NEW FEATURES FROM *Code_Saturne* V3.0 to V4.0

Code_Saturne development team

April 2, 2015

Overview

Overview

Simplify data setting

Put an end to the name/label madness

- In the GUI, mathematical expressions now use field names instead of labels (this makes things more consistent with user routines, and should avoid many bugs due to name/lable confusion);
- The GUI automatically updates previous setups when opening files;
- Labels are now used only to provide alternate names for logging and postprocessing output.

Simplify data setting

Put an end to the name/label madness

- In the GUI, mathematical expressions now use field names instead of labels (this makes things more consistent with user routines, and should avoid many bugs due to name/lable confusion);
- The GUI automatically updates previous setups when opening files;
- Labels are now used only to provide alternate names for logging and postprocessing output.

Simplify data setting

Put an end to the name/label madness

- In the GUI, mathematical expressions now use field names instead of labels (this makes things more consistent with user routines, and should avoid many bugs due to name/lable confusion);
- The GUI automatically updates previous setups when opening files;
- Labels are now used only to provide alternate names for logging and postprocessing output.

Simplify data setting

Rename and merge user subroutines

- User scalars are not declared in `usinsc` (in `cs_user_parameters.f90`) anymore, but in `cs_user_model` (in `cs_user_parameters.c`). They are not identified by number, but their name is defined by by the user. Using unique names instead of ordinal numbers make it easier to combine data setups.
- Remove the `cs_user_field_parameters` subroutine from `cs_user_parameters.f90`. User code in that subroutine may now be placed in `usipsu` or `usipes`. The `usipsc` subroutine is also removed, and scalar variable diffusivity behaviour may be activated through `usipsu`.

Simplify data setting

Rename and merge user subroutines

- User scalars are not declared in `usinsc` (in `cs_user_parameters.f90`) anymore, but in `cs_user_model` (in `cs_user_parameters.c`). They are not identified by number, but their name is defined by by the user. Using unique names instead of ordinal numbers make it easier to combine data setups.
- Remove the `cs_user_field_parameters` subroutine from `cs_user_parameters.f90`. User code in that subroutine may now be placed in `usipsu` or `usipes`. The `usipsc` subroutine is also removed, and scalar variable diffusivity behaviour may be activated through `usipsu`.

Simplify data setting

Rename and merge user subroutines

- Merge (usebu1, uslwc1, usd3pt1, uscpl1, user_coal_ini1, user_fuel_ini1): in cs_user_combustion (in cs_user_paramters.f90)
- Split and rename usray1 in cs_user_radiative_transfer (in cs_user_parameters.f90). (Note that the declaration of the use of radiative transfer (iirayo = 1) is in usppmo in cs_user_parameters.f90, as the other specific physics models).
- Rename usray2 in cs_user_radiative_transfer_bcs.
- Merge usalin into usipph (in cs_user_parameters.f90)
- Rename ustsma into cs_user_mass_source_terms.
- Rename uskpdc into cs_user_head_losses.

Simplify data setting

Rename and merge user subroutines

- Merge (`usebu1, uslwc1, usd3pt1, uscpl1, user_coal_ini1, user_fuel_ini1`): in `cs_user_combustion` (in `cs_user_paramters.f90`)
- Split and rename `usray1` in `cs_user_radiative_transfer` (in `cs_user_parameters.f90`). (Note that the declaration of the use of radiative transfer (`iirayo = 1`) is in `usppmo` in `cs_user_parameters.f90`, as the other specific physics models).
- Rename `usray2` in `cs_user_radiative_transfer_bcs`.
- Merge `usalin` into `usipph` (in `cs_user_parameters.f90`)
- Rename `ustsma` into `cs_user_mass_source_terms`.
- Rename `uskpdc` into `cs_user_head_losses`.

Simplify data setting

Rename and merge user subroutines

- Merge (usebu1, uslwc1, usd3pt1, uscpl1, user_coal_ini1, user_fuel_ini1): in cs_user_combustion (in cs_user_paramters.f90)
- Split and rename usray1 in cs_user_radiative_transfer (in cs_user_parameters.f90). (Note that the declaration of the use of radiative transfer (iirayo = 1) is in usppmo in cs_user_parameters.f90, as the other specific physics models).
- Rename usray2 in cs_user_radiative_transfer_bcs.
- Merge usalin into usipph (in cs_user_parameters.f90)
- Rename ustsma into cs_user_mass_source_terms.
- Rename uskpdc into cs_user_head_losses.

Simplify data setting

Rename and merge user subroutines

- Merge (`usebu1, uslwc1, usd3pt1, uscpl1, user_coal_ini1, user_fuel_ini1`): in `cs_user_combustion` (in `cs_user_paramters.f90`)
- Split and rename `usray1` in `cs_user_radiative_transfer` (in `cs_user_parameters.f90`). (Note that the declaration of the use of radiative transfer (`iirayo = 1`) is in `usppmo` in `cs_user_parameters.f90`, as the other specific physics models).
- Rename `usray2` in `cs_user_radiative_transfer_bcs`.
- Merge `usalin` into `usipph` (in `cs_user_parameters.f90`)
- Rename `ustsma` into `cs_user_mass_source_terms`.
- Rename `uskpdc` into `cs_user_head_losses`.

Simplify data setting

Rename and merge user subroutines

- Merge (usebu1, uslwc1, usd3pt1, uscpl1, user_coal_ini1, user_fuel_ini1): in cs_user_combustion (in cs_user_paramters.f90)
- Split and rename usray1 in cs_user_radiative_transfer (in cs_user_parameters.f90). (Note that the declaration of the use of radiative transfer (iirayo = 1) is in usppmo in cs_user_parameters.f90, as the other specific physics models).
- Rename usray2 in cs_user_radiative_transfer_bcs.
- Merge usalin into usipph (in cs_user_parameters.f90)
- Rename ustsma into cs_user_mass_source_terms.
- Rename uskpdc into cs_user_head_losses.

Simplify data setting

Rename and merge user subroutines

- Merge (usebu1, uslwc1, usd3pt1, uscpl1, user_coal_ini1, user_fuel_ini1): in cs_user_combustion (in cs_user_paramters.f90)
- Split and rename usray1 in cs_user_radiative_transfer (in cs_user_parameters.f90). (Note that the declaration of the use of radiative transfer (iirayo = 1) is in usppmo in cs_user_parameters.f90, as the other specific physics models).
- Rename usray2 in cs_user_radiative_transfer_bcs.
- Merge usalin into usipph (in cs_user_parameters.f90)
- Rename ustsma into cs_user_mass_source_terms.
- Rename uskpdc into cs_user_head_losses.

Simplify data setting

Temporal moment improvement

- Rewrite of temporal moments handling. Moments handling is now more modular, and allows for vector and tensor fields, as well as variances in addition to means. Also, numerically stable recurrence relations are used to update moments, whose values are now directly usable at any given time. Weight accumulators are now handled inside the module, and not seen as fields anymore. Also, user functions allow evaluating any expression in addition to products of fields. Currently, the GUI only exposes the legacy setup, but the new functionality is available using `cs_user_time_moments` (in `cs_user_parameters.c`) (partial in V3.3, improved in V4.0).

Lising(S) restructuring

(see also the setup.log and performance.log)

listing file

- Information on cell and boundary face based fields

	champ	minimum	maximum	moy. ensemble	moy. spatiale
v	Velocity[X]	0.99827	4.9424	2.1241	2.1241
v	Velocity[Y]	0	0	0	0
v	Velocity[Z]	0	0	0	0
v	||Velocity||	0.99827	4.9424	2.1241	2.1241
v	Pressure	-11.56	0.34631	-2.4015	-2.4015
v	scalar1	2.551e-13	0.016417	0.0012164	0.0012164

- Information on convergence

	Variable	Rhs norm	N_iter	Norm. residual	drift	Time residual
c	Velocity	0.68140E+01	14	0.37066E-09	0.61141E-19	0.99487E-10
c	Velocity[X]				0.61141E-19	
c	Velocity[Y]				0.00000E+00	
c	Velocity[Z]				0.00000E+00	
c	Pressure	0.56078E-09	15	0.92442E-12	0.10619E-08	0.69568E-11
c	scalar1	0.76607E-12	0	0.00000E+00	0.00000E+00	0.00000E+00

Lising(S) restructuring

(see also the setup.log and performance.log)

listing file

- Information on cell and boundary face based fields

	champ	minimum	maximum	moy. ensemble	moy. spatiale
v	Velocity[X]	0.99827	4.9424	2.1241	2.1241
v	Velocity[Y]	0	0	0	0
v	Velocity[Z]	0	0	0	0
v	‖Velocity‖	0.99827	4.9424	2.1241	2.1241
v	Pressure	-11.56	0.34631	-2.4015	-2.4015
v	scalar1	2.551e-13	0.016417	0.0012164	0.0012164

- Information on convergence

	Variable	Rhs norm	N_iter	Norm. residual	drift	Time residual
c	Velocity	0.68140E+01	14	0.37066E-09	0.61141E-19	0.99487E-10
c	Velocity[X]				0.61141E-19	
c	Velocity[Y]				0.00000E+00	
c	Velocity[Z]				0.00000E+00	
c	Pressure	0.56078E-09	15	0.92442E-12	0.10619E-08	0.69568E-11
c	scalar1	0.76607E-12	0	0.00000E+00	0.00000E+00	0.00000E+00

Listing(S) restructuring
(see also the setup.log)

performance.log file

- Information on linear solvers

```
Summary of resolutions for "radiation_003"

  Solver type:                   Block Gauss-Seidel
  Number of setups:                     1
  Number of calls:                      1
  Minimum number of iterations:         2
  Maximum number of iterations:         2
  Mean number of iterations:            2
  Total setup time:                 0.000
  Total solution time:              0.005
```

- Information on convergence

```
Summary of resolutions for "radiation_003"

  Solver type:                   Jacobi
  Number of setups:                     1
  Number of calls:                      1
  Minimum number of iterations:       321
  Maximum number of iterations:       321
  Mean number of iterations:          321
  Total setup time:                 0.000
  Total solution time:              0.125
```

Listing(S) restructuring
(see also the setup.log)

`performance.log` file

- Information on linear solvers

```
Summary of resolutions for "radiation_003"

  Solver type:                   Block Gauss-Seidel
  Number of setups:                       1
  Number of calls:                        1
  Minimum number of iterations:           2
  Maximum number of iterations:           2
  Mean number of iterations:              2
  Total setup time:                   0.000
  Total solution time:                0.005
```

