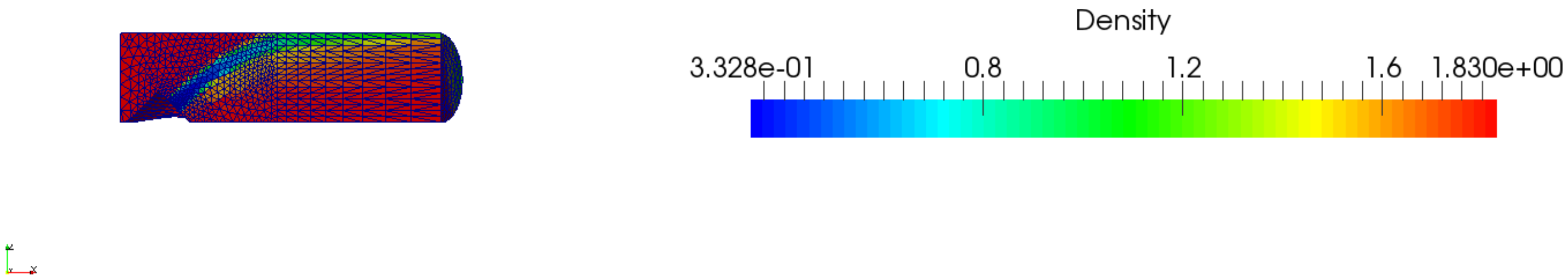
□ Subdivide warped faces

□ Mesh smoothing

Pre-processing

Extrusion

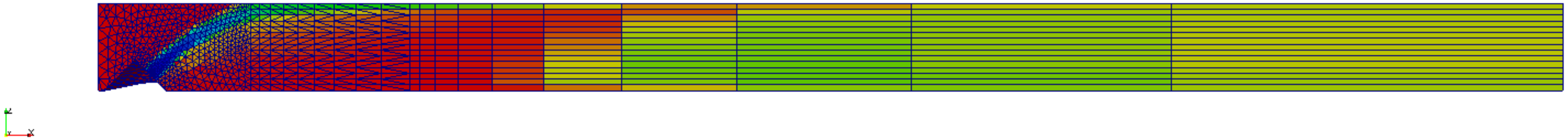
- Add selected mesh boundary extrusion algorithm to extend a mesh (V4.3).
  - Available through simple and advanced user functions and through the GUI (for the simple variant).
  - done by the solver, so works in parallel
  - in case of periodicity, rebuilding the periodicity in a later preprocessing stage may be necessary).



Pre-processing

Extrusion

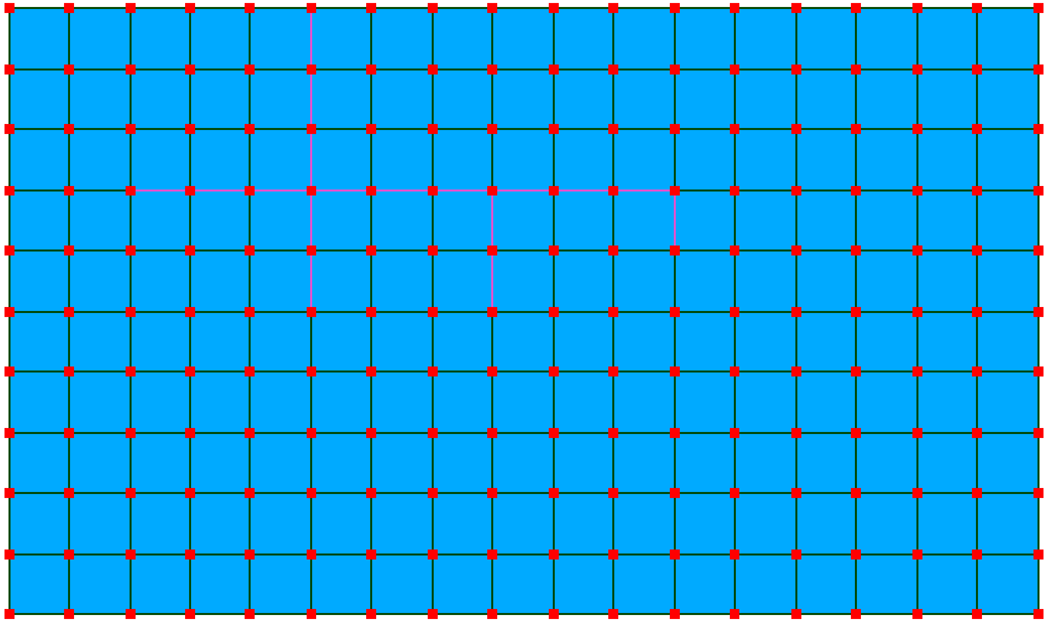
- Add selected mesh boundary extrusion algorithm to extend a mesh (V4.3).
  - Available through simple and advanced user functions and through the GUI (for the simple variant).
  - done by the solver, so works in parallel
  - in case of periodicity, rebuilding the periodicity in a later preprocessing stage may be necessary).



Some details about pre-processing

Interior to boundary faces

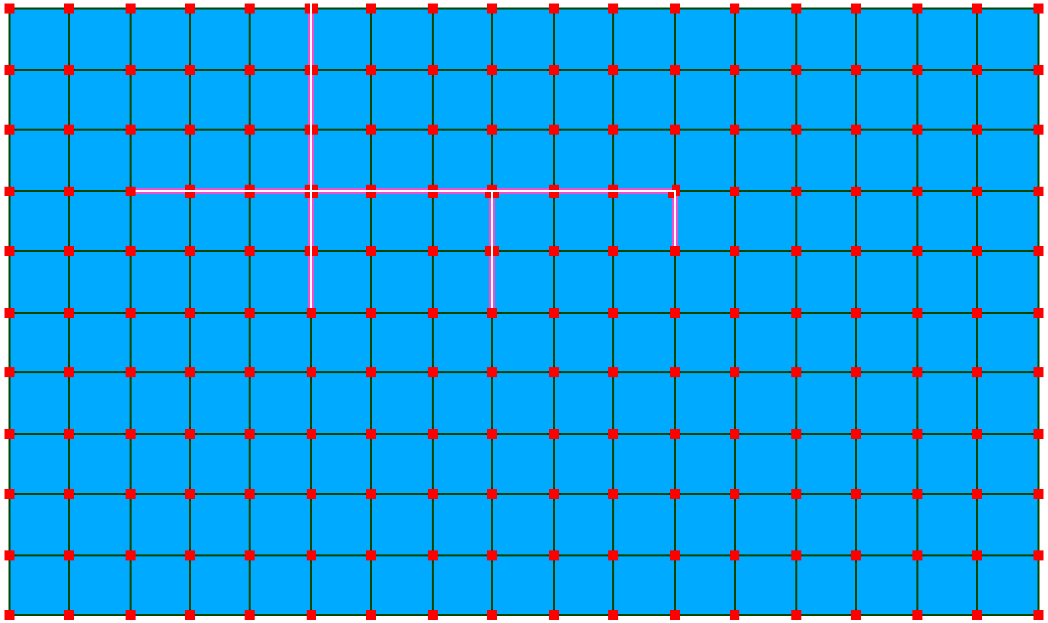
- Selected interior faces may be transformed into boundary faces.
  - Previously available in part through user functions as “thin walls”..
- Vertices, not just faces, are now duplicated.
  - handles selection boundaries (shared vertices) and intersections (leading to more than 2 vertices) correctly;
  - so faces are topologically different;
  - in case of deforming mesh, both sides must be handled;
  - compatible with vertex-based discretizations, such as CDO.



Some details about pre-processing

Interior to boundary faces

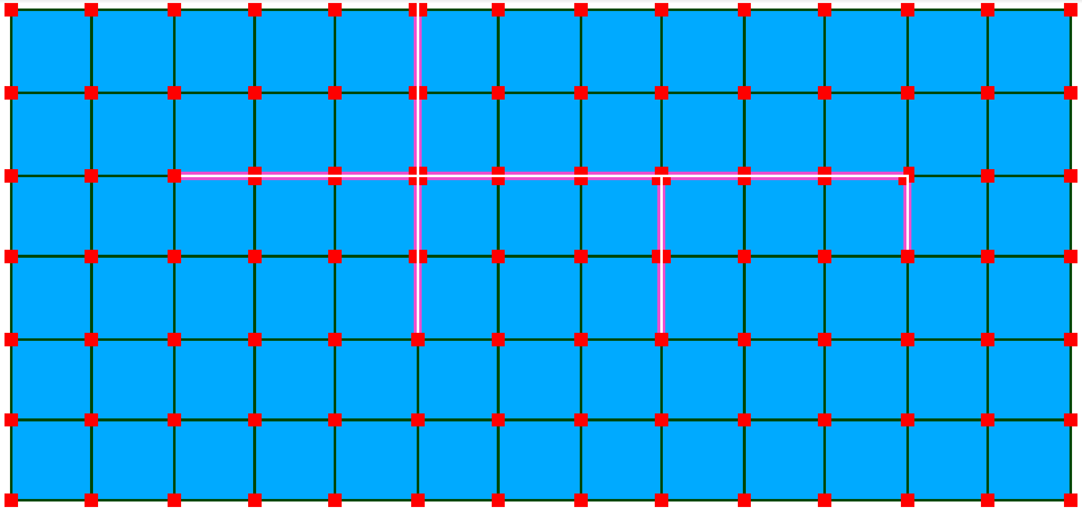
- Selected interior faces may be transformed into boundary faces.
  - Previously available in part through user functions as “thin walls”..
- Vertices, not just faces, are now duplicated.
  - handles selection boundaries (shared vertices) and intersections (leading to more than 2 vertices) correctly;
  - so faces are topologically different;
  - in case of deforming mesh, both sides must be handled;
  - compatible with vertex-based discretizations, such as CDO.



Some details about pre-processing

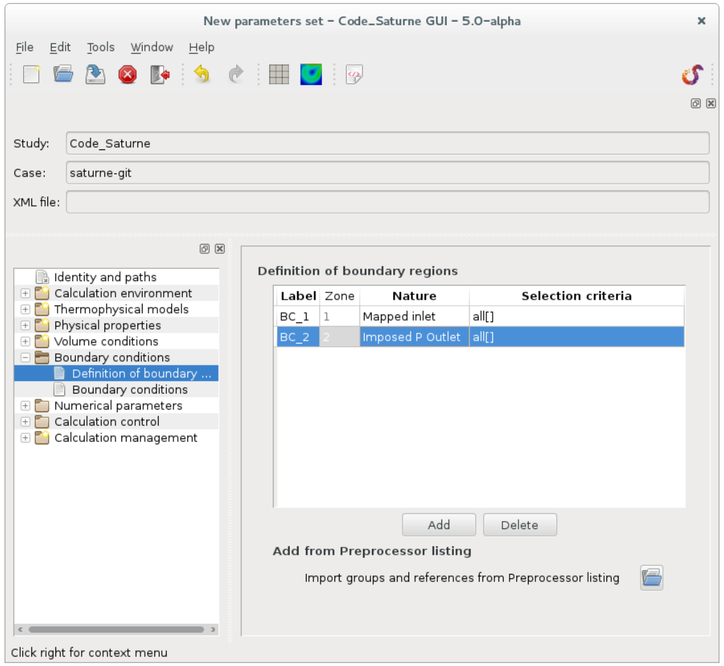
Interior to boundary faces

- Selected interior faces may be transformed into boundary faces.
  - Previously available in part through user functions as “thin walls”..
- Vertices, not just faces, are now duplicated.
  - handles selection boundaries (shared vertices) and intersections (leading to more than 2 vertices) correctly;
  - so faces are topologically different;
  - in case of deforming mesh, both sides must be handled;
  - compatible with vertex-based discretizations, such as CDO.



New boundary conditions in the GUI

- Add "mapped inlet" boundary condition (for recycled inlets) (V5.0)
- Add "imposed pressure" outlet boundary condition (V5.0)



Study: Code_Saturne

Case: saturne-git

XML file:

Definition of boundary regions

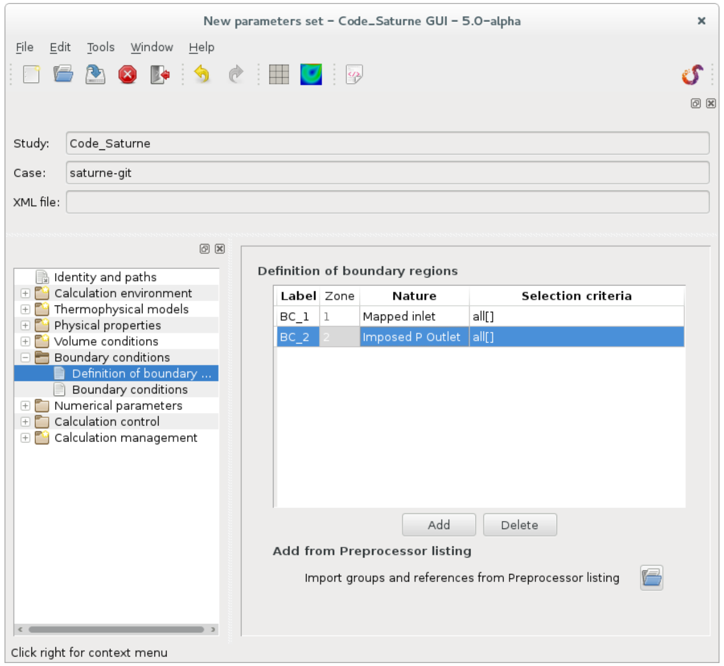
Label	Zone	Nature	Selection criteria
BC_1	1	Mapped inlet	all[]
BC_2	2	Imposed P Outlet	all[]

Add from Preprocessor listing

Import groups and references from Preprocessor listing

New boundary conditions in the GUI

- Add "mapped inlet" boundary condition (for recycled inlets) (V5.0)
- Add "imposed pressure" outlet boundary condition (V5.0)



New parameters set – Code_Saturne GUI – 5.0-alpha

Study: Code_Saturne

Case: saturne-git

XML file:

Definition of boundary regions

Label	Zone	Nature	Selection criteria
BC_1	1	Mapped inlet	all[]
BC_2	2	Imposed P Outlet	all[]

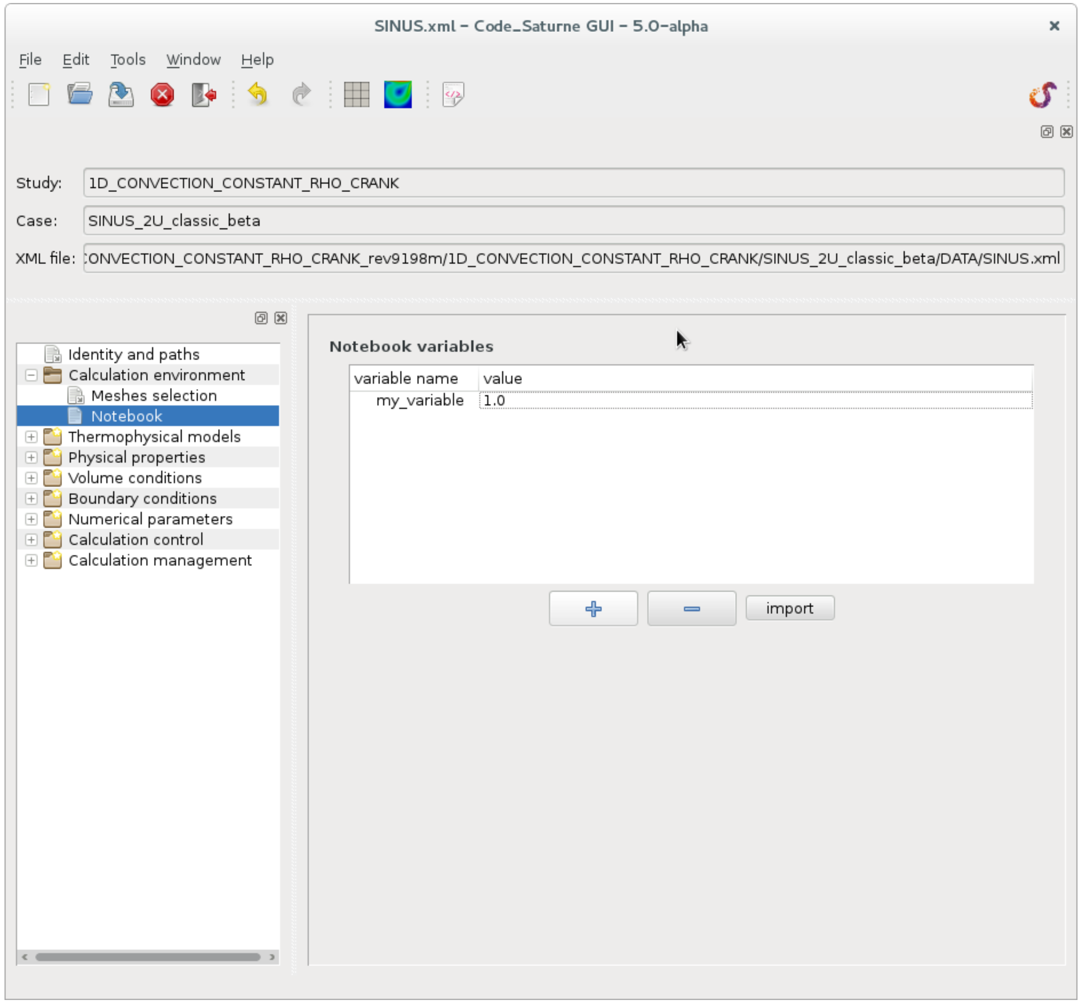
Add from Preprocessor listing

Import groups and references from Preprocessor listing

Click right for context menu

Notebook for global variables in the GUI

- Add a notebook to add global variables to be used in mathematical expressions (such as variables in the physical laws) (V5.0)



Study: 1D_CONVECTION_CONSTANT_RHO_CRANK

Case: SINUS_2U_classic_beta

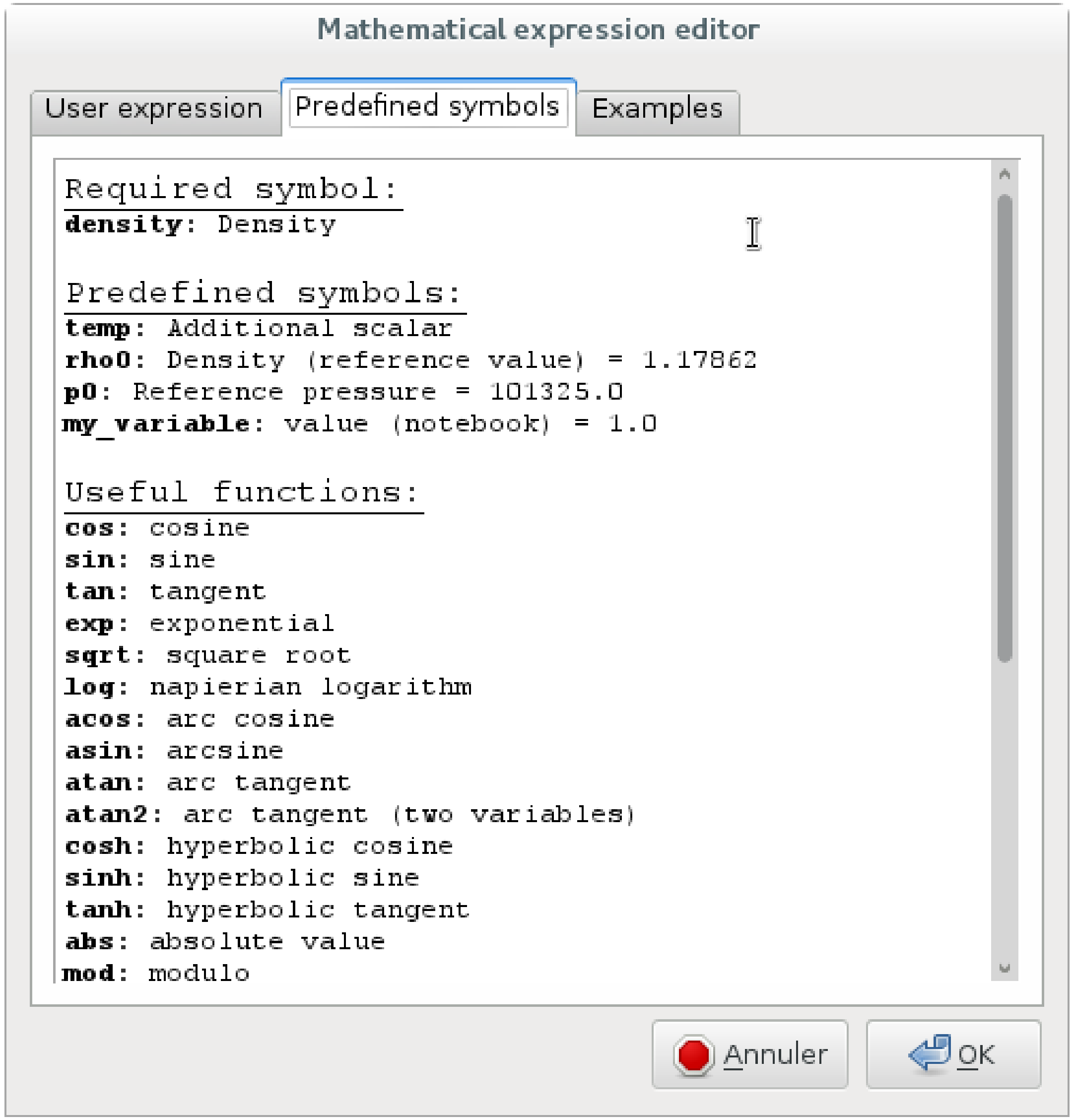
XML file: :ONVECTION_CONSTANT_RHO_CRANK_rev9198m/1D_CONVECTION_CONSTANT_RHO_CRANK/SINUS_2U_classic_beta/DATA/SINUS.xml