- Information on convergence

```
Summary of resolutions for "radiation_003"

  Solver type:                   Jacobi
  Number of setups:                       1
  Number of calls:                        1
  Minimum number of iterations:         321
  Maximum number of iterations:         321
  Mean number of iterations:            321
  Total setup time:                   0.000
  Total solution time:                0.125
```

Mapped inlet boundary conditions

Mapping an inlet profile to a profile inside the domain or an another boundary

- allows defining pseudo-periodic conditions;
- works transparently in parallel;
- multiple interpolation and normalization options;
- see example in `cs_user_boundary_conditions-mapped_inlet.f90`;
- more general than V3.0's `cs_user_boundary_conditions-auto_inlet.f90`;

Mapped inlet boundary conditions

Mapping an inlet profile to a profile inside the domain or an another boundary

- allows defining pseudo-periodic conditions;
- works transparently in parallel;
- multiple interpolation and normalization options;
- see example in `cs_user_boundary_conditions-mapped_inlet.f90`;
- more general than V3.0's `cs_user_boundary_conditions-auto_inlet.f90`;

Mapped inlet boundary conditions

Mapping an inlet profile to a profile inside the domain or an another boundary

- allows defining pseudo-periodic conditions;
- works transparently in parallel;
- multiple interpolation and normalization options;
- see example in `cs_user_boundary_conditions-mapped_inlet.f90`;
- more general than V3.0's `cs_user_boundary_conditions-auto_inlet.f90`;

Mapped inlet boundary conditions

Mapping an inlet profile to a profile inside the domain or an another boundary

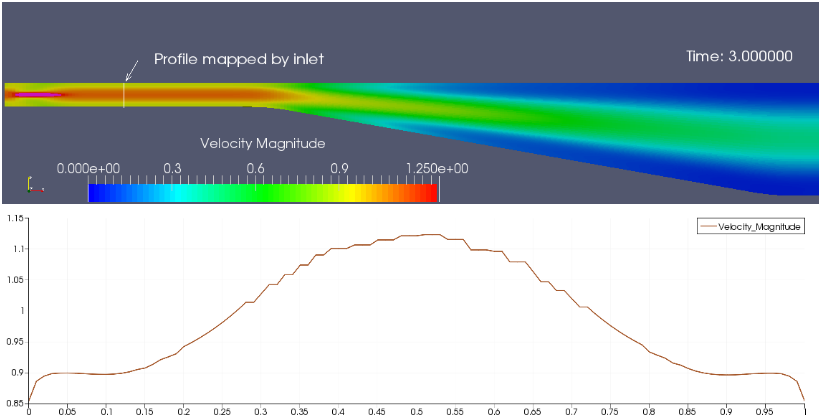
- allows defining pseudo-periodic conditions;
- works transparently in parallel;
- multiple interpolation and normalization options;
- see example in `cs_user_boundary_conditions-mapped_inlet.f90`;
- more general than V3.0's `cs_user_boundary_conditions-auto_inlet.f90`;

Mapped inlet boundary conditions

Mapping an inlet profile to a profile inside the domain or an another boundary

- allows defining pseudo-periodic conditions;
- works transparently in parallel;
- multiple interpolation and normalization options;
- see example in `cs_user_boundary_conditions-mapped_inlet.f90`;
- more general than V3.0's `cs_user_boundary_conditions-auto_inlet.f90`;

Mapped inlet boundary conditions



Architecture changes

- Remove `rtp` and `rtpa` arrays. Variables are defined and accessed using the field structures (V4.0).
- Lagrangian particle data is now shared between Fortran and C parts. Fortran arrays have been replaced by pointers, which map to the C data. arrays `itepa`, `tepa`, `ettp`, and `ettpa` are replaced respectively by pointers `ipepa`, `pepa`, `eptp`, `eptpa`, which use interleaved data (V4.0);
  - Allocation and resizing is automatic, so the maximum number of particles does not need to be predefined.
- Restart files now use a new section naming field, at least for field data. this allows more automated handling of variables and properties in checkpoint/restart (V4.0).
- For non-batch systems, handling of the number of MPI ranks is based on a `code_saturne run` option, `--nprocs`, and is set in the runcase file, not in the XML file anymore (V3.3).
- Hybrid parallelism using MPI + OpenMP (disabled by default, enabled in future versions).

Architecture changes

- Remove `rtp` and `rtpa` arrays. Variables are defined and accessed using the field structures (V4.0).
- Lagrangian particle data is now shared between Fortran and C parts. Fortran arrays have been replaced by pointers, which map to the C data. arrays `itepa`, `tepa`, `ettp`, and `ettpa` are replaced respectively by pointers `ipepa`, `pepa`, `eptp`, `eptpa`, which use interleaved data (V4.0);
  - Allocation and resizing is automatic, so the maximum number of particles does not need to be predefined.
- Restart files now use a new section naming field, at least for field data. this allows more automated handling of variables and properties in checkpoint/restart (V4.0).
- For non-batch systems, handling of the number of MPI ranks is based on a `code_saturne run` option, `--nprocs`, and is set in the runcase file, not in the XML file anymore (V3.3).
- Hybrid parallelism using MPI + OpenMP (disabled by default, enabled in future versions).

Architecture changes

- Remove rtp and rtpa arrays. Variables are defined and accessed using the field structures (V4.0).
- Lagrangian particle data is now shared between Fortran and C parts. Fortran arrays have been replaced by pointers, which map to the C data. arrays itepa, tepa, ettp, and ettpa are replaced respectively by pointers ipepa, pepa, eptp, eptpa, which use interleaved data (V4.0);
  - Allocation and resizing is automatic, so the maximum number of particles does not need to be predefined.
- Restart files now use a new section naming field, at least for field data. this allows more automated handling of variables and properties in checkpoint/restart (V4.0).
- For non-batch systems, handling of the number of MPI ranks is based on a code_saturne run option, --nprocs, and is set in the runcase file, not in the XML file anymore (V3.3).
- Hybrid parallelism using MPI + OpenMP (disabled by default, enabled in future versions).

Architecture changes

- Remove rtp and rtpa arrays. Variables are defined and accessed using the field structures (V4.0).
- Lagrangian particle data is now shared between Fortran and C parts. Fortran arrays have been replaced by pointers, which map to the C data. arrays itepa, tepa, ettp, and ettpa are replaced respectively by pointers ipepa, pepa, eptp, eptpa, which use interleaved data (V4.0);
  - Allocation and resizing is automatic, so the maximum number of particles does not need to be predefined.
- Restart files now use a new section naming field, at least for field data. this allows more automated handling of variables and properties in checkpoint/restart (V4.0).
- For non-batch systems, handling of the number of MPI ranks is based on a `code_saturne run` option, `--nprocs`, and is set in the runcase file, not in the XML file anymore (V3.3).
- Hybrid parallelism using MPI + OpenMP (disabled by default, enabled in future versions).

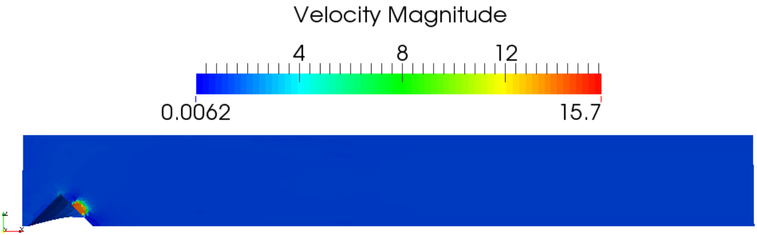
Architecture changes

- Remove rtp and rtpa arrays. Variables are defined and accessed using the field structures (V4.0).
- Lagrangian particle data is now shared between Fortran and C parts. Fortran arrays have been replaced by pointers, which map to the C data. arrays itepa, tepa, ettp, and ettpa are replaced respectively by pointers ipepa, pepa, eptp, eptpa, which use interleaved data (V4.0);
  - Allocation and resizing is automatic, so the maximum number of particles does not need to be predefined.
- Restart files now use a new section naming field, at least for field data. this allows more automated handling of variables and properties in checkpoint/restart (V4.0).
- For non-batch systems, handling of the number of MPI ranks is based on a `code_saturne run` option, `--nprocs`, and is set in the runcase file, not in the XML file anymore (V3.3).
- Hybrid parallelism using MPI + OpenMP (disabled by default, enabled in future versions).

Extra instrumentation

- Add a `slope_test_upwind_id` field keyword (see `cs_user_parameters-output.f90`), allowing post-processing output of the contribution of slope tests to convected variables. Visualize:

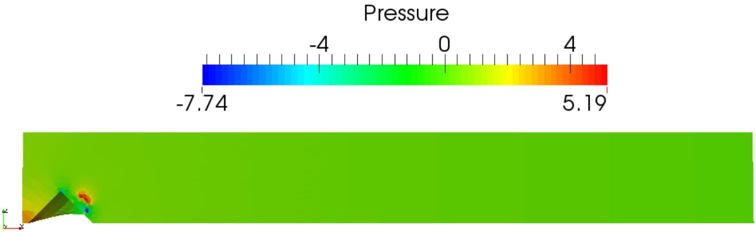
$$\texttt{indicator}_i = \sum_{f \in \mathcal{F}_i} |\dot{m}_f| \, \texttt{Is the slope test activated?}$$



Extra instrumentation

- Add a `slope_test_upwind_id` field keyword (see `cs_user_parameters-output.f90`), allowing post-processing output of the contribution of slope tests to convected variables. Visualize:

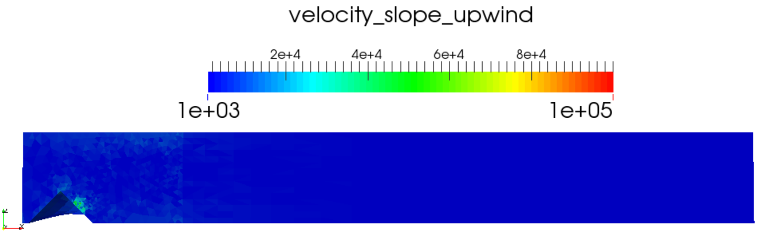
$$\texttt{indicator}_i = \sum_{f \in \mathcal{F}_i} |\dot{m}_f| \, \texttt{Is the slope test activated?}$$



Extra instrumentation

- Add a `slope_test_upwind_id` field keyword (see `cs_user_parameters-output.f90`), allowing post-processing output of the contribution of slope tests to convected variables. Visualize:

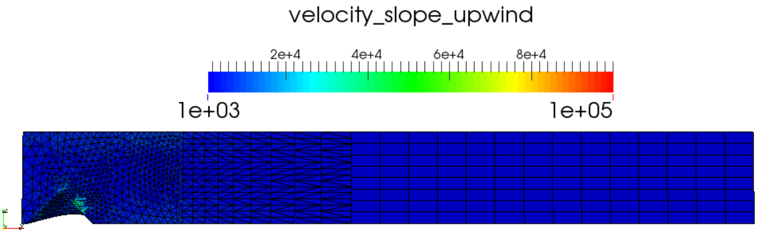
$$\texttt{indicator}_i = \sum_{f \in \mathcal{F}_i} |\dot{m}_f| \, \texttt{Is the slope test activated?}$$



Extra instrumentation

- Add a `slope_test_upwind_id` field keyword (see `cs_user_parameters-output.f90`), allowing post-processing output of the contribution of slope tests to convected variables. Visualize:

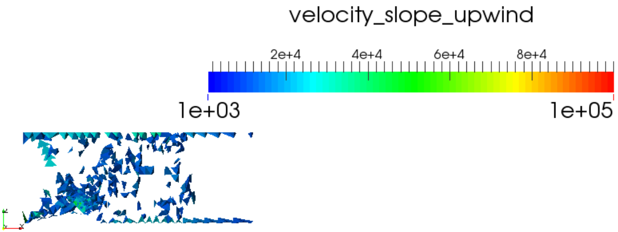
$$\texttt{indicator}_i = \sum_{f \in \mathcal{F}_i} \left| \dot{m}_f \right| \texttt{Is the slope test activated?}$$



Extra instrumentation

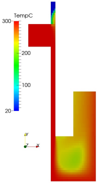
- Add a `slope_test_upwind_id` field keyword (see `cs_user_parameters-output.f90`), allowing post-processing output of the contribution of slope tests to convected variables. Visualize:

$$\texttt{indicator}_i = \sum_{f \in \mathcal{F}_i} |\dot{m}_f| \, \texttt{Is the slope test activated?}$$



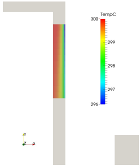
Automatic balance by zone: an example with the energy

Balance of "`temperature`" on "`box[−0.5, 1.3, 0, 1, 1.9, 1]`".



Whole domain

```
  ** SCALAR BALANCE BY ZONE at iteration     121
  ---------------------------------------------
------------------------------------------------------------
  SCALAR: temperature
  ZONE SELECTION CRITERIA: "box[-0.5, 1.3, 0., 1., 1.9, 1.]"
------------------------------------------------------------
  Unst. term    Inj. Mass.    Suc. Mass.
  -1.0308e+03    0.0000e+00    0.0000e+00
------------------------------------------------------------
  IB inlet      IB outlet
   2.9729e+07   -2.9728e+07
------------------------------------------------------------
  Inlet         Outlet
   0.0000e+00    0.0000e+00
------------------------------------------------------------
  Sym.          Smooth W.     Rough W.
   0.0000e+00    0.0000e+00    0.0000e+00
------------------------------------------------------------
  Coupled       Undef. BC
   0.0000e+00    0.0000e+00
------------------------------------------------------------
  Total         Instant. norm. total
  -5.5312e+01   -3.7962e-06
------------------------------------------------------------
```



Zoom on the selected zone

An example of turbo-machinery and Lagrangian modelling

New programming features
(see the setup.log)

Field structure (for facilitating access to quantities)

- V2.0:

```
 call grdcel                                                &
 !==========
( ifinia , ifinra ,                                         &
  ndim   , ncelet , ncel   , nfac   , nfabor , nfml   , nprfml , &
  nnod   , lndfac , lndfbr , ncelbr , nphas  ,               &
  nideve , nrdeve , nituse , nrtuse ,                        &
  ivar   , imrgra , inc    , iccocg , nswrgp , imligp , iphydp , &
  iwarnp , nfecra ,                                          &
  epsrgp , climgp , extrap ,                                 &
  ifacel , ifabor , ifmfbr , ifmcel , iprfml ,               &
  ipnfac , nodfac , ipnfbr , nodfbr ,                        &
  idevel , ituser , ia     ,                                 &
  xyzcen , surfac , surfbo , cdgfac , cdgfbo , xyznod , volume , &
  ra(itravx) , ra(itravx) , ra(itravx) ,                     &
  rtp(1,ivar) , coefa(1,iclvar) , coefb(1,iclvar) ,          &
  ra(igradx) , ra(igrady) , ra(igradz) ,                     &
!  ----------   ----------   ----------
  ra(itravx) , ra(itravy) , ra(itravz) ,                     &
  rdevel , rtuser , ra     )
```

- is now in V4.0

```
call field_gradient_scalar(ivarfl(ivar), iprev, imrgra, inc, iccocg, &
                           grad)
```

New programming features
(see the setup.log)

Field structure (for facilitating access to quantities)

- V2.0:

```
 call grdcel                                                           &
 !==========
( ifinia , ifinra ,                                                    &
  ndim   , ncelet , ncel   , nfac   , nfabor , nfml   , nprfml ,       &
  nnod   , lndfac , lndfbr , ncelbr , nphas  ,                         &
  nideve , nrdeve , nituse , nrtuse ,                                  &
  ivar   , imrgra , inc    , iccocg , nswrgp , imligp , iphydp ,       &
  iwarnp , nfecra ,                                                    &
  epsrgp , climgp , extrap ,                                           &
  ifacel , ifabor , ifmfbr , ifmcel , iprfml ,                         &
  ipnfac , nodfac , ipnfbr , nodfbr ,                                  &
  idevel , ituser , ia     ,                                           &
  xyzcen , surfac , surfbo , cdgfac , cdgfbo , xyznod , volume ,       &
  ra(itravx) , ra(itravx) , ra(itravx) ,                               &
  rtp(1,ivar) , coefa(1,iclvar) , coefb(1,iclvar) ,                    &
  ra(igradx) , ra(igrady) , ra(igradz) ,                               &
! ----------   ----------   ----------
  ra(itravx) , ra(itravy) , ra(itravz) ,                               &
  rdevel , rtuser , ra     )
```

- is now in V4.0

```
call field_gradient_scalar(ivarfl(ivar), iprev, imrgra, inc, iccocg, &
                           grad)
```

New programming features

Updated slides of the *Code_Saturne* Training days are provided.

Catalyst co-processing

Configuration

1. Install ParaView 4.2 or 4.3 with Catalyst support (prefer OSMesa[a]),
2. Install *Code_Saturne* with configure option `--with-catalyst=/PATH/`
3. Load CatalystGeneratorPlugin (Tools → Manage Plugins) in ParaView

You may have to load some PYTHONPATH (use a `saturne_rc` file which is loaded automatically if set in `CSINSTALL/etc/code_saturne.cfg`):

```
export PYTHONPATH=/.../usr/arch/calibre7/lib/python2.6/site-packages:$PYTHONPATH
export PYTHONPATH=/.../lib/paraview-4.3/site-packages/vtk:$PYTHONPATH
export PYTHONPATH=/.../lib/paraview-4.3/site-packages/paraview:$PYTHONPATH
export PYTHONPATH=/.../lib/paraview-4.3/site-packages:$PYTHONPATH

export LD_LIBRARY_PATH=/.../lib/paraview-4.3:$LD_LIBRARY_PATH
```

[a] Off screen rendering, contact `saturne-support@edf.fr`

Catalyst co-processing

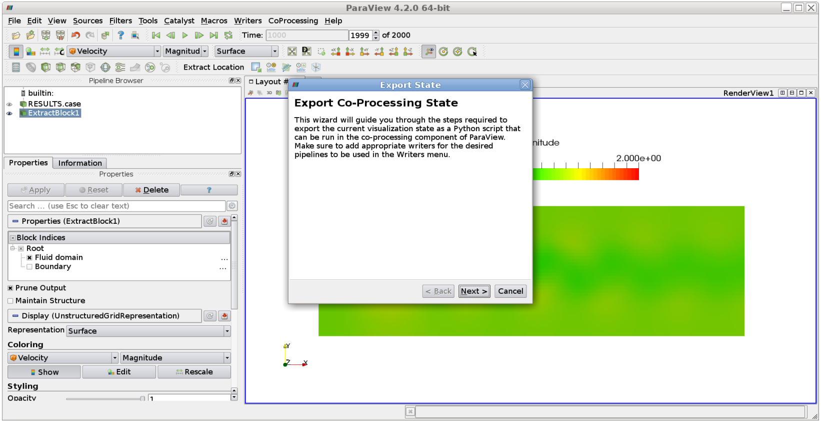
Installation of ParaView

```
H81256@dsp0698225: ~/Code_Saturne/opt/ParaView-v4.2.build
Fichier  Édition  Affichage  Terminal  Aide
                                          Page 1 of 1
 BUILD_DOCUMENTATION              OFF
 BUILD_EXAMPLES                   OFF
 BUILD_SHARED_LIBS                ON
 BUILD_TESTING                    OFF
 CMAKE_BUILD_TYPE                 RelWithDebInfo
 CMAKE_INSTALL_PREFIX             /usr/local
 GMVReader_GMVREAD_LIB_DIR        /home/H81256/Code_Saturne/opt/ParaView-v4.2.0-source/Utilities/VisItBridge/databases/GMV
 GMVReader_SKIP_DATARANGE_CALCU   OFF
 MPI_C_INCLUDE_PATH               /usr/lib/openmpi/include;/usr/lib/openmpi/include/openmpi
 MPI_C_LIBRARIES                  /usr/lib/openmpi/lib/libmpi.so;/usr/lib/openmpi/lib/libopen-rte.so;/usr/lib/openmpi/lib/libopen-pal.so;/usr/lib/libdl.so;/u
 PARAVIEW_BUILD_QT_GUI            ON
 PARAVIEW_ENABLE_CATALYST         ON
 PARAVIEW_ENABLE_FFMPEG           OFF
 PARAVIEW_ENABLE_PYTHON           ON
 PARAVIEW_INSTALL_DEVELOPMENT_F   ON
 PARAVIEW_USE_MPI                 ON
 PARAVIEW_USE_PISTON              OFF
 PARAVIEW_USE_UNIFIED_BINDINGS    OFF
 PARAVIEW_USE_VISITBRIDGE         OFF
 SURFACELIC_PLUGIN_TESTING        ON
 VTK_RENDERING_BACKEND            OpenGL
 VTK_SMP_IMPLEMENTATION_TYPE      Sequential
 VTK_USE_LARGE_DATA               OFF
 XDMF_USE_BZIP2                   OFF
 XDMF_USE_GZIP                    OFF

BUILD_DOCUMENTATION: Build the VTK documentation
Press [enter] to edit option                                                                                   CMake Version 3.1.0
Press [c] to configure
Press [h] for help           Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)
```