Notebook variables

variable name	value
my_variable	1.0

Notebook for global variables in the GUI

- Add a notebook to add global variables to be used in mathematical expressions (such as variables in the physical laws) (V5.0)



Required symbol:
density: Density

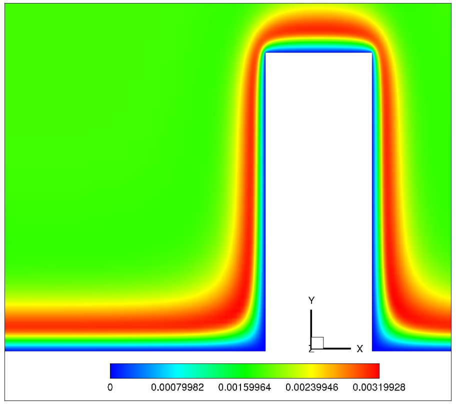
Predefined symbols:
temp: Additional scalar
rho0: Density (reference value) = 1.17862
p0: Reference pressure = 101325.0
my_variable: value (notebook) = 1.0

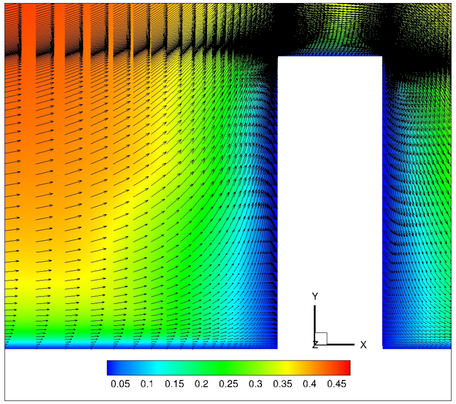
Useful functions:
cos: cosine
sin: sine
tan: tangent
exp: exponential
sqrt: square root
log: napierian logarithm
acos: arc cosine
asin: arcsine
atan: arc tangent
atan2: arc tangent (two variables)
cosh: hyperbolic cosine
sinh: hyperbolic sine
tanh: hyperbolic tangent
abs: absolute value
mod: modulo

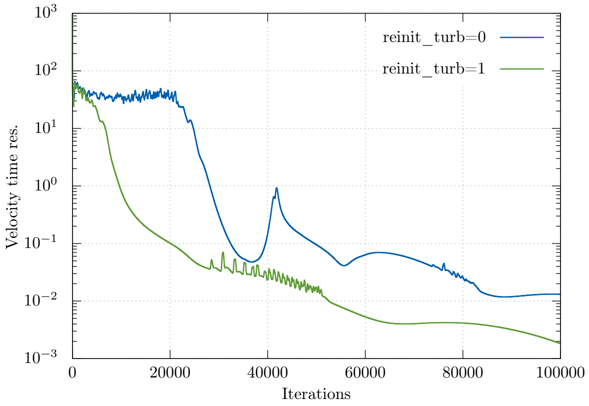
Simplify data setting, Initialisation ...

Automatic initialization of the Turbulence for EBRSM and $k - \omega$ models (V4.2)

From a reference velocity (`uref`), the turbulence profiles are reset after the first iteration. The velocity magnitude is also changed so that the Reichard profile is imposed next to walls. Activate it with `reinit_turb=1` (in `usipsu`); Provided by R. Manceau (Uni. of Pau).







- Add handling of multiple compute builds through the GUI (V5.0).
- Add the verbosity mode for transported variables (V5.0).

Simplify data setting, Initialisation ...

Automatic initialization of the Turbulence for EBRSM and $k - \omega$ models (V4.2)

From a reference velocity (`uref`), the turbulence profiles are reset after the first iteration. The velocity magnitude is also changed so that the Reichard profile is imposed next to walls. Activate it with `reinit_turb=1` (in `usipsu`); Provided by R. Manceau (Uni. of Pau).

Volume or boundary settings

- Allow zone-based definitions for condensation model (recommended). The examples are updated as well, though single-zone setups remain compatible.
- Add a boundary condition code (`icodcl=11`), allowing to easily impose a boundary face value of the form detailed below (used for the wall pressure in the compressible module) (V4.1).

- Add handling of multiple compute builds through the GUI (V5.0).
- Add the verbosity mode for transported variables (V5.0).

Simplify data setting, Initialisation ...

Automatic initialization of the Turbulence for EBRSM and $k-\omega$ models (V4.2)

From a reference velocity (`uref`), the turbulence profiles are reset after the first iteration. The velocity magnitude is also changed so that the Reichard profile is imposed next to walls. Activate it with `reinit_turb=1` (in `usipsu`); Provided by R. Manceau (Uni. of Pau).

Volume or boundary settings

- Allow zone-based definitions for condensation model (recommended). The examples are updated as well, though single-zone setups remain compatible.
- Add a boundary condition code (`icodcl=11`), allowing to easily impose a boundary face value of the form detailed below (used for the wall pressure in the compressible module) (V4.1).

$$P_{f_b} = \alpha P_{I'}^{n+1} + \beta,\ (\alpha, \beta) \text{ defined by the user}$$

- Add handling of multiple compute builds through the GUI (V5.0).
- Add the verbosity mode for transported variables (V5.0).

Simplify data setting, Initialisation ...

Automatic initialization of the Turbulence for EBRSM and $k - \omega$ models (V4.2)

From a reference velocity (`uref`), the turbulence profiles are reset after the first iteration. The velocity magnitude is also changed so that the Reichard profile is imposed next to walls. Activate it with `reinit_turb=1` (in `usipsu`); Provided by R. Manceau (Uni. of Pau).

Volume or boundary settings

- Allow zone-based definitions for condensation model (recommended). The examples are updated as well, though single-zone setups remain compatible.
- Add a boundary condition code (`icodcl=11`), allowing to easily impose a boundary face value of the form detailed below (used for the wall pressure in the compressible module) (V4.1).

- Add handling of multiple compute builds through the GUI (V5.0).
- Add the verbosity mode for transported variables (V5.0).

Simplify data setting, Initialisation ...

Automatic initialization of the Turbulence for EBRSM and $k-\omega$ models (V4.2)

From a reference velocity (`uref`), the turbulence profiles are reset after the first iteration. The velocity magnitude is also changed so that the Reichard profile is imposed next to walls. Activate it with `reinit_turb=1` (in `usipsu`); Provided by R. Manceau (Uni. of Pau).

Volume or boundary settings

- Allow zone-based definitions for condensation model (recommended). The examples are updated as well, though single-zone setups remain compatible.
- Add a boundary condition code (`icodcl=11`), allowing to easily impose a boundary face value of the form detailed below (used for the wall pressure in the compressible module) (V4.1).

- Add handling of multiple compute builds through the GUI (V5.0).
- Add the verbosity mode for transported variables (V5.0).

Simplify data setting, Initialisation ...

Automatic initialization of the Turbulence for EBRSM and $k-\omega$ models (V4.2)

From a reference velocity (`uref`), the turbulence profiles are reset after the first iteration. The velocity magnitude is also changed so that the Reichard profile is imposed next to walls. Activate it with `reinit_turb=1` (in `usipsu`); Provided by R. Manceau (Uni. of Pau).

Volume or boundary settings

- Allow zone-based definitions for condensation model (recommended). The examples are updated as well, though single-zone setups remain compatible.
- Add a boundary condition code (`icodcl=11`), allowing to easily impose a boundary face value of the form detailed below (used for the wall pressure in the compressible module) (V4.1).

- Add handling of multiple compute builds through the GUI (V5.0).
- Add the verbosity mode for transported variables (V5.0).

Simplify data setting, Boundary Conditions ...

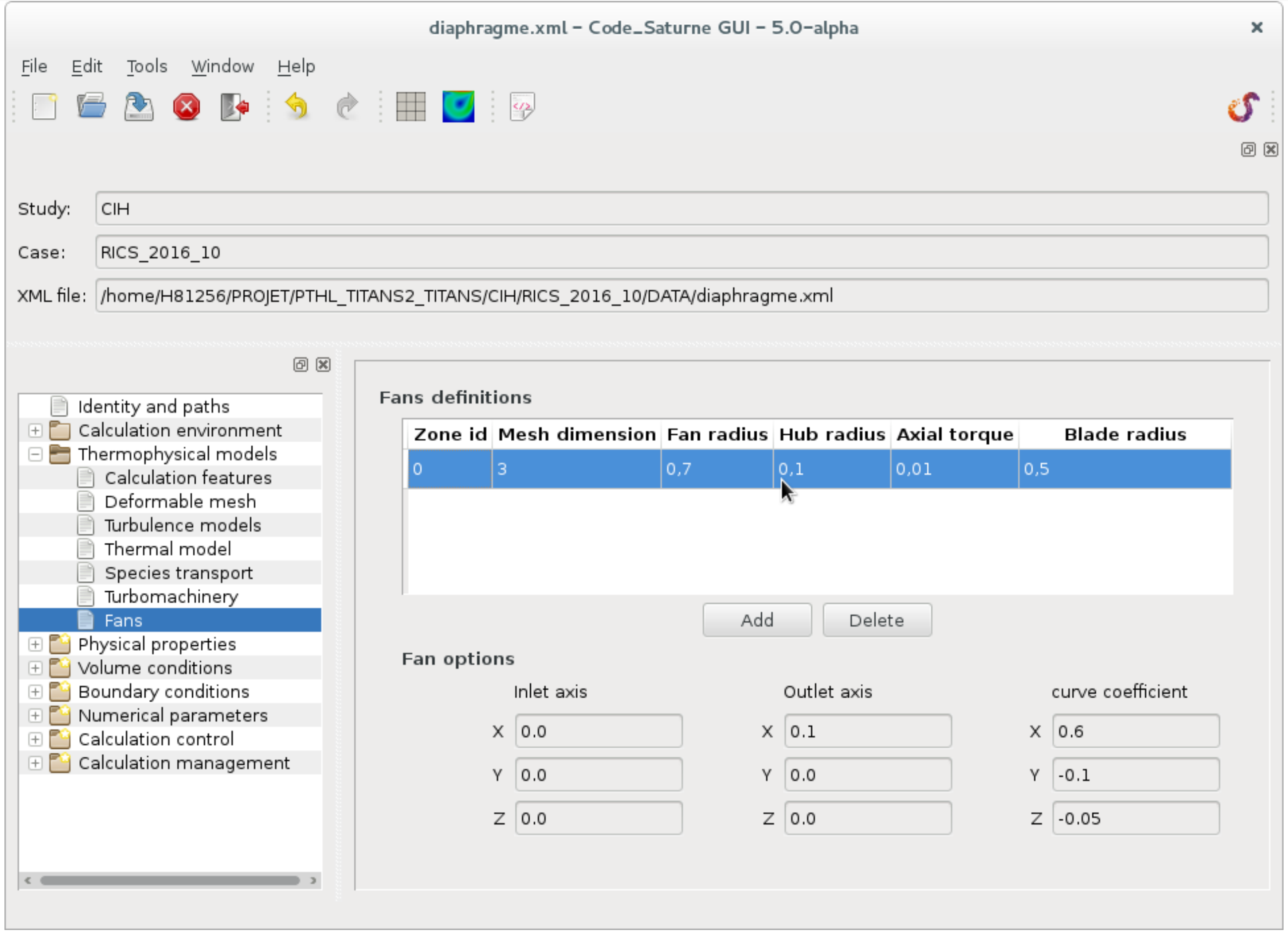
Volume settings

- Add simple fan effects modelling as explicit momentum source term in regions defined by fan characteristics (see `cs_user_fans.f90`) (V4.1).
- Add fans modelling in the V5.0.

Simplify data setting, Boundary Conditions ...

Volume settings

- Add simple fan effects modelling as explicit momentum source term in regions defined by fan characteristics (see `cs_user_fans.f90`) (V4.1).
- Add fans modelling in the V5.0.



Study: CIH

Case: RICS_2016_10

XML file: /home/H81256/PROJET/PTHL_TITANS2_TITANS/CIH/RICS_2016_10/DATA/diaphragme.xml

Fans definitions

Zone id	Mesh dimension	Fan radius	Hub radius	Axial torque	Blade radius
0	3	0,7	0,1	0,01	0,5

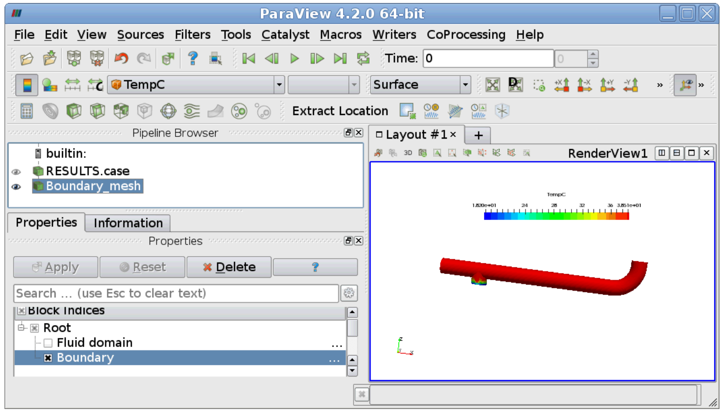
Fan options

	Inlet axis	Outlet axis	curve coefficient
X	0.0	0.1	0.6
Y	0.0	0.0	-0.1
Z	0.0	0.0	-0.05

Simplify data setting, Post-processing

Boundary post-processing

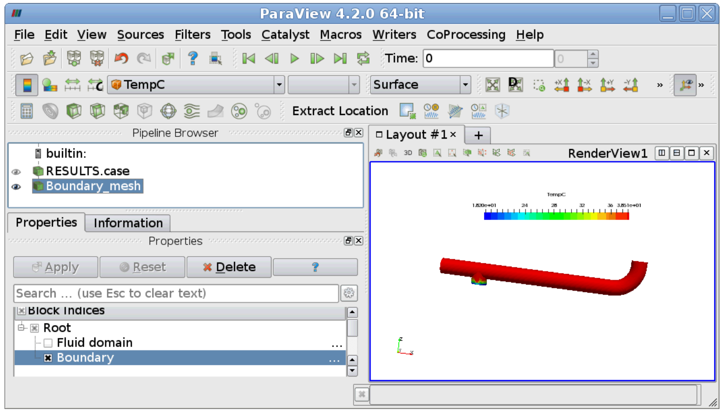
- Merge general boundary temperature handling with the radiative ”wall temperature”, for unified logging and post-processing (V4.2).
- Added optional saving of scalar variable boundary values as fields (also done for temperature when a property) (V4.2).



Simplify data setting, Post-processing

Boundary post-processing

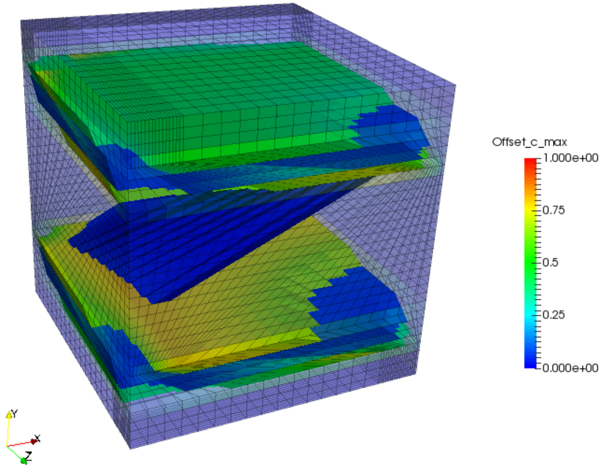
- Merge general boundary temperature handling with the radiative "wall temperature", for unified logging and post-processing (V4.2).
- Added optional saving of scalar variable boundary values as fields (also done for temperature when a property) (V4.2).

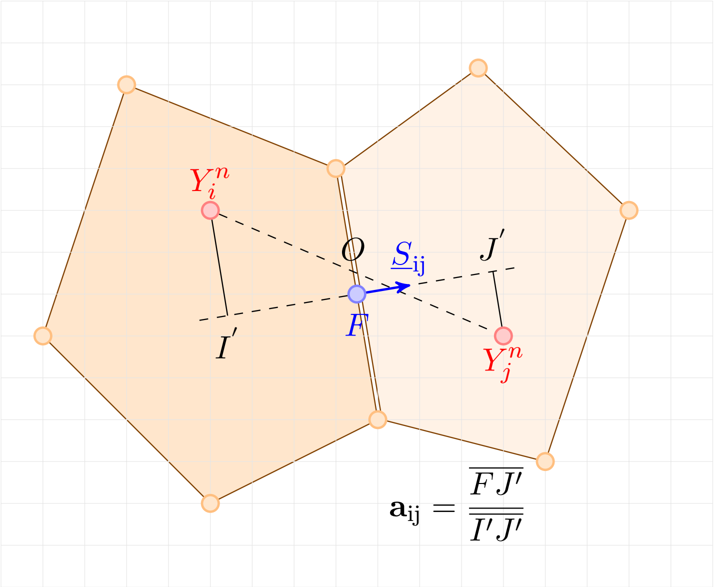


Simplify data setting, Post-processing

Volume post-processing

- Add boundary cell thickness computation to mesh quality criteria (V4.2).
- Renamed 'efforts' to 'stress', which should be less confusing (V4.1).
- Add higher level functions for turbulent boundary condition settings. This allows moving tests on the current turbulence model inside the user-callable functions, for more concise and safer programming (V4.1).
- Merge the bad cell and the mesh quality criterion for offsetting (V4.2).
- Add "iterative process error estimators" in the GUI in "Volume solution contol". pannel (V5.0).

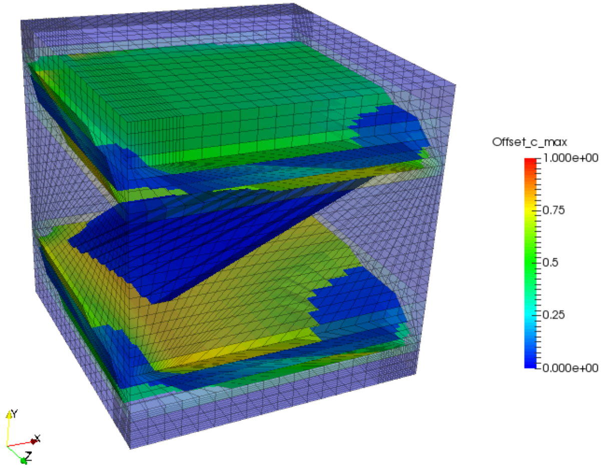


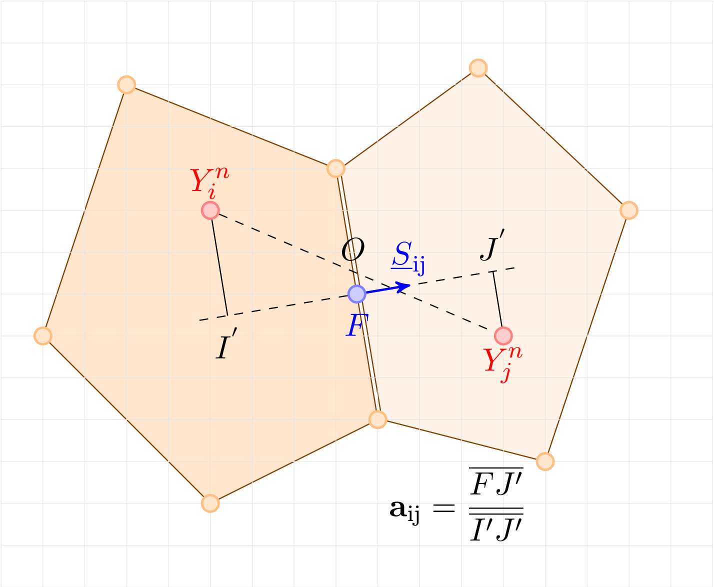


Simplify data setting, Post-processing

Volume post-processing

- Add boundary cell thickness computation to mesh quality criteria (V4.2).
- Renamed 'efforts' to 'stress', which should be less confusing (V4.1).
- Add higher level functions for turbulent boundary condition settings. This allows moving tests on the current turbulence model inside the user-callable functions, for more concise and safer programming (V4.1).
- Merge the bad cell and the mesh quality criterion for offsetting (V4.2).
- Add "iterative process error estimators" in the GUI in "Volume solution contol". pannel (V5.0).

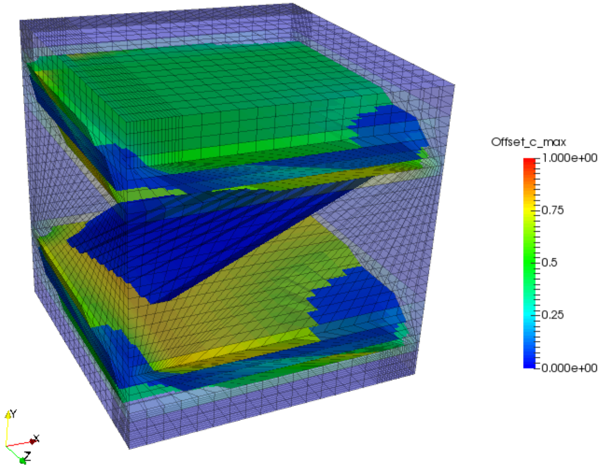


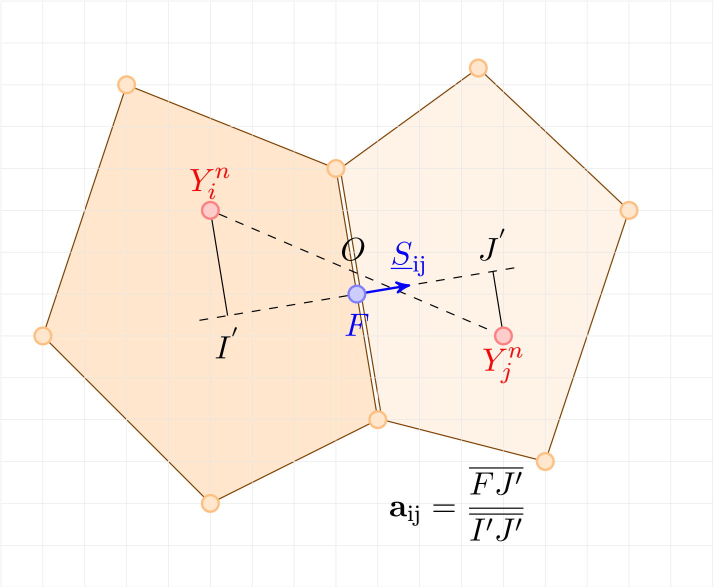


Simplify data setting, Post-processing

Volume post-processing

- Add boundary cell thickness computation to mesh quality criteria (V4.2).
- Renamed 'efforts' to 'stress', which should be less confusing (V4.1).
- Add higher level functions for turbulent boundary condition settings. This allows moving tests on the current turbulence model inside the user-callable functions, for more concise and safer programming (V4.1).
- Merge the bad cell and the mesh quality criterion for offsetting (V4.2).
- Add "iterative process error estimators" in the GUI in "Volume solution contol". pannel (V5.0).

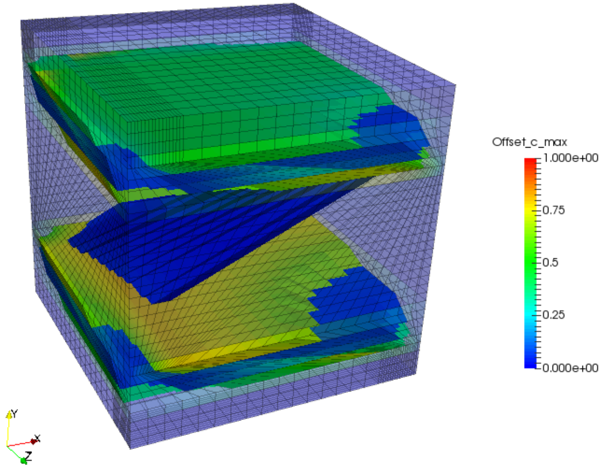


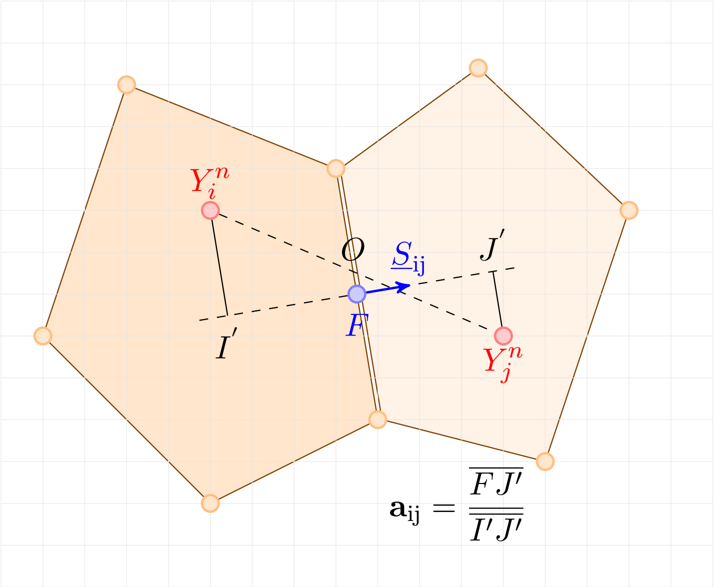


Simplify data setting, Post-processing

Volume post-processing

- Add boundary cell thickness computation to mesh quality criteria (V4.2).
- Renamed 'efforts' to 'stress', which should be less confusing (V4.1).
- Add higher level functions for turbulent boundary condition settings. This allows moving tests on the current turbulence model inside the user-callable functions, for more concise and safer programming (V4.1).
- Merge the bad cell and the mesh quality criterion for offsetting (V4.2).
- Add "iterative process error estimators" in the GUI in "Volume solution contol". pannel (V5.0).

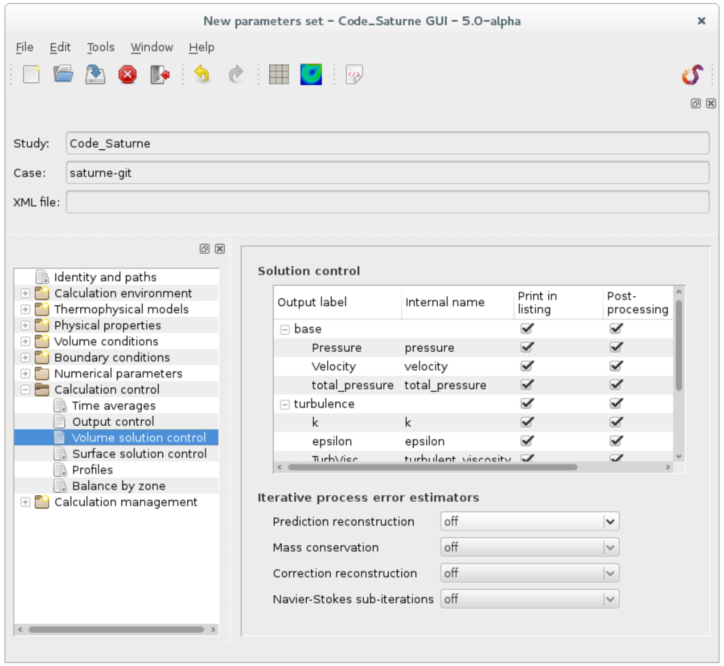




Simplify data setting, Post-processing

Volume post-processing

- Add boundary cell thickness computation to mesh quality criteria (V4.2).
- Renamed 'efforts' to 'stress', which should be less confusing (V4.1).
- Add higher level functions for turbulent boundary condition settings. This allows moving tests on the current turbulence model inside the user-callable functions, for more concise and safer programming (V4.1).
- Merge the bad cell and the mesh quality criterion for offsetting (V4.2).
- Add "iterative process error estimators" in the GUI in "Volume solution contol". pannel (V5.0).



Study: Code_Saturne

Case: saturne-git

XML file:

Solution control

Output label	Internal name	Print in listing	Post-processing
base		✓	✓
Pressure	pressure	✓	✓
Velocity	velocity	✓	✓
total_pressure	total_pressure	✓	✓
turbulence		✓	✓
k	k	✓	✓
epsilon	epsilon	✓	✓
TurbVisc	turbulent_viscosity	✓	✓

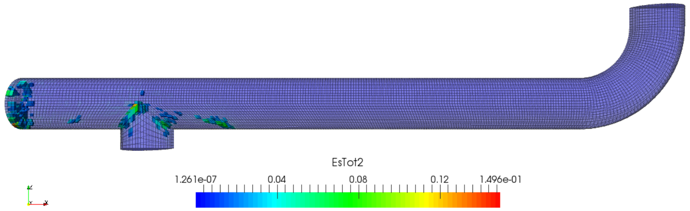
Iterative process error estimators

Prediction reconstruction off

Mass conservation off

Correction reconstruction off

Navier-Stokes sub-iterations off

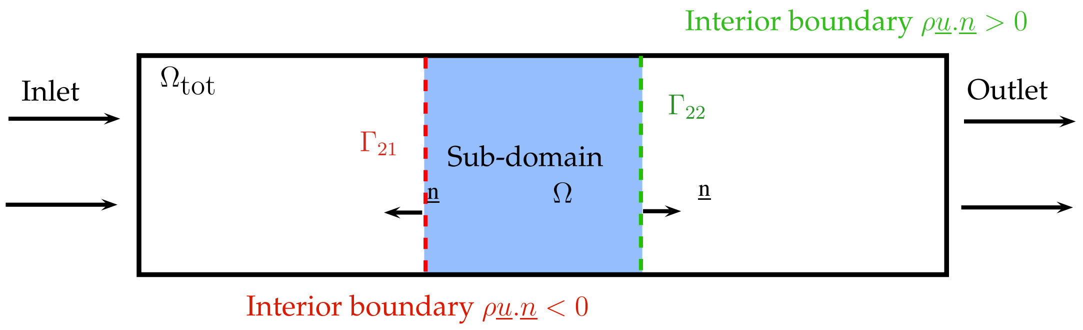


Pressure drop balance by zone

Head transport equation integrated over a sub-domain $\Omega \subset \Omega_{tot}$:

$$\int_{\Omega} \frac{\partial \rho u^2/2}{\partial t} d\Omega + \int_{\Omega} \mathrm{div}\left(\left[p + \rho \frac{u^2}{2} - \rho \underline{g} \cdot \underline{x}\right] \underline{u}\right) d\Omega = \int_{\Omega} \mathrm{div}\left(\underline{\underline{\tau}}_{viscous} \cdot \underline{u}\right) d\Omega - \int_{\Omega} \underline{\underline{\tau}}_{viscous} : \underline{\underline{S}} d\Omega$$

- unsteady term
- pressure term
  $\longrightarrow$ detailed over interior and exterior boundary
- redistribution term
- dissipation term.


Interior boundary $\rho \underline{u}.\underline{n} > 0$
Inlet
Ω_{tot}
Γ_{21}
Sub-domain
Ω
Γ_{22}
n
n
Outlet
Interior boundary $\rho \underline{u}.\underline{n} < 0$

How to perform a head loss balance?

Feature available in GUI or user subroutine (examples provided, see Doxygen).

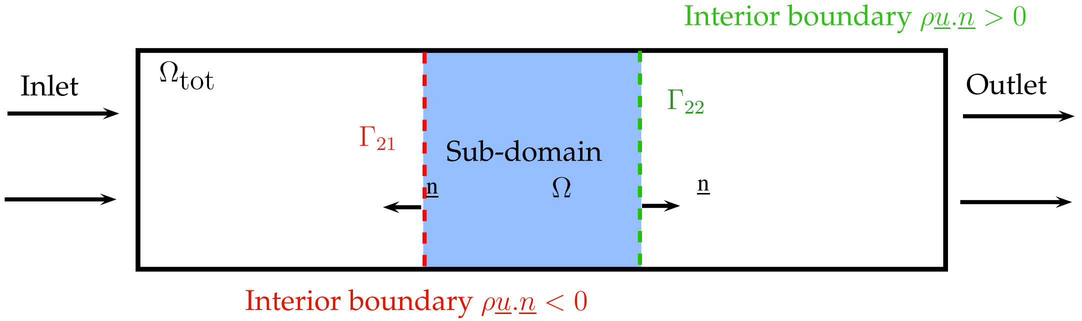
The detailed balance information will appear in the `listing` file.

Pressure drop balance by zone

Head transport equation integrated over a sub-domain $\Omega \subset \Omega_{tot}$:

$$\int_\Omega \frac{\partial \rho u^2/2}{\partial t} d\Omega + \int_\Omega \text{div}\left(\left[p + \rho\frac{u^2}{2} - \rho \underline{g} \cdot \underline{x}\right] \underline{u}\right) d\Omega = \int_\Omega \text{div}\left(\underline{\underline{\tau}}_{viscous} \cdot \underline{u}\right) d\Omega - \int_\Omega \underline{\underline{\tau}}_{viscous} : \underline{\underline{S}} d\Omega$$

- unsteady term
- pressure term
  ⟶ detailed over interior and exterior boundary
- redistribution term
- dissipation term.


Interior boundary $\rho \underline{u}.\underline{n} > 0$
Inlet
Ω_{tot}
Γ_{21}
Sub-domain
Ω
n
Γ_{22}
n
Outlet
Interior boundary $\rho \underline{u}.\underline{n} < 0$

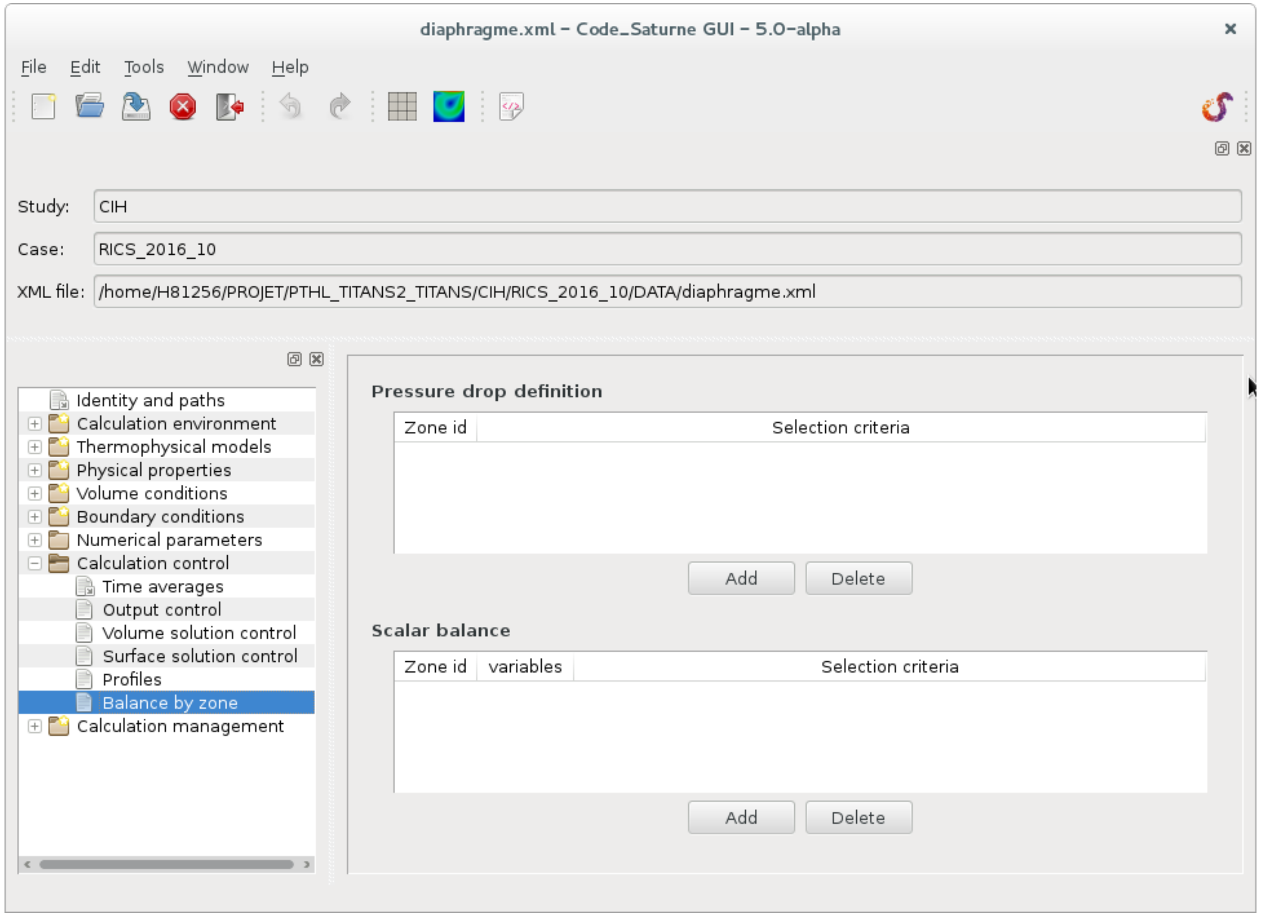
How to perform a head loss balance?

Feature available in GUI or user subroutine (examples provided, see Doxygen).

The detailed balance information will appear in the `listing` file.