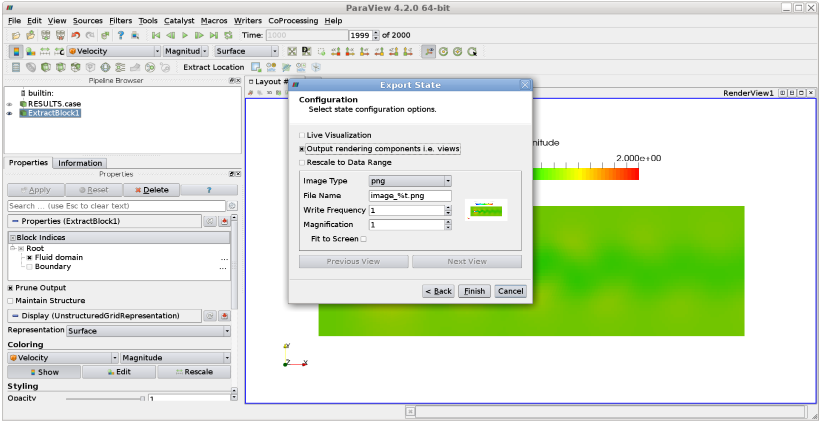
Catalyst co-processing

Configuration of ParaView state



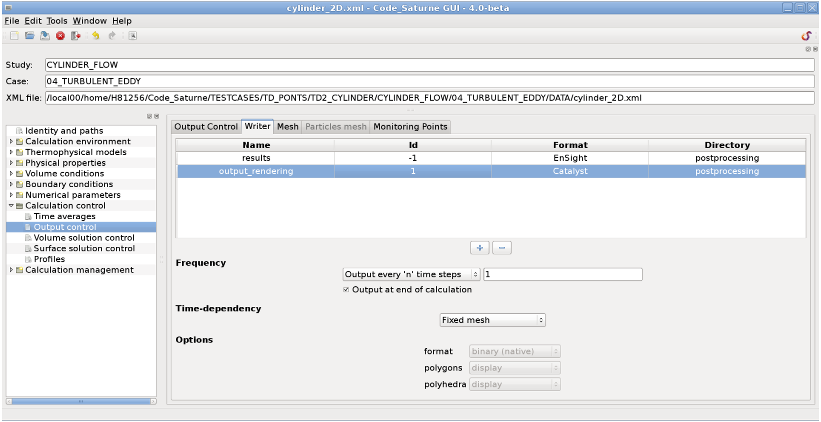
Catalyst co-processing

Configuration of ParaView state: output rendering or live visualization



Catalyst co-processing

Add a *Code_Saturne* `Writer` with the same name as the ParaView `state.py`



New numerical features

- Generalize double backward implicit Euler time scheme for all variables. It can be activated with the keyword `ibdtso(ivar) = 1` (V4.0). Second order backward Euler scheme in time for velocity prediction since V3.3.
- Turbo-machinery modelling: enable multiple rotors for rotor-stator model based on mesh joining (V4.0). Turbo-machinery modelling: added a rotor-stator model based on mesh joining (V3.2).
- By default, do not force use of an iterative gradient reconstruction method for pressure gradients, or other gradients deriving from a potential. To force it, a negative value of the `imrgra` (-1, -2, -3) keyword may be used (V4.0).
- Add cell and face renumbering options to improve performance (V4.0).

New numerical features

- Generalize double backward implicit Euler time scheme for all variables. It can be activated with the keyword `ibdtso(ivar) = 1` (V4.0). Second order backward Euler scheme in time for velocity prediction since V3.3.
- Turbo-machinery modelling: enable multiple rotors for rotor-stator model based on mesh joining (V4.0).
  Turbo-machinery modelling: added a rotor-stator model based on mesh joining (V3.2).
- By default, do not force use of an iterative gradient reconstruction method for pressure gradients, or other gradients deriving from a potential. To force it, a negative value of the `imrgra` (-1, -2, -3) keyword may be used (V4.0).
- Add cell and face renumbering options to improve performance (V4.0).

New numerical features

- Generalize double backward implicit Euler time scheme for all variables. It can be activated with the keyword `ibdtso(ivar) = 1` (V4.0). Second order backward Euler scheme in time for velocity prediction since V3.3.
- Turbo-machinery modelling: enable multiple rotors for rotor-stator model based on mesh joining (V4.0). Turbo-machinery modelling: added a rotor-stator model based on mesh joining (V3.2).
- By default, do not force use of an iterative gradient reconstruction method for pressure gradients, or other gradients deriving from a potential. To force it, a negative value of the `imrgra` (-1, -2, -3) keyword may be used (V4.0).
- Add cell and face renumbering options to improve performance (V4.0).

New numerical features

- Generalize double backward implicit Euler time scheme for all variables. It can be activated with the keyword `ibdtso(ivar) = 1` (V4.0). Second order backward Euler scheme in time for velocity prediction since V3.3.
- Turbo-machinery modelling: enable multiple rotors for rotor-stator model based on mesh joining (V4.0).
  Turbo-machinery modelling: added a rotor-stator model based on mesh joining (V3.2).
- By default, do not force use of an iterative gradient reconstruction method for pressure gradients, or other gradients deriving from a potential. To force it, a negative value of the `imrgra` (-1, -2, -3) keyword may be used (V4.0).
- Add cell and face renumbering options to improve performance (V4.0).

Major changes for solvers (V4.0)

- Unify handling of linear solvers, so as to allow finer user control, and enable future additions of solver options and user-defined or external solvers.
- Single-reduction conjugate gradient is now an option rather than a separate solver. This allows switching automatically from one to the other based on computation vs. communication cost.
- Added a BiCGstab2 linear solver.
- Better handling of BiCGstab breakdown (non-convergence instead of error).
- Change default solver for pure diffusion problems (from conjugate gradient to multi-grid solver).

Major changes for solvers (V4.0)

- Unify handling of linear solvers, so as to allow finer user control, and enable future additions of solver options and user-defined or external solvers.
- Single-reduction conjugate gradient is now an option rather than a separate solver. This allows switching automatically from one to the other based on computation vs. communication cost.
- Added a BiCGstab2 linear solver.
- Better handling of BiCGstab breakdown (non-convergence instead of error).
- Change default solver for pure diffusion problems (from conjugate gradient to multi-grid solver).

Major changes for solvers (V4.0)

- Unify handling of linear solvers, so as to allow finer user control, and enable future additions of solver options and user-defined or external solvers.
- Single-reduction conjugate gradient is now an option rather than a separate solver. This allows switching automatically from one to the other based on computation vs. communication cost.
- Added a BiCGstab2 linear solver.
- Better handling of BiCGstab breakdown (non-convergence instead of error).
- Change default solver for pure diffusion problems (from conjugate gradient to multi-grid solver).

Major changes for solvers (V4.0)

- Unify handling of linear solvers, so as to allow finer user control, and enable future additions of solver options and user-defined or external solvers.
- Single-reduction conjugate gradient is now an option rather than a separate solver. This allows switching automatically from one to the other based on computation vs. communication cost.
- Added a BiCGstab2 linear solver.
- Better handling of BiCGstab breakdown (non-convergence instead of error).
- Change default solver for pure diffusion problems (from conjugate gradient to multi-grid solver).

Major changes for solvers (V4.0)

- Unify handling of linear solvers, so as to allow finer user control, and enable future additions of solver options and user-defined or external solvers.
- Single-reduction conjugate gradient is now an option rather than a separate solver. This allows switching automatically from one to the other based on computation vs. communication cost.
- Added a BiCGstab2 linear solver.
- Better handling of BiCGstab breakdown (non-convergence instead of error).
- Change default solver for pure diffusion problems (from conjugate gradient to multi-grid solver).

Parallel block Gauß-Seidel linear solver

- Really a Jacobi (inter-rank) - Gauß-Seidel (intra-rank) hybrid (V4.0)
- May be accelerated for "upwind" type systems by a matrix line ordering
- Used (by default) for the DOM radiation module, using the ordering defined by the radiation direction, for a factor of 2 to 4 4 improvement over the Jacobi solver (tested on a small number of MPI ranks; factor $>$ 5 for iterations, but each iteration more costly due to indirection and cache behavior).

Parallel block Gauß-Seidel linear solver

- Really a Jacobi (inter-rank) - Gauß-Seidel (intra-rank) hybrid (V4.0)
- May be accelerated for "upwind" type systems by a matrix line ordering
- Used (by default) for the DOM radiation module, using the ordering defined by the radiation direction, for a factor of 2 to 4 4 improvement over the Jacobi solver (tested on a small number of MPI ranks; factor >5 for iterations, but each iteration more costly due to indirection and cache behavior).

Parallel block Gauß-Seidel linear solver

- Really a Jacobi (inter-rank) - Gauß-Seidel (intra-rank) hybrid (V4.0)
- May be accelerated for "upwind" type systems by a matrix line ordering
- Used (by default) for the DOM radiation module, using the ordering defined by the radiation direction, for a factor of 2 to 4 4 improvement over the Jacobi solver (tested on a small number of MPI ranks; factor > 5 for iterations, but each iteration more costly due to indirection and cache behavior).

New numerical features

Boundary Conditions (BCs)

- Fix in the wall boundary conditions for the viscous boundary term (the viscous boundary term is not always parallel to the wall). This is mainly impacting for verification test-cases.
- Add a new Boundary Condition type for free inlet (`itypfb(ifac) = ifrent`), this BC can be used for natural convective flows in free atmosphere for instance (plumes, flame, etc.).
- Add a new Boundary Condition type for imposed mass flux at the inlet (`itypfb(ifac) = i_convective_inlet`), this BC can be used for imposing total ingoing mass of a scalars (V4.0).

New numerical features

Boundary Conditions (BCs)

- Fix in the wall boundary conditions for the viscous boundary term (the viscous boundary term is not always parallel to the wall). This is mainly impacting for verification test-cases.
- Add a new Boundary Condition type for free inlet (`itypfb(ifac) = ifrent`), this BC can be used for natural convective flows in free atmosphere for instance (plumes, flame, etc.).
- Add a new Boundary Condition type for imposed mass flux at the inlet (`itypfb(ifac) = i_convective_inlet`), this BC can be used for imposing total ingoing mass of a scalars (V4.0).