Head loss balance for a flow through an orifice plate

Balance on "`z > 0.05 and z < 0.175`".



diaphragme.xml – Code_Saturne GUI – 5.0-alpha

Study: CIH

Case: RICS_2016_10

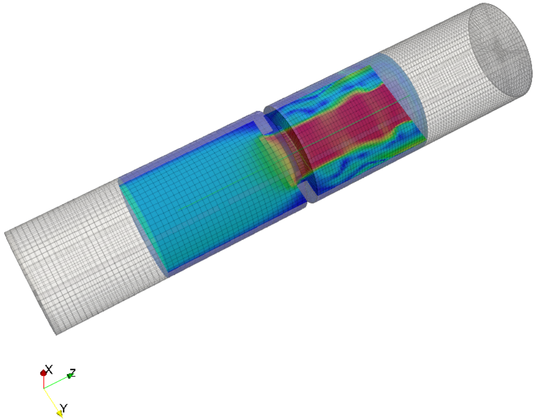
XML file: /home/H81256/PROJET/PTHL_TITANS2_TITANS/CIH/RICS_2016_10/DATA/diaphragme.xml

Pressure drop definition

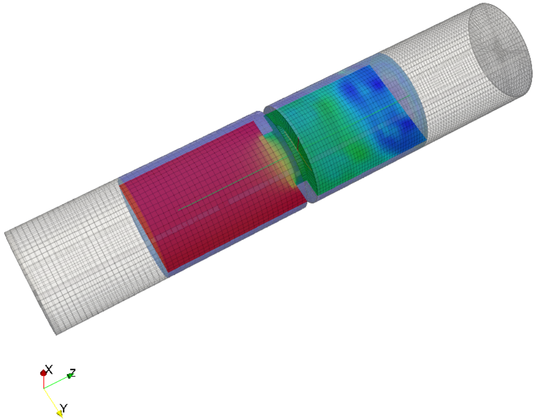
Zone id	Selection criteria

Scalar balance

Zone id	variables	Selection criteria



Velocity field on selected zone.



Pressure field on selected zone.

Overview

2 Physical modelling

- Compressible module
- Volume of Fluid module
- Cooling Tower module
- Lagrangian module
- Turbulence modelling
- Atmospheric module
- Internal coupling
- Others

News in the compressible module

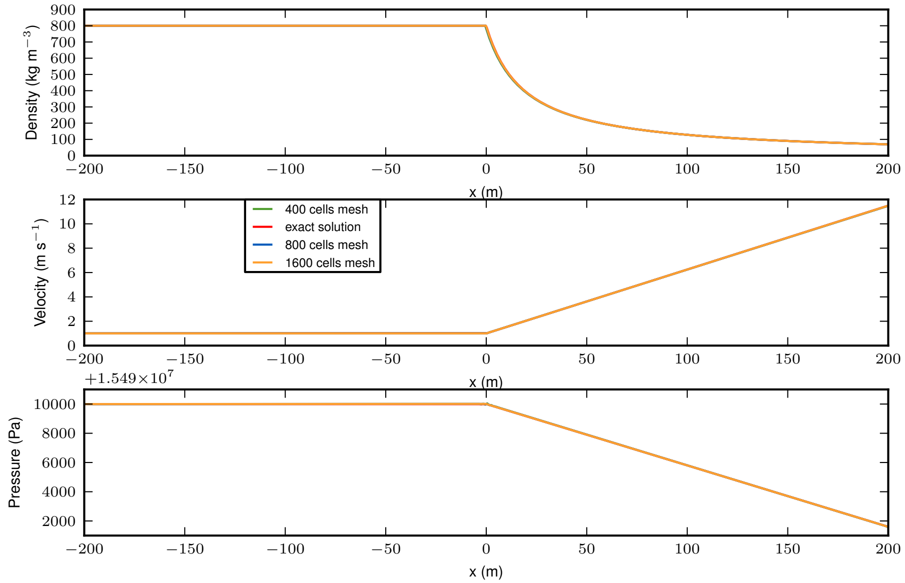
- Mass source terms are now usable. They are now correctly taken into account by the pressure step of the algorithm (V4.2).
- Add thermodynamic law for a perfect gas mix (V4.2):
    - gas mix (`igmix`, available in V4.0) and compressible (`icompf`) specific physics are used together
    - add property field for deduced mass fraction (`iddgas`) and mixture molar mass (`igmxml`)
    - add Sutherland behavior law for viscosity and thermal conductivity of gas mix (`ivsuth` option)
    - add one gas mix composed of helium, N_2 and O_2 (i.e. Helium+Air), O_2 is the deduced species.
- Add stiffened gas thermodynamic law (`ieos=2`) (V4.1). Set the new parameters `gammasg` (”pseudo” specific heat ratio) and `psginf` (infinite pressure) in `uscfx2`.

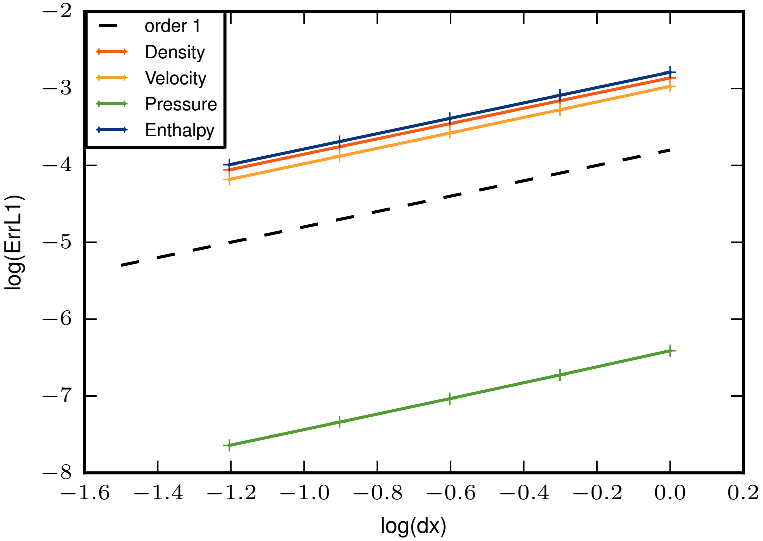
News in the compressible module

- Mass source terms are now usable. They are now correctly taken into account by the pressure step of the algorithm (V4.2).
- Add thermodynamic law for a perfect gas mix (V4.2):
    - gas mix (`igmix`, available in V4.0) and compressible (`icompf`) specific physics are used together
    - add property field for deduced mass fraction (`iddgas`) and mixture molar mass (`igmxml`)
    - add Sutherland behavior law for viscosity and thermal conductivity of gas mix (`ivsuth` option)
    - add one gas mix composed of helium, N_2 and O_2 (i.e. Helium+Air), O_2 is the deduced species.
- Add stiffened gas thermodynamic law (`ieos=2`) (V4.1). Set the new parameters `gammasg` ("pseudo" specific heat ratio) and `psginf` (infinite pressure) in `uscfx2`.

News in the compressible module

- Mass source terms are now usable. They are now correctly taken into account by the pressure step of the algorithm (V4.2).
- Add thermodynamic law for a perfect gas mix (V4.2):
  - gas mix (`igmix`, available in V4.0) and compressible (`icompf`) specific physics are used together
  - add property field for deduced mass fraction (`iddgas`) and mixture molar mass (`igmxml`)
  - add Sutherland behavior law for viscosity and thermal conductivity of gas mix (`ivsuth` option)
  - add one gas mix composed of helium, N_2 and O_2 (i.e. Helium+Air), O_2 is the deduced species.
- Add stiffened gas thermodynamic law (`ieos=2`) (V4.1). Set the new parameters `gammasg` (”pseudo” specific heat ratio) and `psginf` (infinite pressure) in `uscfx2`.





Volume of Fluid module

Model developed in collab. with RENUDA (V5.0) TALK

- mixture dynamic - incompressible Navier-Stokes equations:

$$\frac{\partial \rho}{\partial t} + \text{div}\ (\rho \underline{u}) = 0$$

$$\frac{\partial}{\partial t}(\rho \underline{u}) + \underline{\text{div}}\ (\underline{u} \otimes \rho \underline{u}) = -\underline{\nabla} P + \underline{\text{div}}\, \underline{\underline{\tau}}$$

- homogeneous mixture:
  $\rho = \alpha \rho_v + (1-\alpha)\rho_l$ and $\mu = \alpha \mu_v + (1-\alpha)\mu_l$

- void fraction pure convection:

$$\frac{\partial \alpha}{\partial t} + \text{div}\,(\alpha \underline{u}) = 0$$

with Compressive Interface Capturing Scheme for Arbitrary Meshes (CICSAM)

New physical models

Refurbished the Cooling Tower module in collab. with RENUDA (upcoming in V5.0)

- Use scalar with drift for the packing zones. TALK

News in the Lagrangian module

- Add precipitation/dissolution modelling for particle tracking (V4.2).
- Add the added-mass term in particle tracking (`iadded_mass=1`, V4.2).
- Changes in Lagrangian Particle tracking (V4.1).
  - update the multi-layer model
  - compute cell porosity from the mean deposition height at boundary faces
  - influence of deposited layers on the flow (`iflow = 1` option) (with additional head losses);
  - Energy barrier used for deposition computed for smooth and rough walls.

News in the Lagrangian module

- Add precipitation/dissolution modelling for particle tracking (V4.2).
- Add the added-mass term in particle tracking (`iadded_mass=1`, V4.2).
- Changes in Lagrangian Particle tracking (V4.1).
  - update the multi-layer model
  - compute cell porosity from the mean deposition height at boundary faces
  - influence of deposited layers on the flow (`iflow = 1` option) (with additional head losses);
  - Energy barrier used for deposition computed for smooth and rough walls.

News in the Lagrangian module

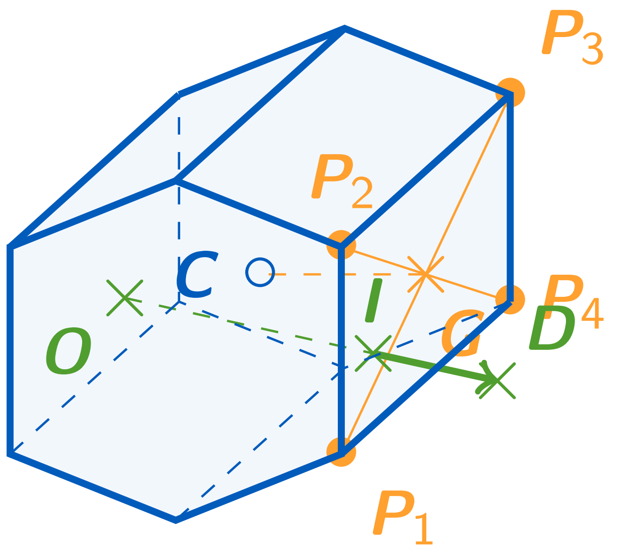
- Add precipitation/dissolution modelling for particle tracking (V4.2).
- Add the added-mass term in particle tracking (`iadded_mass=1`, V4.2).
- Changes in Lagrangian Particle tracking (V4.1).
  - update the multi-layer model
  - compute cell porosity from the mean deposition height at boundary faces
  - influence of deposited layers on the flow (`iflow = 1` option) (with additional head losses);
  - Energy barrier used for deposition computed for smooth and rough walls.

News in the Lagrangian module

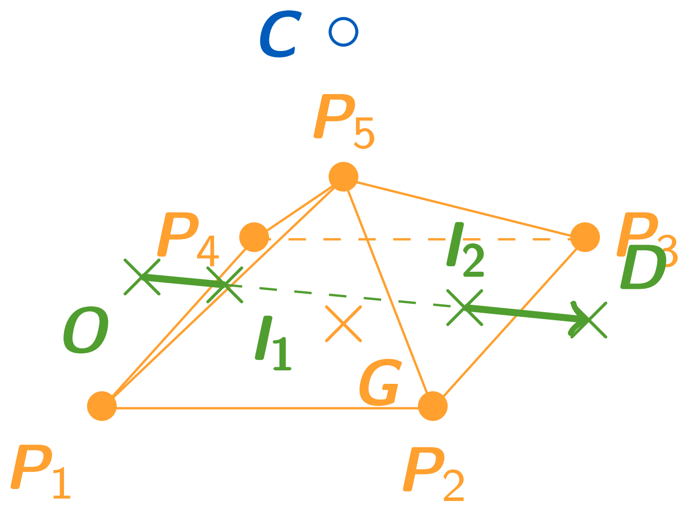
- Add deposition and resuspension models on internal faces. The user can the impose the motion of deposited particles. If integral approach for porous modelling is set up (`iporos=3`), then the internal fluid section is reduced by particle deposition (V4.3).
- New trajectory algorithm which does not loose particles even for warpped faces (V4.3).

News in the Lagrangian module

- Add deposition and resuspension models on internal faces. The user can the impose the motion of deposited particles. If integral approach for porous modelling is set up (`iporos=3`), then the internal fluid section is reduced by particle deposition (V4.3).
- New trajectory algorithm which does not loose particles even for warpped faces (V4.3).



Particle displacement from O to D within a cell.



Particle displacement from O to D going through a warped face.

News in the Lagrangian module

- injection is now pseudo-continuous when injecting at every time step. To revert to the previous behavior, the `CS_LAGR_RESIDENCE_TIME` value must be set to 0 for newly injected particles (V5.0).
- 2 new attributes, `CS_LAGR_TR_TRUNCATE` and `CS_LAGR_TR_REPOSITION`, may be used to visualize particles with trajectory errors, rather than remove them. Particles are now only removed when "completely lost", which should never happen (V5.0) (TALK).

Turbulence modelling

Turbulence modelling

- Add 2-scales wall function (with V. Driest mixing length) and its consistant wall function on scalars (keyword `iwallf` in the doc., V4.1).
- Add a wall function for the velocity based on scalable wall function which is valid for both rough and smooth walls (activate it with `iwallf=6` in `cs_user_parameters.f90`). Moreover, the continuous wall function based on Van Driest (`iwallf=5`) is extended to Eddy Viscosity Models, and the use of roughness is allowed. For both wall functions, roughness must be specified in the field nammed `boundary_roughness` in `cs_user_boundary_conditions.f90` for instance (V4.2).
- Modification of the LES dynamic Smagorinsky clippings (V5.0).
- Elliptic Blending Differential Flux Models for scalars (EBDFM, upcoming in V5.0). POSTER

Turbulence modelling

Turbulence modelling

- Add 2-scales wall function (with V. Driest mixing length) and its consistant wall function on scalars (keyword `iwallf` in the doc., V4.1).
- Add a wall function for the velocity based on scalable wall function which is valid for both rough and smooth walls (activate it with `iwallf=6` in `cs_user_parameters.f90`). Moreover, the continuous wall function based on Van Driest (`iwallf=5`) is extended to Eddy Viscosity Models, and the use of roughness is allowed. For both wall functions, roughness must be specified in the field nammed `boundary_roughness` in `cs_user_boundary_conditions.f90` for instance (V4.2).
- Modification of the LES dynamic Smagorinsky clippings (V5.0).
- Elliptic Blending Differential Flux Models for scalars (EBDFM, upcoming in V5.0). POSTER

TODO Atmo video

Turbulence modelling

Turbulence modelling

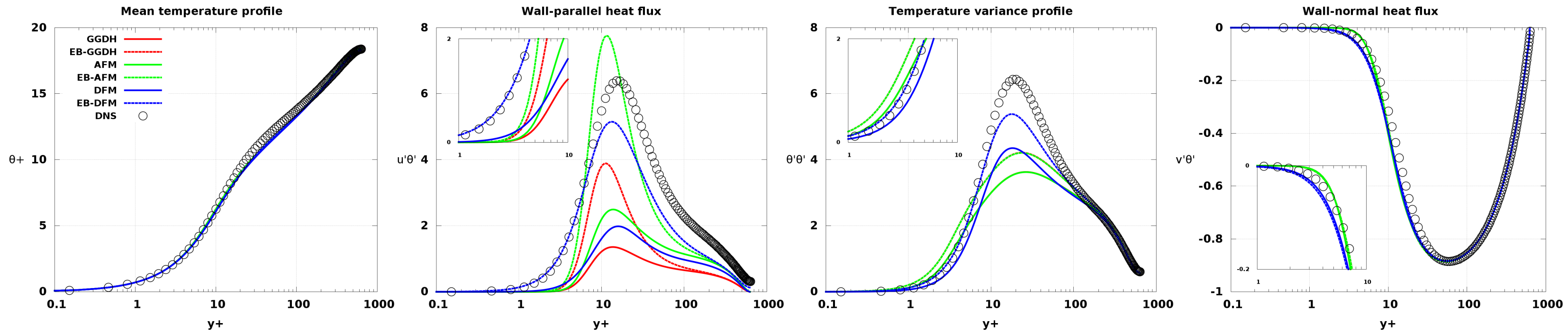
- Add 2-scales wall function (with V. Driest mixing length) and its consistant wall function on scalars (keyword `iwallf` in the doc., V4.1).
- Add a wall function for the velocity based on scalable wall function which is valid for both rough and smooth walls (activate it with `iwallf=6` in `cs_user_parameters.f90`). Moreover, the continuous wall function based on Van Driest (`iwallf=5`) is extended to Eddy Viscosity Models, and the use of roughness is allowed. For both wall functions, roughness must be specified in the field nammed `boundary_roughness` in `cs_user_boundary_conditions.f90` for instance (V4.2).
- Modification of the LES dynamic Smagorinsky clippings (V5.0).
- Elliptic Blending Differential Flux Models for scalars (EBDFM, upcoming in V5.0). POSTER

Turbulence modelling

Turbulence modelling

- Add 2-scales wall function (with V. Driest mixing length) and its consistant wall function on scalars (keyword `iwallf` in the doc., V4.1).
- Add a wall function for the velocity based on scalable wall function which is valid for both rough and smooth walls (activate it with `iwallf=6` in `cs_user_parameters.f90`). Moreover, the continuous wall function based on Van Driest (`iwallf=5`) is extended to Eddy Viscosity Models, and the use of roughness is allowed. For both wall functions, roughness must be specified in the field nammed `boundary_roughness` in `cs_user_boundary_conditions.f90` for instance (V4.2).
- Modification of the LES dynamic Smagorinsky clippings (V5.0).
- Elliptic Blending Differential Flux Models for scalars (EBDFM, upcoming in V5.0). POSTER

Thermal turbulent fluxes and temperature variance predictions in channel flow



Atmospheric module

- Improve robustness of rough boundary conditions for wall functions of scalars. Mainly impact Atmospheric flows.
- Add data assimilation feature to atmospheric module (optimal interpolation and nudging):
  - copy LU utilities to `cs_math` and keep static inline version of them in `cs_sles_it`
  - an optimal interpolation structure is created
  - interpol grid and measures set structures are used as well
  - multidimensional analysis are computed for multidimensional variables.

New physical models

Internal coupling between domains

- Add internal coupling for scalars of two domains (for instance temperature between solid and liquid, or enthalpy for electric arcs between plasma and weldpool). See `cs_user_parameters.c` (V5.0) POSTER, TALKS

New physical models

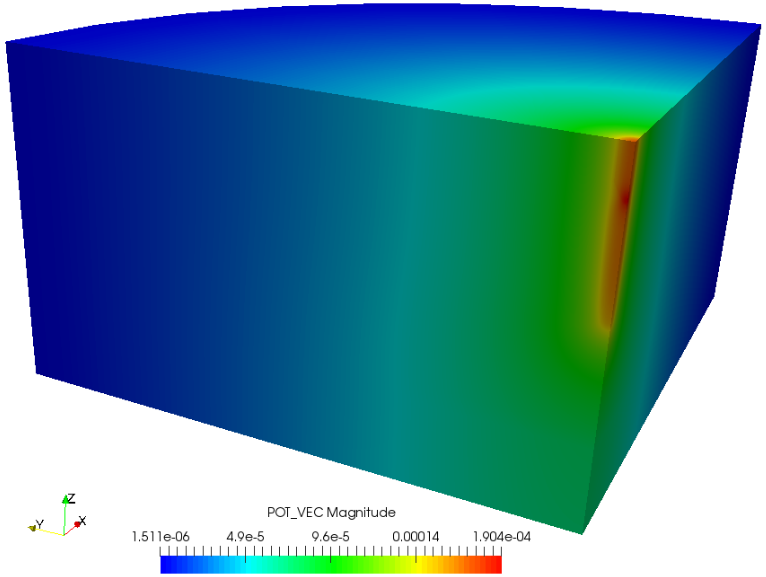
Other

- Add ADF models for radiative transfers (V4.1).
- Add a convection-diffusion equation solver for additional vector variables (V5.0).
- Add sorption model treating non-equilibrium between solid and liquid phases to the ground water flow module (V5.0).

New physical models

Other

- Add ADF models for radiative transfers (V4.1).
- Add a convection-diffusion equation solver for additional vector variables (V5.0).
- Add sorption model treating non-equilibrium between solid and liquid phases to the ground water flow module (V5.0).

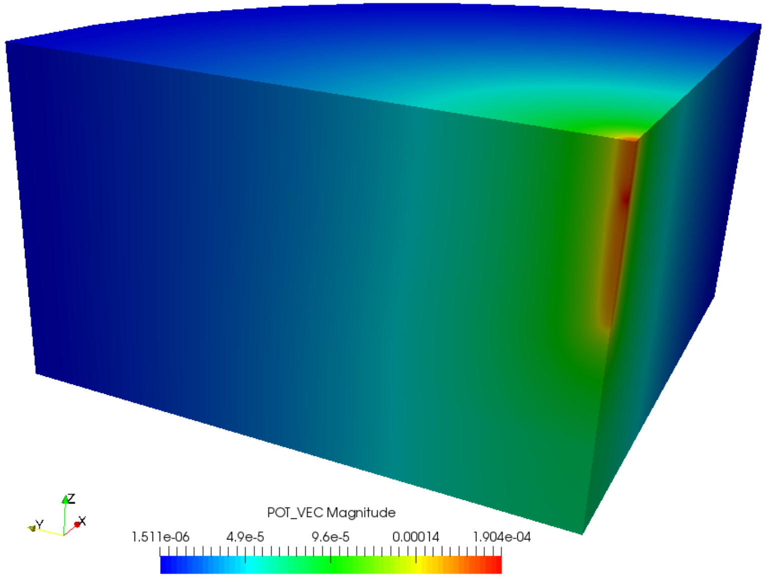


Coupled vector potential in electric arcs module

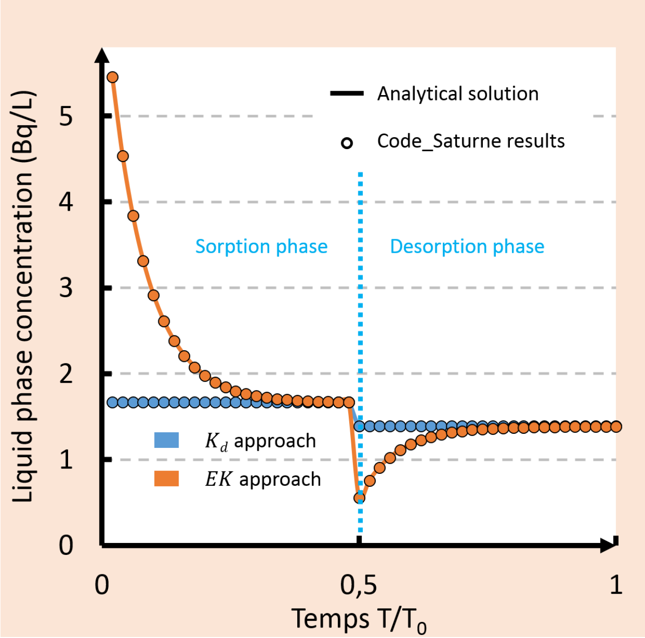
New physical models

Other

- Add ADF models for radiative transfers (V4.1).
- Add a convection-diffusion equation solver for additional vector variables (V5.0).
- Add sorption model treating non-equilibrium between solid and liquid phases to the ground water flow module (V5.0).



Coupled vector potential in electric arcs module



Non-equilibrium models in GWF module

Physical modules improvements

Turbomachinery

- To ensure correct restart behavior, the joined mesh is now also handled using checkpoint/restart.
- Added some turbomachinery post-processing utility functions (torque and manometric head). See `cs_user_extra_operations.c` Doxygen examples.

- Allow coupling of radiative transfer with 1d wall thermal module.
- Extend automatic postprocessing output to fields defined at vertices.
- Improve robustness of rough boundary conditions for wall functions of scalars. Mainly impact Atmospheric flows.

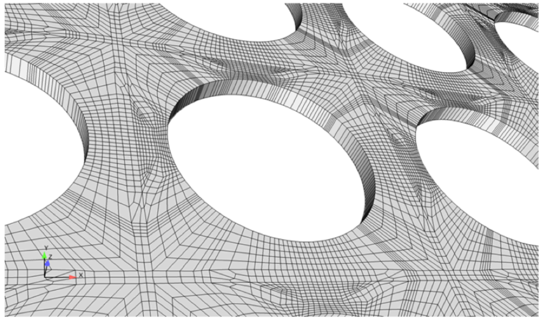
Overview

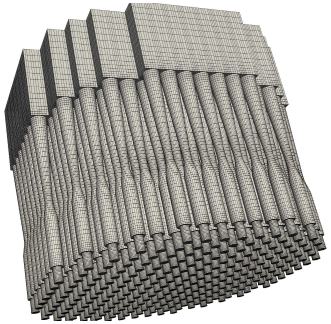
3 Numerics and linear solvers

- Compatible Discrete Operator (CDO) schemes
- Iterative solvers
- Others

CDO schemes: Newly available in *Code_Saturne* (V4.2)

- Design to be robust on polyhedral/distorted meshes
- State-of-the-art discretization schemes mixing FE and FV ideas
- V&V process completed



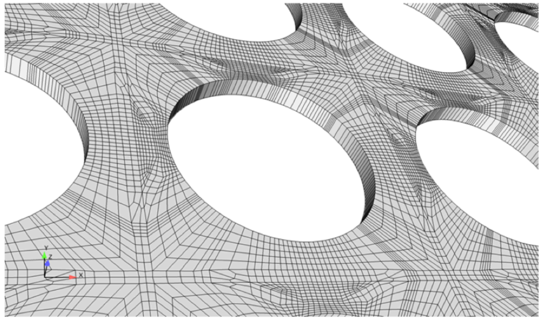


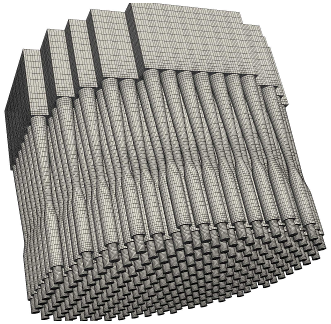
New features

- Add new CDO schemes for scalar transport equations
  - Degrees of freedom at vertices (V4.2) and at cells/vertices (V5.0)
  - Several diffusion/convection schemes and boundary enforcement – *Acknowledgment to P. Cantin (PhD)*
- Improve the modularity/integration of CDO schemes (V5.0)
  - New probe/profile mechanism
  - Monitoring (log files, timer stats. . .)

CDO schemes: Newly available in *Code_Saturne* (V4.2)

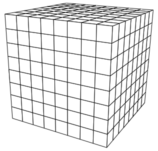
- Design to be robust on polyhedral/distorted meshes
- State-of-the-art discretization schemes mixing FE and FV ideas
- V&V process completed



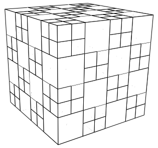


New features

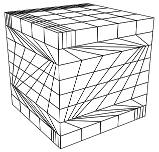
- Add new CDO schemes for scalar transport equations
  - Degrees of freedom at vertices (V4.2) and at cells/vertices (V5.0)
  - Several **diffusion**/convection schemes and boundary enforcement – *Acknowledgment to P. Cantin (PhD)*
- Improve the modularity/integration of CDO schemes (V5.0)
  - New probe/profile mechanism
  - Monitoring (log files, timer stats. . .)



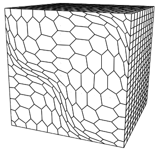
Hexa



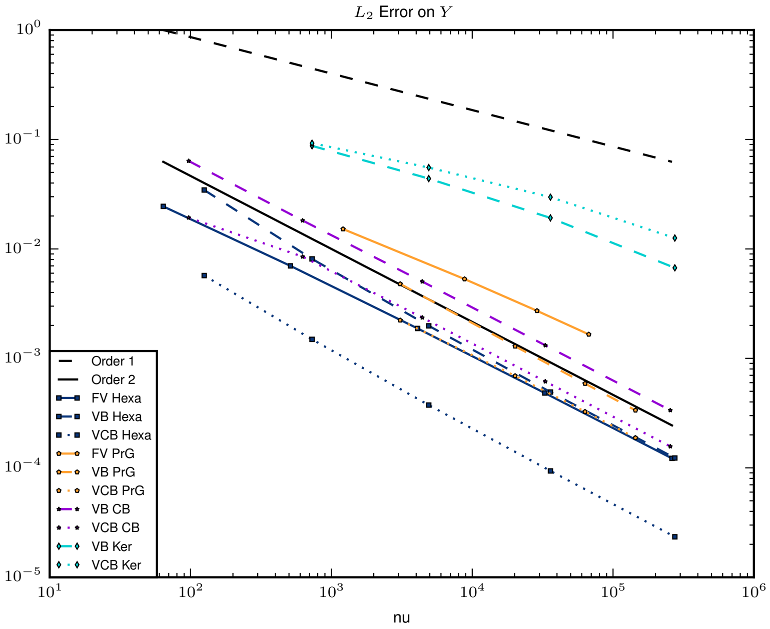
CB



Kershaw

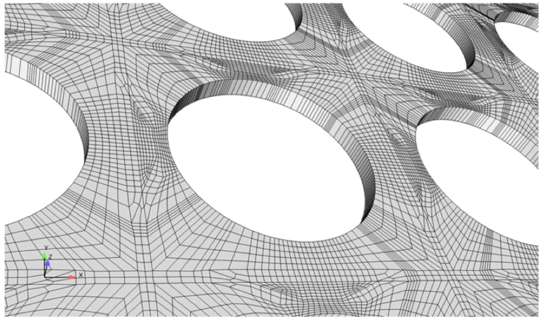


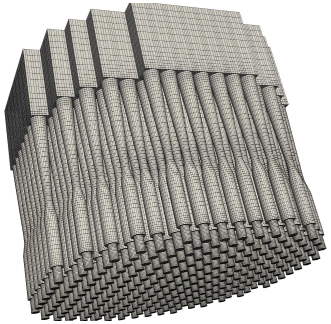
PrG



CDO schemes: Newly available in *Code_Saturne* (V4.2)

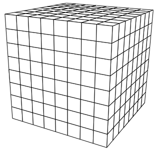
- Design to be robust on polyhedral/distorted meshes
- State-of-the-art discretization schemes mixing FE and FV ideas
- V&V process completed



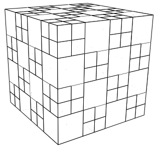


New features

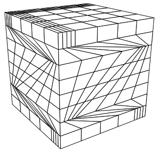
- Add new CDO schemes for scalar transport equations
  - Degrees of freedom at vertices (V4.2) and at cells/vertices (V5.0)
  - Several diffusion/**convection** schemes and boundary enforcement – *Acknowledgment to P. Cantin (PhD)*
- Improve the modularity/integration of CDO schemes (V5.0)
  - New probe/profile mechanism
  - Monitoring (log files, timer stats. . .)



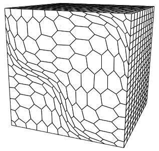
Hexa



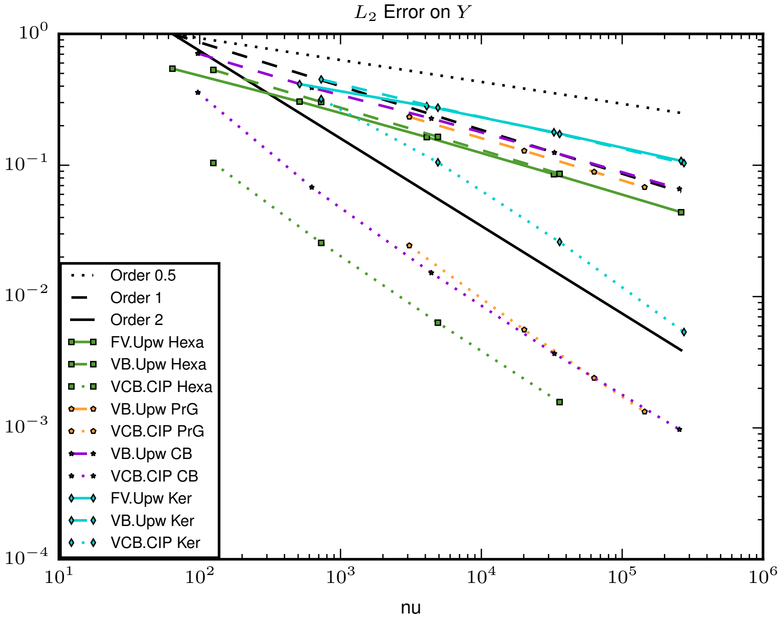
CB



Kershaw

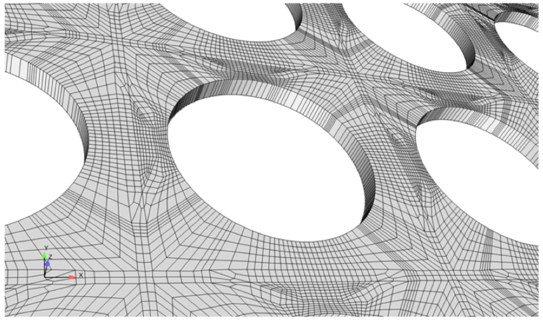


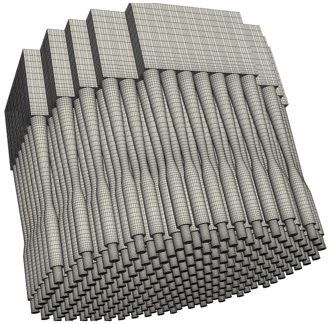
PrG



CDO schemes: Newly available in *Code_Saturne* (V4.2)

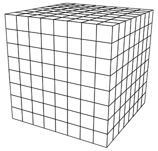
- Design to be robust on polyhedral/distorted meshes
- State-of-the-art discretization schemes mixing FE and FV ideas
- V&V process completed

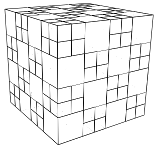


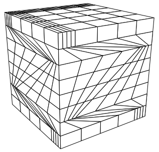


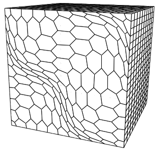
New features

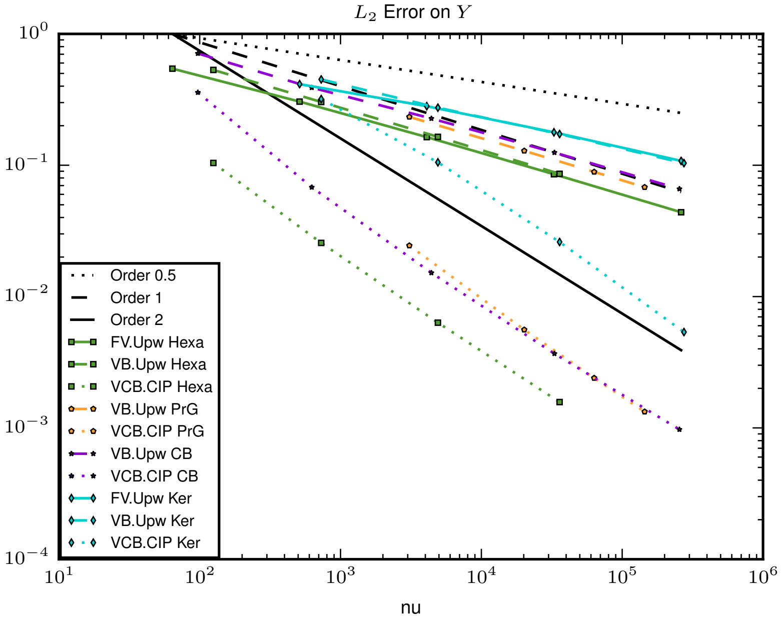
- Add new CDO schemes for scalar transport equations
  - Degrees of freedom at vertices (V4.2) and at cells/vertices (V5.0)
  - Several diffusion/convection schemes and boundary enforcement – *Acknowledgment to P. Cantin (PhD)*
- Improve the modularity/integration of CDO schemes (V5.0)
  - New probe/profile mechanism
  - Monitoring (log files, timer stats. . .)