New numerical features

Boundary Conditions (BCs)

- Fix in the wall boundary conditions for the viscous boundary term (the viscous boundary term is not always parallel to the wall). This is mainly impacting for verification test-cases.
- Add a new Boundary Condition type for free inlet (`itypfb(ifac) = ifrent`), this BC can be used for natural convective flows in free atmosphere for instance (plumes, flame, etc.).
- Add a new Boundary Condition type for imposed mass flux at the inlet (`itypfb(ifac) = i_convective_inlet`), this BC can be used for imposing total ingoing mass of a scalars (V4.0).

A word about convective inlet

To impose the ingoing scalar mass flux

- Transport equation:

$$\frac{\partial \rho Y}{\partial t} + \operatorname{div}\left(Y \rho \underline{u}\right) = \operatorname{div}\left(K \underline{\nabla} Y\right) + ST_Y$$

$$\frac{\rho Y^{n+1} - \rho Y^{n}}{\Delta t} + \sum_f Y_f \dot{m}_f = \sum_f \underbrace{D_f\left(K, Y\right)}_{0 \text{ for } \texttt{i\_convective\_inlet} \text{ faces}} + ST_Y$$

A word about convective inlet

To impose the ingoing scalar mass flux

- Transport equation:

$$\begin{aligned}
\frac{\partial \rho Y}{\partial t} + \quad \operatorname{div}\left(Y \rho \underline{u}\right) &= \operatorname{div}\left(K \underline{\nabla} Y\right) && + ST_Y \\
\frac{\rho Y^{n+1} - \rho Y^{n}}{\Delta t} + \quad \sum_f Y_f \dot{m}_f &= \sum_f \underbrace{D_f\left(K, Y\right)}_{0 \text{ for } \texttt{i_convective_inlet} \text{ faces}} && + ST_Y
\end{aligned}$$

A word about Free Bernoulli Entrance

see the theory guide

- Bernoulli's relation:

$$
\begin{aligned}
P_f - \rho_f \underline{g} \cdot (\underline{x}_f - \underline{x}_0) + \frac{1+K}{2} \rho_f \underline{u}_f \cdot \underline{u}_f \\
= \\
P_\infty - \rho_\infty \underline{g} \cdot (\underline{x}_\infty - \underline{x}_0) + \frac{1}{2} \rho_\infty \underline{u}_\infty \cdot \underline{u}_\infty,
\end{aligned}
\tag{1}
$$

- K is a possible head loss of the fluid between the infinity and the boundary face entrance (which the user may play with to model the non-computed domain). K should be given in `rcodcl(ifac,ipr,2)`.
- The prediction-correction velocity-pressure coupling algorithm requires boundary conditions on the pressure increment (computed in the correction step), and therefore relation (1) is derived to obtain boundary conditions on the pressure increment δP

A word about Free Bernoulli Entrance

see the theory guide

- Bernoulli's relation:

$$P_f - \rho_f \underline{g} \cdot (\underline{x}_f - \underline{x}_0) + \frac{1+K}{2} \rho_f \underline{u}_f \cdot \underline{u}_f = P_\infty - \rho_\infty \underline{g} \cdot (\underline{x}_\infty - \underline{x}_0) + \frac{1}{2} \rho_\infty \underline{u}_\infty \cdot \underline{u}_\infty, \quad (1)$$

- K is a possible head loss of the fluid between the infinity and the boundary face entrance (which the user may play with to model the non-computed domain). K should be given in `rcodcl(ifac,ipr,2)`.
- The prediction-correction velocity-pressure coupling algorithm requires boundary conditions on the pressure increment (computed in the correction step), and therefore relation (1) is derived to obtain boundary conditions on the pressure increment δP

A word about Free Bernoulli Entrance

see the theory guide

- Bernoulli's relation:

$$P_f - \rho_f \underline{g} \cdot (\underline{x}_f - \underline{x}_0) + \frac{1+K}{2} \rho_f \underline{u}_f \cdot \underline{u}_f = P_\infty - \rho_\infty \underline{g} \cdot (\underline{x}_\infty - \underline{x}_0) + \frac{1}{2} \rho_\infty \underline{u}_\infty \cdot \underline{u}_\infty, \quad (1)$$

- K is a possible head loss of the fluid between the infinity and the boundary face entrance (which the user may play with to model the non-computed domain). K should be given in `rcodcl(ifac,ipr,2)`.
- The prediction-correction velocity-pressure coupling algorithm requires boundary conditions on the pressure increment (computed in the correction step), and therefore relation (1) is derived to obtain boundary conditions on the pressure increment δP

A word about Free Bernoulli Entrance

2D Helium plume

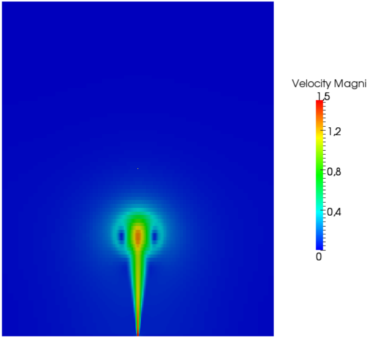


Figure: Velocity field

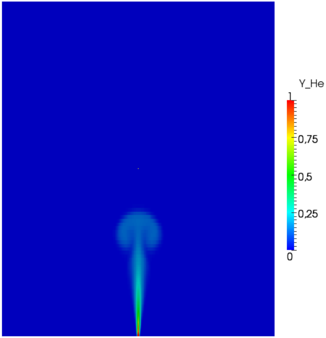


Figure: Helium fraction

Standard free outlet with $\dfrac{\partial u}{\partial n} = 0$ on the vertical sides

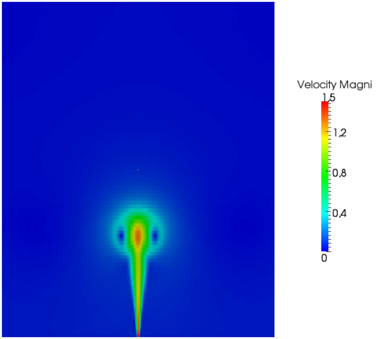


Figure: Velocity field

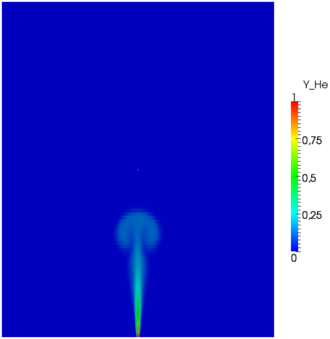


Figure: Helium fraction

Bernoulli free-inlet with $\dfrac{\partial u}{\partial n} = 0$ on the vertical sides

News for turbulence

- Move velocity wall functions to C (to share them with NCFD) (V3.2)
- Changes for *RSM* models:
  - The Daly Harlow model on the diffusive term is now by default for the SSG model (`iturb=31`) (V3.2)

$$\underline{\underline{\text{div}}}\left(\underline{\underline{Q}}_R\right) = \underline{\underline{\text{div}}}\left(C_R \frac{k}{\varepsilon} \underline{\underline{R}} \cdot \underline{\underline{\nabla}}\, \underline{\underline{R}}\right) \qquad (2)$$

  - The previous Shir model (isotropic diffusion) available with the `idirsm=0` (V3.3)

$$\underline{\underline{\text{div}}}\left(\underline{\underline{Q}}_R\right) = \underline{\underline{\text{div}}}\left(C_R \frac{k^2}{\varepsilon} \underline{\underline{\nabla}}\, \underline{\underline{R}}\right) \qquad (3)$$

  - The generic tensor diffusion brick is used for $R_{ij} - \epsilon$ (mesh robustness gain) (V3.2)
- Improve robustness of the time-stepping of the $k - \omega$ (`iturb=60`) model for low y^+ (V3.1)

News for turbulence

- Move velocity wall functions to C (to share them with NCFD) (V3.2)
- Changes for *RSM* models:
  - The Daly Harlow model on the diffusive term is now by default for the SSG model (`iturb=31`) (V3.2)

$$\underline{\underline{\operatorname{div}}} \left(\underline{\underline{Q}}_R \right) = \underline{\underline{\operatorname{div}}} \left(C_R \frac{k}{\varepsilon} \underline{\underline{R}} \cdot \underline{\underline{\nabla}}\, \underline{\underline{R}} \right) \tag{2}$$

  - The previous Shir model (isotropic diffusion) available with the `idirsm=0` (V3.3)

$$\underline{\underline{\operatorname{div}}} \left(\underline{\underline{Q}}_R \right) = \underline{\underline{\operatorname{div}}} \left(C_R \frac{k^2}{\varepsilon} \underline{\underline{\nabla}}\, \underline{\underline{R}} \right) \tag{3}$$

  - The generic tensor diffusion brick is used for $R_{ij} - \epsilon$ (mesh robustness gain) (V3.2)
- Improve robustness of the time-stepping of the $k - \omega$ (`iturb=60`) model for low y^+ (V3.1)

News for turbulence

- Move velocity wall functions to C (to share them with NCFD) (V3.2)
- Changes for *RSM* models:
  - The Daly Harlow model on the diffusive term is now by default for the SSG model (`iturb=31`) (V3.2)

$$\underline{\underline{\text{div}}}\left(\underset{\equiv R}{Q}\right) = \underline{\underline{\text{div}}}\left(C_R \frac{k}{\varepsilon} \underline{\underline{R}} \cdot \underline{\underline{\nabla}}\, \underline{\underline{R}}\right) \qquad (2)$$

  - The previous Shir model (isotropic diffusion) available with the `idirsm=0` (V3.3)

$$\underline{\underline{\text{div}}}\left(\underset{\equiv R}{Q}\right) = \underline{\underline{\text{div}}}\left(C_R \frac{k^2}{\varepsilon} \underline{\underline{\nabla}}\, \underline{\underline{R}}\right) \qquad (3)$$

  - The generic tensor diffusion brick is used for $R_{ij} - \epsilon$ (mesh robustness gain) (V3.2)
- Improve robustness of the time-stepping of the $k - \omega$ (`iturb=60`) model for low y^+ (V3.1)

News for turbulence

- Move velocity wall functions to C (to share them with NCFD) (V3.2)
- Changes for *RSM* models:
  - The Daly Harlow model on the diffusive term is now by default for the SSG model (`iturb=31`) (V3.2)

$$\underline{\underline{\text{div}}} \left(\underline{\underline{Q}}_R \right) = \underline{\underline{\text{div}}} \left(C_R \frac{k}{\varepsilon} \underline{\underline{R}} \cdot \underline{\underline{\nabla}}\, \underline{\underline{R}} \right) \tag{2}$$