Hexa


CB

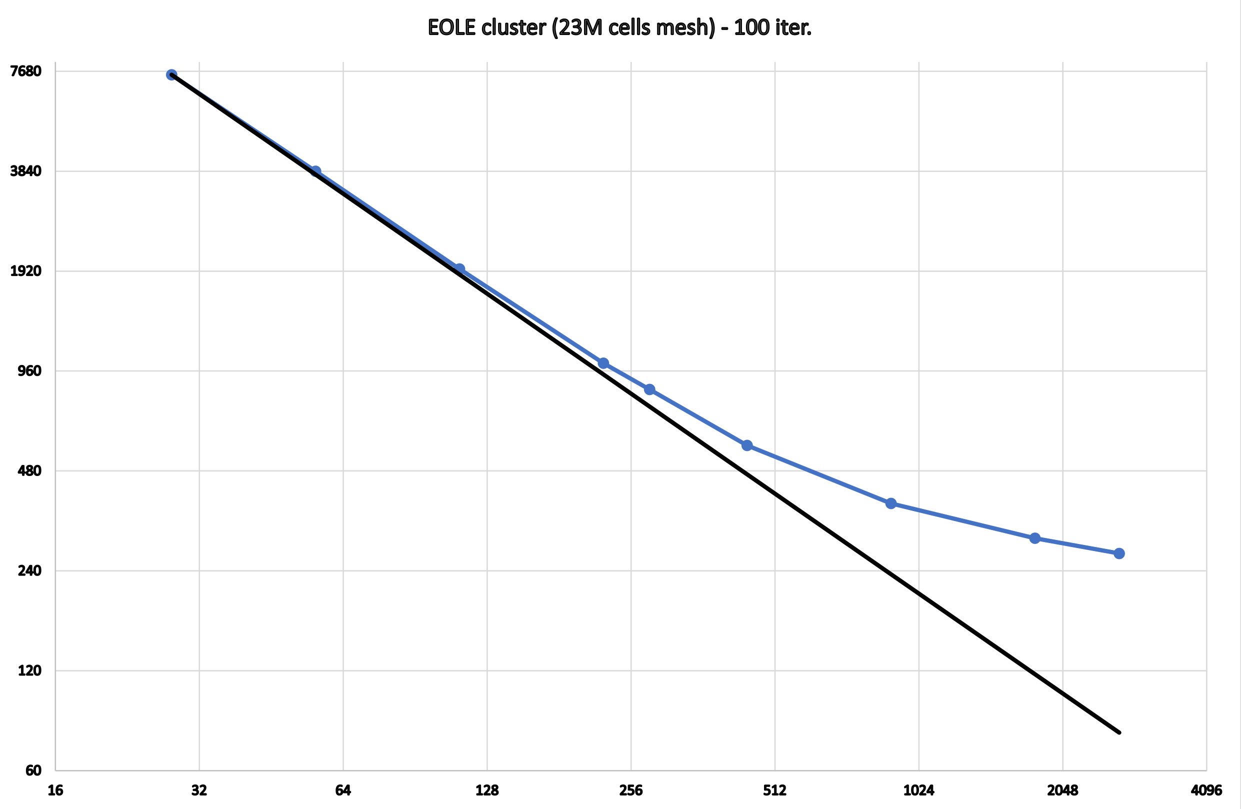

Kershaw


PrG



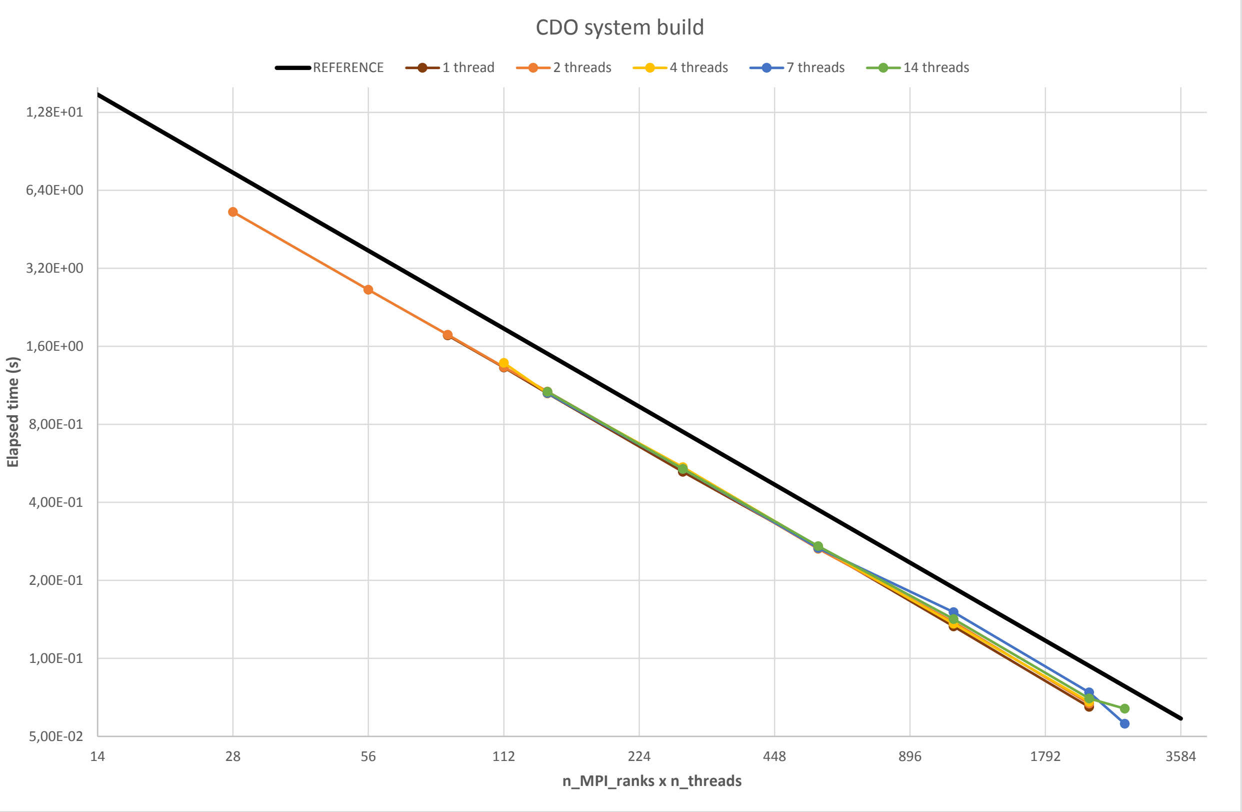
CDO schemes: Performance improvements

1. Two-level parallelism to take a better benefit of modern CPU architecture
   - Distributed memory based on **MPI**
   - Shared memory based on OpenMP
2. More efficient algorithms to build the linear system
   - Cellwise approach being more cache-friendly
   - New **parallel assembler mechanism**



CDO schemes: Performance improvements

1. Two-level parallelism to take a better benefit of modern CPU architecture
   - Distributed memory based on MPI
   - Shared memory based on **OpenMP**
2. More efficient algorithms to build the linear system
   - **Cellwise approach being more cache-friendly**
   - New parallel assembler mechanism

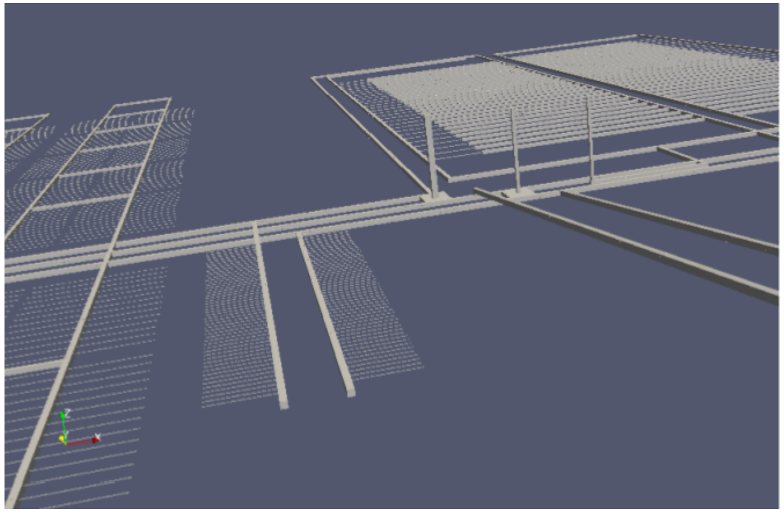


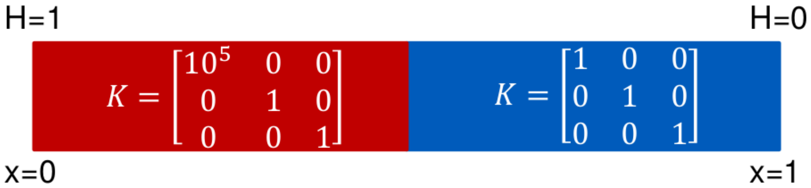
CDO schemes: Groundwater flow module

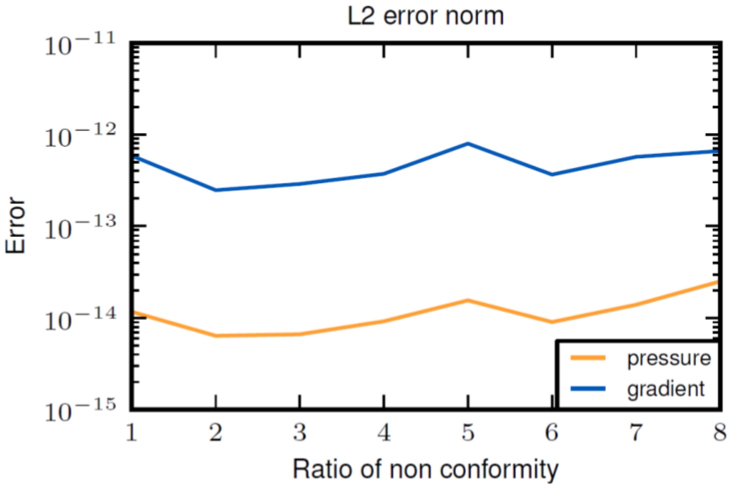
First industrial application

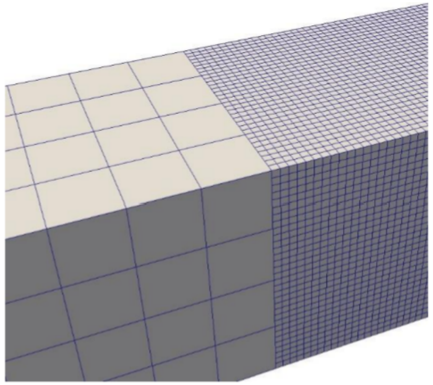
- Solve the Richards equation (time-dependent, highly heterogeneous and anisotropic diffusion equation)
- Induced a velocity field for scalar transport equations

- Complex geometries
- Complex soil structure




H=1
H=0
$K = \begin{bmatrix} 10^5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
$K = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
x=0
x=1



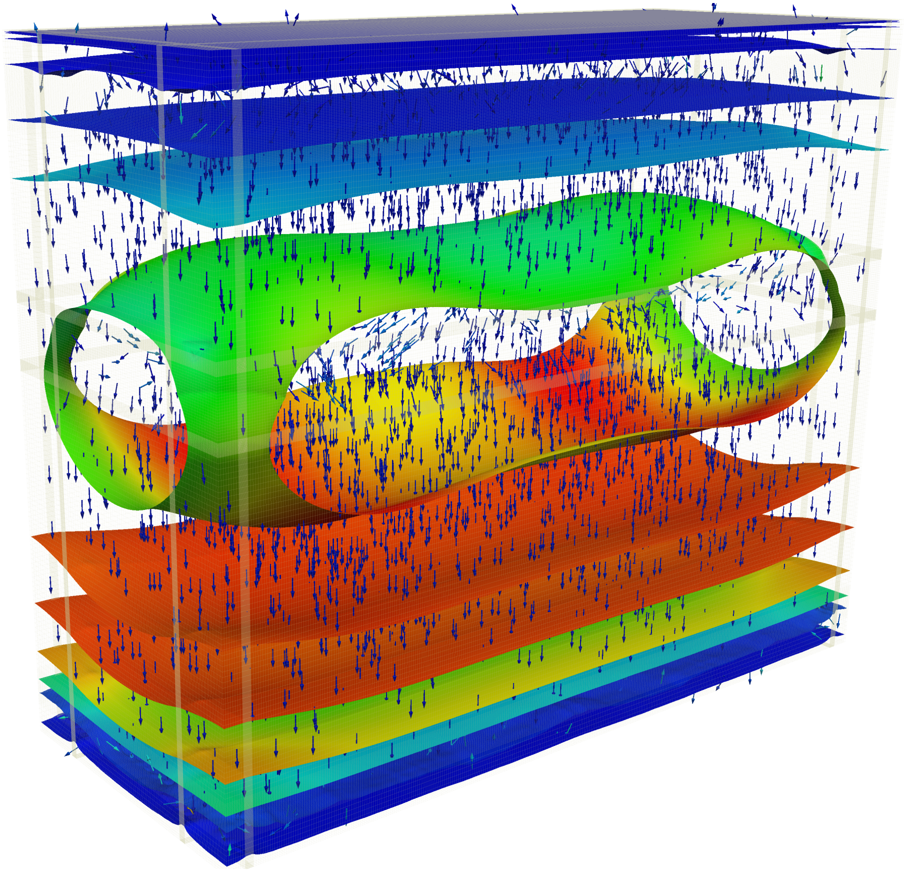


CDO schemes: Groundwater flow module

First industrial application

- Solve the Richards equation (time-dependent, highly heterogeneous and anisotropic diffusion equation)
- Induced a velocity field for scalar transport equations

- Study of the transport of radionuclides in a nuclear waste storage unit
- Simulation run with 23M mesh cells
- Test up to 83M



News for iterative solvers

New solver

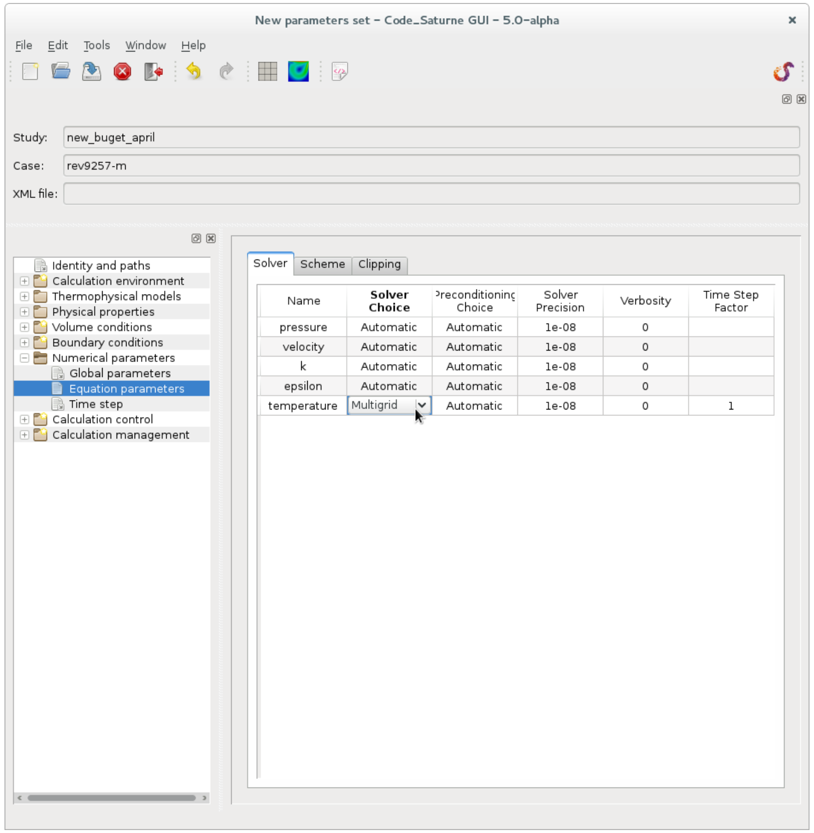
- 3-layer conjugate residuals and parallel block Gauß-Seidel algorithms (V4.1).
- Added support for PETSc in linear solvers (V4.1).
- Add optional multigrid solver for scalars with convection and diffusion, based on the PhD work of Sana Khelifi (V5.0).

Better monitoring

Plot convergence of linear solvers in CSV files (V4.1).

Improve performance

Generalize preconditioning to enable multigrid as preconditioner (V4.2), now the default for all symmetric systems (V4.3); 2x to 4x speed improvement for this stage on average.



Study: new_buget_april

Case: rev9257-m

XML file:

Name	**Solver Choice**	?reconditioning Choice	Solver Precision	Verbosity	Time Step Factor
pressure	Automatic	Automatic	1e-08	0	
velocity	Automatic	Automatic	1e-08	0	
k	Automatic	Automatic	1e-08	0	
epsilon	Automatic	Automatic	1e-08	0	
temperature	Multigrid	Automatic	1e-08	0	1

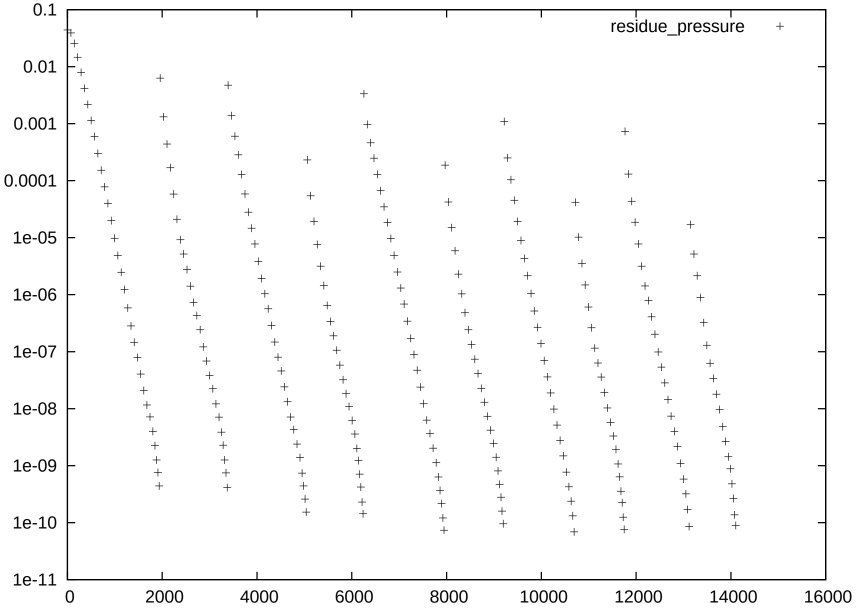
News for iterative solvers

New solver

- 3-layer conjugate residuals and parallel block Gauß-Seidel algorithms (V4.1).
- Added support for PETSc in linear solvers (V4.1).
- Add optional multigrid solver for scalars with convection and diffusion, based on the PhD work of Sana Khelifi (V5.0).

Better monitoring

Plot convergence of linear solvers in CSV files (V4.1).



Improve performance

Generalize preconditioning to enable multigrid as preconditioner (V4.2), now the default for all symmetric systems (V4.3); 2x to 4x speed improvement for this stage on average.

News for iterative solvers

New solver

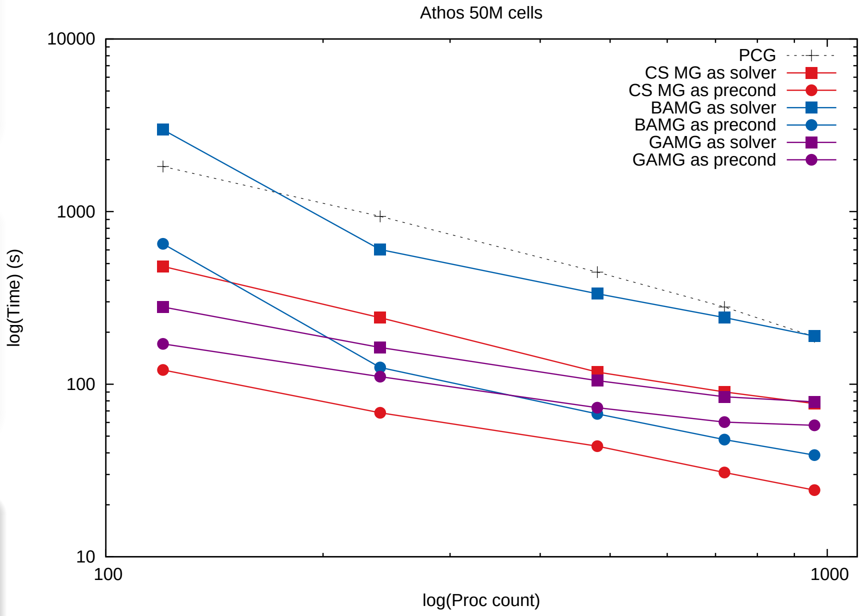
- 3-layer conjugate residuals and parallel block Gauß-Seidel algorithms (V4.1).
- Added support for PETSc in linear solvers (V4.1).
- Add optional multigrid solver for scalars with convection and diffusion, based on the PhD work of Sana Khelifi (V5.0).

Better monitoring

Plot convergence of linear solvers in CSV files (V4.1).

Improve performance

Generalize preconditioning to enable multigrid as preconditioner (V4.2), now the default for all symmetric systems (V4.3); 2x to 4x speed improvement for this stage on average.

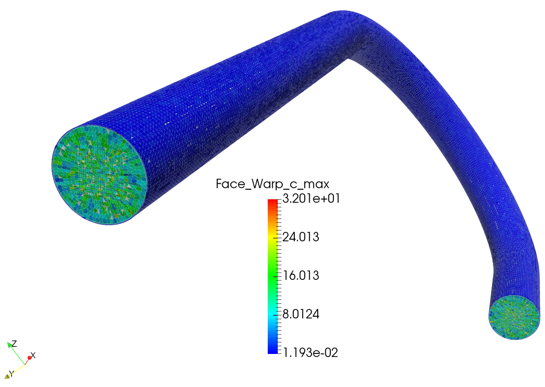


New numerical features

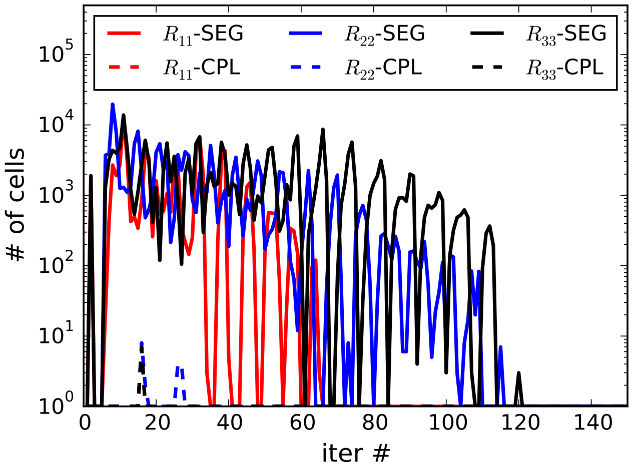
- Improve the robustness of classical FV schemes in case of heterogeneous and anisotropic diffusion (V4.2, V5.0)
- Add a coupled solver for the components of symmetric tensors in Reynolds stress model (`irijco=1`, V4.2) and improve time stepping (V5.0). POSTER
- Add several flux limiters for convective schemes (V4.3 and V5.0):
  - Add an *ad hoc* limiter, which ensures that, for any convective schemes and any time step values, the variable remains between `min_scalar` and `max_scalar` (to be given by the user).
  - Add Roe-Sweby Limiters for all convective schemes

New numerical features

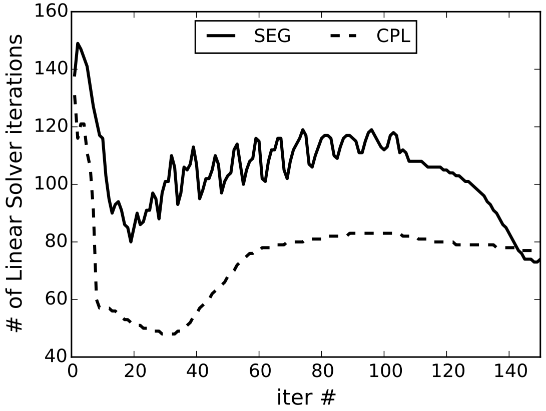
- Improve the robustness of classical FV schemes in case of heterogeneous and anisotropic diffusion (V4.2, V5.0)
- Add a coupled solver for the components of symmetric tensors in Reynolds stress model (`irijco=1`, V4.2) and improve time stepping (V5.0). POSTER
- Add several flux limiters for convective schemes (V4.3 and V5.0):
  - Add an *ad hoc* limiter, which ensures that, for any convective schemes and any time step values, the variable remains between `min_scalar` and `max_scalar` (to be given by the user).
  - Add Roe-Sweby Limiters for all convective schemes



U-Bend configuration



Number of clippings

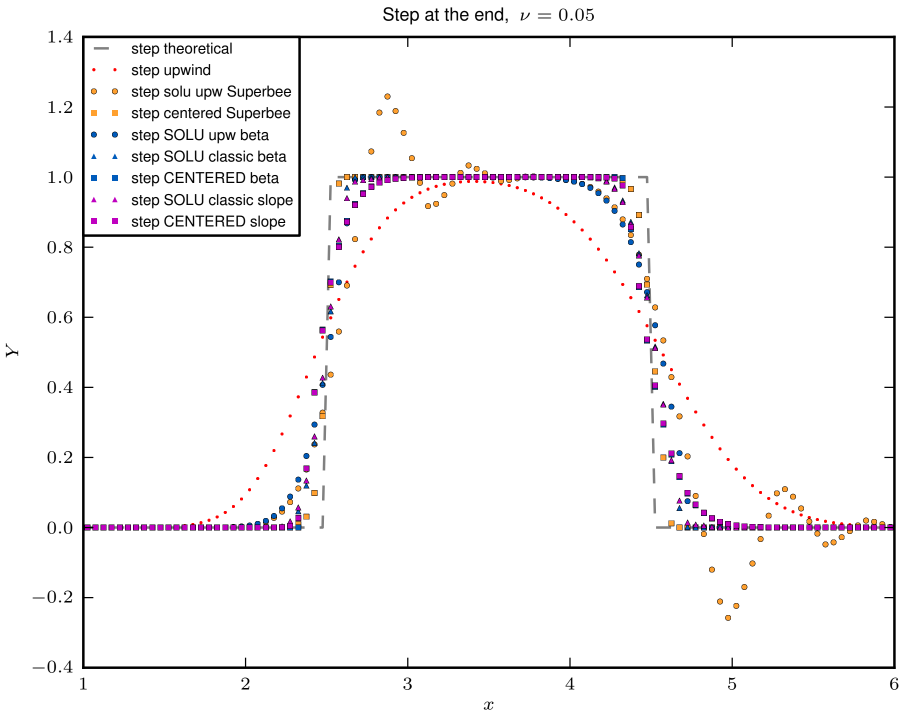


Convergence

(# linear solver iterations)

New numerical features

- Improve the robustness of classical FV schemes in case of heterogeneous and anisotropic diffusion (V4.2, V5.0)
- Add a coupled solver for the components of symmetric tensors in Reynolds stress model (`irijco=1`, V4.2) and improve time stepping (V5.0). POSTER
- Add several flux limiters for convective schemes (V4.3 and V5.0):
  - Add an *ad hoc* limiter, which ensures that, for any convective schemes and any time step values, the variable remains between `min_scalar` and `max_scalar` (to be given by the user).
  - Add Roe-Sweby Limiters for all convective schemes



Overview

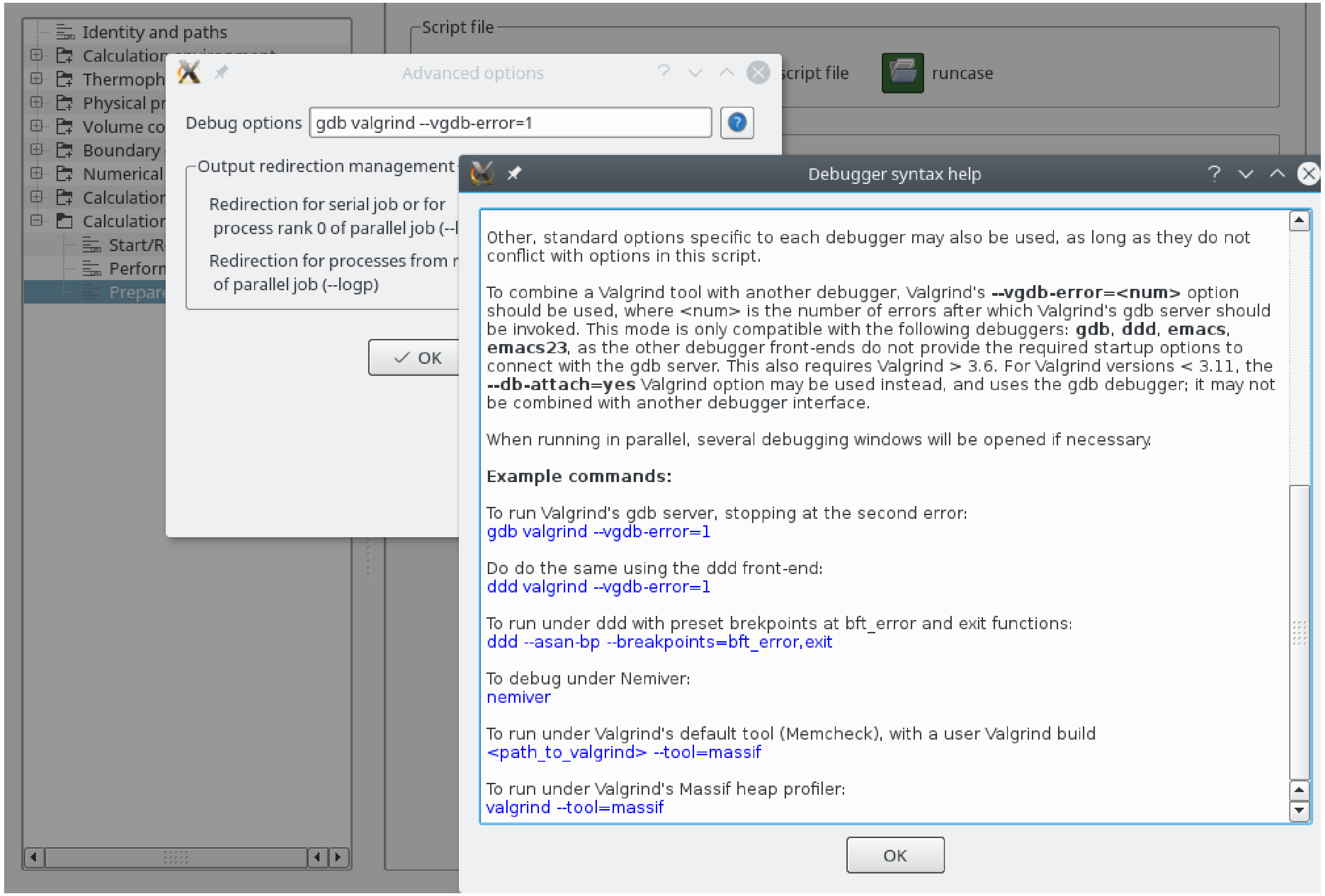
4 Architecture

Debugging help

Add cs_debug_wrapper.py script to simplify running under a debugger (V4.2).

- usable with gdb and Valgrind, allowing additional user interfaces such as ddd, Emacs, KDevelop and Nemiver.
- launch several parallel instances and add breakpoints with a single command, even combined with Valgrind's gdb server.
- not an alternative to TotalView or DDT, but practical for most debugging tasks.
- usable both from the GUI and main user scripts, extending and replaces the previous Valgrind option)
- usable as a standalone script

Example of use of debugger wrapper



Debug options: gdb valgrind --vgdb-error=1

Other, standard options specific to each debugger may also be used, as long as they do not conflict with options in this script.

To combine a Valgrind tool with another debugger, Valgrind's **--vgdb-error=<num>** option should be used, where <num> is the number of errors after which Valgrind's gdb server should be invoked. This mode is only compatible with the following debuggers: **gdb**, **ddd**, **emacs**, **emacs23**, as the other debugger front-ends do not provide the required startup options to connect with the gdb server. This also requires Valgrind > 3.6. For Valgrind versions < 3.11, the **--db-attach=yes** Valgrind option may be used instead, and uses the gdb debugger; it may not be combined with another debugger interface.

When running in parallel, several debugging windows will be opened if necessary.

Example commands:

To run Valgrind's gdb server, stopping at the second error:
gdb valgrind --vgdb-error=1

Do do the same using the ddd front-end:
ddd valgrind --vgdb-error=1

To run under ddd with preset brekpoints at bft_error and exit functions:
ddd --asan-bp --breakpoints=bft_error,exit

To debug under Nemiver:
nemiver

To run under Valgrind's default tool (Memcheck), with a user Valgrind build
<path_to_valgrind> --tool=massif

To run under Valgrind's Massif heap profiler:
valgrind --tool=massif

Architecture changes

Computing Environment

- Builds use OpenMP thread parallelism by default (V4.1)
  - best performance on n cores usually obtained with 2 OpenMP threads per rank and n/2 MPI ranks.
  - even when performance is similar, memory usage is reduced when exanging MPI processes for threads.
- Do not build with BLAS by default, as only MKL is used outside of unit tests, and using it requires providing its path to `--with-blas` anyways.
- Complete Python 3 compatibility (V4.1)
  - minimum Python version: 2.6 (may become 2.7 in the future)
  - builds with SALOME should use the same Python version as SALOME
- PyQt 5 compatible (V4.3)
  - for systems with both PyQt 4 and 5, defining `QT_SELECT=4` or `QT_SELECT=5` at the `configure` stage allows chosing between the two
- did we mention the optional PETSc support already?

Architecture changes

Computing Environment

- Builds use OpenMP thread parallelism by default (V4.1)
  - best performance on n cores usually obtained with 2 OpenMP threads per rank and n/2 MPI ranks.
  - even when performance is similar, memory usage is reduced when exanging MPI processes for threads.
- Do not build with BLAS by default, as only MKL is used outside of unit tests, and using it requires providing its path to `--with-blas` anyways.
- Complete Python 3 compatibility (V4.1)
  - minimum Python version: 2.6 (may become 2.7 in the future)
  - builds with SALOME should use the same Python version as SALOME
- PyQt 5 compatible (V4.3)
  - for systems with both PyQt 4 and 5, defining `QT_SELECT=4` or `QT_SELECT=5` at the `configure` stage allows chosing between the two
- did we mention the optional PETSc support already?

Architecture changes

Computing Environment

- Builds use OpenMP thread parallelism by default (V4.1)
  - best performance on n cores usually obtained with 2 OpenMP threads per rank and n/2 MPI ranks.
  - even when performance is similar, memory usage is reduced when exanging MPI processes for threads.
- Do not build with BLAS by default, as only MKL is used outside of unit tests, and using it requires providing its path to `--with-blas` anyways.
- Complete Python 3 compatibility (V4.1)
  - minimum Python version: 2.6 (may become 2.7 in the future)
  - builds with SALOME should use the same Python version as SALOME
- PyQt 5 compatible (V4.3)
  - for systems with both PyQt 4 and 5, defining `QT_SELECT=4` or `QT_SELECT=5` at the `configure` stage allows chosing between the two
- did we mention the optional PETSc support already?

Architecture changes

Computing Environment

- Builds use OpenMP thread parallelism by default (V4.1)
  - best performance on n cores usually obtained with 2 OpenMP threads per rank and n/2 MPI ranks.
  - even when performance is similar, memory usage is reduced when exanging MPI processes for threads.
- Do not build with BLAS by default, as only MKL is used outside of unit tests, and using it requires providing its path to `--with-blas` anyways.
- Complete Python 3 compatibility (V4.1)
  - minimum Python version: 2.6 (may become 2.7 in the future)
  - builds with SALOME should use the same Python version as SALOME
- PyQt 5 compatible (V4.3)
  - for systems with both PyQt 4 and 5, defining `QT_SELECT=4` or `QT_SELECT=5` at the `configure` stage allows chosing between the two
- did we mention the optional PETSc support already?

Architecture changes

Computing Environment

- Builds use OpenMP thread parallelism by default (V4.1)
  - best performance on n cores usually obtained with 2 OpenMP threads per rank and n/2 MPI ranks.
  - even when performance is similar, memory usage is reduced when exanging MPI processes for threads.
- Do not build with BLAS by default, as only MKL is used outside of unit tests, and using it requires providing its path to `--with-blas` anyways.
- Complete Python 3 compatibility (V4.1)
  - minimum Python version: 2.6 (may become 2.7 in the future)
  - builds with SALOME should use the same Python version as SALOME
- PyQt 5 compatible (V4.3)
  - for systems with both PyQt 4 and 5, defining `QT_SELECT=4` or `QT_SELECT=5` at the `configure` stage allows chosing between the two
- did we mention the optional PETSc support already?

Architecture changes

Parallel algorithms

- Implement handling of point and element tags so as to enable face or cell coupling within a single computation (V4.3).
- Refactor EnSight, MED and CGNS output, replacing serialized slice gathers by parallel block redistribution logic (V4.1).
  - Added parallel MED output when MED and HDF5 libraries are built with parallel IO (V4.3).
- Major changes to `all_to_all` API, so as to make its usage easier (V4.3).
  - new API is similar to `cs_part_to_block` / `cs_block_to_part`, as this may align with sparse or neighborhod collectives in the future.
  - allows instrumentation (timing) and choice of algorithms.
  - only partially deployed at this stage, most refactoring work remains.

Architecture changes

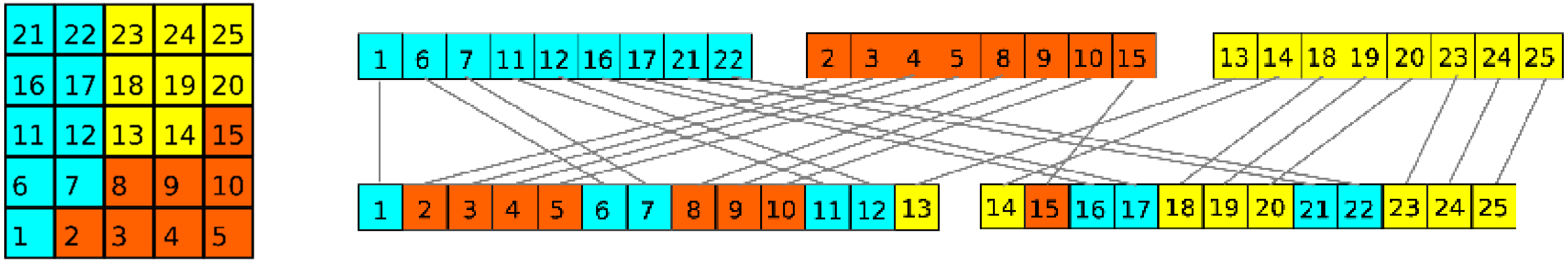
Parallel algorithms

- Implement handling of point and element tags so as to enable face or cell coupling within a single computation (V4.3).
- Refactor EnSight, MED and CGNS output, replacing serialized slice gathers by parallel block redistribution logic (V4.1).
  - Added parallel MED output when MED and HDF5 libraries are built with parallel IO (V4.3).
- Major changes to `all_to_all` API, so as to make its usage easier (V4.3).
  - new API is similar to `cs_part_to_block` / `cs_block_to_part`, as this may align with sparse or neighborhod collectives in the future.
  - allows instrumentation (timing) and choice of algorithms.
  - only partially deployed at this stage, most refactoring work remains.

Architecture changes

Parallel algorithms

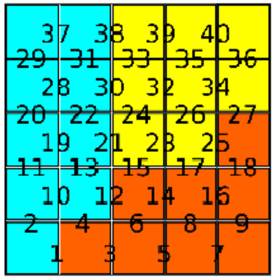
- Implement handling of point and element tags so as to enable face or cell coupling within a single computation (V4.3).
- Refactor EnSight, MED and CGNS output, replacing serialized slice gathers by parallel block redistribution logic (V4.1).
  - Added parallel MED output when MED and HDF5 libraries are built with parallel IO (V4.3).
- Major changes to `all_to_all` API, so as to make its usage easier (V4.3).
  - new API is similar to `cs_part_to_block` / `cs_block_to_part`, as this may align with sparse or neighborhod collectives in the future.
  - allows instrumentation (timing) and choice of algorithms.
  - only partially deployed at this stage, most refactoring work remains.

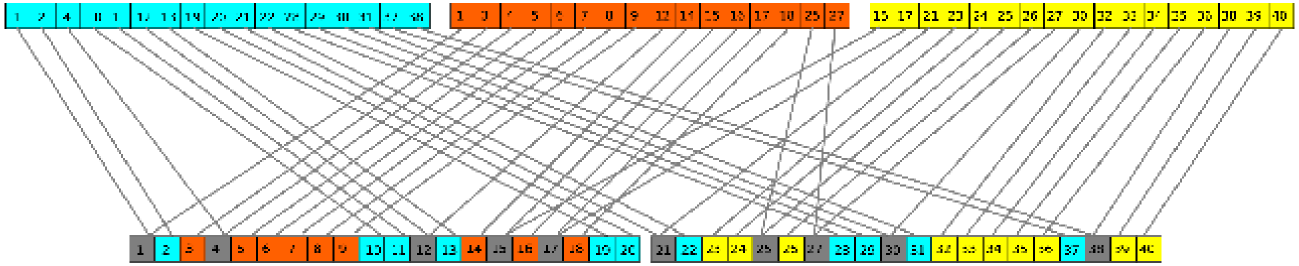


Architecture changes

Parallel algorithms

- Implement handling of point and element tags so as to enable face or cell coupling within a single computation (V4.3).
- Refactor EnSight, MED and CGNS output, replacing serialized slice gathers by parallel block redistribution logic (V4.1).
  - Added parallel MED output when MED and HDF5 libraries are built with parallel IO (V4.3).
- Major changes to `all_to_all` API, so as to make its usage easier (V4.3).
  - new API is similar to `cs_part_to_block` / `cs_block_to_part`, as this may align with sparse or neighborhod collectives in the future.
  - allows instrumentation (timing) and choice of algorithms.
  - only partially deployed at this stage, most refactoring work remains.