  - The previous Shir model (isotropic diffusion) available with the `idirsm=0` (V3.3)

$$\underline{\underline{\text{div}}} \left(\underline{\underline{Q}}_R \right) = \underline{\underline{\text{div}}} \left(C_R \frac{k^2}{\varepsilon} \underline{\underline{\nabla}}\, \underline{\underline{R}} \right) \tag{3}$$

  - The generic tensor diffusion brick is used for $R_{ij} - \epsilon$ (mesh robustness gain) (V3.2)
- Improve robustness of the time-stepping of the $k - \omega$ (`iturb=60`) model for low y^+ (V3.1)

News for turbulence

- Move velocity wall functions to C (to share them with NCFD) (V3.2)
- Changes for *RSM* models:
  - The Daly Harlow model on the diffusive term is now by default for the SSG model (iturb=31) (V3.2)

$$\underline{\underline{\text{div}}}\left(\underline{\underline{Q}}_R\right) = \underline{\underline{\text{div}}}\left(C_R \frac{k}{\varepsilon} \underline{\underline{R}} \cdot \underline{\underline{\nabla}}\, \underline{\underline{R}}\right) \tag{2}$$

  - The previous Shir model (isotropic diffusion) available with the idirsm=0 (V3.3)

$$\underline{\underline{\text{div}}}\left(\underline{\underline{Q}}_R\right) = \underline{\underline{\text{div}}}\left(C_R \frac{k^2}{\varepsilon} \underline{\underline{\nabla}}\, \underline{\underline{R}}\right) \tag{3}$$

  - The generic tensor diffusion brick is used for $R_{ij} - \epsilon$ (mesh robustness gain) (V3.2)
- Improve robustness of the time-stepping of the $k - \omega$ (iturb=60) model for low y^+ (V3.1)

News for turbulence

- Move velocity wall functions to C (to share them with NCFD) (V3.2)
- Changes for *RSM* models:
  - The Daly Harlow model on the diffusive term is now by default for the SSG model (`iturb=31`) (V3.2)

$$\underline{\underline{\text{div}}}\left(\underline{\underline{Q}}_R\right) = \underline{\underline{\text{div}}}\left(C_R \frac{k}{\varepsilon} \underline{\underline{R}} \cdot \underline{\underline{\nabla}}\, \underline{\underline{R}}\right) \tag{2}$$

  - The previous Shir model (isotropic diffusion) available with the `idirsm=0` (V3.3)

$$\underline{\underline{\text{div}}}\left(\underline{\underline{Q}}_R\right) = \underline{\underline{\text{div}}}\left(C_R \frac{k^2}{\varepsilon} \underline{\underline{\nabla}}\, \underline{\underline{R}}\right) \tag{3}$$

  - The generic tensor diffusion brick is used for $R_{ij} - \epsilon$ (mesh robustness gain) (V3.2)
- Improve robustness of the time-stepping of the $k - \omega$ (`iturb=60`) model for low y^+ (V3.1)

New physical models

Fire

- Add a new dilatable (non conservative) algorithm for fire modelling. Activate it with `idilat=4` (the formulation is in "div $(\underline{u})$" instead of "div $(\rho \underline{u})$". You can access to the previous dedicated algorithm by setting `idilat=5` (V4.0).

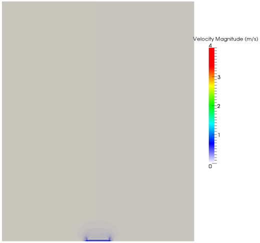


Figure: Velocity field

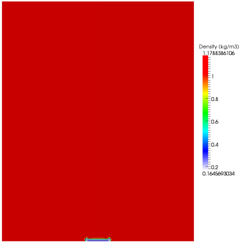


Figure: density field

New physical models

Water in unsaturated soils flows

- Add a new module solving the Richards equation with Darcy law. It can be activated setting the keyword `usppmo(idarcy) = 1`. This path includes new developments to improve gradient reconstruction calculation with heterogeneous diffusion coefficients. This feature is only available for standard least squares gradients and can be activated with the keyword `iwgrec(ivar) = 1` (V4.0).

New physical models

Coal combustion

- Add drift modelling for coal combustion, and clean up the module:
  - A model with a transported particle velocity per class is added (in fact, this velocity is handled as 3 scalars) (V4.0)
  - Now, the enthalpy of the continuous phase (gas phase) is transported rather than deduced (V4.0)
  - The convective flux for the gas phase is deduced from the convective fluxes of the particle classes and the convective flux of the bulk: therefore, the algorithm is fully conservative over time and space (V4.0)
  - Some coal combustion fields are renamed (V4.0)

New physical models

Coal combustion

- Add drift modelling for coal combustion, and clean up the module:
  - A model with a transported particle velocity per class is added (in fact, this velocity is handled as 3 scalars) (V4.0)
  - Now, the enthalpy of the continuous phase (gas phase) is transported rather than deduced (V4.0)
  - The convective flux for the gas phase is deduced from the convective fluxes of the particle classes and the convective flux of the bulk: therefore, the algorithm is fully conservative over time and space (V4.0)
  - Some coal combustion fields are renamed (V4.0)

New physical models

Coal combustion

- Add drift modelling for coal combustion, and clean up the module:
  - A model with a transported particle velocity per class is added (in fact, this velocity is handled as 3 scalars) (V4.0)
  - Now, the enthalpy of the continuous phase (gas phase) is transported rather than deduced (V4.0)
  - The convective flux for the gas phase is deduced from the convective fluxes of the particle classes and the convective flux of the bulk: therefore, the algorithm is fully conservative over time and space (V4.0)
  - Some coal combustion fields are renamed (V4.0)

New physical models

Coal combustion

- Add drift modelling for coal combustion, and clean up the module:
  - A model with a transported particle velocity per class is added (in fact, this velocity is handled as 3 scalars) (V4.0)
  - Now, the enthalpy of the continuous phase (gas phase) is transported rather than deduced (V4.0)
  - The convective flux for the gas phase is deduced from the convective fluxes of the particle classes and the convective flux of the bulk: therefore, the algorithm is fully conservative over time and space (V4.0)
  - Some coal combustion fields are renamed (V4.0)

New physical models

Coal combustion

- Add drift modelling for coal combustion, and clean up the module:
  - A model with a transported particle velocity per class is added (in fact, this velocity is handled as 3 scalars) (V4.0)
  - Now, the enthalpy of the continuous phase (gas phase) is transported rather than deduced (V4.0)
  - The convective flux for the gas phase is deduced from the convective fluxes of the particle classes and the convective flux of the bulk: therefore, the algorithm is fully conservative over time and space (V4.0)
  - Some coal combustion fields are renamed (V4.0)

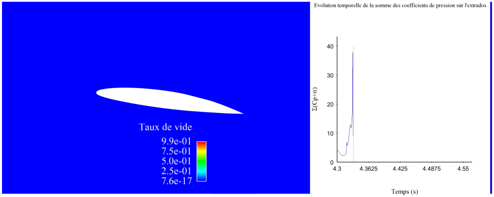
New physical models

Atmospheric module (V3.2)

- Add gaseous chemistry models.
- Plug the SIze REsolved Aerosol Model (SIREAM).

Cavitation

- Add cavitation models. See the documentation (theory, user, Doxygen) for more details. You can activate it in `cs_user_parameters.f90` with `icavit=1`.



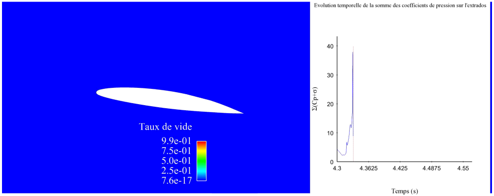
New physical models

Atmospheric module (V3.2)

- Add gaseous chemistry models.
- Plug the SIze REsolved Aerosol Model (SIREAM).

Cavitation

- Add cavitation models. See the documentation (theory, user, Doxygen) for more details. You can activate it in `cs_user_parameters.f90` with `icavit=1`.



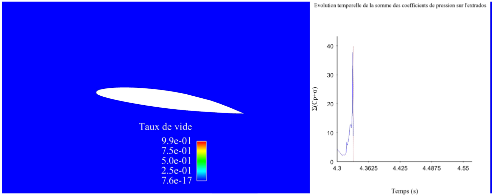
New physical models

Atmospheric module (V3.2)

- Add gaseous chemistry models.
- Plug the SIze REsolved Aerosol Model (SIREAM).

Cavitation

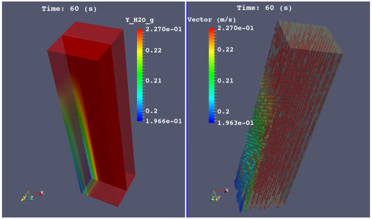
- Add cavitation models. See the documentation (theory, user, Doxygen) for more details. You can activate it in `cs_user_parameters.f90` with `icavit=1`.

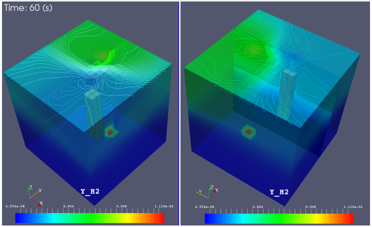


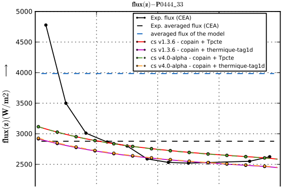
New physical models

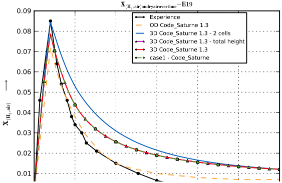
Severe accident simulation

- ALL the functionality of the V1.3-based extensions are available in V4.0, with clean user data defintitions (i.e. not requiring modification of non-user sources).