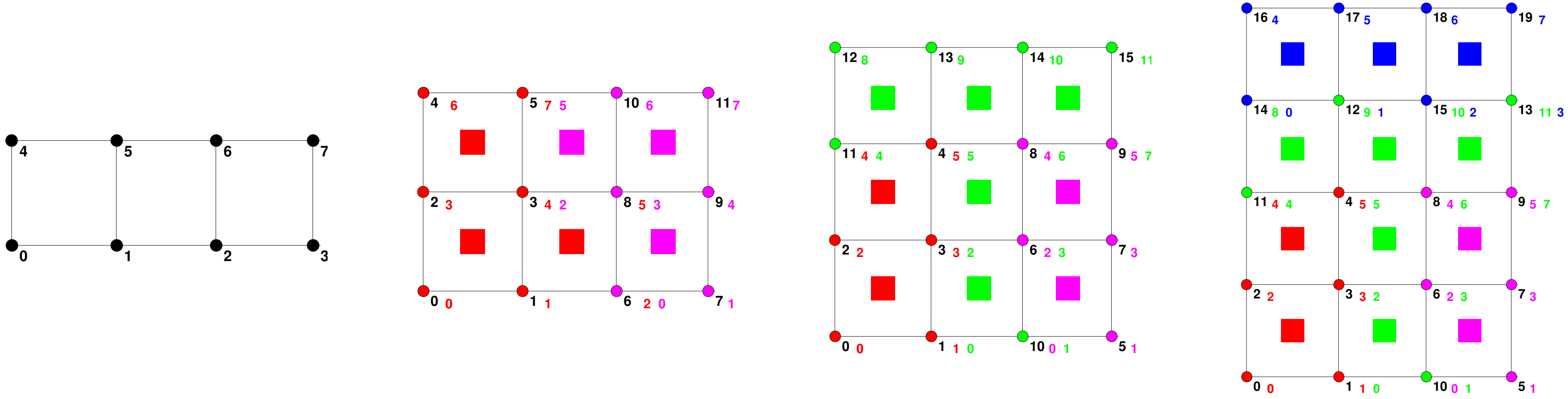




Architecture changes

Parallel linear algebra

- Implement algebraic construction of sparse matrixes based on global row and column ids, using the `cs_matrix_assembler_...(*)` API (V5.0).
  - the associated `cs_range_set_...(*)` API allows handling of an owning rank for distributed entities with some shared elements, such as vertices on parallel boundaries.
  - this is the basis for parallelizing vertex or face-based CDO schemes, while maintaining a compatiility with optional external linear algebra libraries
  - in the future, this may also be used to allow various forms or internal or reinforced couplings
  - unit tests are included



Architecture changes

User interface improvements

- Add `--import-only` option to `code_saturne create` command so as to rebuild SaturneGUI and runcase scripts for a case which was copied from a different system (V4.1).
- Add `--compute-build` option to `code_saturne run` command to allow choosing one of several compute builds at runtime (V4.2).
  - choosing between builds (such as production and debug) using the GUI is now possible (V5.0).
- Add `code_saturne submit` command to submit a batch job. The GUI automatically uses this to prepare data upon submission.
  - run directory is created, data is copied, and user subroutines are compiled before the job is effectively enqueued.
  - if user subroutines don't build, the job is not enqueued (of course, this never happens, as you always check with `code_saturne compile -t` first, don't you?)
  - if you modify your setup before the job starts, the job still starts with the setup as it was at submission time.

Architecture changes

User interface improvements

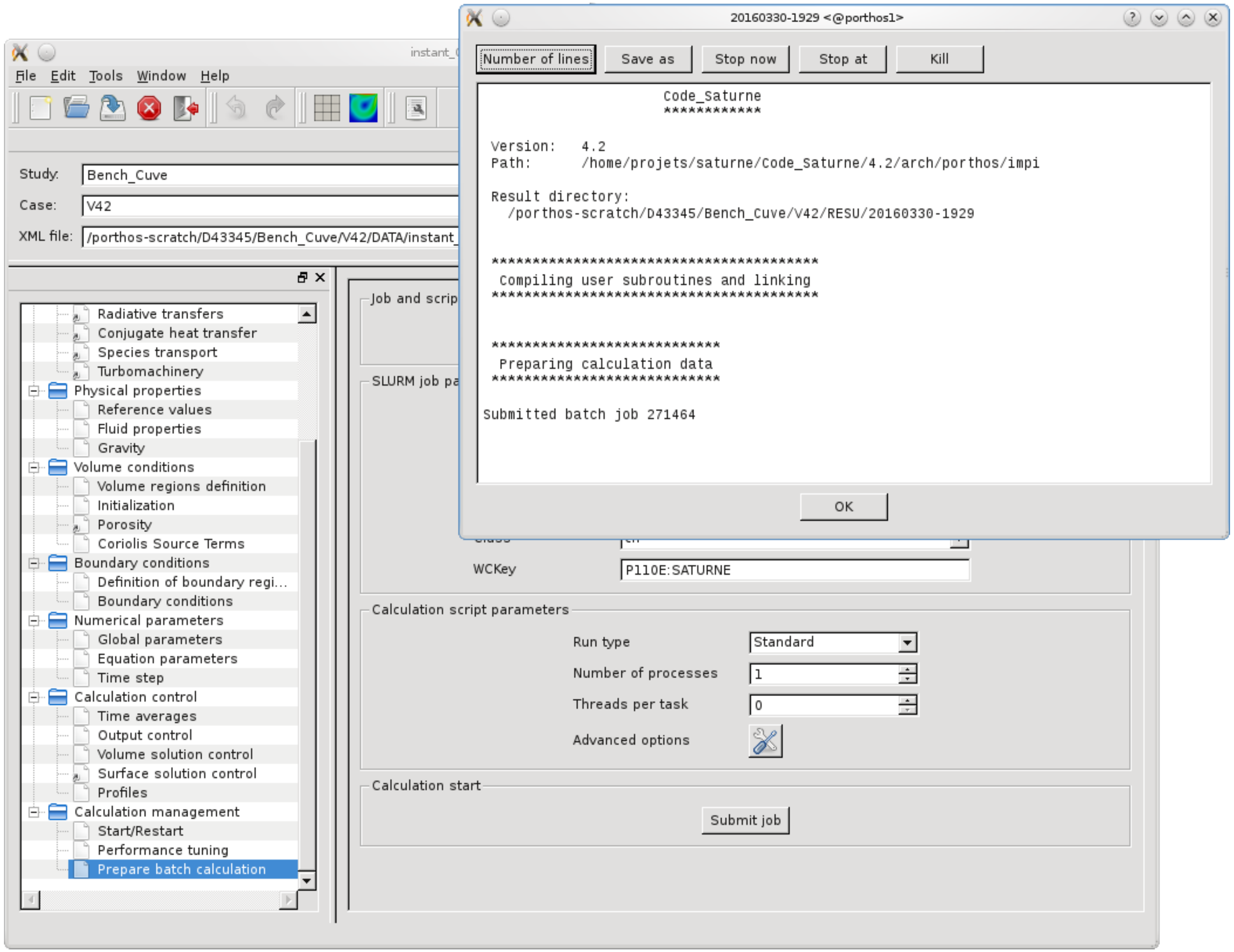
- Add `--import-only` option to `code_saturne create` command so as to rebuild SaturneGUI and runcase scripts for a case which was copied from a different system (V4.1).
- Add `--compute-build` option to `code_saturne run` command to allow choosing one of several compute builds at runtime (V4.2).
  - choosing between builds (such as production and debug) using the GUI is now possible (V5.0).
- Add `code_saturne submit` command to submit a batch job. The GUI automatically uses this to prepare data upon submission.
  - run directory is created, data is copied, and user subroutines are compiled before the job is effectively enqueued.
  - if user subroutines don't build, the job is not enqueued (of course, this never happens, as you always check with `code_saturne compile -t` first, don't you?)
  - if you modify your setup before the job starts, the job still starts with the setup as it was at submission time.

Architecture changes

User interface improvements

- Add `--import-only` option to `code_saturne create` command so as to rebuild SaturneGUI and runcase scripts for a case which was copied from a different system (V4.1).
- Add `--compute-build` option to `code_saturne run` command to allow choosing one of several compute builds at runtime (V4.2).
  - choosing between builds (such as production and debug) using the GUI is now possible (V5.0).
- Add `code_saturne submit` command to submit a batch job. The GUI automatically uses this to prepare data upon submission.
  - run directory is created, data is copied, and user subroutines are compiled before the job is effectively enqueued.
  - if user subroutines don't build, the job is not enqueued
    (of course, this never happens, as you always check with `code_saturne compile -t` first, don't you?)
  - if you modify your setup before the job starts, the job still starts with the setup as it was at submission time.

Example of job submission



Study: Bench_Cuve

Case: V42

XML file: /porthos-scratch/D43345/Bench_Cuve/V42/DATA/instant_

WCKey: P110E:SATURNE

Run type: Standard

Number of processes: 1

Threads per task: 0

```
                    Code_Saturne
                    ************

 Version:   4.2
 Path:      /home/projets/saturne/Code_Saturne/4.2/arch/porthos/impi

 Result directory:
   /porthos-scratch/D43345/Bench_Cuve/V42/RESU/20160330-1929


 ****************************************
  Compiling user subroutines and linking
 ****************************************


 ****************************
  Preparing calculation data
 ****************************

Submitted batch job 271464
```

Architecture changes

C translation

- Removed all the remaining mappings between fields and propce array:
  - removed propce and ipproc arrays and nproce
  - removed iprpfl indirection array between field indices and properties numbers; iprpfl is kept for compatibility but is just an identity function (hence still known in user subroutines);
  - irom, iviscl, etc.. are now directly field indices (hence starting from 0)
  - test on variability of specific heat (icp), isochoric specific heat (icv) and volumetric viscosity (iviscv) are consequently shifted of 1 (-1 : uniform field, $>=$ 0: non uniform)
  - mesh viscosity now a 3-dimensional field when strictly orthotropic
  - removed iroma, ivisla, ivista, icpa variables; previous values of these fields now accessed by field_get_val_prev subroutine.
- Convert Lagrangian and radiative module implementations to C. This affects the associated user subroutines (V4.3).
- Add cs_user_initialization.c, cs_user_physical_properties.c (V4.3).
- Translation of cs_user_extra_op and cs_user_parameters to C in VnV base:
  - removed Fortran arrays corresponding to members of cs_var_cal_opt_t C structure
  - deployed use of get/set var_cal_opt everywhere
  - added cs_post_util functions (R_{ij} post-pro for EVM models on a subset of cells)
  - moved izfppp to C (cs_glob_face_zone) and map Fortran izfppp to it
  - added a cs_user_output function in cs_user_parameters.c (usipes equ.)
  - added many C model structures (turbulence, gas mix, ...)

Architecture changes

C translation

- Removed all the remaining mappings between fields and propce array:
  - removed propce and ipproc arrays and nproce
  - removed iprpfl indirection array between field indices and properties numbers; iprpfl is kept for compatibility but is just an identity function (hence still known in user subroutines);
  - irom, iviscl, etc.. are now directly field indices (hence starting from 0)
  - test on variability of specific heat (icp), isochoric specific heat (icv) and volumetric viscosity (iviscv) are consequently shifted of 1 (-1 : uniform field, $>=$ 0: non uniform)
  - mesh viscosity now a 3-dimensional field when strictly orthotropic
  - removed iroma, ivisla, ivista, icpa variables; previous values of these fields now accessed by

    field_get_val_prev subroutine.
- Convert Lagrangian and radiative module implementations to C. This affects the associated user subroutines (V4.3).
- Add cs_user_initialization.c, cs_user_physical_properties.c (V4.3).
- Translation of cs_user_extra_op and cs_user_parameters to C in VnV base:
  - removed Fortran arrays corresponding to members of cs_var_cal_opt_t C structure
  - deployed use of get/set var_cal_opt everywhere
  - added cs_post_util functions (R_{ij} post-pro for EVM models on a subset of cells)
  - moved izfppp to C (cs_glob_face_zone) and map Fortran izfppp to it
  - added a cs_user_output function in cs_user_parameters.c (usipes equ.)
  - added many C model structures (turbulence, gas mix, ...)

Architecture changes

C translation

- Removed all the remaining mappings between fields and `propce` array:
  - removed `propce` and `ipproc` arrays and `nproce`
  - removed `iprpfl` indirection array between field indices and properties numbers; `iprpfl` is kept for compatibility but is just an identity function (hence still known in user subroutines);
  - `irom`, `iviscl`, etc.. are now directly field indices (hence starting from 0)
  - test on variability of specific heat (`icp`), isochoric specific heat (`icv`) and volumetric viscosity (`iviscv`) are consequently shifted of 1 (-1 : uniform field, $>=$ 0: non uniform)
  - mesh viscosity now a 3-dimensional field when strictly orthotropic
  - removed `iroma`, `ivisla`, `ivista`, `icpa` variables; previous values of these fields now accessed by `field_get_val_prev` subroutine.
- Convert Lagrangian and radiative module implementations to C. This affects the associated user subroutines (V4.3).
- Add `cs_user_initialization.c`, `cs_user_physical_properties.c` (V4.3).
- Translation of `cs_user_extra_op` and `cs_user_parameters` to C in VnV base:
  - removed Fortran arrays corresponding to members of `cs_var_cal_opt_t` C structure
  - deployed use of get/set `var_cal_opt` everywhere
  - added `cs_post_util` functions (R_{ij} post-pro for EVM models on a subset of cells)
  - moved `izfppp` to C (`cs_glob_face_zone`) and map Fortran `izfppp` to it
  - added a `cs_user_output` function in `cs_user_parameters.c` (`usipes` equ.)
  - added many C model structures (turbulence, gas mix, ...)

Architecture changes

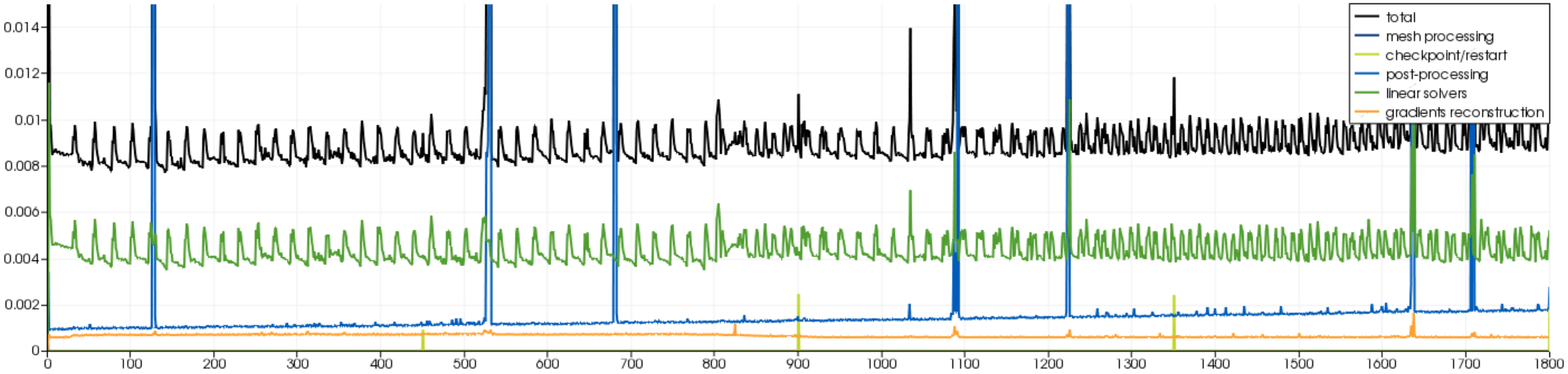
C translation

- Removed all the remaining mappings between fields and propce array:
  - removed propce and ipproc arrays and nproce
  - removed iprpfl indirection array between field indices and properties numbers; iprpfl is kept for compatibility but is just an identity function (hence still known in user subroutines);
  - irom, iviscl, etc.. are now directly field indices (hence starting from 0)
  - test on variability of specific heat (icp), isochoric specific heat (icv) and volumetric viscosity (iviscv) are consequently shifted of 1 (-1 : uniform field, $>=$ 0: non uniform)
  - mesh viscosity now a 3-dimensional field when strictly orthotropic
  - removed iroma, ivisla, ivista, icpa variables; previous values of these fields now accessed by field_get_val_prev subroutine.
- Convert Lagrangian and radiative module implementations to C. This affects the associated user subroutines (V4.3).
- Add cs_user_initialization.c, cs_user_physical_properties.c (V4.3).
- Translation of cs_user_extra_op and cs_user_parameters to C in VnV base:
  - removed Fortran arrays corresponding to members of cs_var_cal_opt_t C structure
  - deployed use of get/set var_cal_opt everywhere
  - added cs_post_util functions (R_{ij} post-pro for EVM models on a subset of cells)
  - moved izfppp to C (cs_glob_face_zone) and map Fortran izfppp to it
  - added a cs_user_output function in cs_user_parameters.c (usipes equ.)
  - added many C model structures (turbulence, gas mix, ...)

Architecture changes

Output

- Add timer statistics and plots for different stages and operators (V4.1).
  - Base timers for each time step (with initialization as time step 0) are now available in the `timer_stats.csv` file (or `timer_stats.dat` if the default format is changed).
  - Final performance data is also moved from `listing` to `performance.log`.



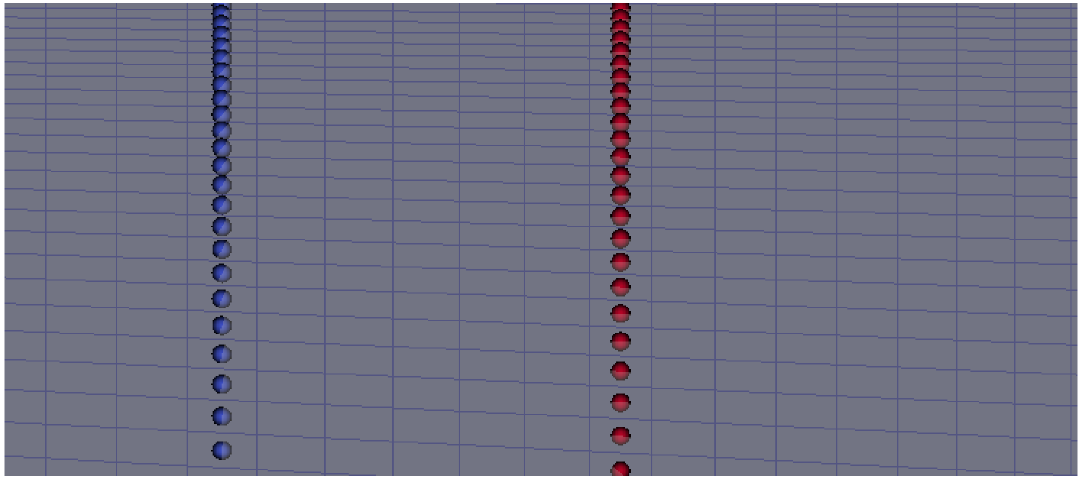
Architecture changes

Output

- Improvements to output update behavior (V4.2)
  - 'listing' and probes output is usually buffered, so it is sometimes difficult to know how a run is making progress.
  - creating an empty `control_file` in the run directory (for example, running "`touch control_file`" in a terminal) forces update of those files at the beginning of the next time step.
- Improvements to postprocessing writer handling (V4.3, V5.0)
  - writers can now use the 'separate_meshes' option to create a separate format-specific writer per mesh.
  - add 'plot' and 'time_plot' writer types
  - probes and user-function defined profiles use this mechanism; GUI-defined profiles still use the old mechanism.
  - profiles along a line segment may use the local mesh resolution rather than uniform user-defined sampling.

Architecture changes (V5.0)

- Add functions for definition of mesh groups during mesh preprocessing.
- Replaced `icond` keyword by `icondb`, `icondv` to allow to enable wall condensation and condensation on internals at the same time.
- Probes output activation is now based on field `post_vis` keyword, and does not allow fine-grained per-variable probes selection anymore (this being little-used, and feasible through use of additional probe sets).
- Added range set structure, to ease operations related to handling of an owning rank for distributed entities.
- Refactored parallel numbering for space-filling curves and added numberings based on a 1D series of real values (used in parallel output of profiles).
- Add configuration options `--with-med=salome`, `--with-hdf5=salome`, and `--with-cgns=salome` to use Salome libraries when `--with-salome=...` is defined



Grand challenge on EOLE EDF cluster: 450 nodes

Billion cell LES to get pressure load on rods

Take home messages

Keep using!

WebSite

Take home messages

Keep feed-backing!

WebSite

BugTracker

Take home messages

Keep coding!

WebSite

BugTracker

Forum

Thank you for your attention.

Any question?