Overview

Tutorial documentation

- Moved tutorials outside the code-base, and into a separate base. This allows looser synchronization with the code base, as tutorials may be updated somewhat less frequently.
- Access to the new base:
  `git svn clone https://noeyy727.noe.edf.fr/mfee/saturne-doc/trunk saturne-doc`

Tutorial documentation

- Moved tutorials outside the code-base, and into a separate base. This allows looser synchronization with the code base, as tutorials may be updated somewhat less frequently.
- Access to the new base:

```
git svn clone
https://noeyy727.noe.edf.fr/mfee/saturne-doc/trunk
saturne-doc
```

html Doxygen documentation

Complete the Doxygen documentation of

- Fortran modules (all keywords from the User doc are documented)
- User examples (interactive display of advanced user data setting)
- Fortran routines headers (automatically translated from the current format, quality of the comments checked by the compilation)
- Fortran-C Naming reference

Available from

- The preconfigured sources (`make install-doc`)
- Launch it: `code_saturne info -g Doxygen`
- From the internet web-site `http://code-saturne.org/doxygen/src/index.html`

html Doxygen documentation

Complete the Doxygen documentation of

- Fortran modules (all keywords from the User doc are documented)
- User examples (interactive display of advanced user data setting)
- Fortran routines headers (automatically translated from the current format, quality of the comments checked by the compilation)
- Fortran-C Naming reference

Available from

- The preconfigured sources (`make install-doc`)
- Launch it: `code_saturne info -g Doxygen`
- From the internet web-site `http://code-saturne.org/doxygen/src/index.html`

html Doxygen documentation

Complete the Doxygen documentation of

- Fortran modules (all keywords from the User doc are documented)
- User examples (interactive display of advanced user data setting)
- Fortran routines headers (automatically translated from the current format, quality of the comments checked by the compilation)
- Fortran-C Naming reference

Available from

- The preconfigured sources (`make install-doc`)
- Launch it: `code_saturne info -g Doxygen`
- From the internet web-site `http://code-saturne.org/doxygen/src/index.html`

html Doxygen documentation

Complete the Doxygen documentation of

- Fortran modules (all keywords from the User doc are documented)
- User examples (interactive display of advanced user data setting)
- Fortran routines headers (automatically translated from the current format, quality of the comments checked by the compilation)
- Fortran-C Naming reference

Available from

- The preconfigured sources (`make install-doc`)
- Launch it: `code_saturne info -g Doxygen`
- From the internet web-site `http://code-saturne.org/doxygen/src/index.html`

html Doxygen documentation

Complete the Doxygen documentation of

- Fortran modules (all keywords from the User doc are documented)
- User examples (interactive display of advanced user data setting)
- Fortran routines headers (automatically translated from the current format, quality of the comments checked by the compilation)
- Fortran-C Naming reference

Available from

- The preconfigured sources (`make install-doc`)
- Launch it: `code_saturne info -g Doxygen`
- From the internet web-site `http://code-saturne.org/doxygen/src/index.html`

html Doxygen documentation

Complete the Doxygen documentation of

- Fortran modules (all keywords from the User doc are documented)
- User examples (interactive display of advanced user data setting)
- Fortran routines headers (automatically translated from the current format, quality of the comments checked by the compilation)
- Fortran-C Naming reference

Available from

- The preconfigured sources (`make install-doc`)
- Launch it: `code_saturne info -g Doxygen`
- From the internet web-site `http://code-saturne.org/doxygen/src/index.html`

html Doxygen documentation

Complete the Doxygen documentation of

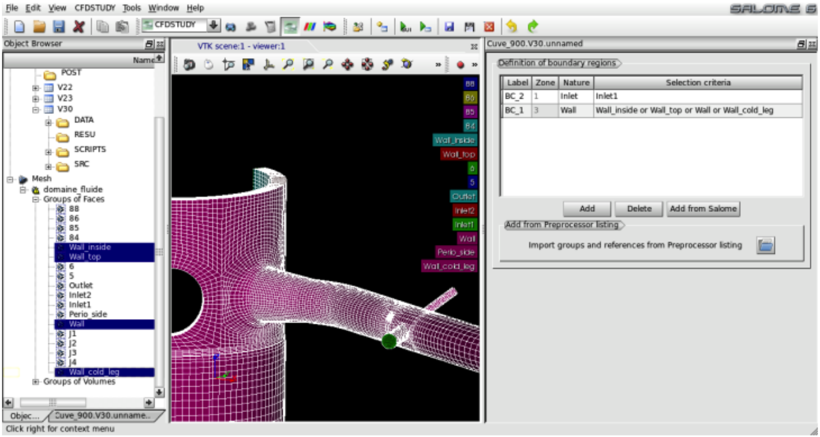
- Fortran modules (all keywords from the User doc are documented)
- User examples (interactive display of advanced user data setting)
- Fortran routines headers (automatically translated from the current format, quality of the comments checked by the compilation)
- Fortran-C Naming reference

Available from

- The preconfigured sources (`make install-doc`)
- Launch it: `code_saturne info -g Doxygen`
- From the internet web-site `http://code-saturne.org/doxygen/src/index.html`

Overview

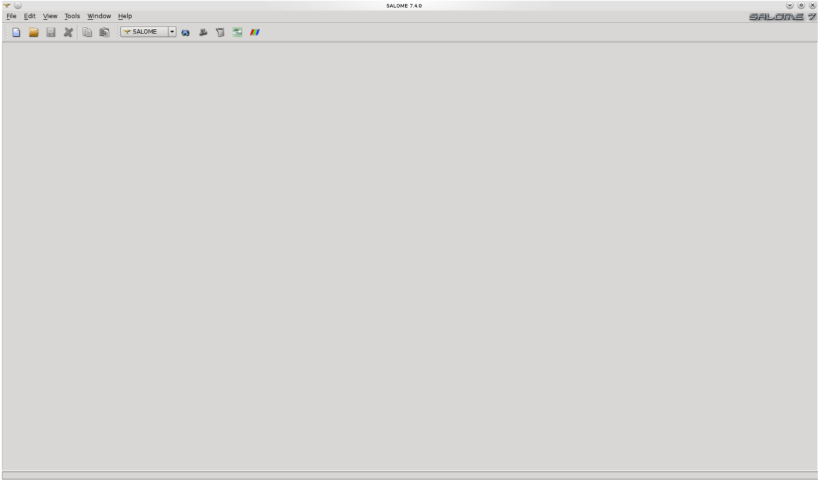
Graphical User Interface – SALOME _CFD



A common kernel with:

- Modules: GEOM, MESH, PARAVIS, YACS, ADAO, HOMARD, JOBMANAGER, OPENTURNS, ...
- Prerequisites: METIS, SCOTCH, HDF5, MED, CGNS, libccmio, ...
- SYRTHES, *Code_Saturne*

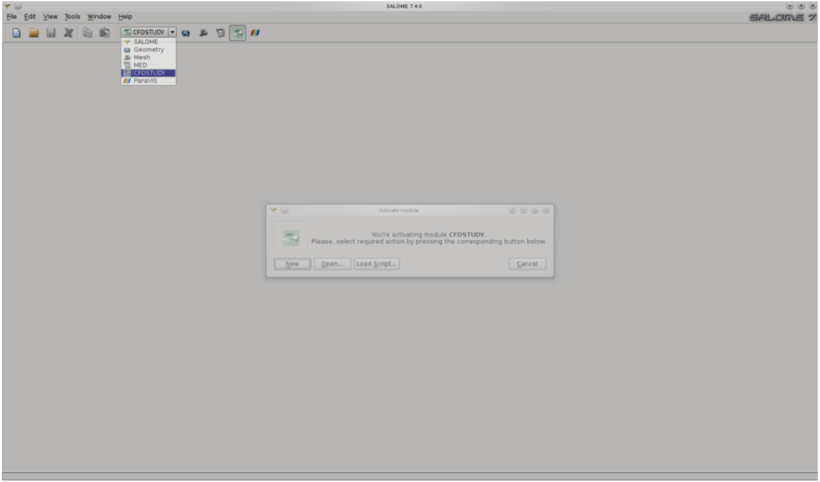
Graphical User Interface – SALOME _CFD



A common kernel with:

- Modules: GEOM, MESH, PARAVIS, YACS, ADAO, HOMARD, JOBMANAGER, OPENTURNS, ...
- Prerequisites: METIS, SCOTCH, HDF5, MED, CGNS, libccmio, ...
- SYRTHES, *Code_Saturne*

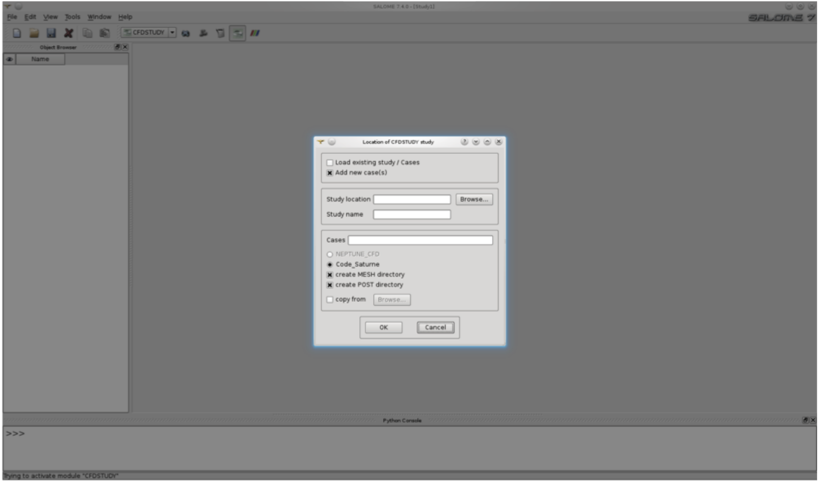
Graphical User Interface – SALOME _CFD



A common kernel with:

- Modules: GEOM, MESH, PARAVIS, YACS, ADAO, HOMARD, JOBMANAGER, OPENTURNS, ...
- Prerequisites: METIS, SCOTCH, HDF5, MED, CGNS, libccmio, ...
- SYRTHES, *Code_Saturne*

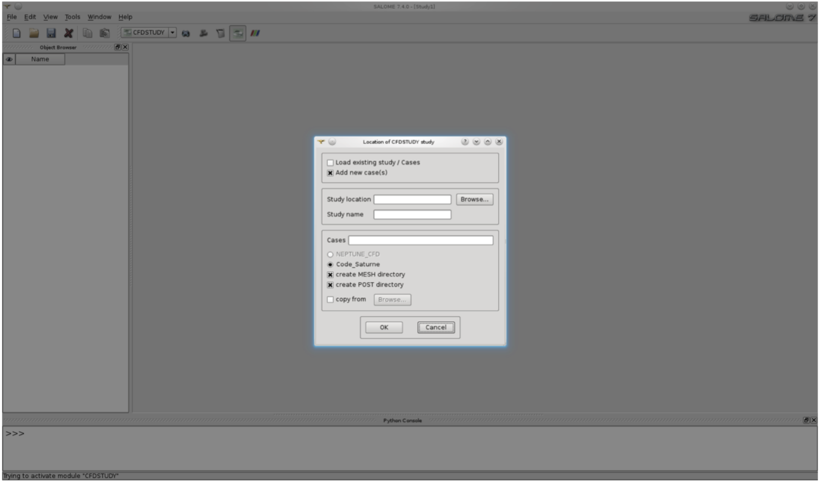
Graphical User Interface – SALOME _CFD



A common kernel with:

- Modules: GEOM, MESH, PARAVIS, YACS, ADAO, HOMARD, JOBMANAGER, OPENTURNS, ...
- Prerequisites: METIS, SCOTCH, HDF5, MED, CGNS, libccmio, ...
- SYRTHES, *Code_Saturne*

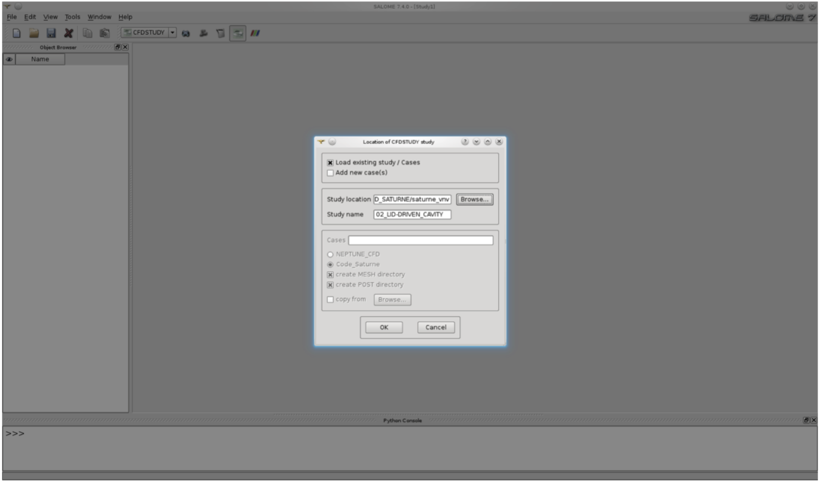
Graphical User Interface – SALOME _CFD



A common kernel with:

- Modules: GEOM, MESH, PARAVIS, YACS, ADAO, HOMARD, JOBMANAGER, OPENTURNS, ...
- Prerequisites: METIS, SCOTCH, HDF5, MED, CGNS, libccmio, ...
- SYRTHES, *Code_Saturne*

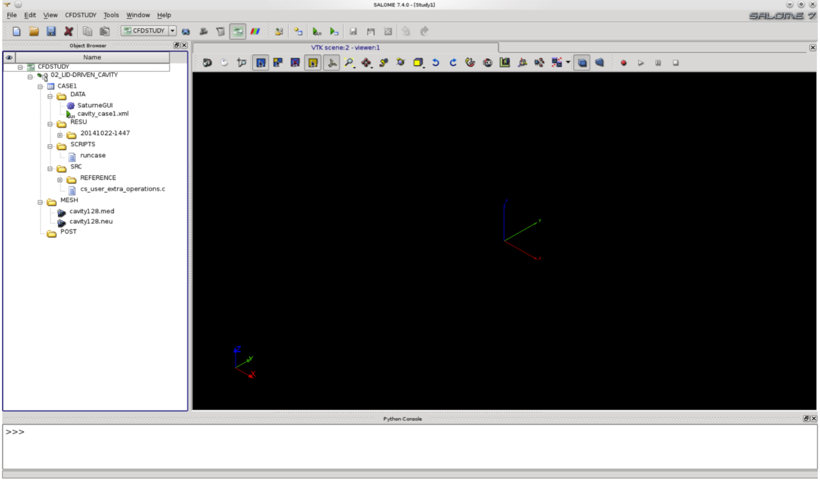
Graphical User Interface – SALOME _CFD



A common kernel with:

- Modules: GEOM, MESH, PARAVIS, YACS, ADAO, HOMARD, JOBMANAGER, OPENTURNS, ...
- Prerequisites: METIS, SCOTCH, HDF5, MED, CGNS, libccmio, ...
- SYRTHES, *Code_Saturne*

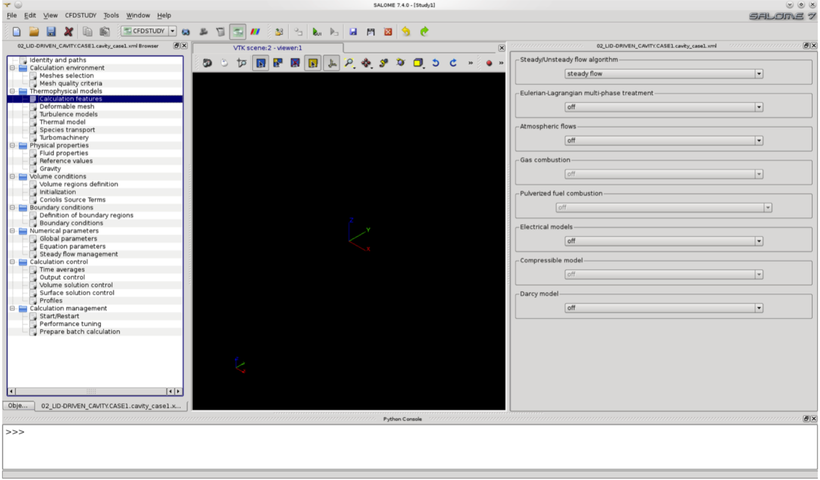
Graphical User Interface – SALOME _CFD



A common kernel with:

- Modules: GEOM, MESH, PARAVIS, YACS, ADAO, HOMARD, JOBMANAGER, OPENTURNS, ...
- Prerequisites: METIS, SCOTCH, HDF5, MED, CGNS, libccmio, ...
- SYRTHES, *Code_Saturne*

Graphical User Interface – SALOME _CFD



A common kernel with:

- Modules: GEOM, MESH, PARAVIS, YACS, ADAO, HOMARD, JOBMANAGER, OPENTURNS, ...
- Prerequisites: METIS, SCOTCH, HDF5, MED, CGNS, libccmio, ...
- SYRTHES, *Code_Saturne*

Windows port: available for testing

- Built using cross-compilation
  - Cygwin environment, cxfreeze, innosetup installer
  - installs with a wizard, familiar to Windows users
  - 32 and 64 bit versions
- Using most libraries used on Linux workstations and clusters
  - GUI (libxml2, Python-QT4)
  - MPI (from MS HPC pack, MPICH-based)
  - HDF5, MED, CGNS
  - Scotch
  - Doxygen and pdf documentation
- Validation
  - using .bat scripts for validation
  - Full test case suite runs at least on 10 iterations in 64 bit build
  - One Lagrangian cases crashes in 32 bit build
  - using MiKTeX for report generation works
- Yet to be done
  - generation of Windows SALOME-CFD build
  - *Code_Saturne*/ *Code_Saturne*/ Syrthes coupling

Windows port: available for testing

- Built using cross-compilation
  - Cygwin environment, cxfreeze, innosetup installer
  - installs with a wizard, familiar to Windows users
  - 32 and 64 bit versions
- Using most libraries used on Linux workstations and clusters
  - GUI (libxml2, Python-QT4)
  - MPI (from MS HPC pack, MPICH-based)
  - HDF5, MED, CGNS
  - Scotch
  - Doxygen and pdf documentation
- Validation
  - using .bat scripts for validation
  - Full test case suite runs at least on 10 iterations in 64 bit build
  - One Lagrangian cases crashes in 32 bit build
  - using MiKTeX for report generation works
- Yet to be done
  - generation of Windows SALOME-CFD build
  - *Code_Saturne*/ *Code_Saturne*/ Syrthes coupling

Windows port: available for testing

- Built using cross-compilation
  - Cygwin environment, cxfreeze, innosetup installer
  - installs with a wizard, familiar to Windows users
  - 32 and 64 bit versions
- Using most libraries used on Linux workstations and clusters
  - GUI (libxml2, Python-QT4)
  - MPI (from MS HPC pack, MPICH-based)
  - HDF5, MED, CGNS
  - Scotch
  - Doxygen and pdf documentation
- Validation
  - using .bat scripts for validation
  - Full test case suite runs at least on 10 iterations in 64 bit build
  - One Lagrangian cases crashes in 32 bit build
  - using MiKTeX for report generation works
- Yet to be done
  - generation of Windows SALOME-CFD build
  - *Code_Saturne*/ *Code_Saturne*/ Syrthes coupling

Windows port: available for testing

- Built using cross-compilation
  - Cygwin environment, cxfreeze, innosetup installer
  - installs with a wizard, familiar to Windows users
  - 32 and 64 bit versions
- Using most libraries used on Linux workstations and clusters
  - GUI (libxml2, Python-QT4)
  - MPI (from MS HPC pack, MPICH-based)
  - HDF5, MED, CGNS
  - Scotch
  - Doxygen and pdf documentation
- Validation
  - using .bat scripts for validation
  - Full test case suite runs at least on 10 iterations in 64 bit build
  - One Lagrangian cases crashes in 32 bit build
  - using MiKTeX for report generation works
- Yet to be done
  - generation of Windows SALOME-CFD build
  - *Code_Saturne*/ *Code_Saturne*/ Syrthes coupling

Overview

Verification (and Validation)

Automatic tool

- Access to *Code_Saturne* GUI functions in external preprocessing scripts (automatically set the PYTHONPATH variable)
- Merge the preprocessing and case running steps
- Global post-processing possibility to a study
- Possible run of the same case several times (prescribe the name of results directory)
- Now access to case description

Verification (and Validation)

Automatic tool

- Access to *Code_Saturne* GUI functions in external preprocessing scripts (automatically set the PYTHONPATH variable)
- Merge the preprocessing and case running steps
- Global post-processing possibility to a study
- Possible run of the same case several times (prescribe the name of results directory)
- Now access to case description

Verification (and Validation)

Automatic tool

- Access to *Code_Saturne* GUI functions in external preprocessing scripts (automatically set the PYTHONPATH variable)
- Merge the preprocessing and case running steps
- Global post-processing possibility to a study
- Possible run of the same case several times (prescribe the name of results directory)
- Now access to case description

Verification (and Validation)

Automatic tool

- Access to *Code_Saturne* GUI functions in external preprocessing scripts (automatically set the PYTHONPATH variable)
- Merge the preprocessing and case running steps
- Global post-processing possibility to a study
- Possible run of the same case several times (prescribe the name of results directory)
- Now access to case description

Verification (and Validation)

Automatic tool

- Access to *Code_Saturne* GUI functions in external preprocessing scripts (automatically set the PYTHONPATH variable)
- Merge the preprocessing and case running steps
- Global post-processing possibility to a study
- Possible run of the same case several times (prescribe the name of results directory)
- Now access to case description

Overview

Thank you for your attention...
Any question?